Semantic Region Retrieval from Spatial RDF Data

Dingming Wu[1(✉)], Can Hou[1], Erjia Xiao[1], and Christian S. Jensen[2]

[1] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
dingming@szu.edu.cn,
{houcan2017,xiaoerjia}@email.szu.edu.cn
[2] Department of Computer Science, Aalborg University, Aalborg, Denmark
csj@cs.aau.dk

Abstract. The top-k most relevant Semantic Place retrieval (kSP) query on spatial RDF data combines keyword-based and location-based retrieval. The query returns semantic places that are subgraphs rooted at a place entity with an associated location. The relevance to the query keywords of a semantic place is measured by a looseness score that aggregates the graph distances between the place (root) and the occurrences of the keywords in the nodes of the tree. We observe that kSP queries may retrieve semantic places that are spatially close to the query location, but with very low keyword relevance. When any single nearby place has low relevance, returning instead multiple relevant places maybe helpful. Hence, we propose a generalization of semantic place retrieval, namely semantic region (SR) retrieval. An SR query aims to return multiple places that are spatially close to the query location such that each place is relevant to one or more query keywords. An algorithm and optimization techniques are proposed for the efficient processing of SR queries. Extensive empirical studies with two real datasets offer insight into the performance of the proposals.

Keywords: Semantic region · Spatial RDF data · Query processing

1 Introduction

Large knowledge bases like DBpedia [1] and YAGO [2] typically adopt the Resource Description Framework (RDF) data model, which represents data as collections of (*subject*, *predicate*, *object*) triples. Currently, YAGO includes knowledge of more than 10 million entities (persons, organizations, cities, etc.) and contains more than 120 million facts about these entities. The English version of DBpedia describes 4.58 million entities, including 1,445,000 persons,

This work is supported in part by grant No. 2019A1515011721 from Natural Science Foundation of Guangdong, China and the DiCyPS project, funded by Innovation Fund Denmark.

Y. Nah et al. (Eds.): DASFAA 2020, LNCS 12113, pp. 415–431, 2020.
https://doi.org/10.1007/978-3-030-59416-9_25

735,000 places, 411,000 creative works, 241,000 organizations, 251,000 species, and 6,000 diseases.

RDF data is traditionally accessed using a structured query language, like SPARQL [20,25,30]. However, it is not friendly to common users, since query issuers need to understand the language itself and to be aware of the data domain. Hence, SPARQL limits data access mostly to domain experts. This leaves room for a *keyword search* model on RDF data [8,18,27]. This model allows common users to access RDF knowledge bases using ad-hoc keyword queries. RDF data can be modeled as a directed graph with subjects and objects as vertices and predicates as edges. In the keyword search model [18], outgoing edges from subjects that connect to types or literals are removed, and all the keywords in the URIs, types, and literals of such entities are collected to form a document of each vertex. A keyword query retrieves a set of (small) subgraphs, where the vertices in each subgraph collectively cover all the given keywords.

The recent *top-k relevant Semantic Place retrieval* (kSP) query [29] on spatial RDF data takes a query location and a set of query keywords as parameters and combines keyword-based and location-based retrieval. The query returns k *semantic places* that are subgraphs rooted at place entities that have associated locations. Specifically, a kSP query returns the top-k *Tightest Qualified Semantic Places* (TQSP) according to a scoring function that considers both the spatial distance of a semantic place to the query location and the graph proximity of the occurrences of the query keywords in the RDF graph of a place. A *qualified semantic place* satisfies two conditions: (i) it is a tree rooted at a *place entity* (i.e., a vertex in the RDF graph associated with a spatial location), (ii) the documents associated with the vertices in the tree collectively cover all query keywords. A *looseness* score of a qualified semantic place is an aggregate of the graph distances between the place (root) and the occurrences of the query keywords covered by the nodes of the tree rooted at the place [8,18,27]. The kSP query returns the k places with the smallest *combined looseness and spatial distance* with respect to the query parameters. However, we observe that kSP queries may retrieve places that are spatially close to the query, but with poor looseness score, or places that have good looseness score, but far from the query. Consider the example kSP query q in Fig. 1 with keywords "childhood" and "scientific." The top-1 semantic place, shown in Fig. 3, is rooted at p_1. Although p_1 is spatially close to the query location in Fig. 1, the looseness of the semantic place rooted at p_1 is large, i.e., "scientific" is six edges away and "childhood" is eight edges away from p_1, which means that p_1 may not satisfy the user's intent. When a single place cannot satisfy a user's intent, returning instead several relevant places that are close to each other may be helpful. In practice, they can be considered as one place, since users can easily visit them. Take again query q in Fig. 1 with keywords "childhood" and "scientific" as an example. Returning two spatially close places p_2 and p_3 is more helpful than returning p_1. Places p_2 and p_3 collectively satisfy the user's intent, since "childhood" is close to p_2 and "scientific" is close to p_3 on the RDF graph, shown in Fig. 2.

Motivated by the above observation, we propose a generalization of semantic place retrieval called semantic region (SR) retrieval. Specifically, an SR query takes a spatial range and a set of query keywords as arguments and returns the qualified semantic region that minimizes a scoring function. The semantic region is composed of a subgraph $T(r, P)$ connecting a set of places that are in the query spatial range and the so-called keyword-relevant paths of the query keywords. The scoring function considers both the graph proximity of the occurrences of query keywords in the RDF graph to the places and the graph proximity among the places. An SR query aims to retrieve multiple places that are spatially close to the query location such that each place is relevant to one or more the query keywords.

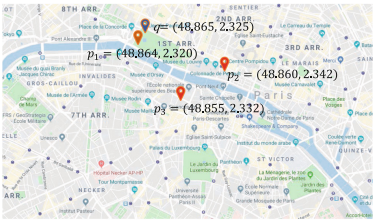


Fig. 1. Map of places in Figs. 2 and 3 and query location.

