Hindawi
Scientific Programming
Volume 2020, Article ID 5060635, 10 pages
https://doi.org/10.1155/2020/5060635

*Research Article*

# Place Retrieval in Knowledge Graph

**Xin Shan,[1,2] Jingyi Qiu,[3] Bo Wang,[2] Yongcheng Dang,[3] Tingxiang LU,[2] and Yiming Zheng[2]**

[1]*School of Energy and Electrical Engineering, Hohai University, Nanjing 210000, China*
[2]*NARI Group Corporation (State Grid Electric Power Research Institute), Nanjing 210000, China*
[3]*School of Computer Science and Engineering, Southeast University, Nanjing 211189, China*

Correspondence should be addressed to Bo Wang; wwwbo1981@163.com

Received 24 October 2019; Accepted 24 February 2020; Published 1 July 2020

Guest Editor: Guandong Xu

With the rapid development of Internet and big data, place retrieval has become an indispensable part of daily life. However, traditional retrieval technology cannot meet the semantic needs of users. Knowledge graph has been introduced into the new-generation retrieval systems to improve retrieval performance. Knowledge graph abstracts things into entities and establishes relationships among entities, which are expressed in the form of triples. However, with the expansion of knowledge graph and the rapid increase of data volume, traditional place retrieval methods on knowledge graph have low performance. This paper designs a place retrieval method in order to improve the efficiency of place retrieval. Firstly, perform data preprocessing and problem model building in the offline stage. Meanwhile, build semantic distance index, spatial quadtree index, and spatial semantic hybrid index according to semantic and spatial information. At the same time, in the online retrieval stage, this paper designs an efficient query algorithm and ranking model based on the index information constructed in the offline stage, aiming at improving the overall performance of the retrieval system. Finally, we use experiment to verify the effectiveness and feasibility of the place retrieval method based on knowledge graph in terms of retrieval accuracy and retrieval efficiency under the real data.

## 1. Introduction

In recent years, retrieval related to geographic information has drawn increasing attention from academic fields due to its indispensable applications in social life. Currently, place retrieval is ubiquitous in tourism query, peripheral life, and other aspects. According to relevant investigation and statistics, more than 28 percent of the results are related to geographic information such as zip core addresses, Internet IP addresses, provinces, and cities when users input a relevant query on Internet search engines.

The traditional retrieval approaches for keywords cannot achieve high accuracy and efficiency because they fail to capture the semantic requirement of users well. Therefore, the new generation of big data retrieval is based on knowledge graph [1]. Unlike traditional retrieval methods, the purpose of the knowledge graph is to describe and abstract entities and concepts which exist in the real world and express the relationship between entities and concepts. It can better reflect the semantics and requirements of users through association relation and improve the accuracy and conformity of retrieval. Meanwhile, the previous graph search algorithms [2, 3] focus more on the graph structure rather than the semantic content, which is not sufficient to improve the accuracy and efficiency of search obviously.

In recent years, keyword query based on RDF data has developed rapidly. RDF is a special type of graph data. Queries based on RDF data are structured query languages such as SPARQL [4]. In general, there are broadly two categories of retrieval based on knowledge graph: one is the research on the optimization of structured query language based on SPARQL. At present, SPARQL query research mainly focuses on relational algebra and Top-K Join query algorithm [5, 6]. Although query results can be more accurate by the above studies on query optimization of SPARQL Top-K, it is not appropriate to directly apply them on the large-scale RDF data set. In addition, users need to understand query semantics and RDF storage mode; the location and other information of users are not considered. Another one is based on the RDF graph query method. This approach allows capturing more information from RDF graph structure such as subgraphs [7–10], neighbour [11], path [5], and distance [12]. These ways can express the semantic needs of users and the results are retrieved quickly. Nevertheless, they have low efficiency for massive data query inherently.

At present, the place retrieval of knowledge graph is still in the preliminary stage and there are many research difficulties: (1) The place retrieval needs to have a better understanding of the user's actual retrieval semantics and needs by extracting information from knowledge graph. (2) A large amount of spatial information in the data is ignored, resulting in that the query results cannot better understand the users' retrieval needs in terms of geographical location. The retrieval results need to combine the geographic location, time, and other information for personalized recommendation. (3) The existing methods lack the ability to conduct efficient query on the massive graph data for its large storage and calculation costs. At the same time, the previous researches do not explicitly compute the cost of ranking the retrieval results, which affect the accuracy of retrieval results.

Considering the shortcomings of the above research status, it is essential to design a place retrieval method to overcome these limitations. This paper proposes KPR system, an effective system of knowledge graph place retrieval to enhance the speed and accuracy of retrieval. Specifically, we conduct experiments on different data sets, and empirical results demonstrate that our method significantly outperforms traditional retrieval.

It is worthwhile to highlight the following contributions of this paper:

(1) In the offline stage, we introduce a spatial semantic hybrid index to preserve both location feature and semantic feature. Furthermore, by jointly optimizing the construction process of spatial index and semantic distance index, hybrid index not only satisfies the requirements of location and semantics but also shortens the execution time of query.

(2) In the online stage, we present a ranking model based on ESH sorting and an online query algorithm to enhance the accuracy and efficiency of KPR system. The results indicate that our proposed method overcomes the low efficiency for big data retrieval.

This paper is organized as follows. Section 2 proposes preprocessing and index building methods. Section 3 introduces query algorithm and ranking model. In Section 4, index building and online query algorithm are tested to verify their effectiveness. Section 5 summarizes and prospects the research work.

## 2. Preprocessing and Index Building

This paper establishes the problem model through the preprocessing part. At the same time, this paper constructs semantic distance index and spatial index based on space and semantic information, and this paper optimizes separately to reduce the cost of storage. Finally, this paper will build a hybrid index structure.

