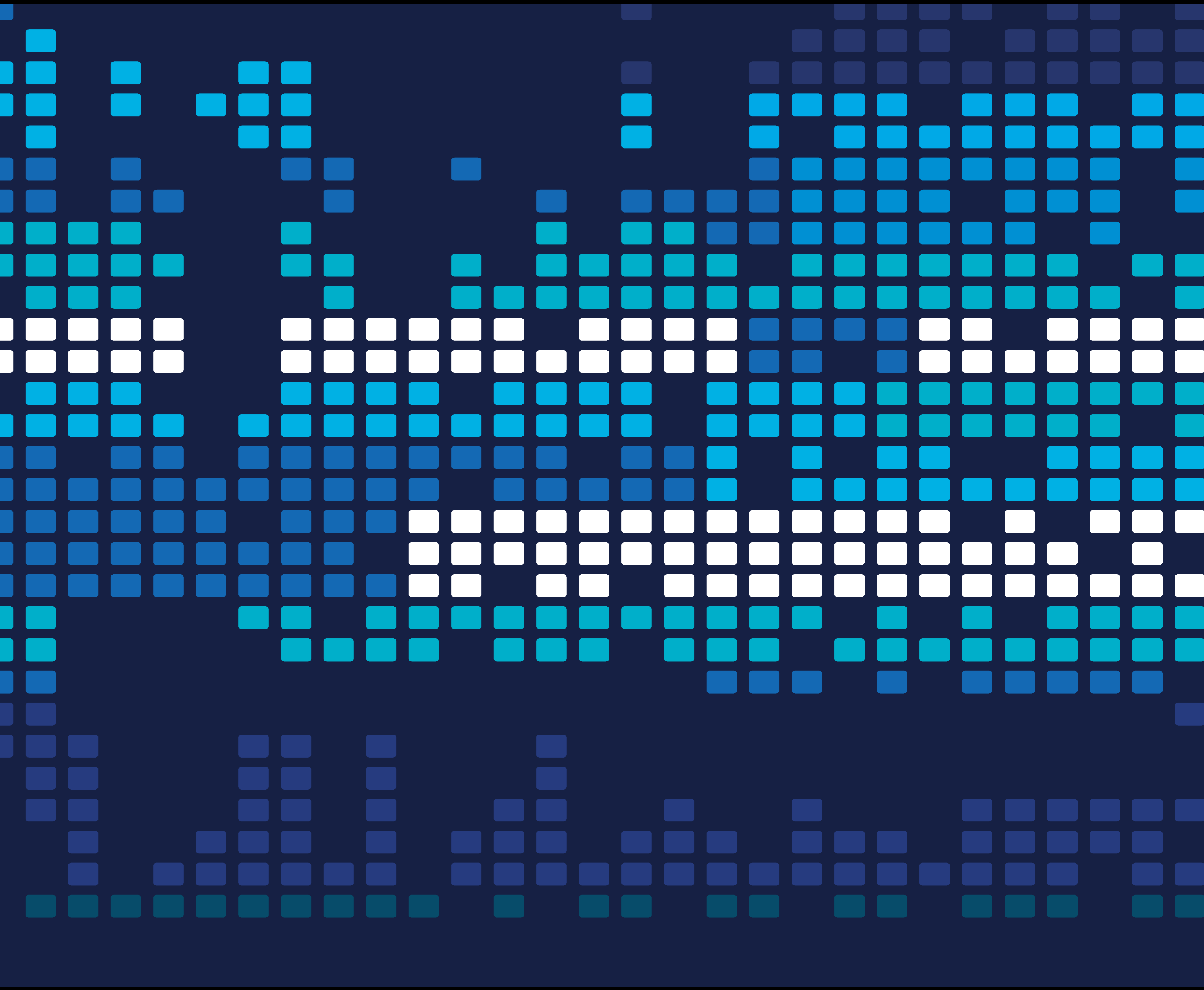
Scientific Programming

Big Data Management and Analytics in Scientific Programming

Lead Guest Editor: Aibo Song
Guest Editors: Guandong Xu, Jianming Bian, Jinghui Zhang, and Zhiang Wu



Big Data Management and Analytics in Scientific Programming

Big Data Management and Analytics in Scientific Programming

Lead Guest Editor: Aibo Song
Guest Editors: Guandong Xu, Jianming Bian,
Jinghui Zhang, and Zhiang Wu