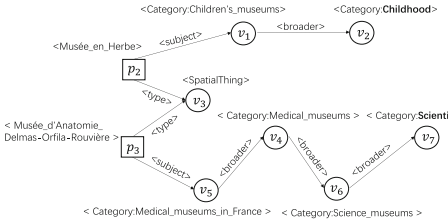


Fig. 2. Semantic region.

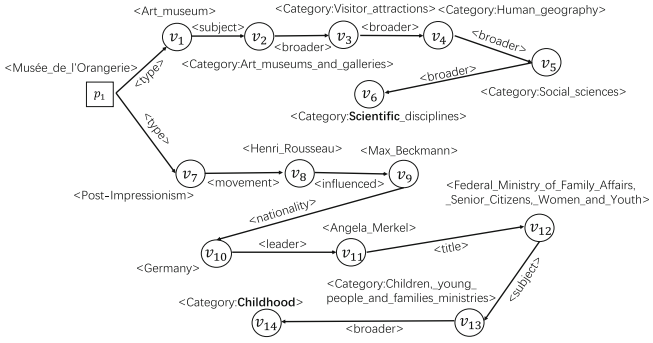


Fig. 3. Semantic place.

A straightforward method to process the SR query is to first find the place set P_0 in the query spatial range and then to enumerate all the subsets of P_0. The next step is to construct the semantic region for each subset of places and to compute its score. The semantic region with the smallest score is returned. However, this method is inefficient, since considering all subsets of P_0 is time consuming and unnecessary. In addition, constructing the semantic region for

a subset of places is expensive. It involves finding the graph proximity of the query keywords to each place and computing the graph proximity among the places. We propose the SRRA algorithm for efficiently computing SR queries. A lemma guarantees that the number of places in the semantic region of an SR query cannot exceed the number of query keywords, thus reducing the number of subsets of places to be computed. Also, it first finds a candidate semantic region of the query and derives a lower bound on the scores of un-computed subsets. Then, the subsets with bound no less than the candidate can be pruned. In addition, a lower bound is derived on the graph proximity among the places. During the process of constructing $T(r, P)$, this bound is used to prune unpromising semantic regions. To further improve the performance of the SRRA algorithm, two additional pruning rules are proposed. The SRRA algorithm first considers subsets of places of size 2 and then expands small subsets to larger subsets by adding places. The proposed pruning rules use the places already in the processed subsets to prune the places to be added, so that more subsets of places can be pruned.

Outline. Section 2 defines the semantic region retrieval query and relevant concepts. The SRRA algorithm for computing SR queries is presented in Sect. 3, and optimizations of SRRA are covered in Sect. 4. Our empirical study is the subject of Sect. 5. Related work is reviewed in Sect. 6, and we conclude in Sect. 7.

2 Problem Definition

An RDF data set is a collection of (*subject, predicate, object*) triples, where *subjects* are entities linked to *objects* (other entities, types, or literals) via *predicates*. Such a data set can be modeled as a directed graph $G = (V, E)$, where vertices refer to entities, and edges represent connections between entities based on *predicates*. Some entities are associated with spatial coordinates λ. We call such entities places. We use v to denote any vertex in an RDF graph, while p denotes a place vertex. In accordance with previous work on querying spatial RDF data [29], we construct, for each entity, a document ψ from the entity's URI and literals. In addition, for each triple, the description of the predicate is added to the document of the object entity.

Definition 1 ***Keyword-Relevant Path.*** *Given an RDF graph $G = (V, E)$, a keyword w, and a place p, a w-relevant path of p is a path with the fewest edges from p to a vertex whose document contains w. Formally, let $\gamma(p, v)$ be the shortest path from place p to vertex v, and let $d(p, v)$ be the length (number of edges) of $\gamma(p, v)$. Let $V(w)$ be the set of vertices whose documents contain keyword w. A* ***w-relevant path of*** *p is defined as $\Gamma_w(p) = \gamma(p, v^*)$, where $v^* = \arg\min_{v \in V(w)} d(p, v)$.*

Definition 2 ***Keyword-Distance of a Place.*** *In an RDF graph $G = (V, E)$, the distance between a place p and a keyword w, denoted by $d_g(p, w)$, is the number of edges in $\Gamma_w(p)$ (a w-relevant path of p).*

According to Definitions 1 and 2, a place p is relevant to a keyword w (denoted as $p \sim w$) if it is connected to a vertex whose document contains the keyword. A small keyword-distance of a place indicates that this place is relevant to the keyword. If no documents of any of the vertices connected to a place p (including the document of p itself) contain keyword w, the keyword-distance $d_g(p, w)$ is undefined. In this case, place p is irrelevant to keyword w.

Definition 3 ***Keyword-Distance of a set of Places.*** *In an RDF graph $G = (V, E)$, the distance between a keyword w and a set of places P is defined as $d_g(P, w) = \min_{p \in P} d_g(p, w)$.*

Example 1. Figure 4 shows an example RDF graph. Figure 5 shows (part of) the documents attached to the vertices in Fig. 4. Considering keyword w_3, there are two paths from p_2 to the vertices (i.e., v_1 and v_6) whose documents contain w_3. The w_3-relevant path of p_2 is the shorter path, i.e., $\Gamma_{w_3}(p_2) = p_2 \rightarrow v_5 \rightarrow v_6$. The distance between p_2 and w_3 is $d_g(p_2, w_3) = 2$. Similarly, the w_3-relevant path of p_1 is $\Gamma_{w_3}(p_1) = p_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_1$. The distance between p_1 and w_3 is $d_g(p_1, w_3) = 3$. Consider the set of places $P = \{p_1, p_2\}$. The distance between w_3 and P is $d_g(P, w_3) = 2$.

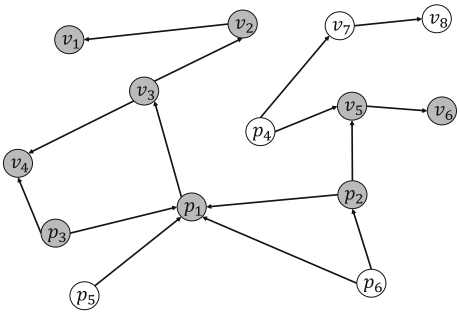


Fig. 4. RDF graph.