### 2.1. Related Definition

*Definition 1* (KSP query). User semantic Top-K query is based on spatial RDF data set. The user inputs the geographic location $q$ of the current query, the type of the location to be queried, the number of results that need to be returned $k$, and a series of keyword description sets $t$. These will return Top-K locations that best match the user's query needs. These locations take into account the cost function of semantic distance and spatial distance to complete evaluation ranking. The following two requirements are met: (1) This location is a place entity in the RDF graph. (2) Nodes in the subtree rooted at the vertex of the entity contain all the keywords entered by users. Figure 1 shows an example of KSP query.

*Definition 2* (Semantic distance $L(T_p)$). Quantitative calculation by semantic distance is used to express the size of semantic relevance. The smaller the semantic distance, the greater the semantic relevance. This means that the greater the relevance of the current query location to the user query semantics requirement. To suppose the user enters a set of query keywords as $t = \{t_1, t_2, t_3, \ldots, t_i, \ldots, t_m\}$, given a place entity that satisfies the query $T_p$, $d_G(p, t_i)$ is the shortest distance from the entity $p$ to the keyword $t_i$ in the RDF graph $G$. Then, the semantic distance $L(T_p)$ is the sum of the shortest distance from $p$ to all keywords:

$$L(T_p) = \min \sum_{i=1}^{m} (d_G(p, t_i))|_{t_i \in t}. \quad (1)$$

*Definition 3* (Spatial distance $S(q, p)$). The spatial distance represents the Euclidean distance of the spatial geographic location between the user query location $q$ and the place entity $p$. This equation is $S(q,p) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. $(x_1, y_1)$ are the spatial geographic coordinates of user query location $q$. $(x_2, y_2)$ are the spatial geographic coordinates of the place entity $p$.

*2.2. Problem Model.* Assuming that users are in Boston, users will query "Universities that have cultivated Nobel and Turing prizes." In Figure 1, two subtrees contain the keywords "Nobel Prize" and "Turing Prize", and root nodes are "Harvard University" and "Massachusetts Institute of Technology" separately. The purpose of KSP query is to find a set of place entities that satisfy the user to query keywords.

Considering the semantic and spatial information factors, the target model of KSP query is constructed:

$$\begin{aligned} G &= \mathrm{Top}_k\left(f\left(L\left(T_p\right), S(q,p)\right)\right) \\ &= \mathrm{Top}_k\left(f\left(\min \sum_{i=1}^{m} d_G(p, t_i)\right)|_{t_i \in t}, S(q,p)\right). \end{aligned} \quad (2)$$

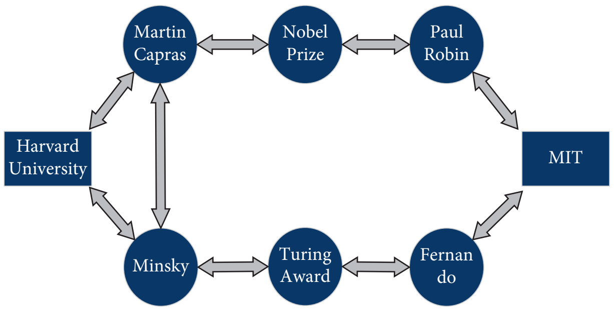

Figure 1: KSP query example.

In equation (2), place entity $p \in$ place type $C$, $q$ is the user query location, $t$ is user query keyword set, and $f$ is the sorting function of synthetic semantic and spatial information.

Because the entity description contains a huge amount of keyword information and there are a lot of duplicates, this paper constructs an inverted index with the keyword Key in the preprocessing stage and converts the calculation of semantic distance into calculations between entity nodes. According to the definition of semantic distance, to satisfy all the query keywords, it is necessary to calculate the minimum value of the sum of the shortest distances of the current entity to be queried $p$ to all the physical nodes $v_i$ in each set $V$:

$$L\left(T_p\right) = \min \sum_{v_i \in V} \left(d_G\left(p, v_i\right)\right). \tag{3}$$

Figure 2 is the problem model, supposing that the user inputs the query keyword as {subject, dedication, roman}. Finding the set of nodes corresponding to the keyword according to the inverted index and to find nodes into {{v1, v4}, {v2}, {v1, v2, v5}}, after the node sets are arranged and combined, the set of nodes that need to be searched is obtained as {v1, v2}, {v1, v2, v5}, {v4, v2}, {v4, v2, v5}, {v4, v2, v1}. The semantic distance is obtained according to the shortest distance between nodes.

## 2.3. Semantic Distance Index

*2.3. Semantic Distance Index.* According to the problem model established in the previous section, in order to quickly obtain the semantic distance by calculating the shortest distance between the physical nodes, this section constructs an efficient semantic distance index structure.

*Definition 4* (Symbol $D(v)$). $D$ represents the shortest distance from an entity to another set of entities in the RDF graph. Storing a vertex pair $(u, \delta_{uv})$, $u$ is the any entity node and $\delta_{uv}$ represents the shortest distance from entity node $u$ to entity node $v$.