Contents

Hindawi
Scientific Programming
Volume 2020, Article ID 7084958, 9 pages
https://doi.org/10.1155/2020/7084958

Research Article

Multiview Translation Learning for Knowledge Graph Embedding

Chenzhong Bin,[1,2] Saige Qin,[2] Guanjun Rao,[2] Tianlong Gu,[2] and Liang Chang[2]

[1]*School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China*
[2]*Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China*

Correspondence should be addressed to Chenzhong Bin; binchenzhong@163.com and Tianlong Gu; cctlgu@guet.edu.cn

Received 14 October 2019; Accepted 9 July 2020; Published 25 August 2020

Academic Editor: Zhiang Wu

Recently, knowledge graph embedding methods have attracted numerous researchers' interest due to their outstanding effectiveness and robustness in knowledge representation. However, there are still some limitations in the existing methods. On the one hand, translation-based representation models focus on conceiving translation principles to represent knowledge from a global perspective, while they fail to learn various types of relational facts discriminatively. It is prone to make the entity congestion of complex relational facts in the embedding space reducing the precision of representation vectors associating with entities. On the other hand, parallel subgraphs extracted from the original graph are used to learn local relational facts discriminatively. However, it probably causes the relational fact damage of the original knowledge graph to some degree during the subgraph extraction. Thus, previous methods are unable to learn local and global knowledge representation uniformly. To that end, we propose a multiview translation learning model, named MvTransE, which learns relational facts from global-view and local-view perspectives, respectively. Specifically, we first construct multiple parallel subgraphs from an original knowledge graph by considering entity semantic and structural features simultaneously. Then, we embed the original graph and construct subgraphs into the corresponding global and local feature spaces. Finally, we propose a multiview fusion strategy to integrate multiview representations of relational facts. Extensive experiments on four public datasets demonstrate the superiority of our model in knowledge graph representation tasks compared to state-of-the-art methods.

1. Introduction

Knowledge graphs [1] are a sort of directed graphs consisting of entities as nodes and relations between entities as edges. And each relational fact of knowledge graphs is stored as a triplet (*head*, *relation*, *tail*), abbr. (h, r, t), where h and t represent head and tail entities, respectively, and r is a relationship from h to t. With the advent of the big data era, the scale of knowledge graph continues to grow; diverse large-scale knowledge graphs (e.g., WordNet [2] and Freebase [3]) have appeared. Despite the large scale of current knowledge graphs, they are still far from the knowledge completeness. For example, 75% of people in Freebase lack nationality information and 71% lack birthplace [4]. Therefore, it is necessary to design approaches to automatically complete or infer missing relational facts of the existing knowledge graph.

Recently, embedding-based approaches present strong feasibility and robustness in terms of knowledge graph completion, which project entities and relations of knowledge graphs into a dense, continuous, and low-dimensional vector space. Among the existing approaches, translation-based approaches have attracted numerous researchers' interests due to the outstanding effectiveness and robustness in the knowledge representation. The first translation-based method was proposed by Bordes et al., named TransE [5]. For each triplet (h, r, t), TransE treats a relation r as a translation operation from h to t in a vector space. If (h, r, t) holds, the translation principle $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ should be satisfied in the vector space, where $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$ are vector representations of h, r, and t. TransE is a simple yet effective translation model in processing 1-to-1 simple relational facts, which stands for each single head entity connecting only one tail entity via a specific relation. It achieves state-of-the-art performance on link prediction. However, the translation principle is too rigid to deal with complex relational facts, including 1-to-*N*, *N*-to-1, and *N*-to-*N* facts. Technically, it may cause spatial congestion of entities when many head entities (or tail entities) are projecting at only one point of the vector space.

To eliminate the weakness of TransE in representing complex relational facts, a series of improved models were proposed, such as TransH [6], TransR [7], TransD [8], and TranSparse [9]. Essentially, the above methods focused on designing various translation principles to learn complex relational facts more precisely. However, they still embed a complete knowledge graph into a single vector space from a global perspective while they fail to learn the various type of relational facts discriminatively. That is, each entity and relation of knowledge graphs are learned as corresponding unique representing vectors in their spaces. Hence, there are still vector congestions in the translation spaces. For a real world example, the head entities of the triples (*Obama, President, the United States*) and (*Trump, President, the United States*), i.e., Obama and Trump, are projected closely in the vector space due to the same social status. But, they are quite different in other perspectives.