Vertices	Document
v_1	w_2, w_3, w_5
v_4	w_1, w_4, w_5
v_6	w_3, w_6
v_8	w_1, w_5, w_6

Fig. 5. Documents.

A *semantic region* $R(P) = (T(r, P), \{\Gamma_w(p) \mid p \in P\})$ is a connected subgraph of the RDF graph. It consists of a tree $T(r, P)$ with root r and the places in P as leaves, and the keyword-relevant paths of the places in P. A special case is a semantic region with only one place, i.e., $|P| = 1$, which is called a semantic place [29]. This paper studies the case where a semantic region contains at least two places, i.e., $|P| > 1$.

A **Semantic Region (SR) retrieval query** q takes two parameters: a spatial range $q.r$ and a keyword set $q.\psi$. Given an SR query q, a qualified semantic region $R(P)$ of q satisfies two properties: (i) all the places in $R(P)$ are in the query spatial range $q.r$, and (ii) for each keyword w in $q.\psi$, there exists a relevant place p in $R(P)$. The qualified semantic region w.r.t. an SR query is formally defined in Definition 4.

Definition 4 ***Qualified Semantic Region (QSR).*** *Given an SR query q, a qualified semantic region $R_q(P) = (T(r, P), \{\Gamma_w(p) \mid p \in P, w \in q.\psi\})$ satisfies $\forall p \in P\ (p.\lambda \in q.r) \wedge \forall w \in q.\psi\ \exists p \in P\ (\Gamma_w(p) \neq \emptyset)$.*

Definition 5 ***Semantic Region Retrieval.*** *Given an SR query q on an RDF graph G, the result of q is the qualified semantic region $R_q(P) = (T(r, P), \{\Gamma_w(p) \mid p \in P, w \in q.\psi\})$ that minimizes the following scoring function.*

$$f(R_q(P)) = \alpha \cdot \frac{\min(cscore(T(r,P)), L)}{L} + (1-\alpha) \cdot \frac{\min(kscore(P), L)}{L}$$

$$cscore(T(r,P)) = \sum_{p \in P} d(r,p) \qquad kscore(P) = \max_{w \in q.\psi} d_g(P,w)$$

where L is the maximum allowed number of edges (e.g., L=10), $cscore(T(r, P))$ is the graph proximity of the places in P, $kscore(P)$ indicates the relevance of the set of places P w.r.t. the query keywords, and parameter α is used to balance the importance of $cscore(T(r, P))$ versus $kscore(P)$.

Given an SR query, if there exist multiple QSRs that minimize the scoring function, the firstly found such QSR is returned as the answer for simplicity.

Example 2. Consider an SR query q with keywords $q.\psi = \{w_1, w_2, w_3\}$. Figure 5 shows (part of) the documents attached to the vertices in Fig. 4. The omitted content does not contain any of the query keywords. Assuming that all the places in Fig. 4 are in the query spatial range $q.r$, the result of q is the qualified semantic region $R_q(P) = (T(r, P), \{\Gamma_w(p)\})$ (the gray part in Fig. 4), where $P = \{p_1, p_2, p_3\}$, $r = p_1$, $\Gamma_{w_1}(p_3) = p_3 \rightarrow v_4$, $\Gamma_{w_2}(p_1) = p_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_1$, and $\Gamma_{w_3}(p_2) = p_2 \rightarrow v_5 \rightarrow v_6$. Given $\alpha = 0.5$, its score is $0.5 \times 2/10 + 0.5 \times 3/10 = 0.25$, where $cscore(T(r, P)) = 2$ and $kscore(P) = 3$.

3 Semantic Region Retrieval Algorithm

3.1 Data Structures

The semantic region retrieval algorithm (SRRA) uses three main data structures. An R*-tree indexes the locations of all places in the RDF graph. A disk-resident inverted index I indexes the documents of all vertices in the RDF graph. This index consists of two main components: (1) a vocabulary of all distinct terms in the collection of documents and (2) a posting list for each term t in the vocabulary. The posting list for term t is a list of the *ids* of all vertices v whose document ψ contains term t. An additional disk-resident inverted index I^α indexes the α-radius word neighborhoods of all places in the RDF graph [29]. The α-radius word neighborhood $WN(p)$ of a place p contains the set of word-distance pairs $\{(w_i, d_g(p, w_i))\}$, where the shortest graph distance from p to w_i is no larger than α, i.e., $d_g(p, w_i) \leq \alpha$.

Algorithm 1. SRRA$(q, Rtree, G, I, I^\alpha)$

```
1: for each keyword w_i in q.ψ do
2:     Load posting list pl_i of w_i from I
3:     Load posting list pl_i^α of w_i from I^α
4: P_0 ← Rtree.RangeQuery(q.r)
5: for each keyword w in q.ψ do
6:     v ← arg_p min d_g(p, w)
7:     Add v to P_c
8: Compute f(R_q(P_c))
9: R^c ← R_q(P_c)
10: j ← 2
11: while j ≤ |q.ψ| do                                   ▷ Lemma 1
12:     if P ← GenerateCover(R^c, j) ≠ NULL then
13:         for each P_i in P do
14:             Compute {Γ_w(p) | p ∈ P_i, w ∈ q.ψ}
15:             Compute T(r, P_i)          ▷ Lemma 4 is applied for pruning.
16:             Compute f(R_q(P_i))
17:             if f(R^c) > f(R_q(P_i)) then
18:                 R^c ← R_q(P_i)
19:     j++
20: return R^c
```

3.2 Algorithm SRRA

The semantic region retrieved by the *SR* query q is formed by the subset of places in the query spatial range that minimizes the scoring function $f(R_q(P))$. A straightforward way of finding the result is to compute the scores of the qualified semantic regions formed by each subset of the places in the query spatial range. This method is inefficient. First, the number of the subsets of places in the query spatial range may be large; second, constructing the semantic region formed by a set of places P is expensive as it involves computing the keyword-relevant path of each place $\Gamma_w(p)$ for each query keyword and the subgraph $T(r, P)$ that connects all the places in P. Algorithm SRRA tries to reduce the search space in three ways. (i) We show that the number of places in the result region cannot exceed $|q.\psi|$, so SRRA prunes subsets of places larger than $|q.\psi|$. (ii) SRRA first computes a candidate QSR with the best $kscore(P)$ using index structure I^α. It also derives bounds on the scores of other subsets of places. Then, subsets with bounds larger than the score of the candidate QSR are pruned. (iii) SRRA derives a lower bound on the $cscore(T(r, P))$ of the QSRs. During the process of constructing $T(r, P)$, this bound is used for early termination of the computation of unpromising QSRs.