With regard to every entity node $v$, from query node $s$ to node $t$ is the shortest distance $\delta_{st}$; $\mathrm{QUERY}(s, t, L)$ is

$$\mathrm{QUERY}(s, t, L) = \min\left\{\delta_{vs} + \delta_{vt}\right\}. \tag{4}$$

First, sort all current place entity ID numbers. Secondly, for each node $u$, initialization flag $D_0(u)$ is empty. Finally, perform breadth-first-search for d-layer in order of place

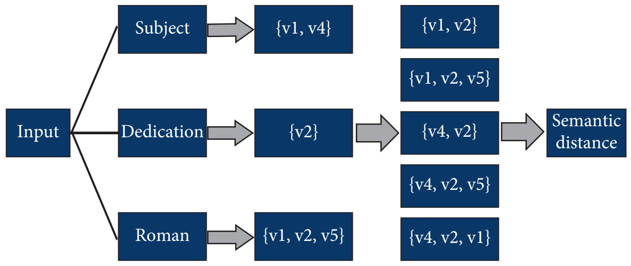

Figure 2: Problem model.

entity ID. Add node distance from $v_k$ to departure when the breadth-first-search is performed from node 1 at the $k$th time. It can be displayed as $D_k(u) = D_{k-1}(u) \cup \{(v_k, d_G(v_k, u))\}$. Suppose the distance between $u$ and traversal is $\delta$. If $\mathrm{QUERY}(v_k, u, D_{k-1}) \le \delta$, then cut $u$ and no longer put $(v_k, \delta)$ into the mark $D_k(u)$. In addition, no longer try to associate all edges with $u$. Index information is stored in the form of < place entity ID, node ID, shortest distance >. Figure 3 describes the process of semantic distance index construction.

## 2.4. Spatial Index

*2.4. Spatial Index.* Spatial place is another important factor affecting place retrieval results. Therefore, this section focuses on how to build an effective index based on spatial place information and quickly query place entities.

*Definition 5* (Regional rectangle coordinates (RRC)). Regional rectangle coordinates (RRC) represent and store information of spatial area information. Suppose a space has $m$ areas, and $T_i$ represents the spatial region coordinate RRC of the $i$_th region. The coordinates of the upper left corner and the coordinates of the lower right corner are expressed as

$$\left\{\begin{array}{l} T_i \mid T_i \text{ the upper left corner is } \left(a_x^i, a_y^i\right), \\ T_i \text{ the bottom right corner is } \left(b_x^i, b_y^i\right) \end{array}\right\}. \tag{5}$$

According to the area RRC of $T_i$ and the spatial coordinate $(x_1, y_1)$ of the location $q$, it can be determined whether the location is within the area $T_i$:

$$\begin{cases} \text{if } \left(a_x^i \le x_1 \le b_x^i\right) \cap \left(b_y^i \le y_1 \le a_y^i\right) & \text{then } q \exists T_i, \\ \text{else } q \nexists T_i. \end{cases} \tag{6}$$

*Definition 6* (Linear quadtree). A linear quadtree meets the following conditions: (1) Nonleaf nodes have 4 subtrees. (2) The intermediate node is also a linear quadtree. (3) Save the nonempty leaf nodes of the quadtree in a one-dimensional auxiliary object space based on disk and each of these nodes can be encoded by space filling techniques. (4) In the case of ensuring that all nodes in a region are divided into four rectangular subregions, try to ensure that the number of nodes in each subarea is even.

The generation algorithm of quadtree is a top-down iterative process. (1) First, calculate the RRC of the RDF graph that contains all the place entity nodes, as $I$ of the root

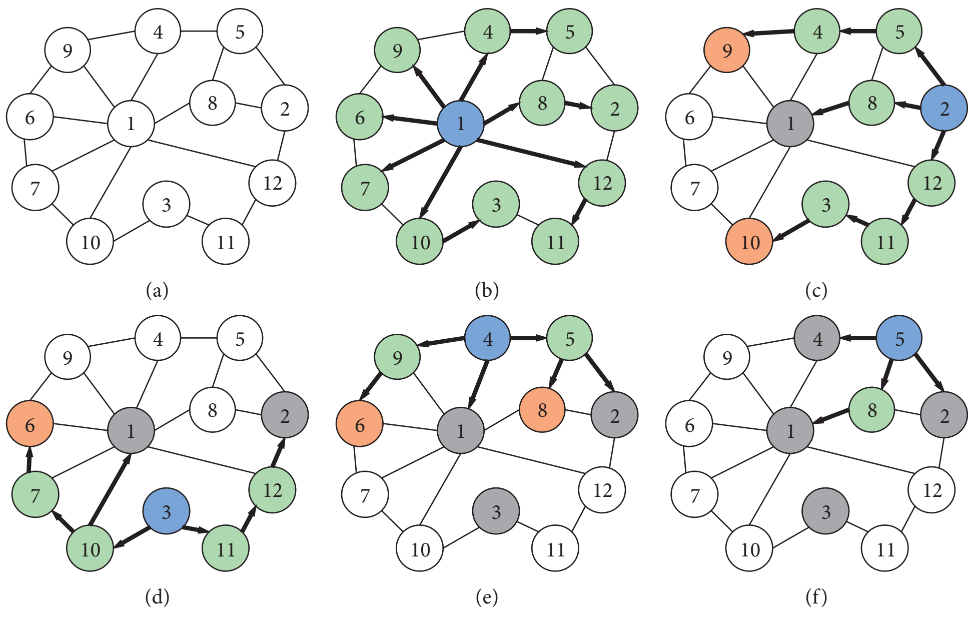

Figure 3: Semantic distance index construction.

node of the linear quadtree, and as the entire data space simultaneously. (2) The data space *I* is divided into 4 parts vertically or in parallel in the *x*-axis direction or the *y*-axis direction according to the distribution of the nodes. Each part contains an almost equal number of place entity nodes. (3) To calculate the *RRC* of the four subareas separately, establish a link to the parent node. Moreover, perform iterative execution until the number of physical nodes in each area reaches the threshold *m*. (4) Do split-sequence-based encoding for quadtrees when the area is divided. Split in order of space is as follows: SW (southwest), SE (southeast), NW (northwest), NE (northeast). These spaces represent binary 00, 01, 10, and 11 encoding separately, which can be obtained by splitting sequence links for each subpartition. The node paths and specific region codes in a linear quadtree are globally unique, which facilitates linear storage of data. (5) Map the region coding with the RRC of the corresponding region < region coding, RRC>, which can make it easy to locate a location quickly. It can be quickly determined in a certain node area of the linear quadtree according to the geographical location of the place.

