International Journal of Mobile Computing and Multimedia Communications
Volume 13 • Issue 1

Large-Scale Commodity Knowledge Organization and Intelligent Query Optimization

Ya Zhou, Hunan International Economics University, China*

ABSTRACT

The internet is facing the era of knowledge interconnection Web 3.0, and its goal is to realize a more intelligent network that can be understood by both humans and machines. In this environment, various types of knowledge graphs have emerged. Because of the heterogeneity of knowledge, commodity knowledge makes its management more challenging. A large-scale product knowledge organization framework is designed, objective product classification knowledge is combined with subjective user perspectives in the framework, a neural network-based learning index technology is proposed to improve query efficiency. According to the properties of the knowledge structure and the characteristics of query requirements, a connection strategy is realized based on sub-variable combination. The experimental results show that, compared with the existing knowledge management system, the proposed method has a significant improvement in the retrieval efficiency of large-scale commodity knowledge.

KEYWORDS

Knowledge Organization, Learned Index, Product Knowledge Graph, Query Optimization

INTRODUCTION

With the continuous development of Web technology, the Internet is moving from the Web 2.0 era characterized by the interconnection of people to the Web 3.0 era of knowledge interconnection (Sheth, A., & Thirunarayan, K. 2013). Its goal is to realize the Internet that can be understood by both humans and machines. It makes the network more intelligent. In this context, how to knowledge and efficiently manage the massive data on the Web so that it can provide users with higher-quality information services has become a hot issue that academia and industry are concerned about. In 2012, Google took the lead in launching the knowledge graph, and it was used as an auxiliary knowledge base to enhance its search function and build a next-generation intelligent search engine. Subsequently, various types of knowledge graphs have been launched, such as Wikipedia-based YAGO (Suchanek, F. M., et al. 2008; Hoffart, J., et al. 2013; Mahdisoltan,i F., Biega, J., & Suchanek, F. M. 2015), Dbpedia (Auer, S. O. R., et al. 2007) and Freebase (Bollacker, K. D., et al. 2008).

At the same time, commodity-related data on the Internet is also increasing rapidly, while the demand for accurate acquisition of commodity information by upper-level applications/users is difficult to meet. The contradiction between the two has not been alleviated, and there is a situation that is getting worse. The main reason for this contradiction is that, on the one hand, most of the data carrying product information exists in an unstructured form, which severely limits their automated and

DOI: 10.4018/IJMCMC.297965 *Corresponding Author intelligent applications; on the other hand, these large-scale information lack efficient data management mechanism, it causes users to directly face fragmented and highly redundant information, which further exacerbates the problem of information overload. Knowledgeable and structured processing of massive data containing commodity information on the Web, and achieving unified and efficient management can not only effectively resolve the contradiction, but also provide users with more comprehensive and accurate information services (Kim, H. 2017; Bartalesi, V., & Meghin,i C. 2017). How to efficiently retrieve large-scale commodity knowledge has become an important issue. There are many knowledge retrieval systems that support large-scale knowledge graphs, such as SW-store (Abadi, D. J., et al. 2009; Abadi, D. J., Marcus, A., Madden, S., et al. 2007), RDF-3x (Neumann, T., & Weikum, G. 2008; Neumann, T., & Weikum, G. 2010; Neumann, T., et al., 2010), Hexastore (Weiss, C., Karras, P., & Bernstein, A. 2008) and gStore (Zou, L., et al. 2014; Zou, L., Mo, J., et al. 2011; Zeng, L., & Zou, L. 2018). When storing data, the URI (Uniform Resource Identifier) text in the knowledge graph is converted into an ID value through a mapping dictionary, thereby reducing the cost of data storage and query. Knowledge retrieval systems based on graph models, such as gStore, can use the graph structure characteristics of the knowledge graph to process knowledge retrieval, and there is high query efficiency.

When processing large-scale queries, a large amount of text needs to be converted into corresponding ID values, which leads to frequent access to the mapping dictionary. At this time, the time cost of the mapping dictionary cannot be ignored. In addition, the retrieval system based on the graph model cannot make full use of the structural features of the product knowledge graph when processing product knowledge queries, resulting in low performance when querying product viewpoint knowledge, and it cannot meet the requirements of product knowledge retrieval performance. This paper focuses on the above-mentioned problems and the main contributions are as follows:

- A commodity knowledge graph framework that combines objective and subjective knowledge is proposed to realize the unified organization of objective commodity information and user viewpoint information.
- A mapping dictionary with the learning index is proposed to improve the query speed of the mapping dictionary (Kraska, T., et al. 2018), and the prefix tree is used to further reduce the retrieval time. Experiments show that this method improves the retrieval efficiency of the mapping dictionary.
- A combination strategy of commodity attribute viewpoint variables is proposed. This method is based on the structural characteristics of commodity knowledge graphs. Experiments prove that this method effectively reduces the time of commodity knowledge retrieval.

RELATED WORK

RDF is a description form of knowledge graph/semantic network/ontology database data, describing entities, attributes, relationships, etc., of course, it can also be understood as a file-based knowledge base storage method. The flexibility offered by the Resource Description Framework (RDF) has led it to become a very popular standard for representing data with an undefined or variable schema using the concept of triples. Its success has resulted in many large scale multidisciplinary datasets, that have prompted the development of efficient RDF processing systems. Current approaches can be distinguished into two groups: the first, adopting the relational model storing the triples in tables, and the second creating data structures that model RDF data as a graph. The strategies of the first group are more easily scalable since they apply optimization strategies from the relational model like indexing and fragmentation. However, these approaches suffer many overheads when dealing with complex queries (e.g. compounded SPARQL graphs involving filters) persistent in existing applications. Three lists construct a table with three column attributes, and insert RDF triple data directly into the table. Although this method is simple to implement, it contains a lot of self-join operations when processing complex queries. The horizontal table regards all the attributes in the RDF data as the column attributes of the table, avoiding the self-join operation, but leading to a large number of null values. Attribute tables reduce the generation of null values by clustering attributes (Wilkinson, K., et al. 2003), but clustering needs to be handed over to data experts. When data is updated, re-clustering may be required, resulting in huge table maintenance costs. Each attribute is split vertically to construct a two-list (Abadi, D. J., et al. 2009; Abadi, D. J., et al. 2007), store the subject and the object separately, RDF triples are classified according to the attributes, and stored in the corresponding table, with good performance. But when the attribute is a variable, all tables need to be traversed, resulting in a sharp drop in performance. The full permutation index is to perform the full permutation and combination of the elements in the RDF triples, and build the index (Neumann, T., & Weikum, G. 2008; Neumann, T., & Weikum, G. 2010; Neumann, T., et al., 2010; Weiss, C., Karras, P., & Bernstein, A. 2008), which solves the problem of low efficiency of vertical partition query attributes, but the storage space cost is higher. An RDF data distribution method is presented (Schroeder, R., et al. 2021), which overcomes the shortcomings of the current approaches in order to scale RDF storage both on the volume of data and query processing. A Hetero-GCD2RDF data retrieval approach is proposed (Velu, A., & Thangavelu, M. 2021), it focuses on two aspects (1) Extraction of records from satellite data and represent it as linked data namely Resource Description Framework (RDF) and (2) Implementation of SPARQL query engine to the resultant RDF for data retrieval.