Algorithm 1 shows the pseudo code of SRRA. Given an *SR* query q, it first loads the posting lists $\{pl_i\}$ and $\{pl_i^\alpha\}$ of each query keyword w_i from both I and I^α. Then the places P_0 in the query range $q.r$ are retrieved from the R*-tree via a range query. Next, a candidate QSR R^c is constructed by using the nearest places of each query keyword, i.e., $v \leftarrow \arg_p \min d_g(p, w)$. Ties are broken

arbitrarily. Having the candidate QSR R^c, the algorithm tries to find a better QSR in terms of $f(R_q(P))$ from the places in P_0. If one is found, R^c is replaced. In the end, R^c is returned as the answer to query q.

To find a better QSR, algorithm SRRA generates place sets of size j, $j \in [2, |q.\psi|]$, from P_0, since an SR query aims to retrieve a result containing at least two places and at most $|q.\psi|$ places (according to Lemma 1). However, considering all place sets of size j is time consuming. Algorithm SRRA instead calls Algorithm 2 to obtain promising place sets. Algorithm 2 derives a lower bound $kd^B(P^j)$ (Lemma 2) on the keyword distance of each place set and a lower bound $f^B(R_q(P^j))$ (Lemma 3) on the score of the QSR formed by each place set. The place sets $\mathcal{P}$ whose lower bounds on their scores are smaller than the score of the candidate QSR are returned to Algorithm 1 to compute the corresponding QSRs $R_q(P_i)$. To compute the QSR for a set of places P_i, the keyword-relevant path of each place in P_i for each query keyword is calculated using a variant of Dijkstra's Algorithm [29]. Next, $T(r, P)$ is computed using an existing Algorithm [18]. Since $T(r, P)$ is constructed incrementally, Lemma 4 is applied to terminate the computation early if the current QSR cannot obtain a better score than the candidate.

Lemma 1. *The result of an SR query q contains at most $|q.\psi|$ places.*

Proof. Suppose the result $R_q(P)$ of an *SR* query q contains $|q.\psi| + n$ places, i.e., $|P| = |q.\psi| + n$. Create a subset P' of P in the following way: for each query keyword $w_i \in q.\psi$, let p_i be the place in P having the shortest keyword distance, i.e., $p_i = \arg\min_{p_i \in P} d_g(p, w)$; add p_i to P'. Then, we have $cscore(T(r, P)) > cscore(T(r, P'))$, since $P' \subset P$. Also, by construction, $kscore(P) \geq kscore(P')$. Hence, we obtain $f(R(P)) > f(R(P'))$, which leads to a contradiction. In addition, $|P'| \leq |q.\psi|$. The equality holds when each place added to P' is distinct. This proves the lemma.

Algorithm 2. GenerateCover(R^c, j)

```
1: for each P^j do
2:     Compute kd^B(P^j)
3:     Compute f^B(R_q(P^j))                          ▷ Pruning Rule 1
4:     if f^B(R_q(P^j)) < f(R^c) then
5:         Add P^j to P
6: return P
```

Lemma 2. *Given a place set P^j, for each place $p \in P^j$ and for each keyword $w \in q.\psi$, if p does not have the word-distance pair $\{(w, d_g(p, w))\}$ in α-radius word neighborhood $WN(p)$, we set $d_g(p, w) = \alpha + 1$. Otherwise, $d_g(p, w)$ is the value in $WN(p)$. A lower bound on $kscore(P^j)$ is as follows:*

$$kd^B(P^j) = \max_{w \in q.\psi} d_g(P, w) = \max_{w \in q.\psi} \min_{p \in P} d_g(p, w)$$

Proof. For each place $p \in P^j$ and for each keyword $w \in q.\psi$, if p does not have the word-distance pair $\{(w, d_g(p,w))\}$ in α-radius word neighborhood $WN(p)$, it must hold that $d_g(p,w) \geq \alpha + 1$. When computing $kd^B(P^j)$ for such places, $\alpha+1$ is used instead of their real keyword distances. And $kd^B(P^j)$ is computed based on all real keyword distances. Obviously, $kd^B(P^j) \leq kscore(P^j)$.

Lemma 3. *Given a place set P^j, a lower bound on the score of semantic region $R_q(P^j)$ is as follows:*

$$f^B(R_q(P^j)) = \alpha \cdot \frac{\min(|P^j| - 1, L)}{L} + (1-\alpha) \cdot \frac{\min(kd^B(P^j), L)}{L}$$

Proof. Given a place set P^j, $cscore(T(r,P))$ takes its minimum value $|P^j| - 1$ when one of the places is the root and all the other places connect to the root via one edge. According to Lemma 2, $kd^B(P^j) \leq kscore(P^j)$. Hence, we obtain $f^B(R_q(P^j)) \leq f(R_q(P^j))$.

Pruning Rule 1. *Given an SR query q, let P be the subset of the places in the query spatial range generated by Algorithm 2. According to Lemma 3, if $f^B(R_q(P)) \geq f(R^c)$, P cannot be the result and is pruned.*

Lemma 4. *Let R^c be the candidate QSR. A lower bound on $cscore(T(r,P))$ is defined as follows (normalization parameter L is ignored for convenience):*

$$cscore^B(P) = \frac{f(R^c) - (1-\alpha) \cdot kscore(P)}{\alpha}$$

Proof. First, $f(R^c)$ is the score of the candidate QSR. Second, $kscore(P)$ is the keyword distances of the places in P. Then $cscore^B(P)$ is derived according the scoring function using $f(R^c)$ and $kscore(P)$, which is the worst allowed value for $R(P)$ to become the result.