### 2.5. Spatial Semantic Hybrid Index

*Definition 7* (Regional semantic distance (RSD) index). There are still a large number of place entities in the area *s* after spatial segmentation. With regard to semantic distance index of every place entity, entity can be sequentially stored according to entity ID.

According to the definition of KSP query, the semantic distance of each place entity needs to be calculated, and the closer it is to the users' location, the more it meets the user's needs. Therefore, it is necessary to combine spatial index information with semantic distance index information to form a hybrid index structure based on space and semantics. This structure is region semantic index. The semantic distance index information is stored in the leaf nodes of the spatial quadtree index, and each leaf node represents the corresponding spatial area and stores the mark D(v) of all the place entities in the area. Besides, D(v) is the shortest distance information of the place entity in the area from other entity nodes. Figure 4 shows the structure of the regional semantic distance (RSD) index, and Table 1 shows the storage information of the RSD.

## 3. Query Algorithm and Ranking Model

*3.1. Ranking Model.* For a set of location results of user query, a ranking function model needs to be established. Additionally, combine the semantic distance and spatial distance information to sort the results, and select Top-K results to return to the user.

The traditional linear ranking model has large defects. First of all, it is less sensitive to influencing factors. Secondly, it can be affected by weight parameters. Thirdly, it is easy to cause extreme value, which leads to inaccurate query results. This paper generates Skyline's exponential ranking model. Figure 5 shows the Skyline coordinate.

For Exponent-based Skyline Hierarchical Sorting Algorithm (ESH), the result set of the place retrieval is mapped to the Skyline two-dimensional coordinate system, and the horizontal and vertical axes represent the spatial distance index and the semantic distance index, respectively. To calculate data in 2D coordinates to get different contour levels, data in the same contour level has the same ranking priority. The lower the contour level, the higher the ranking priority. For example, the data in CL(k-1) is more advanced than the data in CL(k). With regard to the data in the same contour level, compliance distance F(p) will be calculated by equation (7). The smaller $F(p)$ is, the more advanced the

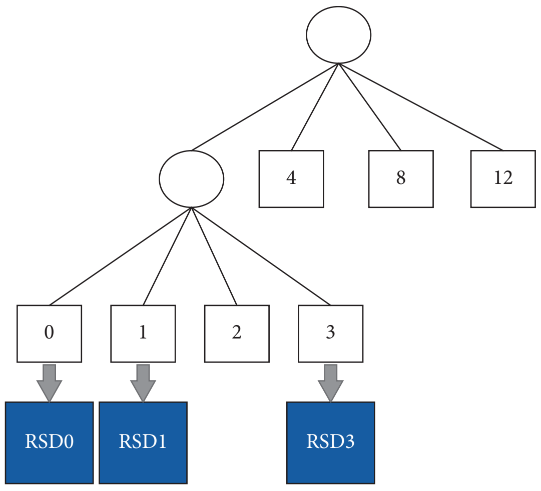

Figure 4: Spatial semantic hybrid index structure.

Table 1: RSD storage information.

| Regional ID | Mark $D(v)$ | Entity node $u$ | Entity linkage $d_G(v, u)$ |
|---|---|---|---|
| 4 (0100) | L (Harvard) | Capras | 1 |
| | L (MIT) | Minsky | 1 |
| | | Nobel Prize | 2 |
| | | Paul Robin | 1 |
| | | Fernando | 1 |
| | | Turing Award | 2 |

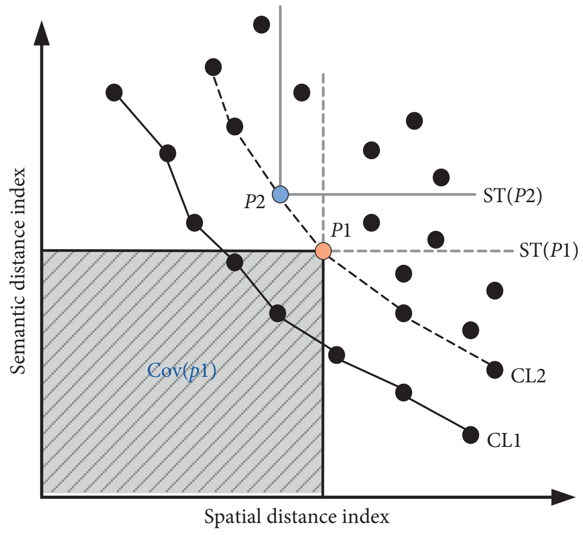

Figure 5: Skyline coordinate.

ranking is, which indicates that the current place entity is more in line with the actual needs of the user:

$$F(p) = \mathrm{Cov}(p) \times \mathrm{ST}(p) = \frac{\mathrm{Cov}(p)}{\mathrm{CT}(p)} = \frac{\beta^{L(T_p)} * \beta^{S(q,p)}}{\mathrm{CT}(p)} = \frac{\beta^{L(T_p)+S(q,p)}}{\mathrm{CT}(p)}. \tag{7}$$

*3.2. Query Algorithm.* This section presents an efficient online retrieval query algorithm; the specific steps are as follows: (1) find a set of nodes corresponding to the keyword set according to the keyword description set input by users. (2) According to the query location information of the user, the RRC of each layer node is searched from the top in the linear quadtree index for comparison with the place space coordinate, and the minimum area to which the location belongs is located. (3) Read the RSD information in this area. Query the tag information of the set of demand nodes according to the user input keyword, and calculate the semantic distance. Then, the spatial coordinates of the user query location and the place entity to be queried are obtained by the distance calculation formula to obtain the spatial distance. The Top-K result set of integrated spatial and semantic information is calculated according to the cost function model. The result set is maintained by the method of the minimal heap and updated by the comparison of the $k$-th result until the end of the query. The online query algorithm is shown in Algorithm 1.