On the other hand, graph-based systems that use more complex data structures fail to efficiently manage the main memory and are not scalable in computer hardware with limited resources. The graph model method is a data graph converted from RDF data. Unlike the relational model method, it can retain the original graph structure. The knowledge query will be transformed into a subgraph matching problem (Zou, L., et al. 2014). Graph division divides the graph into several subgraphs (Yan, Y., et al. 2008; Yan, Y., Wang, C., Zhou, A., et al. 2009), and a bloom filter is constructed for the nodes contained in each subgraph. GRIN uses the distance between nodes to divide the graph by clustering (Udrea, O., Pugliese, A., & Subrahmanian, V. S. 2007), and then constructs a tree index according to the center node and radius of each cluster. In gStore, each entity node is signed (Signature) and encoded, a binary-based signature graph (Signature Graph) is constructed, and a VS-tree index is established on this basis, the search space is reduced. A novel approach is proposed to perform queries (Basic Graph Patterns, Wildcards, Aggregations and Sorting) on RDF data (Khelil, A., et al. 2021). RDF graph exploration is combined with physical fragmentation of triples. SPARQL 1.1 offers a type of navigational query for RDF systems, called regular path query (RPQ). A regular path query allows for retrieving node pairs with the paths between them satisfying regular expressions. Regular path queries are always difficult to be evaluated efficiently because of the possible large search space. There has been no scalable and practical solution. Leon+, an in-memory distributed framework, is presented to address the RPQ problem in the context of the knowledge graph (Guo, X. T., Gao H., & Zou Z. N., 2021). To reduce search space and mitigate mounting communication costs, Leon+ takes advantage of join-ahead pruning via a novel RDF summarization technique together with a path partitioning strategy. The description of resources and their relationships is an essential task on the web. Generally, the web users do not share the same interests and viewpoints. Each user wants that the web provides data and information according to their interests and specialty. The existing query languages, which allow querying data on the web, cannot take into consideration the viewpoint of the user. Introducing viewpoint is proposed in the description of the resources (Djama, O. 2021). The Resource Description Framework (RDF) represents a common framework to share data and describe resources. A View-Point Resource Description Framework (VP-RDF) is proposed as an extension of RDF by adding new elements. VP-RDF can be useful in intelligent systems on the web.

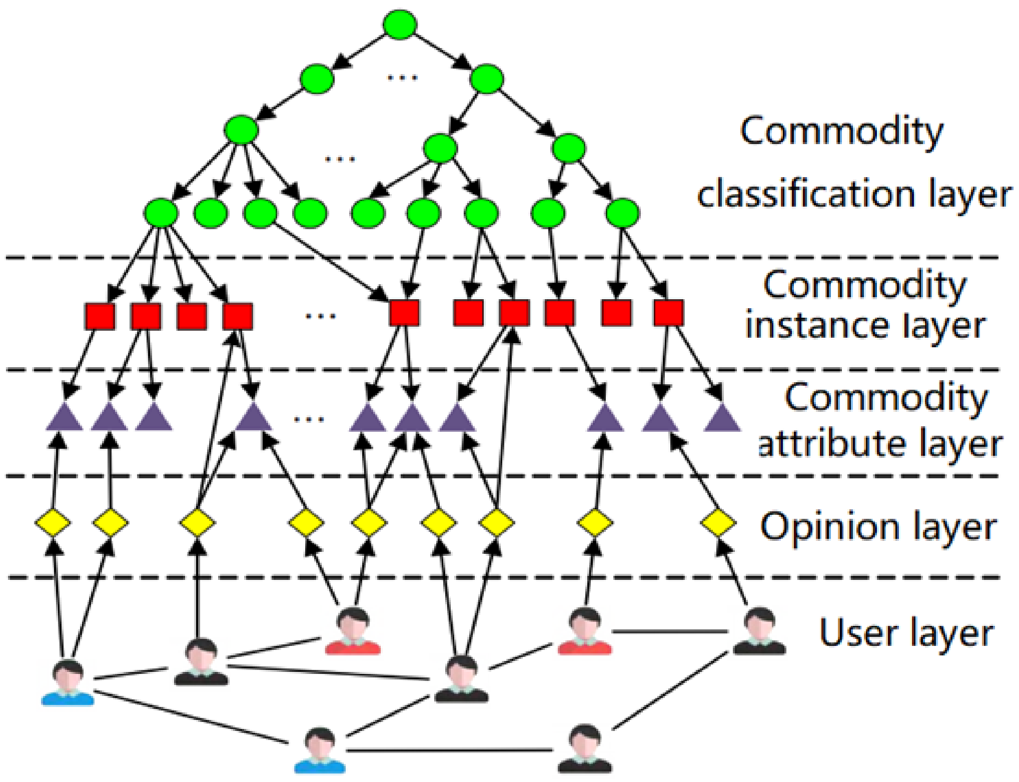
In general, the relational mode method destroys the original graph structure of RDF data, resulting in additional storage or query costs. The graph model method can effectively save the graph structure of RDF data and provide better query performance. However, depending on the graph structure of the data, the existing methods have lower performance when retrieving product knowledge.

ORGANIZATION OF LARGE-SCALE COMMODITY KNOWLEDGE

Overall Framework Overview

The commodity knowledge graph assists users to provide knowledge-based retrieval functions. Therefore, the framework should be consistent with the data structure of the online shopping platform. Its outline structure is shown in Figure 1.

Figure 1. Commodity knowledge graph framework



The commodity knowledge graph is divided into 5 layers in total, of which the top two layers are objective knowledge, and the remaining three layers are subjective knowledge. The knowledge contained in each layer is as follows:

- **Commodity classification layer.** Online shopping malls can manage large-scale products by categorizing products and constructing classification trees. Potential consumers can search for products in catalogue style through product classification.
- **Commodity instance layer.** This layer mainly contains product examples and objective information provided by online shopping platforms, such as product name, brand, price, and product attributes provided by the website.
- **Product attribute layer.** This layer mainly includes the attributes of commodities. The attributes here are not the attributes provided by the platform, but the attributes are described by the opinions in the user comments.

- **Opinion layer.** The opinion layer mainly contains the opinions expressed by users, including opinions on commodities or commodity attributes.
- **User layer.** The user layer includes the registered users of the platform, and it is the main body of consumption and opinions on the platform.

Commodity objective knowledge

Objective commodity knowledge is mainly objective information provided by the website, including commodity classification information and commodity information. First, the types are defined for objective knowledge, these are shown in Table 1 .

Table 1. Objective knowledge type

Class	Corresponding entity form	Meaning
Shop	<x,rdf:type,Shop>	x is an entity of online shopping platform
Category	<x,rdf:type,Category>	x is an entity of product classification
Product	<x,rdf:type,Product>	x is an entity of the commodity instance

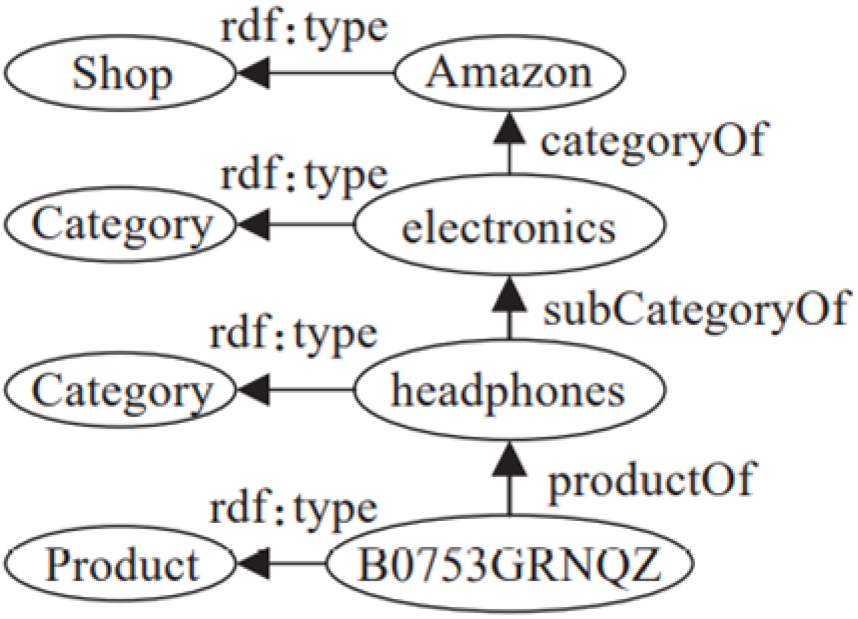
When expressing objective knowledge, the required attributes are shown in Table 2.