Example 3. Consider the SR query q with keywords $q.\psi = \{w_1, w_2, w_3\}$ in the running example in Fig. 4 and 5, assuming that all the places are in the query range $q.r$, i.e., $P_0 = \{p_1, p_2, \cdots, p_6\}$. Inverted index I^α contains the α-radius word neighborhood of all the places. The candidate QSR R^c is formed by place set $P_c = \{p_1, p_2, p_3\}$, and $f(R_q(P_c)) = 0.5 \times 2/10 + 0.5 \times 3/10 = 0.25$, where $\alpha = 0.5$, $cscore(T(r,P_c)) = 2$ and $kscore(P_c) = 3$. Consider place set $P = \{p_2, p_3, p_4\}$. Here, $kd^B(P) = \max\{1,4,2\} = 4$, and $f^B(R_q(P)) = 0.5 \times 2/10 + 0.5 \times 4/10 = 0.3 > f(R_q(P_c))$. According to Pruning Rule 1, set P cannot be the result and is pruned.

4 Optimization

In order to find the subset of places in the query spatial range of an SR query q that minimizes the scoring function, algorithm SRRA generates place sets of size j, $j \in [2, |q.\psi|]$. For each generated place set, it is expensive to compute the

corresponding semantic region. Although algorithm SRRA uses a candidate QSR to prune some subsets of places, many subsets still have to be computed when the number of query keywords is large. Next, we propose two pruning techniques to further prune subsets of places that cannot be the result. The correctness of the pruning rules are guaranteed by Lemmas 5 and 6.

Definition 6. *Place p_i is dominated by place p_j with respect to keyword set ψ, denoted by $p_j \preceq_\psi p_i$, if $\forall w \in \psi\ (d_g(p_j, w) \leq d_g(p_i, w))$.*

Lemma 5. *Given an SR query q, let P_1 and P_2 be subsets of places in the query spatial range, and let $P_1 \subset P_2$. If $\forall p \in P_2 \setminus P_1\ \exists p_0 \in P_1\ (p_0 \preceq_\psi p)$ then $f(R_q(P_1)) \leq f(R_q(P_2))$.*

Proof. Since $P_1 \subset P_2$ and $\forall p \in P_2 \setminus P_1\ \exists p_0 \in P_1\ (p_0 \preceq_\psi p)$, we have $kscore(P_1) \leq kscore(P_2)$. Because $P_1 \subset P_2$, we have $cscore(T(r, P_1)) \leq cscore(T(r, P_2))$. This proves that $f(R_q(P_1)) \leq f(R_q(P_2))$.

Pruning Rule 2. *Given an SR query q, let P be the subset of the places in the query spatial range generated by Algorithm 2. Let $\mathcal{P}$ contain all the sets of places in the query spatial range and $\forall P_i \in \mathcal{P}\ (P \subseteq P_i)$. If $\forall p \in P_i \setminus P\ \exists p_0 \in P\ (p_0 \preceq_\psi p)$, according to Lemma 5, $R_q(P_i)$ cannot have a better score than $R_q(P)$ and can be pruned.*

Example 4. Consider the RDF graph in Fig. 4. Given $\psi = \{w_1, w_2, w_3\}$, let $P_1 = \{p_1, p_2\}$ and $P_2 = \{p_1, p_2, p_5\}$. According to Definition 6, $p_1 \preceq_\psi p_5$, since $\forall w \in \psi\ d_g(p_1, w) \leq d_g(p_5, w)$. According to Pruning Rule 2, P_2 can be pruned.

Definition 7. *Place p_i is dominated by a set of places P with respect to a keyword set ψ, denoted by $P \preceq_\psi p_i$, if $\forall w \in \psi\ \exists p_j \in P\ (d_g(p_j, w) \leq d_g(p_i, w))$.*

Lemma 6. *Given an SR query q, let P_1 and P_2 be subsets of places in the query range, and let $P_1 \subset P_2$. If $\forall p \in P_2 \setminus P_1\ \exists P' \subseteq P_1\ (P' \preceq_\psi p)$, $f(R_q(P_1)) \leq f(R_q(P_2))$.*

Proof. Since $P_1 \subset P_2$ and $\forall p \in P_2 \setminus P_1\ \exists P' \subseteq P_1\ (P' \preceq_\psi p)$, we have $kscore(P_1) \leq kscore(P_2)$. Because $P_1 \subset P_2$, we have $cscore(T(r, P_1)) \leq cscore(T(r, P_2))$. This proves that $f(R_q(P_1)) \leq f(R_q(P_2))$.

Pruning Rule 3. *Given an SR query q, let P be the subset of the places in the query spatial range generated by Algorithm 2. Let $\mathcal{P}$ contain all the sets of places in the query spatial range, and let $\forall P_i \in \mathcal{P}\ (P \subseteq P_i)$. If $\forall p \in P_i \setminus P\ \exists P' \subseteq P\ (P' \preceq_\psi p)$, according to Lemma 6, $R_q(P_i)$ cannot have a better score than $R_q(P)$ and can be pruned.*

Example 5. Consider the RDF graph in Fig. 4. Given $\psi = \{w_1, w_2, w_3\}$, let $P_1 = \{p_1, p_2\}$ and $P_2 = \{p_1, p_2, p_6\}$. According to Definition 7, $P_1 \preceq_\psi p_6$, since $d_g(p_1, w_1) < d_g(p_6, w_1)$, $d_g(p_1, w_2) < d_g(p_6, w_2)$, and $d_g(p_2, w_3) < d_g(p_6, w_3)$. According to Pruning Rule 3, P_2 can be pruned.

5 Empirical Study

This section evaluates the performance of the SRRA algorithm (proposed in Sect. 3) and of SRRA*, which is the SRRA algorithm combined with the optimization techniques presented in Sect. 4.