To solve this problem, puTransE [10] splits knowledge graphs into multiple parallel spaces in the form of subgraphs from a local perspective and achieves the spatial sparsification of complex relational facts. Specifically in puTransE, entities and relations have respective feature representations in different parallel spaces. This approach improves the ability to learn complex relational facts due to avoiding the spatial congestion of entities in complex relational facts. However, puTransE still has two shortcomings. First, puTransE poses excessive sparseness to simple relational facts during the parallel space generation. Consequently, it hardly learns the complete vector representations of simple relational facts within a single subspace. This is because puTransE performs the spatial sparsification not only to complex relational facts but also to simple ones. Second, puTransE randomly selects local knowledge to construct multiple parallel spaces, which is prone to impair relational facts of the original knowledge graph. For example, there is a golden triplet in the original graph that cannot be consisted by entities and relations in any parallel spaces.

In summary, all of the above methods embed knowledge graph from a single perspective, i.e., from a local-view or global-view. Thus, they fail to learn local and global knowledge representation uniformly. To that end, we borrow the idea of multiview learning methods [11, 12] to propose a multiview translation learning model, named MvTransE, which embeds relational facts from global-view and local-view, concurrently. In detail, we first generate multiple parallel subgraphs through semantic and structural perspectives of entities to accurately capture local-view knowledge of the knowledge graph. Then, the original knowledge graph and generated parallel subgraphs are embedded into global-view and local-view spaces, respectively. Finally, we propose a multiview fusion strategy to integrate multiview representations of relational facts. We outline the main contributions of this paper as follows:

(1) We incorporate the idea of multiview learning into our model MvTransE, which can precisely learn relational facts from both global and local views
(2) Our model extracts local knowledge from semantic and structural perspectives to construct multiple parallel subgraphs so as to solve the entity spatial congestion problem by learning local-view representations of relational facts
(3) MvTransE applies a multiview fusion strategy to combine global-view and local-view representations of the knowledge graph, which effectively overcomes the missing of relational facts in parallel spaces
(4) Extensive experiment results demonstrate our method outperforms state-of-the-art models in two knowledge graph completion tasks

2. Related Work

Since the appearance of knowledge graphs, numerous researchers have studied various methods to represent relational facts of the graphs. Initially, some embedding-based models such as Structured Embedding (SE) [13], Semantic Matching Energy (SME) [14, 15], Latent Factor Model (LFM) [16], and Neural Tensor Network (NTN) [17] achieved considerable performance in knowledge representation, while fail to cope with large-scale graphs due to the computing complexity. Recently, translation-based models have attracted lots of attention due to their effective and robust representation abilities.

TransE [5] is the first translation-based method, which treats a relation *r* as a translation from *h* to *t* for a triplet (*h, r, t*). Hence, TransE defines the scoring function as $f_r(h,t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{l1/l2}$, where $\|\cdot\|_{l1/l2}$ stands for norm-1 or norm-2 computation. During the model training, if a triplet (*h, r, t*) holds, the translation principle $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ should be satisfied in the vector space, of which process is illustrated in Figure 1. That is, TransE keeps translation vectors $(\mathbf{h} + \mathbf{r})$ approximate to tail vector $\mathbf{t}$. TransE achieves remarkable performance in representing simple relational facts, i.e., 1-to-1 triplets. However, it has limitations in dealing with complex relationship facts including 1-to-N, N-to-1, and N-to-N triplets due to the rigid translation principle.

TransH [6] tries to solve the problem of TransE by implementing an entity to have unique representations when the entity is involved in different relations. Specifically, for each triplet (*h, r, t*), TransH projects *h* and *t* to a relation *r* specific hyperplane to obtain projected vectors $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^{\mathrm{T}}\mathbf{h}\mathbf{w}_r$ and $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_h^{\mathrm{T}}\mathbf{t}\mathbf{w}_{\mathbf{h}}$. The scoring function is defined as $f_r(h,t) = \|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|_{l1/l2}$.