Table 2. Objective knowledge attributes

Attributes	Corresponding attribute form	Meaning
categoryOf	<x,categoryOf,y>	x is a product category of y
subCategory	<x,subCategory,y>	x is the product subcategory of y
productOf	<x,productOf,y>	x is the product of y category

For example, in the Amazon shopping platform, Amazon website has an electronics (electronic product) product category, headphones is also a product category, a sub-category of electronics, the product number B0753GRNQZ belongs to the headset category, the corresponding RDF data chart is as follows As shown in Figure 2.

Figure 2. Examples of commodities' objective knowledge



User Comments Subjective Knowledge

The subjective knowledge of user reviews is mainly the opinions expressed by users on the attributes of commodities, that is, the commodity attribute-user opinion word pairs are extracted from the reviews. The subjective knowledge of user reviews is a multiple relationship involving users, products, and product attributes. However, RDF triples cannot directly represent multiple relationships. The solution to this problem is to introduce intermediary nodes, and express multiple relationships through intermediary nodes. Since the user's point of view is contained in the comments written by the user, the user comment is used as an intermediary node, the subjective knowledge of the user comment is realized. Subjective knowledge representation will introduce the following types, which are defined in Table 3.

Table 3.Subjective knowledge type

Class	Corresponding entity form	Meaning
User	< x,rdf:type,User >	x is an entity of the user
Feature	< x,rdftype,Feature >	x is an entity of commodity attributes
Review	< x,rdf:type,Review >	x is an entity of user reviews

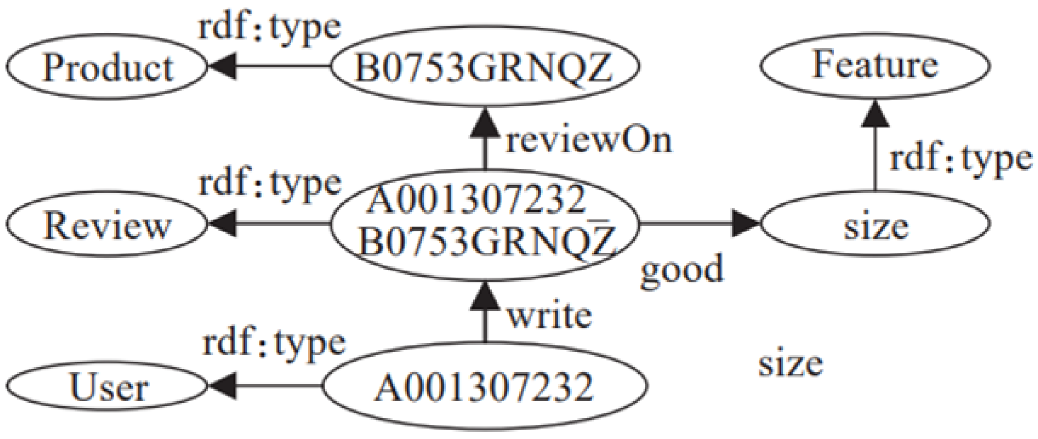
When expressing subjective knowledge, the required attributes are shown in Table 4.

Table 4. Subjective knowledge attributes

Attributes	Corresponding attribute form	Meaning
write	< x,write,y >	User x wrote a review y
reviewOn	< x,reviewOn,y >	Comment x is for product y
opinion	< x,opinion,y >	Comment x comment on attribute y

For example, the user A001307232 thinks that the size of the product B0753GRNQZ is good, and the RDF data graph is used as shown in Figure 3. The opinions expressed by users are regarded as sub-attributes of opinion. In the example, the user opinion good (good) can be expressed as <good, rdfs: subPropertyOf, opinion> through triples.

Figure 3. Examples of commodities' subjective knowledge



LEARNING MAPPING DICTIONARY

When processing large-scale queries at the same time, a large number of URI texts need to be converted into corresponding ID values through a mapping dictionary. The traditional mapping dictionary uses Btree, but as the scale of the mapping dictionary increases, its query efficiency gradually decreases. The learning mapping dictionary is combined with the learning index technology, and the machine learning model is used to fragment the data based on the distribution of the data, thereby improving query efficiency.

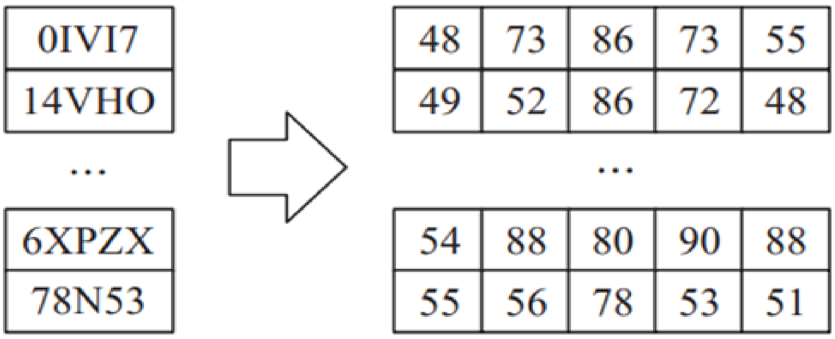
Learning Index

Learning index is to combine traditional index with machine learning technology, the distribution of data is trained to obtain a model that conforms to the data distribution, so that in the traditional index, the query of the data is converted into the prediction of the data location by the model, and the query time is also converted into the execution time of the model in machine learning. At the same time, with the development of high-performance computing hardware such as GPU, TPU and FPGA and its wide application in complex calculations, it is also used in the field of machine learning to improve the execution efficiency of the model, thereby effectively reducing the time cost of retrieval.

Text Distribution

A learning mapping dictionary is constructed for URI text, the distribution of the text needs to be clarified first. Text is a composite data type, which is composed of a combination of multiple characters, which can be converted into a one-dimensional integer array through ASCII encoding, as shown in Figure 4.

Figure 4. The text is converted into an array through ASCII encoding



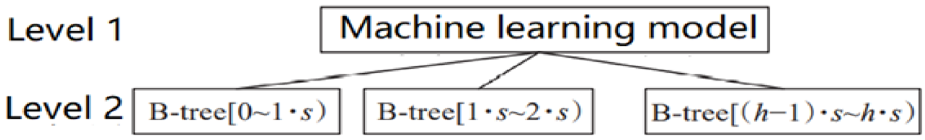
In the commodity knowledge graph, all URI texts are sorted to form an ordered set U $=\{u_1,u_2,\ldots,u_M\}$, and each URI text u_i is converted into an array x_i through ASCII encoding. The first N elements are taken in the array x_i. For the array length n <N, $x^j_i = 0(j> n)$ is set, and the array set $X^N = \{x_1,x_2,\ldots,x_m\}$ is finally obtained. X^N is converted from the URI text set U, the cumulative distribution function is constructed as equation (1).

$$d\left(x\right) = \frac{\sum_{x' \in X^N} x' < x}{M} \tag{1}$$

Learning mapping dictionary structure

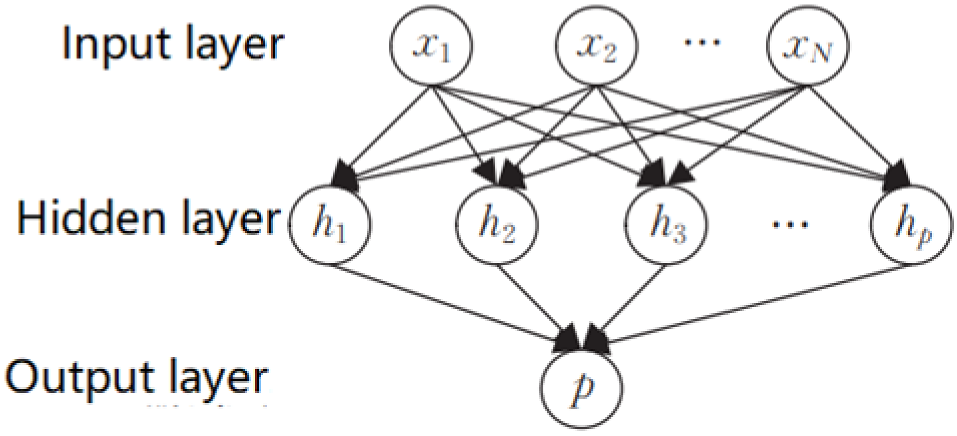
The learning mapping dictionary is divided into two layers. The first layer contains the machine learning model, and the second layer contains the B-tree. In the URI text set, the first layer of machine learning model is used to slice the data, and each data slice saved in the B-tree. Its structure is shown in Figure 5.