5.1 Data and Queries

The data used in our experiments are from DBpedia and Yago (version 2.5). DBpedia's directed RDF graph contains 8,099,955 vertices and 72,193,833 edges, and the dictionary contains 2,927,026 unique words. The documents of all vertices are organized by an inverted index. The average posting list length is 56.46, which means that a word appears on average in the documents of 56.46 vertices in the graph. Among all vertices, 883,665 are places with coordinates. Yago's directed RDF graph has 8,091,179 vertices and 50,415,307 edges, and its dictionary contains 3,778,457 distinct words. The documents of all vertices are organized by an inverted index with average posting list 7.83. A total of 4,774,796 vertices are places with coordinates.

Generating SR queries at random reduces the probability of obtaining any results. To generate more meaningful SR queries, we follow the spatial and keyword distribution of the datasets. For each generated query, its spatial range is a rectangle centered at a point location selected at random from the data. Since the SR query aims to retrieve a semantic region rather than a semantic place, we avoid using keywords that are close to a single place. Specifically, query keywords are generated in the following way. We randomly select a vertex p with low degree[1] from the RDF graph, and we randomly select a keyword from its document. We randomly choose up to $|q.\psi|$ vertices and randomly extract $|q.\psi|$ keywords from the distinct words in the documents of these vertices.

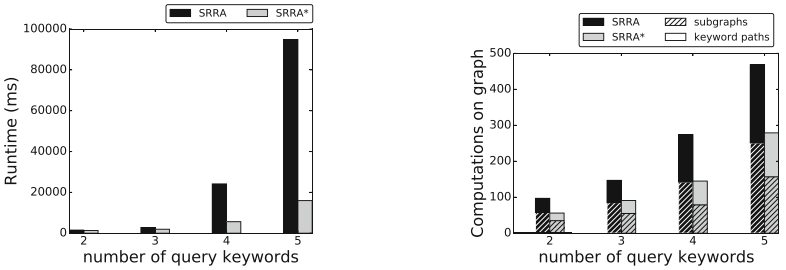
5.2 Setup

The query processing time is evaluated when varying the number of query keywords $|q.\psi|$, α in the scoring function, and the size of the query spatial range $q.r$. We vary one parameter while keeping the others fixed. Table 1 lists the values of the parameters. The values in bold are the (fixed) default values. For each setting, we run 100 queries and measure the average runtime. To evaluate the effectiveness of the proposed techniques, the computations on graph are also reported, including the times of constructing subgraphs connecting places and the times of computing the keyword-relevant paths. All methods were implemented in Java and evaluated on 3.4 GHz quad-core machine running Ubuntu 12.04 with 16 GBytes memory. For both datasets, the RDF graph is assumed to be memory-resident. Although the inverted indexes used can also fit in main memory, we choose to follow the setting of commercial search engines, where the inverted index is disk-resident. This is reasonable because for each query,

[1] Vertices with degree less than 12 on Yago and less than 20 on DBpedia.

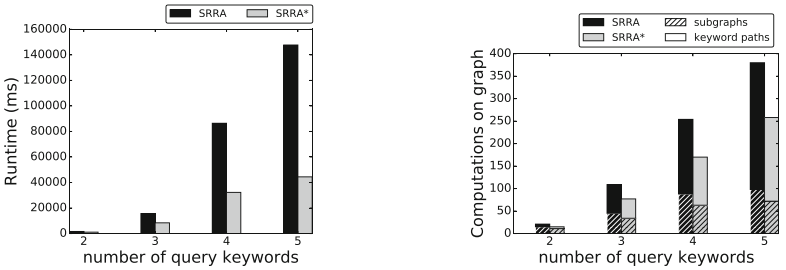
Table 1. Parameter settings.

Parameter	Values
$\|q.\psi\|$	2, **3**, 4, 5
α	0.1, 0.3, **0.5**, 0.7, 0.9
$\|q.r\|$	625 km^2, **784** km^2, 900 km^2, 1600 km^2



(a) Runtime (b) Computations on graph

Fig. 6. Varying the number of query keywords (Yago).



(a) Runtime (b) Computations on graph

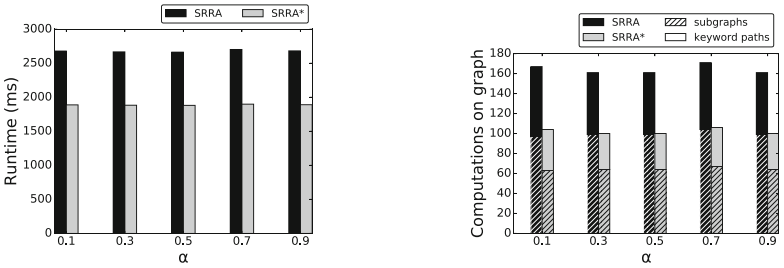
Fig. 7. Varying the number of query keywords (DBpedia).

only a small portion of the inverted index is relevant and needs be kept in main memory. In addition, such a design is scalable when more textual data is added to an RDF knowledge base.

5.3 Performance Evaluation

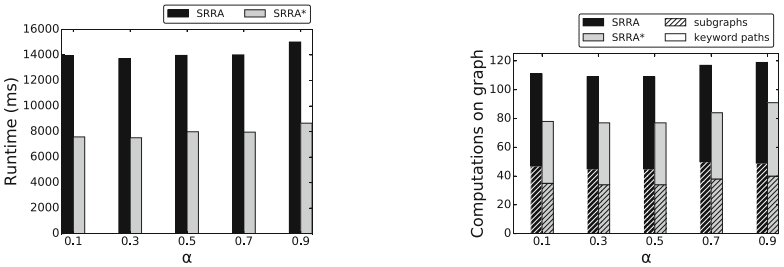
Varying the Number of Query Keywords. Figures 6 and 7 show the cost of both SRRA and SRRA* on datasets Yago and DBpedia, respectively. As expected, the runtime and the computations on graph increase as the number of keywords increases, since more RDF graph vertices need to be explored to discover QSRs covering all the query keywords. SRRA* is significantly faster than SRRA, and the performance gap widens with the number of keywords. For

both algorithms, the cost of constructing QSRs dominates the runtime. SRRA* is more efficient than SRRA, confirming the effectiveness of the pruning techniques proposed in Sect. 4.