With regard to the result set of online queries, suppose that Top-K results are maintained in the current minimum heap. For the $k$-th worst result, if the result of the next query is more expensive than that, there is no need to update the minimum heap. Therefore, the boundary based on semantic distance and spatial distance can be calculated according to the cost function model and the $k$-th result of the current minimum heap. Use this for pruning subsequent queries. Dynamically adjust the corresponding boundary each time when the minimum heap is updated. Assuming that $\theta$ is the cost of the current $k$th result, the maximum bound of the semantic distance of the place entity is $L_m(T_p) = \sqrt[\beta]{\theta - 1}$. This means that no matter how small the spatial distance is, the final cost will not be less than $\theta$ when the subsequent place entity semantic distance is greater than the limit. Therefore, the place entity is removed and the subsequent calculations are not continued. Similarly, the maximum limit of spatial distance is $S_m(q, p) = \sqrt[\beta]{\theta - 1}$. Figure 6 shows an example of a pruning optimization query.

## 4. Experiment Design

In order to verify the index structure in this paper, online query algorithm in this paper, and the improvement of retrieval performance by the ranking model, this paper will test system performance from three aspects: index performance, query algorithm, and query accuracy.

The system module diagram is shown in Figure 7.

*4.1. Experiment Environment and Data Set.* The hardware environment is as follows: the experimental machine in this paper is a dual-channel Intel Core I7-4790 CPU, 64 GB DDR3 1600, 300 GB mechanical hard disk.

The software environment is as follows: Ubuntu14 operating system; JDK version 1.8.0_101; IntelliJ Idea development environment.

The data sets are as follows: YAGO and DBpedia.

The information of data set is listed in Table 2.

```
Input: keyword set query = $\{t_1, t_2, \ldots, t_n\}$,
    the results number $k$, the location of user $q$
Output: The Top-$k$ results heap $H_{\text{top}-k}$
(1) Heap $H_{\text{top}-k} = \varnothing$,
(2) for each keyword $t_i$ in query do
(3)     $t_i \in I$
(4) $\theta = +\infty$
(5) while key = GETKEY$(R, q)$ do
(6)     if $S(q, \text{key}) \geq \theta$ then break
(7)     if $key$ refers to a place $p$ then
(8)     $T_p$ = GETKSP(query, $p$) do
(9)         if $L(T_p) = +\infty$ then continue
(10)            $f(L(T_p), S(q, p))$
(11)        if $f < \theta$ then
(12)            $H_{\text{top}-k}$.add$(T_p, f)$
(13)            Update $\theta$
(14) return $H_{\text{top}-k}$
```

Algorithm 1: Online query algorithm.

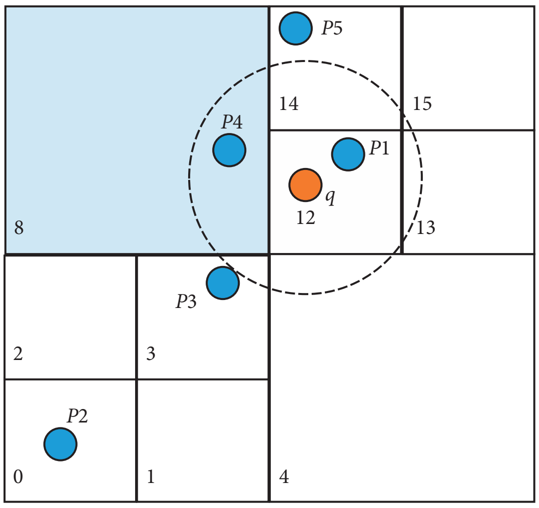

Figure 6: Pruning query example.

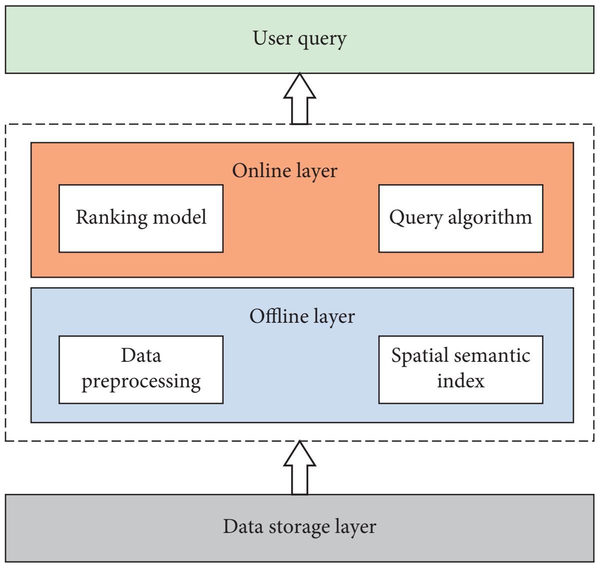

Figure 7: System module diagram.

*4.2. Parameter Confirmation.* Before the experiment, the following parameters need to be confirmed:

(1) The value of $d$: when constructing a semantic distance index, the shortest distance between the place entity and other entity nodes needs to be obtained. This facilitates online calculation of semantic distance. Store the shortest distance between entities into the tag by breadth-first-search. However, considering the storage space and index efficiency, the number of layers $d$ for breadth-first-search needs to be determined experimentally. Figure 8 is an experiment of the effect of parameter $d$ on the semantic distance index storage size under the YAGO data set.

It can be seen that the semantic distance storage size varies from 100M to 800M. Furthermore, the size increases with the value of $d$ becoming larger. When $d = 5$, the index size has increased dramatically. Therefore, in this paper, the parameter $d = 4$ is set, and only the interentity information with the shortest distance of 4 is stored.