Figure 5. Learning mapping dictionary structure



The first layer is the machine learning model. First, the URI text set U is converted into an array set X^N through ASCII encoding, and the neural network is used to train the text distribution, and finally the learning model P(x) is obtained, which realizes the prediction of the position p of the URI text u_i in the data set U. In order to reduce the time cost of the neural network in query execution, a single hidden layer fully connected neural network is used here, as shown in Figure 6, the number of neural units in the input layer is N, and the output value is p.

Figure 6. Single hidden layer fully connected neural network



In the data set U, the position of the URI text ui cannot be negative, and the value span is wide. Therefore, the activation function needs to have a wide output range and avoid the predicted value p £ 0, so in the neural network model, the ReLu function is used as the activation Function, as is shown in equation (2):

$$\mathrm{ReLu} = \begin{cases} 0, x \leq 0 \\ x, x > 0 \end{cases} \tag{2}$$

When training the model, the error value between the real position y of the URI text ui in the data set U and the predicted position p of the model is used, and the variance is used as the loss function, as is shown in equation (3):

$$\mathrm{L} = \frac{1}{\mathrm{M}} \sum_{\mathrm{i}=0}^{\mathrm{M}} \left(\mathrm{p} - \mathrm{y}\right)^2 \tag{3}$$

The second layer contains multiple B-trees. In an ideal state, the first layer model can correctly predict the position of u_i in the text set, but the complexity of the data results in a large error between the predicted position and the true position. Therefore, based on the first layer predicted position, the threshold s is set, the data is divided, that is, when (h-1)×s £ P(x_i) <h ×s, u_i is saved to the h-th data slice, and each slice is saved by using B-tree.

URI Text Compression

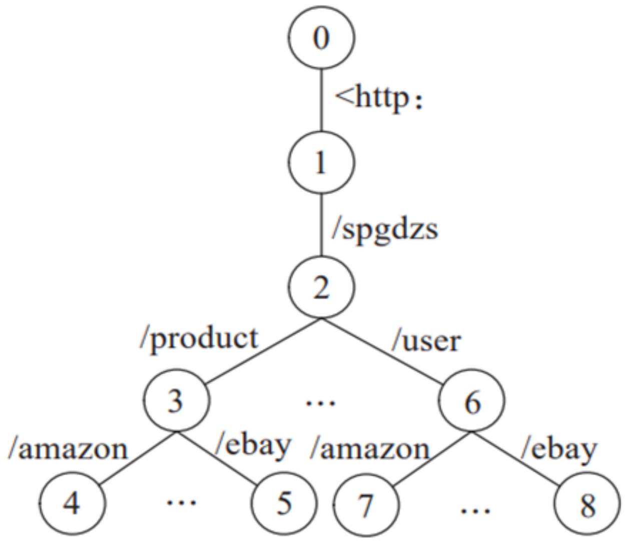
In the commodity knowledge graph, the complete URI text is longer. Table 5 shows the complete URI text corresponding to some types of entities in the commodity knowledge graph. Among them, the URI text of the commodity type entity contains 41 characters, and the first 30 characters are used to represent the namespace. When the number of neural units in the input layer of the neural network N £ 30, for the neural network model, the URIs of all commodity types will be the same, so the number of neural units in the input layer needs to be at least N> 30. In the commodity knowledge graph, the namespace is used as the prefix of the complete URI, and its types are few and fixed, but the text length is generally longer, which causes the neural network model to require more input nodes, this reduces the execution efficiency of the neural network model.

Table 5. Examples of complete URI text corresponding to some types of entities

Entity type	Full URI text
commodity	<http://spgdzs/product/amazon/B00QJDVBFU>
user	<http://spgdzs/user/amazon/AFY2WJ2HD7A4P6TIKGS2F6RUAXGA>

In order to improve the execution efficiency of the machine learning model, the prefix tree is used to compress redundant prefixes in the URI text, and the number of neural units N in the input layer of the neural network model is reduced. The prefix of the URI text is the namespace, the slash (/) symbol is usually used to indicate the directory or classification level of the namespace, so the URI namespace is divided by the slash symbol, and then the prefix tree is constructed. Figure 7 shows the structure of the namespace compression prefix tree. In this structure, the circle represents the node of the prefix tree, the internal number is the number corresponding to the node, and the label of the edge corresponds to the text substring which is obtained after the namespace is divided.

Figure 7. Namespace compression prefix tree



After being compressed by the prefix tree, the complete URI text in Table 5 will be converted to as shown in Table 6. Compared with the complete URI text, the compressed URI text has a shorter length, which reduces the number of neural units N in the input layer of the neural network. This effectively reduces the execution time cost of learning the first layer neural network model of the mapping dictionary.

Table 6. Compressed URI text sample for some types of entities

Entity type	Compress URI text
commodity	4/B00QJDVBFU>
user	7/AFY2WJ2HD7A4P6TIKGS2F6RUAXGA>

COMMODITY KNOWLEDGE QUERY OPTIMIZATION

Connection Cost

After each variable obtains the candidate value set, it is necessary to connect the variables according to the structure of the query graph. The process of connecting a variable is given in Algorithm 1, where IRT represents the intermediate result set, that is, the result set of the subgraph is formed by the connected variables.

```
Algorithm 1: Connect a variable node in algorithm
Input: SPARQL query graph Q, current intermediate result table IRT, variable node v to be connected;
Output: the intermediate result table nIRT after connecting variable node v;
1. Set the new intermediate result set nIRT to be empty, that is, nIRT = ϕ;
2. If the candidate set of variable node v is empty then
3. return nIRT;
4. For each intermediate result record r in IRT do
5. Set the temporary table tmp to be empty, that is, tmp = ϕ;
6. For each element e in intermediate result record r do
7. if e corresponds to the variable node v′ in Q, there is no connected edge between v′ and v, then
8. continue;
9. Obtain another candidate set list of v through e;
10. if tmp is empty then
11. The intersection of list and the candidate set of v is operated, and it is assigned to tmp;
12. else
13. Perform intersection operation between list and tmp, and assign the result to tmp;
14. For each element e in tmp do
15. Create a copy r′ of r and add e to r¢;
16. r¢is added to nIRT;
17. return nIRT;
```

The time cost of the connection operation is mainly composed of two parts: the intersection operation cost and the I/O cost. The intersection operation is affected by the size of the list, and it is difficult to predict its cost before execution. The I/O cost is caused by the operation of accessing the disk. Since the access speed of the disk is several orders of magnitude slower than that of the memory, a huge I/O cost will be caused when the disk is frequently accessed. This causes the I/O cost to become the vast majority of the connection operation cost, so the connection cost can basically be determined by the I/O cost. The I/O cost is caused by connecting a variable in the algorithm, it is determined by the size of the intermediate result set IRT, so the total cost of query graph connection can be jointly determined by the size of the intermediate result set IRT during each step of the connection operation. The query graph contains a set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E = \{e_1, e_2, \ldots, e_m\}$. Assuming that only one edge is connected at a time, the total cost of the connection cost is in equation (4), where S_i represents the size of the intermediate result set IRT and the cost of the i-th step during the i-th connection.:

$$\operatorname{cost}\left(E, V\right) = \sum_{i=1}^{m} S_i \tag{4}$$

Connection Sequence

The cost S_i of the i-th connection is obtained from the completion of the i-1 connection. The cost of each connection operation is obtained from the previous connection operation. Therefore, different connection orders will result in different connection costs. Here, the commodity knowledge data in Table 7 is analyzed.