TransR/CTransR [7] models entities and relations in different vector spaces, respectively, i.e., the entity vector space and the relation vector space. For each relation *r*, it set a projection matrix $\mathbf{M}_r$ to map the vector of entities from the entity vector space to the relation vector space, i.e., $\mathbf{h}_r = \mathbf{h}\mathbf{M}_r$ and $\mathbf{t}_r = \mathbf{t}\mathbf{M}_r$. Its scoring function is $f_r(h,t) = \|\mathbf{h}_r + \mathbf{r} - \mathbf{t}\|_{l1/l2}$.

TransD [8] considers the diversity of entities and relations simultaneously. It uses the product of two vectors of an

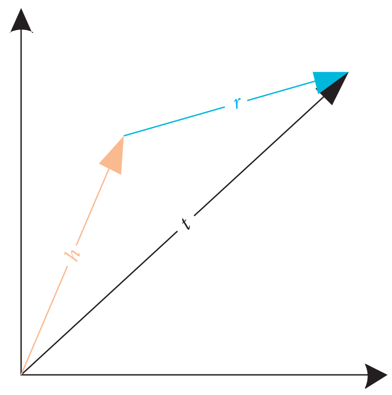


Figure 1: A relational fact translating example of TransE.

entity-relation pair to replace a projection matrix, i.e., $\mathbf{M}_{rh} = \mathbf{r}_p\mathbf{h}_p$ and $\mathbf{M}_{rt} = \mathbf{r}_p\mathbf{t}_p$. TransD is more extensible and can be applied to the large-scale knowledge graphs. Its scoring function is $f_r(h,t) = \|\mathbf{h}\mathbf{M}_{rh} + \mathbf{r} - \mathbf{t}\mathbf{M}_{rt}\|_{l1/l2}$.

TranSparse [9] considers the heterogeneity and imbalance of entities and relations in a knowledge graph, which are generally ignored by previous works. TranSparse constructs adaptive sparse matrices $\mathbf{M}_r^h(\theta_r^h)$ and $\mathbf{M}_r^h(\theta_r^h)$, instead of projection matrices, to concurrently prevent the overfitting of simple relational facts and the underfitting of complex relational facts. Its scoring function is $f_r(h,t) = \|\mathbf{M}_r^h(\theta_r^h)\mathbf{h} + \mathbf{r} - \mathbf{M}_r^t(\theta_r^t)\mathbf{t}_r\|_{l1/l2}$.

FT [18] and DT [19] design the flexible translation principles and the dynamic translation principles, respectively. To some extent, they improve the ability to handle complex relational facts. Essentially, they focus on elaborating various translation principles to learn complex relational facts more accurately. TransAt [17] and GAN-based framework [20] use attention mechanisms and generate adversarial networks to improve the model performance, respectively. However, all of the above methods naturally embed the complete knowledge graph into a uniform vector space from a single perspective, failing to solve the space congestion problem thoroughly.

PuTransE [21] is an online and robust improvement of TransE solving the hyperparameters sensitivity problem and the spatial congestion of entity and relation, as well as the processing of dynamic knowledge graphs. It adopts multiple parallel spaces to learn the vectors of entities and relations, thus avoiding spatial congestion in complex relational facts. Therefore, puTransE achieves state-of-the-art performance on the link prediction task. However, puTransE still has two weaknesses resulting in the performance limitation. First, puTransE causes the excessive sparseness of simple relational facts during the random parallel space generation. It performs knowledge extraction including not only complex relational facts but also sparse simple relational facts. Thus, it probably cannot learn complete vector representations of simple relational facts via a single subgraph. Second, puTransE randomly selects local knowledge to construct multiple parallel spaces, which is prone to impair original facts of knowledge graphs. For example, there is a golden triplet, i.e., a positive sample, in the original graph which cannot be consisted by entities and relation in any parallel spaces. This situation decreases the relational fact prediction accuracy of puTransE.

3. Our Method

In this section, we introduce the details of MvTransE, which embeds relational facts from global-view and local-view, respectively. The workflow of MvTransE mainly consists of three steps. The first step (Section 3.1) is to generate multiple parallel subgraphs so as to extract particular local relational facts of the knowledge graph accurately. The second step (Section 3.2) aims at discriminatively embedding the knowledge graph into multiple parallel spaces to acquire multiview representations of relational facts. The last step (Section 3.3) integrates multiple versions of knowledge representations, i.e., fuses local-view and global-view representations of entities and relations. Figure 2 presents a multiview knowledge learning process of MvTransE.