Table 2: The information of data set.

| Data set | Number of vertices | Number of sides | Keyword | Size (GB) |
|---|---|---|---|---|
| YAGO | 800 ten thousand | 1.2 million | 17 million | 26 |
| DBpedia | 600 ten thousand | 35 million | 22 million | 43 |

(2) The value of $m$: when constructing a spatial linear quadtree index, the regional data threshold $m$ will affect the depth of the linear quadtree, thus affecting the query efficiency. Therefore, $m$ needs to be

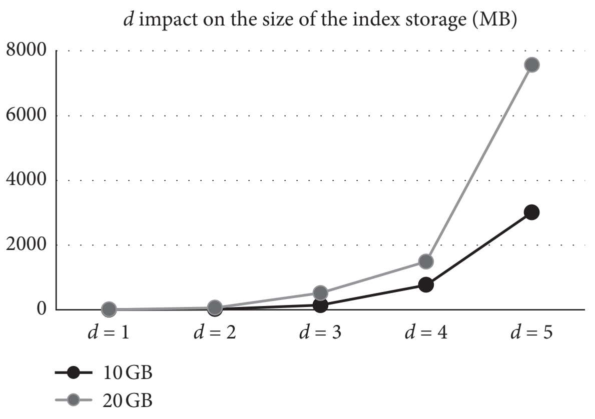

Figure 8: Experiment of determining the parameter *d*.

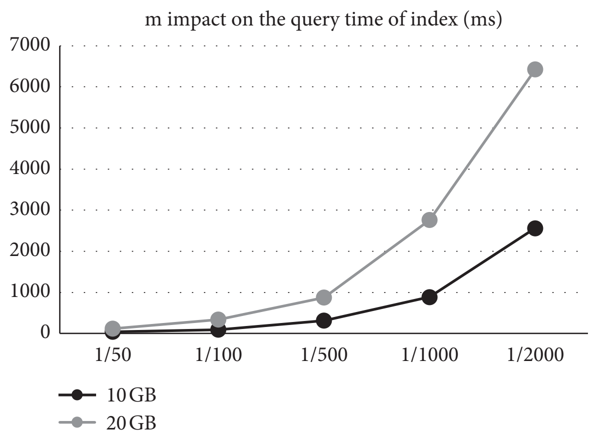

Figure 9: Experiment of determining the parameter *m*.

obtained through experiments. Figure 9 is an experiment of the influence of the parameter $m$ on the spatial index query time under the YAGO data set. $m$ is the percentage of the total number of locations.

It can be seen from this experiment that the spatial index storage size increases significantly as the parameter $m$ decreases. This is because the smaller the threshold $m$, the more regions that need to be divided. The greater the depth of the linear quadtree, the longer the query time. When $m = 1/1000$, the query has a sharp increase. Therefore, this paper sets the parameter $m = 1/500$.

(3) The value $\beta$ of ranking model: $\beta$ represents how sensitive the ranking model is to variables. In the experiment, $\beta$ is equal to 1.5.

*4.3. Construct Index Performance Experiment.* After the parameters are confirmed, a semantic distance index and a spatial linear quadtree index need to be established, and randomly generate 1000 query samples. Adopt the same KSP query algorithm and compare the average query time of indexing and nonindexing under the same query, and then verify the effectiveness of indexing to improve the query efficiency of the retrieval system. Figures 10 and 11 show the experimental results under the YAGO data set and the DBpedia data set, respectively.

From the experimental results, it can be seen that the query time increases greatly with the increase of the amount of data. However, after the index information is built, under the same query input and the same query algorithm, the average query time including the index information is shorter than the average query time without the index, and only about 3/5 ~ 4/5 of the query time without index information is included. As the amount of data increases, the index information improves the query efficiency more obviously.

With regard to semantic distance chipping index construction method in this paper, this method has a significant reduction in the index storage cost compared to the traditional method of directly storing the shortest distance as the semantic distance. Establish traditional

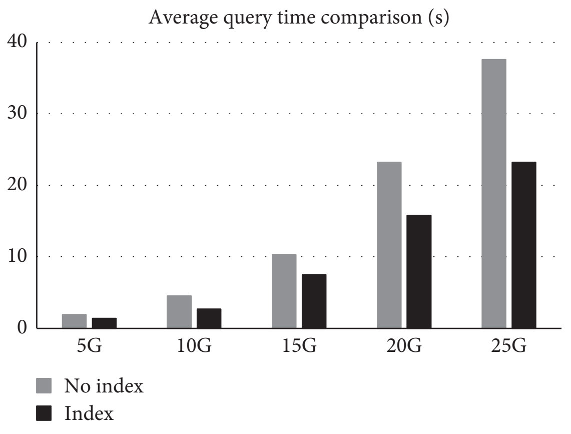

Figure 10: YAGO experimental results.

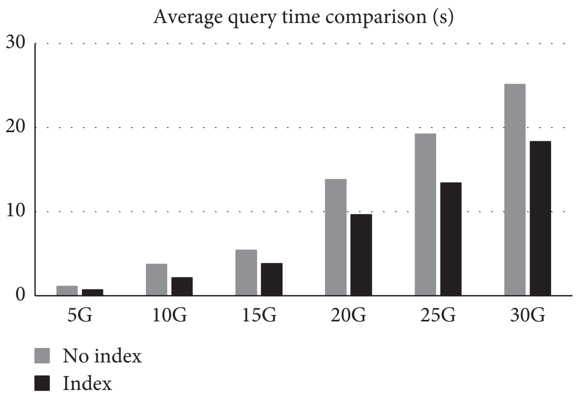

Figure 11: DBpedia experimental results.

shortest distance index and clipping index for different size data of YAGO data set, respectively. Compare the storage size of the index. The experimental results are shown in Figure 12.