(a) Runtime (b) Computations on graph

Fig. 8. Varying α (Yago).



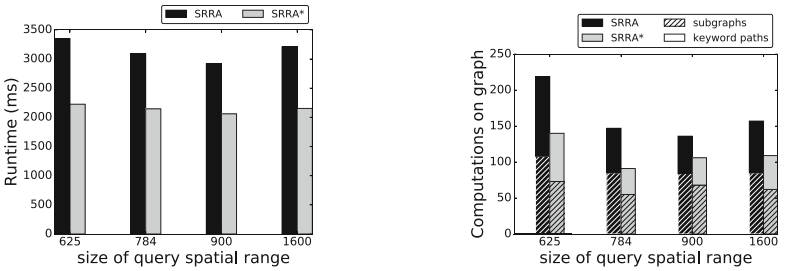
(a) Runtime (b) Computations on graph

Fig. 9. Varying α (DBpedia).

Varying α. Figures 8 and 9 shows the performance of SRRA and SRRA* on Yago and DBpedia when varying α in the scoring function. Large α favor semantic regions where places are closely connected, while small α favor semantic regions where places are close to the query keywords. On both datasets, the performance of the two algorithms are not sensitive to α. The runtime is strongly correlated with the computations on graph. SRRA* outperforms SRRA consistently for all values of α.

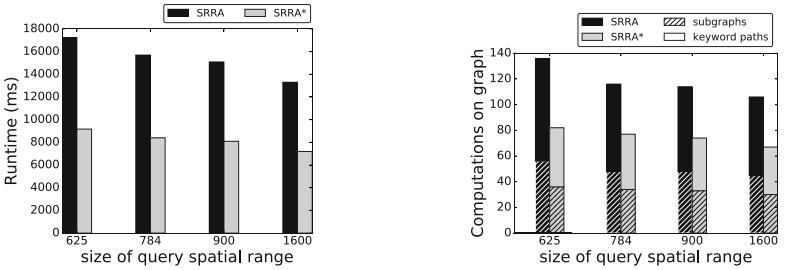
Varying the Size of the Query Spatial Range. Figures 10 and 11 show the computational costs of SRRA and SRRA* on the two datasets. As the size of query spatial range increases, the costs of the algorithms slightly decreases. Intuitively, if the query spatial range is large, a lot of places are involved in the computation, so that the computational cost is expected to be high. However,

we observe from the datasets that a large query spatial range tends to find the semantic region with more places and each place has short keyword relevant path, so that the derived bound on the score is small which is able to prune more sets of places. The amount of computations on graph in Figs. 10b and 11b is the evidence of this observation. Again, SRRA* is more efficient than SRRA in this experiment.



(a) Runtime (b) Computations on graph

Fig. 10. Varying the size of the query spatial range (Yago).



(a) Runtime (b) Computations on graph

Fig. 11. Varying the size of the query spatial range (DBpedia).

6 Related Work

Keyword Search on Graph Data. Traditional graph search algorithms convert queries into searches in feature spaces, such as paths [26], frequent-patterns [31], and sequences [16], which focus predominantly on the structure of the graph rather than on the semantic content of the graph. However, keyword search over graph data [3,6,7,12,14,15,17] takes both the content and the graph structure into account. These two sources of information improve the overall

quality of the results. Moreover, interesting answers that are often difficult to obtain via rigidly-formatted structured queries may be discovered by keyword search. A recent survey covers keyword search on schema graphs (e.g., relational data and XML documents) and schema-free graphs [28]. Keyword-based search is also studied on temporal graphs [22]. Zhong et al. [32] investigate diverse set of most relevant results for a given keyword query on graphs.

Keyword Search on RDF Data. RDF data is traditionally queried using structured query languages, like SPARQL. Due to the advantages of keyword queries over RDF data, SPARQL queries thus have been augmented with keywords for ranked retrieval of RDF data [9]. Elbassuoni et al. study a keyword-based retrieval model over RDF graphs [8] that identifies a set of maximal subgraphs whose vertices contain the query keywords. These subgraphs are ranked based on statistical language models (LMs) [24]. Different from directly searching for keywords on RDF data, Tran et al. [27] first construct a set of k query subgraphs based on the query keywords and then let users choose the appropriate query graph. There are studies focus on the scalable and efficient processing of keyword queries on large RDF graphs [18,27]. These two studies follow the definition of BLINKS [14] for the result subgraphs. Next, k-nearest keyword (k-NK) search on RDF graphs [19] finds the k closest pairs of vertices, (v_i, u_i) that contain two given keywords q and w, respectively. Keyword query interpretation [10] uses a sequence of structured queries to personalize the interpretation of a new query on RDF databases. Personalized keyword search on RDF [11] returns ranked results using the Ranking SVM approach that trains ranking functions based on historical user feedback. Diversified keyword search on RDF graphs [4] finds diversifies results in terms of both the content and the structure of the results. A path-oriented RDF index for keyword search query [5] improves the query processing performance based on associations across RDF paths. A query graph assembly approach [13] converts keyword queries into graph queries. SPARQL and keyword search has been integrated to find SPARQL matches that are closest to all keywords in RDF graphs [23]. Lin et al. [21] translate keyword queries to SPARQL queries using a type-based summary which summarizes all the inter-entity relationships from RDF data.

All these studies concern the querying of general graph and RDF data using keyword-based constraints. Semantic place retrieval [29] from RDF data is the most relevant work, which combines keyword-based and location-based retrieval. The semantic region retrieval studied in this paper is a generalization of semantic place retrieval. It makes up for the shortcomings of semantic place retrieval, addressing the cases where a single place cannot satisfy a users' intent.