3.1. Subgraph Generation. The subgraph generation aims to extract local relational facts from different perspectives, so as to solve spatial congestion of entities by sparsely embedding entities and relations into different parallel vector spaces in the following graph embedding step. Therefore, we construct multiple parallel subgraphs based on different relations of the knowledge graph. Each subgraph mainly contains the local relational facts selected from a specific relation.

Initially, we give the definition of some related symbols in the subgraph generation process. We define a knowledge graph as $G = (E, R, T)$, where E and R denote an entity set and a relation set of graph G, respectively; $T \subseteq E \times R \times E$ represents a triplet set of G. And $\mathbb{G}_{sub}$ is the final generated subgraph set, $\mathcal{G}_i$ is a subgraph that $\mathcal{G}_i \in \mathbb{G}_{\text{sub}}$, and $\mathcal{E}_i$ and $\mathcal{R}_i$ represent an entity set and a relation set of $\mathcal{G}_i$, respectively. Algorithm 1 demonstrates the details of the subgraph generation, which mainly consists of two steps given in the following.

3.1.1. Semantics-Related Entity Selection. To accurately learn local relational facts, we first select relation relevant entities for a subgraph to ensure the semantic consistency of its knowledge as much as possible. That is, entities in a subgraph should be semantically related to each other based on a specific relation. We randomly select a relation r from relation set R then to generate entity set E_r, which consists of extracted entities interconnecting with r. Since E_r is generated via relation r; therefore, r is deemed as the semantic center of the current subgraph $\mathcal{G}_i$.

3.1.2. Structure-Related Subgraph Expansion. In order to learn latent knowledge associating with r more comprehensively, we need to expand each subgraph according to the local graph structure of entities in the set E_r. This step ensures a generated subgraph containing semantic and structural features of local relational facts simultaneously. Specifically, we first randomly select an entity e_i from E_r as a

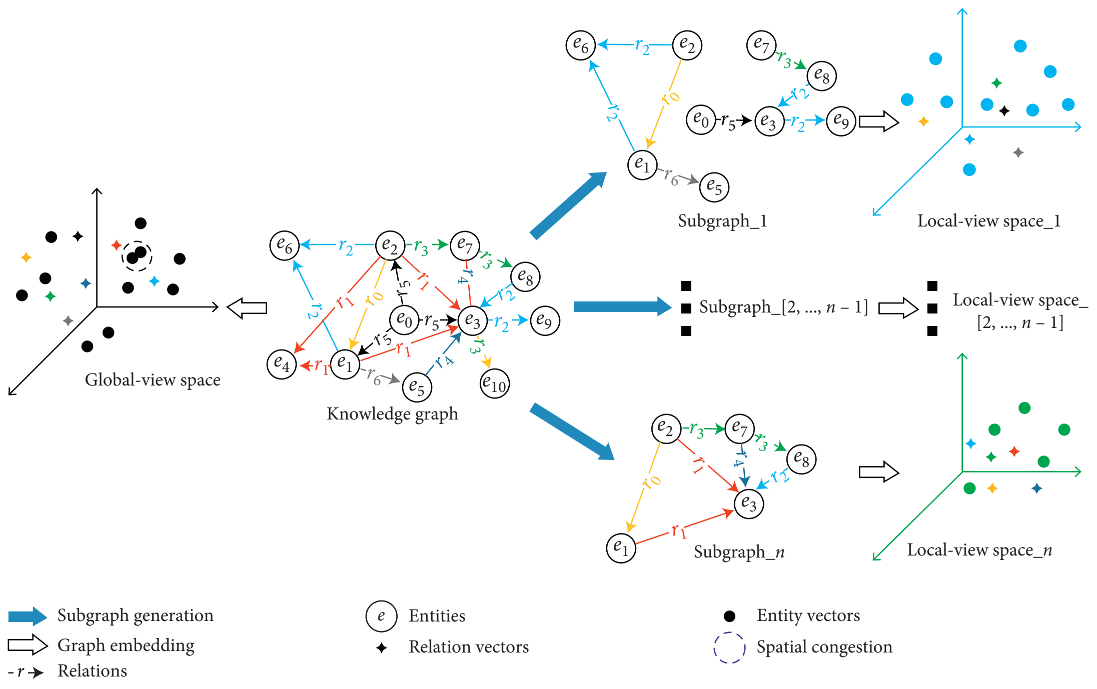


Figure 2: A multiview knowledge learning process of the MvTransE model. The blue arrow indicates a subgraph generation operation for extracting local relational facts from the knowledge graph; and the white arrow indicates a graph embedding process projecting the original knowledge graph and generated subgraphs into a global-view vector space and several local-view vector spaces, respectively. In each vector space, the colored coordinates represent different vector spaces. The solid circles represent entity vectors in a vector space whose color is the same as the color of the space coordinate. The four-angle stars represent relation vectors, and its color corresponds to the color of relations in the knowledge graph.