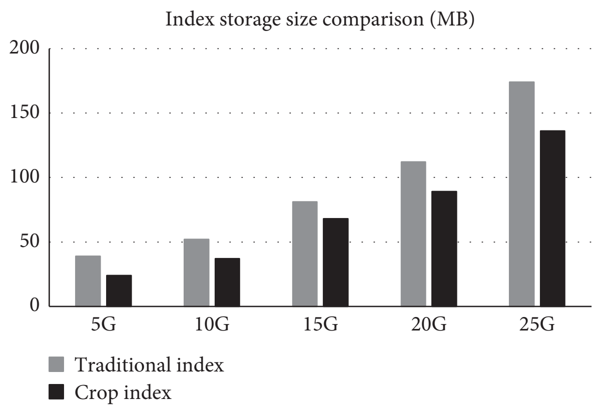

Figure 12: Index construction optimization experiment.

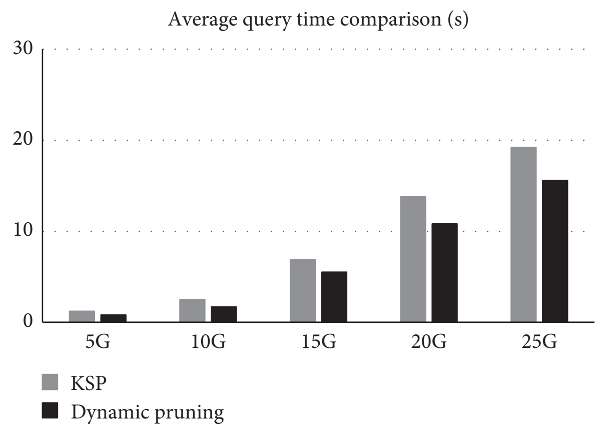

Figure 13: YAGO experimental results.

As can be seen from the experimental results, the index storage size increases as the total amount of data increases. Besides, the method of storing semantic distance information by clipping marks is more space-saving than the traditional way of storing the shortest distance directly. As the overall amount of data increases, the tailoring effect becomes more apparent, and the storage size is approximately 5/7 ~ 6/7 of the conventional method. It is verified that the clipping method of this paper effectively reduces the redundancy of information.

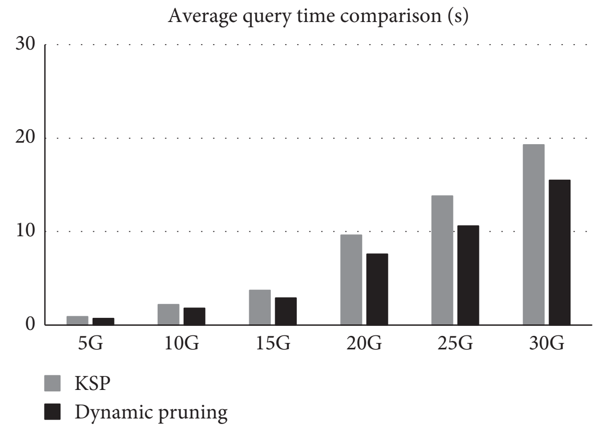

Figure 14: DBpedia experimental results.

*4.4. Query Algorithm Experiment.* After building the index in the offline phase, with regard to online query algorithm, the dynamic pruning query method proposed in this paper can effectively remove unnecessary locations to be inquired, thus shortening the query time. The following experiment verifies the efficiency of the dynamic pruning query algorithm in this paper by comparing different query algorithms. Firstly, randomly generate 1000 query inputs based on the same index information. Secondly, use traditional KSP algorithm and dynamic pruning query algorithm to compare the average query time. Figures 13 and 14 show the results of the comparison experiments under the YAGO dataset and the DBpedia dataset, respectively.

It can be seen from the experimental results that the dynamic pruning query algorithm is more efficient than the traditional KSP algorithm. Moreover, as the amount of data increases, the optimization effect of the dynamic pruning query algorithm is getting better and better.

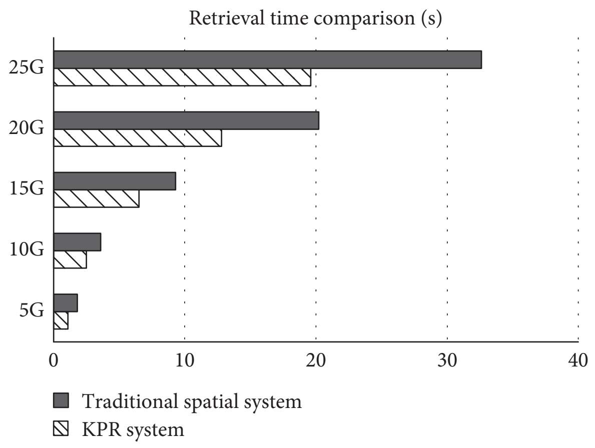

Figure 15: Retrieval efficiency experiment.

*4.5. Retrieval Performance Experiment.* According to the index and online query algorithm proposed in this paper, a complete retrieval system KPR is formed. For experimental comparison with traditional KSP-based retrieval systems, generate 1000 random queries under the YAGO data set. Then, to compare the average query time of different retrieval systems, this experiment will verify the effectiveness of the retrieval system in this paper on the speed of traditional retrieval systems. Figure 15 shows the comparison results of the retrieval system.