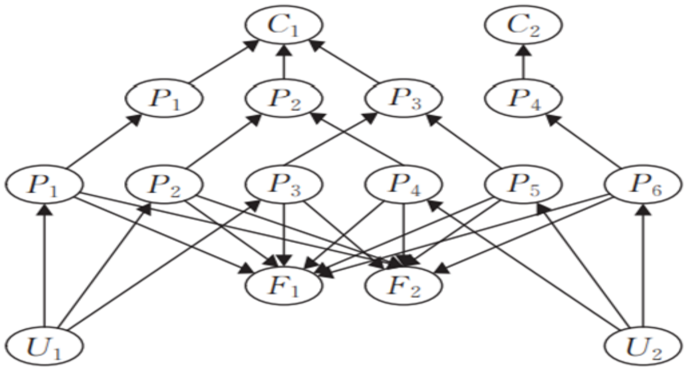
Table 7. RDF triples sample of commodity knowledge graph

Subject	Predicate	Object	Subject	Predicate	Object
P_1	productOf	C_1	R_3	reviewOn	P_3
P_2	productOf	C_1	R_4	reviewOn	P_2
P_3	productOf	C1	R_5	reviewOn	P_3
P_4	productOf	C_2	R_6	reviewOn	P_4
U_1	write	R_1	R_1	O	F_1
U_1	write	R_2	R_1	O	F_2
U_1	write	R_3	R_2	O	F_1
U_2	write	R_4	R_2	O	F_2
U_2	write	R_5	…	…	…
U_2	write	R_6	R_6	O	F_1
R_1	reviewOn	P_1	R_6	O	F_2
R_2	reviewOn	P_2			

Table 7 shows a sample of RDF data for the commodity knowledge graph. For the sake of simplicity, all URIs are expressed in abbreviations, and the first letter of the URI abbreviation is used to indicate the type of the element. For example, the URI starting with P is the product, the URI starting with C is the product category, the URI starting with R is the comment, the URI starting with

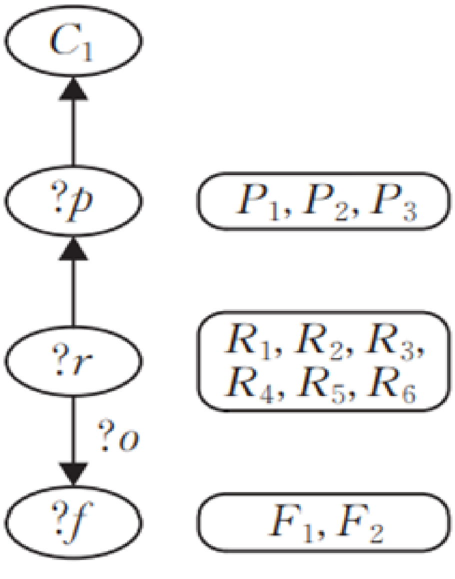
O is the opinion, and the URI starting with F is the product attribute. The data assumes that there are two product categories C_1 and C_2. Among them, there are 3 products under the C_1 category, 1 product under the C_2 category, and two users have posted 3 comments, and each comment contains opinions on features F_1 and F_2. Figure 8 shows the RDF data graph corresponding to this example, where the edge labels are omitted.

Figure 8. Sample RDF data graph of commodity knowledge graph



Suppose query 1 queries the knowledge of C_1 classification of all products and all opinion features. The query graph corresponding to this query is shown in Figure 9. The label of the edge is also omitted, but the label of the variable edge is retained. The set of candidate values for each variable in the figure is marked on the right side of the variable node.

Figure 9. SPARQL query graph and variable candidate nodes



Query 1: All products and all opinion features of C1 category are queried

```
SELECT ?p ?f ?o where {
?p rdf:type Product.
?p productOf C1.
?r rdf:type Review.
?r reviewOn ?p.
?f rdf:type Feature.
?r ?o ?f.
};
```

It can be seen from the figure that the size of the candidate value set of variable ?f is 2, the size of candidate value set of variable ?r is 6, and the size of candidate value set of variable ?p is 3. At present, the latest graph model query system, such as gStore, preferentially select variable nodes with fewer elements in the candidate set for connection when selecting the connection order. Therefore, the query order for this query is ?f ® ?r ® ?p, and the intermediate result set IRT is generated by each step of the connection, it is shown in Table 8.

Table 8. IRT data at each step in the connection process

(a)Initialize	(b) First step		(c) Second step			(d) Third step			
?f	?f	?r	?f	?r	?p	?f	?r	?p	?o
F_1	F_1	R_1	F_1	R_1	P_1	F_1	R_1	P_1	O
F_2	F_1	R_2	F_1	R_2	P_2	F_1	R_2	P_2	O
	…	…	…	…	…	…	…	…	…
	F_1	R_6	F_1	R_5	P_3	F_1	R_5	P_3	O
	F_2	R_1	F_2	R_1	P_1	F_2	R_1	P_1	O
	F_2	R_2	F_2	R_2	P_2	F_2	R_2	P_2	O
	…	…	…	…	…	…	…	…	…
	F_2	R_6	F_2	R_5	P_3	F_2	R_5	P_3	O

For the operations in Table 8, the variable ?r is first selected as the starting node of the connection operation, and the elements in the candidate value set are add to IRT in turn, at this time | IRT | = |C(?f)| = 2; then it connects with the variable ?r. At this time, $S_1 = 2$, after the connection operation ends, | IRT | = 12; it connects with the variable ?p. At this time, $S_2 = 12$, and after the connection is completed, | IRT | =10. Finally, the value of ?o is obtained according to each record variable in IRT, and $S_3 = 10$ at this time. Therefore, the total cost is $S_1 + S_2 + S_3 = 24$.

In the product knowledge graph, user comments are uniquely associated with one product, but they usually contain multiple opinions. According to this data feature, the variable ?p will be selected as the starting node of the connection operation, and the connection sequence is ?p ® ?r ® ?f. The intermediate result set IRT is generated by each step of the connection, it is shown in Table 9.

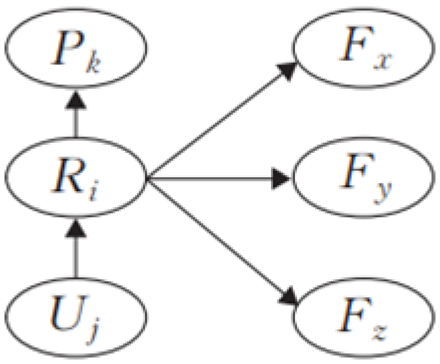
First, the elements in the candidate value set of the variable ?p are added to IRT in turn, at this time | IRT | = |C(?p)| = 3, and then they are connected with the variable ?r, at this time $S_1 = 3$, After the connected completion, | IRT | = 5.Then they are connected with the variable ?f, at this time $S_2 = 5$, after the connection operation is completed, | IRT | = 10. Finally, the value of ?o is obtained according to each record variable in IRT, at this time $S_3 =10$. The total cost of this connection sequence is $S_1 + S_2 + S3 = 18$. In different connection sequence, although the final result is unchanged, but the intermediate result set generated by each step is different.

Table 9. Intermediate result set IRT data for each step in the connection process

(a)Initialize	(b) First step		(c) Second step			(d) Third step			
?p	?p	?r	?p	?r	?f	?p	?r	?f	?o
P_1	P_1	R_1	P_1	R_1	F_1	P_1	R_1	F_1	O
P_2	P_2	R_2	P_1	R_1	F_2	P_1	R_1	F_2	O
P_3	P_2	R_4	P_2	R_2	F_1	P_2	R_2	F_1	O
	P_3	R_3	P_2	R_2	F_2	P_2	R_2	F_2	O
	P_3	R5	…	…	…	…	…	…	…
			P_3	R_5	F_1	P_3	R_5	F_1	O
			P_3	R_5	F_2	P_3	R_5	F_2	O

In the product knowledge graph, each user review R_i is written by a user U_j, and a product P_k connection is described. However, comments will express opinions on multiple product attributes, as is shown in Figure 10.