starting entity to expand the subgraph $\mathcal{G}_i$. And then, we randomly select a triplet whose head or tail entity is the starting entity e_i and add the head or tail entity to the subgraph $\mathcal{G}_i$. Consequently, we can get the local structure information of E_r regarding G by repeating the above two operations for n_s time. Due to the randomly selected entities and triplets in the subgraph expansion, each subgraph expanded from E_r may include different relational facts, which makes generated subgraphs slightly different from each other in terms of semantics and structure. Thereafter, with respect to learning local relational facts discriminately, these random operations ensure MvTransE can learn local knowledge representations from multiple perspectives.

Besides, to make each subgraph focusing on particular relational facts, we need to control the scale of subgraphs for avoiding extracting excessive irrelevant facts. We set hyperparameter n_t to control the number of triplets in subgraphs, set hyperparameter n_s to control the expansion speed of each subgraph, set hyperparameter τ to control the maximum iterations of the triplet selection, and set hyperparameter n to control the number of generated subgraphs.

3.2. Graph Embedding. The goal of this step is to obtain global-view representation and local-view representation of entities and relations. Hence, we perform original knowledge graph G embedding and subgraphs $\mathcal{G}$ embedding to learn global knowledge and local knowledge, respectively. In each vector space, we define the following equation as the scoring function $f_r\ (h, t)$ to translate each triplet (h, r, t):

$$f_r(h,t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{l1/l2}, \tag{1}$$

where **h**, **r**, and **t** are vector representations of h, r, and t, and l_1/l_2 is the l_1-norm or l_2-norm distance.

In MvTransE, we use the margin-based loss function as the optimization target in each vector space, which is defined as follows:

$$L = \sum_{\psi_i \in \Psi} \sum_{(h,r,t)\in T} \sum_{(h',r,t')\in T'} \left[f_r(h,t) + \gamma - f_r(h',t')\right]_+, \tag{2}$$

where Ψ is a set of embedded vector spaces, T is a set of positive triplets in a graph, and T' is a set of negative triplets generated by randomly replacing the head (or tail) of each positive triplet $(h, r, t) \in T$, and γ is a fixed margin distance for distinguishing positive and negative triplets. We use the stochastic gradient descent (SGD) [22] to minimize the loss function.

Algorithm 2 presents the multiview graph embedding process, which aims at respectively embedding each knowledge graph and subgraph into single vector spaces. Thus, we will get $n+1$ vector spaces, including one global-view vector space and n local-view vector spaces. The global-view vector space obtains the global-view representation of all entities and relations regarding the original knowledge graph; the local-view vector spaces differentially learn local-view representation of entities and relations regarding complex relational facts from different semantic and structural perspectives.

```
Input: Knowledge Graph $G = (E, R, T)$, hyperparameters of subgraph $n, \tau$
Output: A subgraphs set $\mathbb{G}_{\text{sub}}$.
(1)  $\mathbb{G}_{\text{sub}} \longleftarrow$ Initialize empty set for subgraph
(2)  while $n > 0$ do
(3)    $\tau \longleftarrow 128$ //Set the maximum iterations
(4)    $r \longleftarrow$ Random sample a relation $r \in R$
(5)    $E_r \longleftarrow$ Select semantics related entities
(6)    $n_s, n_t \longleftarrow$ Generate subgraph scale hyperparameters
(7)    $\mathcal{G} \longleftarrow \varnothing$//Initialize empty set for selected triplets
(8)    while $|\mathcal{G}| < n_t$ and $\tau > 0$ do
(9)      count $\longleftarrow 0$
(10)       while count $< n_s$ do//Control the scale of subgraphs
(11)       $e \longleftarrow$ Randomly sample an entity $e \in E_r$
(12)       $t \longleftarrow$ Randomly select a relevant triplet $t(e)$ from $G$
(13)       $\mathcal{G} \longleftarrow \mathcal{G} \cup t$ //Add a selected triplet
(14)       count $\longleftarrow$ count $+ 1$
(15)       end while
(16)     $E_r \longleftarrow$ All entities collected in $\mathcal{G}$
(17)     $\tau \longleftarrow \tau - 1$
(18)     end while
(19)   $\mathbb{G}_{\text{sub}} \longleftarrow \mathbb{G}_{\text{sub}} \cup \mathcal{G}$ //Add a subgraph
(20)   $n \longleftarrow n - 1$
(21)   end while
```