According to this experiment, the KPR retrieval system proposed in this paper has greatly improved the retrieval efficiency compared with the traditional KSP-based retrieval system. However, this also sacrifices a part of the index. The corresponding parameters were determined by the index cost experiment and the retrieval system will achieve a balance between efficiency improvement and storage cost to

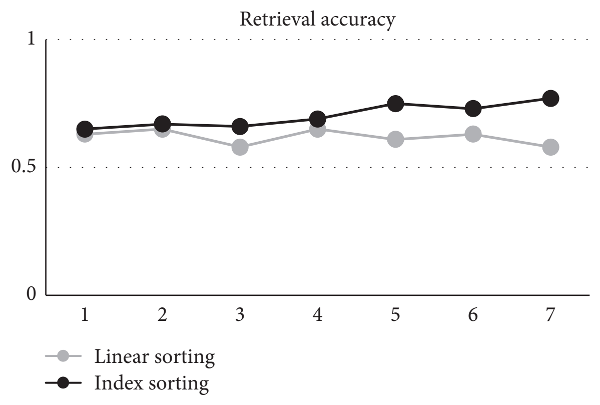

Figure 16: Retrieval accuracy experiment.

guarantee improving the performance of the retrieval system at a lower storage cost.

With regard to different ranking models, firstly, randomly generate 1000 identical queries and take result of Top-K in the query result set. Secondly, calculate the results to obtain the average accuracy. Thirdly, carry out comparative analysis of accuracy under different ranking models. Figure 16 shows an accuracy comparison chart of the traditional linear ranking model and the exponential ranking model under the YAGO data set.

It can be seen from the experimental results that the traditional linear ranking function is inferior to the exponential ranking model in retrieval accuracy. In addition, as the *k* value of Top-K is larger, the accuracy of the index ranking model is getting higher and higher. However, the accuracy of the linear ranking model is less improved. The accuracy gap between the two ranking models is getting bigger and bigger. This is because the linear ordering model is more dependent on the weight value parameter. Moreover, it is easy to generate a set of bad results under extremes due to poor sensitivity to variables. Therefore, the retrieval accuracy is not high.

*4.6. Experimental Summary.* The experiment in this section verifies that not only does the index structure proposed in this paper improve the retrieval performance, but also, compared with the traditional direct index construction method, the index construction method proposed in this paper is lower in the cost of storage space and can be used in the actual production environment. Through the comparison between the query algorithm and the traditional KSP algorithm, the efficiency of the algorithm in this paper is verified. Through the experiment of query accuracy, it is verified that the Top-K results selected by the ranking model in this paper are more accurate than the traditional Top-K results based on the linear mode. All the above experiments verify the effectiveness and feasibility of the method proposed in this paper to improve the place retrieval performance of knowledge graph.

## 5. Conclusion

This paper starts from the problem that the existing knowledge graph place retrieval technology has shortcomings in accuracy and low efficiency under massive data, comprehensively considering semantic and spatial information factors, to build index information in the offline phase, and the efficient query algorithm and ranking model are designed in the online stage to improve the accuracy and efficiency of the place retrieval system.

The future research direction is mainly the distributed and parallelizable knowledge graph retrieval. At present, the retrieval technology of knowledge graph is still based on single-machine memory and disk, which has great limitations in storage space and parallel efficiency. Due to the existence of a large number of entity connections in the knowledge graph, for the distribution and parallelization of knowledge graph, the partition storage of graph and the distributed retrieval algorithms are both hotspots and difficulties in future research. If it can be realized, the performance of knowledge graph retrieval system will be further improved.

## Data Availability

The YAGO data used to support the findings of this study have been deposited in https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/. The DBpedia data used to support the findings of this study have been deposited in https://wiki.dbpedia.org/.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was supported by the Research and Application of Intelligent Technology for Real-time Dispatching of Power Grid Based on Artificial Intelligence (52460817A029).

## References

[1] J. Pujara, H. Miao, L. Getoor et al., "Knowledge graph identification," *International Semantic Web Conference*, Springer-Verlag, Berlin, Germany, 2013.
[2] X. Lian, L. Chen, and Z. Huang, "Keyword search over probabilistic RDF graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1246–1260, 2015.
[3] Z. Wang, J. Zhang, J. Feng et al., *Knowledge Graph and Text Jointly Embedding*, pp. 1591–1601, Association for Computational Linguistics, Stroudsburg, PA, USA, 2014.
[4] M. A. Bornea, J. Dolby, A. Kementsietsidis et al., "Building an efficient RDF store over a relational database," in *Proceedings of the 2013 international conference on Management of data–SIGMOD '13*, pp. 121–132, New York, NY, USA, 2013.
[5] M. Arenas, B. Cuenca Grau, E. Kharlamov, Š. Marciuška, and D. Zheleznyakov, "Faceted search over RDF-based knowledge graphs," *Journal of Web Semantics*, vol. 37-38, pp. 55–74, 2016.
[6] C. Nikolaou and M. Koubarakis, "Querying incomplete information in RDF with SPARQL," *Artificial Intelligence*, vol. 237, pp. 138–171, 2016.
[7] E. Marx, K. Höffner, S. Shekarpour et al., *Exploring Term Networks for Semantic Search over RDF Knowledge Graphs*, Springer International Publishing, Berlin, Germany, 2016.

[8] F. Li, W. Le, S. Duan et al., “Scalable keyword search on large RDF data,” *IEEE Transactions on Knowledge & Data Engineering*, vol. 26, no. 11, pp. 2774–2788, 2014.

[9] H. He, H. Wang, J. Yang et al., “BLINKS: ranked keyword searches on graphs,” in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data–SIGMOD ’07*, pp. 305–316, New York, NY, USA, 2007.

[10] C. Choksuchat, C. Chantrapornchai, M. Haidl et al., *Accelerating Keyword Search for Big RDF Web Data on Many-Core Systems*, Springer, Berlin, Germany, 2015.

[11] H. Arnaout and S. Elbassuoni, *Effective Searching of RDF Knowledge graphs*, Social Science Electronic Publishing, New York, NY, USA, 2017.

[12] J. Shi, D. Wu, and N. Mamoulis, “Top-*k* relevant semantic place retrieval on spatial RDF data,” in *Proceedings of the 2016 International Conference on Management of Data–SIGMOD ’16*, pp. 1977–1990, New York, NY, USA, 2016.