Figure 10. Data graph of a product comment


$$\begin{tikzcd} P_k & & F_x \\ R_i \arrow[u] \arrow[rru] \arrow[rr] \arrow[rrd] & & F_y \\ U_j \arrow[u] & & F_z \end{tikzcd}$$

In the commodity knowledge graph, the number of commodities and users is huge, while the scale of commodity attributes is relatively small, which leads to prioritized connection of commodity attributes with user reviews during connection operations. However, user reviews are only associated with one product, but usually contain multiple opinions. Therefore, in the product knowledge graph, when product attributes and reviews are connected, the intermediate result set will expand, so the operation of connecting reviews and product attributes during the connection process should be performed at the end.

Variable Combination Connection Strategy

Existing query systems based on graph models are mainly for queries whose edge labels are constant in the query graph. When processing a query with a variable edge label, the connection process first connects the node variables and then obtains the value of the variable edge. This connection strategy leads to lower query performance.

However, in the commodity knowledge graph, the user's opinion is an important piece of knowledge, and there are many queries surrounding the user's opinion. Query 1 shows a simple user's opinion query. However, there are often more complex queries around user opinions, such as querying products related to product P_1. These products have common buyers with P_1, and buyers have the same views on certain attributes of the product. The SPARQL is like query 2.

Query 2: Find the SPARQL of P_1 related products

SELECT ?p ?f ?o where{
?r_1 rdf:type Review.
?r_1 reviewOn P_1.
?r_2 rdf:type Review.
?p rdf:type Product.
?p productOf C_1.
?r_2 reviewOn ?p.
?f rdf:type Feature.
?r_1 ?o ?f.
?r_2 ?o ?f.
?u rdf:type User.
?u write ?r_1.
?u write ?r_2.
}

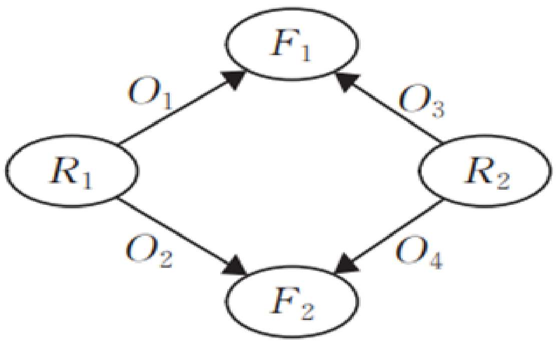
For this query, first exclude variables ?o and ?f, and connect other variables according to the query graph, an intermediate result set is obtained, as is shown in Table 10.

The second result of the intermediate result set is discussed. When R_1 and R_2 respectively published different user opinions on the commodity attributes F_1 and F_2, the RDF data graph is shown in Figure 11.

Table 10. Join intermediate result set of query 2

?r_1	?u	?r_2	?p
R_1	U_1	R_1	P_1
R_1	U_1	R_2	P_2
R_1	U_1	R_3	P_3

Figure 11. A data graph where two user reviews' opinions are different



According to the original connection strategy, the variable ?f will be connected, which will produce the results in Table 11.

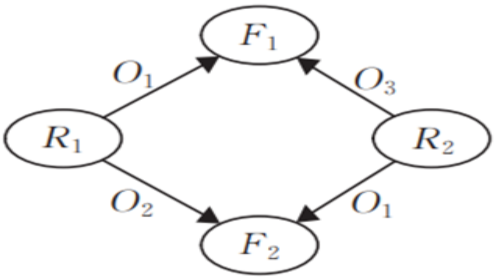
Table 11. Invalid intermediate result list

$?r_1$	?u	$?r_2$	?p	?f
R_1	U_1	R_1	P_2	F_1
R_1	U_1	R_2	P_2	F_2

When finally getting the result of the variable ?o, all the intermediate results in Table 11 are invalid and then deleted. When considering whether the intermediate result needs to be connected to the variable ?f, it is necessary for the values of the variables $?r_1$ and $?r_2$ to have the same edge before they need to be connected. Therefore, when the values of the variables $?r_1$ and $?r_2$ do not have the same edge, the middle result can be deleted directly without the need for further operation.

When two reviews R_1 and R_2 have the same opinion, but for different product attributes, only considering whether there are the same side labels, it will also lead to invalid intermediate results, as is shown in Figure 12.

Figure 12. A data graph where two comments have the same opinion but not the same attribute



The above connection methods are to separate product attributes and user opinions for separate queries, but in the product knowledge graph, product attributes and user opinions are combined and appear in pairs. Therefore, this feature is used in this section to propose a product Aspect and Opinion Combination (AOC) connection strategy, that is, the variable edge ?o and the variable node ?f are combined into a whole variable. First, connect variable nodes other than variables ?f and ?o, and then connect variable combinations. The process of connecting combination variables is shown in Algorithm 2.

```
Algorithm 2: Connect a combined variable algorithm
Input: the current intermediate result table IRT, the combined
variables (e, v) to be connected;
Output: the intermediate result table nIRT after connecting
variable node v;
1. Set the new intermediate result set nIRT to be empty, that is,
nIRT = ϕ;
2. If the candidate set of variable node v is empty then
```

```
3. return nIRT;
4. For each intermediate result record r in IRT do
5. Set the temporary table tmp to be empty, that is, tmp = ϕ;
6. Set the temporary table tmpv to be empty, that is, tmpv = ϕ;
7. for each element ele in the intermediate result record r do
8. If ele and v do not pass through edge e do
9. continue;
10. Add ele to tmpv;
11. Obtain another candidate value set list of v through ele;
12. if tmp is empty then
13. The intersection of list and the candidate set of v is performed, and the result is assigned to tmp;
14. else
15. Intersection operation between list and tmp is performed, and the result is assigned to tmp;
16. if tmp is empty then
17. continue
18. for each element ele in tmp then
19. Obtain ele edge candidate value set tmpe;
20. For each element edge in tmpe then
21. Get the candidate list list through ele and edge
22. if tmpv Í list then
23. Create a copy r¢of r, and add (edge, ele) to r¢;
24. r¢ is added to nIRT;
25. return nIRT;
```

EXPERIMENT AND PERFORMANCE ANALYSIS

Experimental Environment

The hardware environment of this experiment is a single Intel® Core™ i5-6500 @3.20 GHz quad-core processor computer with 16 GB DDR3 memory and 1 TB mechanical hard disk. In order to ensure the fairness of the experiment, hardware acceleration such as GPU, TPU or FPGA is not used. The operating system is a 64-bit Ubuntu 16.04 operating system.

The experimental data will be organized through the commodity knowledge graph framework designed in this paper. The data of the knowledge graph is a combination of real data and simulation data. The overall overview of the data set is shown in Table 12. Among them, product classification, product information, and user knowledge are real data from the Amazon platform. The total number of product entities is 478 626, and the total number of user entities is 1 000 000. User comments and opinion knowledge are simulation data (Mahria, B. B., Chaker, I., & Zahi, A. 2021). Each user randomly selects products and makes comments. The comments contain a random amount of opinion knowledge.