Algorithm 1: Generating subgraph.

```
Input: A set of Knowledge Graph and Subgraphs $\mathbb{G} = \{G, \mathcal{G}_1, \ldots, \mathcal{G}_n\}$, vector dimension $k$, global-view margin $\gamma_0$, global-view
learning rate $\alpha_0$.
Output: A set of generated vector spaces $\psi$.
(1)  $\psi \longleftarrow$ Initialize empty set for vector spaces
(2)  for $\mathcal{G} \in \mathbb{G}$ do
(3)    if $\mathcal{G} = G$ then
(4)      $\gamma, \alpha \longleftarrow$ global-view margin $\gamma_0$, global-view learning rate $\alpha_0$
(5)    else
(6)      $\gamma, \alpha \longleftarrow$ Randomly initialize local-view margin and learning rate
(7)    end if
(8)    $\mathcal{E}, \mathcal{R} \longleftarrow$ All entities and relations in $\mathcal{G}$ respectively
(9)    $\mathbf{e}, \mathbf{r} \longleftarrow$ Initialize uniform $(-6\sqrt{k}, 6\sqrt{k})$ for each $e \in \mathcal{E}$, $r \in \mathcal{R}$
(10)   loop
(11)       $S_{\text{batch}} \longleftarrow$ sample $(S, b)$//Sample a minibatch of size $b$
(12)       $T_{\text{batch}} \longleftarrow \varnothing$ //Initialize the set of pairs of triplets
(13)       for $(h, r, t) \in S_{\text{batch}}$ do
(14)       $(h, r, t) \longleftarrow$ sample $(S_{(h',r,t')})$//Sample corrupted triplet
(15)       $T_{\text{batch}} \longleftarrow T_{\text{batch}} \cup \{(h, r, t), (h', r, t')\}$
(16)       end for
(17)       Update vectors w.r.t
             $\sum_{((h,r,t),(h',r,t')) \in T_{\text{batch}}} \nabla [f_r(h,t) + \gamma - f_r(h',t')]_+$
(18)     end loop
(19)   $\Delta \longleftarrow (\mathcal{G}, \mathbf{e}, \mathbf{r})$//Trained parameters are saved as one vector space
(20)   $\psi \longleftarrow \psi \cup \Delta$ //Add vector space to sets
(21)  end for
```

Algorithm 2: Learning multiview graph embedding.

3.3. Multiview Fusion Strategy. In this section, we propose a multiview fusion strategy that adopts an adaptive selection principle to integrate knowledge representations of global-view vector space and local-view vector space. For each testing triplet (h, r, t), we define a scoring estimation function $S_r(h, t)$ to calculate the distance score of a vector space, and then dynamically select the final representation of the triplet according to the minimum score. The scoring estimation function is defined as follows:

$$S_r(h,t) = \min_{\Delta\in\psi} \|\mathbf{h}\Delta + \mathbf{r}\Delta + \mathbf{t}\Delta\|_{l1/l2}, \tag{3}$$

where Δ is a vector space in ψ which contains h, r, and t; $\mathbf{h}\Delta$, $\mathbf{r}\Delta$ and $\mathbf{t}\Delta$ are vectors of h, r and in a vector space Δ.

Since each parallel vector space generally contains local relational facts related to a particular relation, our model can subtly solve the spatial congestion problem. Additionally, MvTransE constructs a global-view vector space containing complete knowledge representations in a knowledge graph, which makes any testing triplet able to find a knowledge representation at least. Thus, our model significantly improves the performance of learning simple relational knowledge.

4. Experiments

In this section, we study the performance of our model in link prediction and triplet classification tasks under four public datasets, i.e., WN18, WN18RR, WN11, and FB15K-237.