7 Conclusion

We propose a novel semantic region retrieval (SR) query that takes as parameters a query spatial range $q.r$ and a set of query keywords $q.\psi$, and returns the qualified semantic region in an RDF graph that minimizes a scoring function. The scoring function takes the graph proximity of the places in the region and

the graph proximity of the places w.r.t. the query keywords into account. Compared to existing semantic place retrieval, the SR query targets for the situation when a single place cannot satisfy the users' requirements. It retrieves multiple nearby relevant places. In order to support efficiently processing of SR queries, we propose algorithm SRRA that follows the branch-and bound paradigm. The most expensive part of the algorithm is computing keyword-relevant paths and connected subgraphs that cover sets of places. To improve the performance of SRRA, we propose two pruning rules that are able to reduce the number of computed semantic regions. The proposed techniques are evaluated on DBpedia and Yago, two large real RDF data sets. The results show that applying all techniques enables processing SR queries efficiently.

References

1. Dbpedia. http://wiki.dbpedia.org
2. Yago. http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/
3. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: a system for keyword-based search over relational databases. In: ICDE, pp. 5–16 (2002)
4. Bikakis, N., Giannopoulos, G., Liagouris, J., Skoutas, D., Dalamagas, T., Sellis, T.: RDivF: diversifying keyword search on RDF graphs. In: Aalberg, T., Papatheodorou, C., Dobreva, M., Tsakonas, G., Farrugia, C.J. (eds.) TPDL 2013. LNCS, vol. 8092, pp. 413–416. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40501-3_49
5. Cappellari, P., De Virgilio, R., Maccioni, A., Roantree, M.: A path-oriented RDF index for keyword search query processing. In: Hameurlain, A., Liddle, S.W., Schewe, K.-D., Zhou, X. (eds.) DEXA 2011. LNCS, vol. 6861, pp. 366–380. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23091-2_31
6. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: a semantic search engine for XML. In: VLDB, pp. 45–56 (2003)
7. Dalvi, B.B., Kshirsagar, M., Sudarshan, S.: Keyword search on external memory data graphs. PVLDB **1**(1), 1189–1204 (2008)
8. Elbassuoni, S., Blanco, R.: Keyword search over RDF graphs. In: CIKM, pp. 237–242 (2011)
9. Elbassuoni, S., Ramanath, M., Schenkel, R., Weikum, G.: Searching RDF graphs with SPARQL and keywords. IEEE Data Eng. Bull. **33**(1), 16–24 (2010)
10. Fu, H., Anyanwu, K.: Effectively interpreting keyword queries on RDF databases with a rear view. In: Aroyo, L., et al. (eds.) ISWC 2011. LNCS, vol. 7031, pp. 193–208. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25073-6_13
11. Giannopoulos, G., Biliri, E., Sellis, T.: Personalizing keyword search on RDF data. In: Aalberg, T., Papatheodorou, C., Dobreva, M., Tsakonas, G., Farrugia, C.J. (eds.) TPDL 2013. LNCS, vol. 8092, pp. 272–278. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40501-3_27
12. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: ranked keyword search over XML documents. In: SIGMOD, pp. 16–27 (2003)
13. Han, S., Zou, L., Yu, J.X., Zhao, D.: Keyword search on RDF graphs - a query graph assembly approach. In: CIKM, pp. 227–236 (2017)

14. He, H., Wang, H., Yang, J., Yu, P.S.: BLINKS: ranked keyword searches on graphs. In: SIGMOD, pp. 305–316 (2007)
15. Hristidis, V., Papakonstantinou, Y.: DISCOVER: keyword search in relational databases. In: VLDB, pp. 670–681 (2002)
16. Jiang, H., Wang, H., Yu, P.S., Zhou, S.: GString: a novel approach for efficient search in graph databases. In: ICDE, pp. 566–575 (2007)
17. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: VLDB, pp. 505–516 (2005)
18. Le, W., Li, F., Kementsietsidis, A., Duan, S.: Scalable keyword search on large RDF data. TKDE **26**(11), 2774–2788 (2014)
19. Lian, X., Hoyos, E.D., Chebotko, A., Fu, B., Reilly, C.: k-nearest keyword search in RDF graphs. J. Web Semant. **22**, 40–56 (2013)
20. Libkin, L., Reutter, J.L., Soto, A., Vrgoc, D.: TriAL: a navigational algebra for RDF triplestores. ACM Trans. Database Syst. **43**(1), 5:1–5:46 (2018)
21. Lin, X., Ma, Z., Yan, L.: RDF keyword search using a type-based summary. J. Inf. Sci. Eng. **34**(2), 489–504 (2018)
22. Liu, Z., Wang, C., Chen, Y.: Keyword search on temporal graphs, pp. 1807–1808, ICDE (2018)
23. Peng, P., Zou, L., Qin, Z.: Answering top-k query combined keywords and structural queries on RDF graphs. Inf. Syst. **67**, 19–35 (2017)
24. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR, pp. 275–281 (1998)
25. Prud'Hommeaux, E., Seaborne, A., et al.: SPARQL query language for RDF. W3C recommendation 15 (2008)
26. Shasha, D., Wang, J.T.L., Giugno, R.: Algorithmics and applications of tree and graph searching. In: PODS, pp. 39–52 (2002)
27. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In: ICDE, pp. 405–416 (2009)
28. Wang, H., Aggarwal, C.C.: A survey of algorithms for keyword search on graph data. In: Aggarwal, C., Wang, H. (eds.) Managing and Mining Graph Data. Advances in Database Systems, vol. 40, pp. 249–273. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-6045-0_8
29. Wu, D., Zhou, H., Shi, J., Mamoulis, N.: Top-k relevant semantic place retrieval on spatiotemporal RDF data. VLDB J. **29**(4), 893–917 (2020)
30. Wylot, M., Hauswirth, M., Cudré-Mauroux, P., Sakr, S.: RDF data storage and query processing schemes: a survey. ACM Comput. Surv. **51**(4), 84:1–84:36 (2018)
31. Yan, X., Yu, P.S., Han, J.: Substructure similarity search in graph databases. In: SIGMOD, pp. 766–777 (2005)
32. Zhong, M., Wang, Y., Zhu, Y.: Coverage-oriented diversification of keyword search results on graphs. In: DASFAA, pp. 166–183 (2018)