Table 12.Some statistics of the data set

Data parameter	Data value
Number of triples	116174460
Data Format	Turtle
Number of concepts	6
Number of entities	11979407
Number of attributes	10573
Number of opinion words	10566

Experimental Analysis of Learning Mapping Dictionary

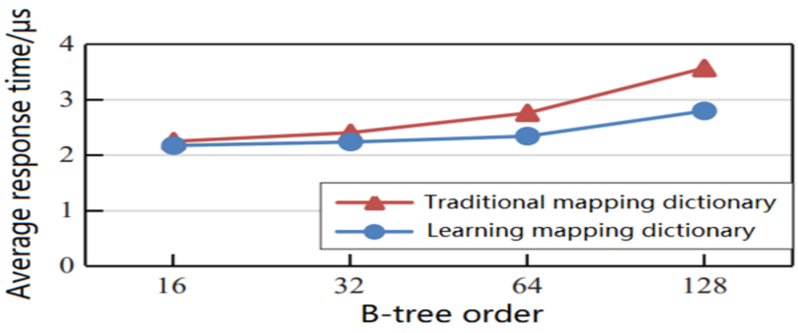
The experiment is analyzed from two aspects: storage space and response time. Since both the learning mapping dictionary and the traditional mapping dictionary adopt the B-tree structure, in order to reduce the difference in the experimental environment, the two use the same B-tree in the experiment. In addition, the order of the B-tree has an impact on the data storage space and query response time, and the order of the B-tree will be discussed. The storage space occupied by the two mapping dictionaries is shown in Table 13.

Table 13. The storage space (MB) is consumed by the data mapping dictionary

Order	Traditional mapping dictionary	Learning mapping dictionary
16	6 673	5 366
32	2 480	2 364
64	1 168	1 121
128	805	783

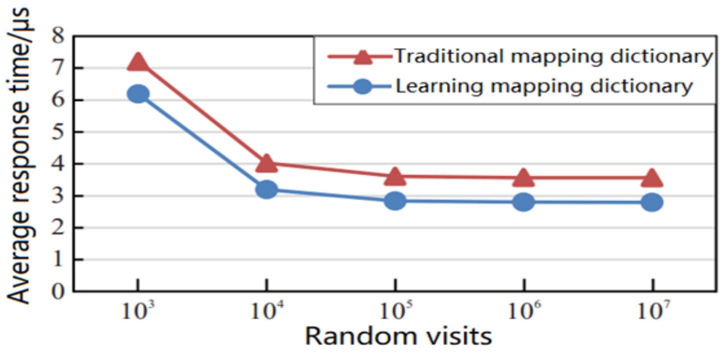
The query efficiency of learning mapping dictionary is further verified. First, the effect of the order of B-tree is discussed on response time. Here randomly query URI text 100,000 times, and then the average response time of single query URI text is calculated, as is shown in Figure 13.

Figure 13. Average response time of different orders' B-tree



For comparison of the number of random queries for URI text, a 128-order B-tree is used here to calculate the average response time of a single query for URI text. The result is shown in Figure 14.

Figure 14. Average query time when B-tree order is 128



Connection strategy experiment

According to the commodity knowledge graph, the experiment designed common commodity knowledge query sentences for the subjective and objective knowledge of commodities, as shown in Table 14..

Table 14. Commodity knowledge query

Numbering	SPARQL query statement
Q1	SELECT ?p WHERE {
	?p rdf: type Product.
	?p productOf C_1.
	?r rdf: type Review.
	?r reviewOn ?p.
	?r O_1 F_1.
	};
Q2	SELECT ?f ?o WHERE {
	?r rdf: type Review.
	?r reviewOn P_1.
	?r ?o ?f.
	};

Table 14 continued on next page

Table 14 continued

Numbering	SPARQL query statement
Q3	SELECT ?f (count(?o) AS ?count) WHERE {
	?p rdf: type Product.
	?p productOf C_1.
	?r rdf: type Review.
	?r reviewOn P_1.
	?r ?o ?f.
	}; GROUP BY ?f
Q4	SELECT ?u ?f ?o WHERE {
	$?r_1$ rdf: type Review.
	U_1 write $?r_1$.
	?p rdf: type Product.
	$?r_1$ reviewOn ?p.
	?f rdf: type Feature.
	$?r_1$?o ?f.
	?u rdf: type User.
	$?r_2$ rdf: type Review.
	?u write ?2.
	$?r_2$ reviewOn ?p.
	$?r_2$?o ?f.
	};

The meaning of the above SPARQL query is as follows:

Q1: Under the product category C_1, find the product whose attribute F_1 has an O_1 opinion.
Q2: Query all user opinions and corresponding product attributes of product P_1.
Q3: Inquire about all the products under a product category C_1, which attributes are more concerned by users. By querying all the products under the product category and the corresponding user opinions and characteristics, the data is aggregated through the product attributes, and then the quantity is counted.
Q4: Query users who have the same hobbies as user U_1. These users not only bought the same product as user U_1, but they also expressed the same opinion on a certain product attribute of the product.

The results of the SPARQL query running time are shown in Table 15.

Table 15. SPARQL query running time

Numbering	Q1	Q2	Q3	Q4
gStore	290 ms	913 ms	>1 h	>1 h
AOC	287 ms	271 ms	2 285 ms	2 389 ms

Performance Analysis

According to the results in **Table 13**, as the order of B-tree increases, the storage space of the two mapping dictionaries tends to decrease. When the two mapping dictionaries adopt the same order, the storage space of the learning mapping dictionary is less than that of the traditional mapping dictionary. The first layer in the learning mapping dictionary only needs to save the parameters of the machine learning model, and its storage space is much smaller than the overall storage space. In this experiment, the model storage space occupies about 1 KB. In the second layer of the learning mapping dictionary, the first layer learning model divides the data into multiple small-scale data fragments, and the height of the B-tree corresponding to each data fragment is not higher than the B-tree in the traditional mapping dictionary at the same time, the total number of keys contained in all B-tree internal nodes is also less than that of B-tree in the traditional mapping dictionary. Since the number of internal node keys occupies most of the storage space, the storage space occupied by the learning mapping dictionary is smaller than that of the traditional mapping dictionary.

The experimental results in **Figure 13** show that when B-tree adopts different orders, the average response time of learning mapping dictionary is shorter than that of traditional mapping dictionary. In addition, with the increase of the B-tree order, the efficiency of learning mapping dictionary query is more obvious.

Experimental results in **Figure 14** show that the average response time of learning mapping dictionary is shorter than that of traditional mapping dictionary when dealing with different random access times.

The learning mapping dictionary uses a machine learning model to divide a large-scale URI text set into multiple small-scale URI text set pieces. When querying URI text, only one of the URI text set pieces needs to be queried. Compared with traditional mapping dictionaries, learning mapping dictionary reduces the number of comparison operations in the query process. At the same time, the machine learning model of the first layer adopts a single hidden layer fully connected neural network model, which has low model complexity and high execution efficiency. CPU processor is used to execute this model, the average single execution time is less than 100 ns, which is less than the overall query time.

The experimental results in **Table 15** show that the query performance is similar in query Q1. Since the query does not include queries on product attributes and user opinions, the optimization strategy has no effect. For queries Q2, Q3, and Q4, these queries all include product attributes and user opinions. At this time, the optimized query strategy comes into play. The system using the AOC strategy has higher query efficiency than gStore. Among them, the query Q2 statement is relatively simple, there is no complicated relationship, and the query is for a product, and the amount of related information is small. The system using the AOC connection strategy has improved the performance of gStore by 2 times. The query Q3 statement is also relatively simple, but the query involves the data of all products in a product category, and the amount of related data is large. The gStore query has not obtained results for more than one hour, and the system using the AOC strategy completes the query in a relatively short time. The generation of intermediate results are reduced, and the efficiency of the query is improved. The language complexity of query Q4 is greater than that of queries Q2 and Q3. gStore also cannot complete the query within 1 h. The system using the AOC strategy completes the query in a relatively short time, which avoids generating invalid intermediate results, thereby improving Query efficiency.

CONCLUSION AND OUTLOOK

The ever-increasing amount of RDF data made available requires data to be partitioned across multiple servers. We have witnessed some research progress made towards scaling RDF query processing based on suitable data distribution methods. In general, they work well for queries matching simple triple patterns, but they are not efficient for queries involving more complex patterns.

In order to unify the management of objective information of commodities and subjective information of users' viewpoints, a commodity knowledge graph framework is designed in this paper, the framework combines objective and subjective knowledge of commodities. Then, in order to improve query efficiency, reduce the cost of converting URI text into corresponding ID values, learning indexes are combined to improve query efficiency, and prefix trees are used to compress redundant prefixes in URI text, query efficiency is further improved. In order to improve the efficiency of commodity knowledge query, the connection strategy of the combination of commodity attribute viewpoint variables is designed, which has higher commodity knowledge query efficiency. Finally, the performance improvement of the method proposed in this paper is verified on the commodity knowledge graph data.

We apply a method that identifies frequent patterns accessed by queries in order to keep related data in the same partition. We deploy our reasoning on a summarized view of data in order to avoid exhaustive analysis on large datasets. As result, partitioning templates are obtained from data items in an RDF structure. In addition, we provide an approach for dynamic data insertions even if new data do not conform to the original RDF structure. Apart from the repartitioning approaches, we use an overflow repository to store data which may not follow the original schema. Our study shows that our method scales well and is effective to improve the overall performance by decreasing the amount of message passing among servers, compared to alternative data distribution approaches for RDF. In terms of principle, neural network is still a supervised traditional machine learning method; in terms of structure, there are many parameters and it is easy to lose spatial information. In future work, we will consider using GPUs to build heterogeneous computing platforms, the efficiency of product knowledge retrieval is further improved.

FUNDING AGENCY

Publisher has waived the Open Access publishing fee.

ACKNOWLEDGMENT

This work was supported by the Scientific Research Project (NO.20B337) of Hunan Provincial Education Department, China & Hunan Natural Science Foundation(Grant No: 2019JJ40154), Hunan Province, China.

REFERENCES

Abadi, D. J., Marcus, A., & Madden, S. (2007). Scalable semantic Web data management using vertical partitioning. *Proceedings of the 33rd International Conference on Very Large Data Bases*, 411-422.

Abadi, D. J., Marcus, A., Madden, S., & Hollenbach, K. (2009). SW-Store: A vertically partitioned DBMS for semantic Web data management. *The VLDB Journal*, *18*(2), 385–406. doi:10.1007/s00778-008-0125-y

Auer, S. O. R., Bizer, C., & Kobilarov, G. (2007). DBpedia: A nucleus for a web of open data. *Proceedings of the 6th International Semantic Web Conference*, 722-735. doi:10.1007/978-3-540-76298-0_52

Bartalesi, V., & Meghini, C. (2017). Using an ontology for representing the knowledge on literary texts: The dante alighieri case study. *Semantic Web*, *8*(3), 385–394. doi:10.3233/SW-150198

Bollacker, K. D., Evans, C., & Paritosh, P. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, 1247-1250. doi:10.1145/1376616.1376746

Djama, O. (2021). VP-RDF: An RDF Based Framework to Introduce the Viewpoint in the Description of Resources. *Applied Computer Systems*, *26*(1), 44–53. doi:10.2478/acss-2021-0006

Guo, X. T., Gao, H., & Zou, Z. N. (2021). Distributed processing of regular path queries in RDF graphs. *Knowledge and Information Systems*, *63*(4), 993–1027. doi:10.1007/s10115-020-01536-2

Hoffart, J., Suchanek, F. M., Berberich, K., & Weikum, G. (2013). YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, *194*, 28–61. doi:10.1016/j.artint.2012.06.001

Khelil, A., Mesmoudi, A., Galicia, J., Bellatreche, L., Hacid, M. S., & Coquery, E. (2021). Combining Graph Exploration and Fragmentation for Scalable RDF Query Processing. *Information Systems Frontiers*, *23*(1), 165–183. doi:10.1007/s10796-020-09998-z

Kim, H. (2017). Towards a sales assistant using a product knowledge graph. *Journal of Web Semantics*, *46-47*, 14–19. doi:10.1016/j.websem.2017.03.001

Kraska, T., Beutel, A., & Chi, E. H. (2018). The case for learned index structures. *Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data*, 489-504. doi:10.1145/3183713.3196909

Mahdisoltan, F., Biega, J., & Suchanek, F. M. (2015). YAGO3: a knowledge base from multilingual wikipedias. *Proceedings of the 7th Biennial Conference on Innovative Data Systems Research.*

Mahria, B. B., Chaker, I., & Zahi, A. (2021). An empirical study on the evaluation of the RDF storage systems. *Journal of Big Data*, *8*(1), 100. Advance online publication. doi:10.1186/s40537-021-00486-y

Neumann, T., & Weikum, G. (2008). RDF-3X: A RISC-style engine for RDF. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, *1*(1), 647–659. doi:10.14778/1453856.1453927

Neumann, T., & Weikum, G. (2010). The RDF-3X engine for scalable management of RDF data. *The VLDB Journal*, *19*(1), 91–113. doi:10.1007/s00778-009-0165-y

Neumann, T., & Weikum, G. (2010). x-RDF- 3X: Fast querying, high update rates, and consistency for RDF databases. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, *3*(1), 256–263. doi:10.14778/1920841.1920877

Schroeder, R., Penteado, R. R. M., & Hara, C. S. (2021). A data distribution model for RDF. *Distributed and Parallel Databases*, *39*(1), 129–167. doi:10.1007/s10619-020-07296-w

Sheth, A., & Thirunarayan, K. (2013). *Semantics empowered Web 3.0 managing enterprise,social,sensor,and cloud- based data and services for advanced applications*. Morgan and Claypool. doi:10.2200/S00433ED1V01Y201207DTM031

Suchanek, F. M., Kasneci, G., & Weikum, G. (2008). YAGO: A large ontology from wikipedia and wordNet. *Journal of Web Semantics*, *6*(3), 203–217. doi:10.1016/j.websem.2008.06.001

Udrea, O., Pugliese, A., & Subrahmanian, V. S. (2007). GRIN: A graph based RDF index. *Proceedings of the 22nd National Conference on Artificial Intelligence*, 1465-1470.

Velu, A., & Thangavelu, M. (2021). Hetero-GCD2RDF: An Interoperable Solution for Geospatial Climatic Data by Deploying Semantic Web Technologies. *Wireless Personal Communications*, *117*(4), 3527–3551. doi:10.1007/s11277-021-08365-8

Weiss, C., Karras, P., & Bernstein, A. (2008). Hexastore: Sextuple indexing for semantic web data management. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, *1*(1), 1008–1019. doi:10.14778/1453856.1453965

Wilkinson, K., Sayers, C., & Kuno, H. A. (2003). Efficient RDF storage and retrieval in jena2. *Proceedings of the 1st International Workshop on Semantic Web and Databases*, 131-150.

Yan, Y., Wang, C., & Zhou, A. (2008). Efficiently querying rdf data in triple stores. *Proceedings of the 17th International Conference on World Wide Web*, 1053-1054. doi:10.1145/1367497.1367652

Yan, Y., Wang, C., & Zhou, A. (2009). Efficient indices using graph partitioning in RDF triple stores. *Proceedings of the 25th International Conference on Data Engineering*, 1263-1266. doi:10.1109/ICDE.2009.216

Zeng, L., & Zou, L. (2018). Redesign of the gStore system. *Frontiers of Computer Science*, *12*(4), 623–641. doi:10.1007/s11704-018-7212-z

Zou, L., Mo, J., Chen, L., Özsu, M. T., & Zhao, D. (2011). gStore: Answering SPARQL queries via subgraph matching. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, *4*(8), 482–493. doi:10.14778/2002974.2002976

Zou, L., Ozsu, M. T., Chen, L., Shen, X., Huang, R., & Zhao, D. (2014). gStore: A graph-based SPARQL query engine. *The VLDB Journal*, *23*(4), 565–590. doi:10.1007/s00778-013-0337-7

Ya Zhou (b. 1981) received the B.S. degree in Tourism and hotel management from Chongqin Business University in 2003 and received the M.S. degree in Management Science and Engineering from Central South University. Now, she is an associate professor at Business School, Hunan University of International Economics, China. Her research interests include hotel management and organizational behavior.