4.1. Datasets. WordNet [2] is a large knowledge graph of English vocabulary which is widely used in graph embedding works. In WordNet, a set of synonyms representing a basic vocabulary concept is taken as an entity, and various semantic relations are established between these synonym sets. In the following experiments, we use three public subsets of WordNet, i.e., WN18, WN18RR, and WN11. WN18 contains 18 relations and 40943 entities. WN18RR is a modified version of WN18 introduced by Dettmers et al. [23], which removes the reversing relational facts avoiding information leakage problem in representation tasks. WN11 consists of 11 relations and 38696 entities. Freebase is a large collaborative knowledge graph storing the general facts of the real world. We use a subset of Freebase, i.e., FB15k-237 [21], which consists of 237 relations and 14541 entities in total. Table 1 presents the statistics of the above datasets.

4.2. Link Prediction. Link prediction aims to predict the missing head entity h or tail entity t of a test triplet (h, r, t). In this experiment, we take the entity h (or t) missed in test triplets as the correct entity, and all other entities are considered as candidate entities. Firstly, we construct candidate triplets by replacing h (or t) of the test triplet. Then, the link prediction score of each triplet is calculated by the scoring function of our model. Finally, candidate entities and the correct entity are sorted in ascending order based on their prediction scores. We adopt two metrics used in [5] to evaluate our model: the average rank of each correct entity (i.e., Mean Rank) and the average number of correct entities ranked in the top 10 (i.e., Hits@10). Obviously, a good prediction performance should achieve a high Hits@10 and a low Mean Rank.

Table 1: Dataset information.

Dataset	#Ent	#Rel	#Train	#Valid	#Test
WN18	40943	18	141442	5000	5000
WN11	38696	11	112581	2609	10544
WN18RR	40943	11	86835	3034	3134
FB15K-237	14541	237	272115	17535	20466

Note that the candidate triplets may already exist in the knowledge graph, so these candidate triplets should be considered as the correct triplets. The scores of these candidate triplets are likely to be lower than the correct triplets. Therefore, we should filter out these candidate triplets that have already appeared in the train, validation, and test sets. We denote an evaluation setting by "Filt" if we filter out these candidate triplets before the test, otherwise denote it by "Raw."

We compare MvTransE with a few state-of-the-art methods in the link prediction task on WN18, WN18RR, and FB15K-237 datasets. On WN18, MvTransE is compared to RESCAL [24], SE, SME, LFM, TransE, TransH, TransR/CTransR, puTransE, and TransAt in Table 2. In Table 3, we compare MvTransE with three competitive methods DistMult [25], ComplEx [26], and ConvE [23] on WN18RR and FB15K-237 datasets both of which do not have information leakage problem found on WN18 dataset. We directly use the results reported in their published papers or in [7] due to the same experimental settings. For MvTransE, our experimental settings are as follows.

4.2.1. Subgraph Generation Setup. On WN18, we set the number of subgraphs $n = 5000$, the expanding speed of subgraphs $n_s \in [50, 450]$, and the size of subgraphs $n_t \in [200, 1500]$. On WN18RR, we set the number of subgraphs $n = 5000$, the expanding speed of subgraphs $n_s \in [50, 300]$, and the size of subgraphs $n_t \in [200, 1000]$. On FB15K-237, we set the number of subgraphs $n = 5000$, the expanding speed of subgraphs $n_s \in [100, 350]$, and the size of subgraphs $n_t \in [1000, 2200]$.

4.2.2. Graph Embedding Setup. The dimension of vectors k is set among {25, 30, 35, 40, 50}, the training phase *epoch* among {200, 500, 1000}. In global-view vector space, we set the learning rate α_0 among {0.01, 0.005, 0.001, 0.0008}, the margin value γ_0 among {2, 3, 3.5, 4, 4.5, 5, 5.5}, the minibatch size B_0 among {100, 200, 500}. For each dataset, we choose the best parameter configuration through validation sets and they are as follows: $k = 35$, epoch $= 1000$, $\alpha_0 = 0.005$, $\gamma_0 = 3.5$, $B_0 = 100$ on WN18; $k = 35$, epoch $= 1000$, $\alpha_0 = 0.0008$, $\gamma_0 = 5.5$, $B_0 = 100$ on WN18RR; $k = 35$, epoch $= 1000$, $\alpha_0 = 0.001$, $\gamma_0 = 3$, $B_0 = 100$ on FB15K-237. In local-view spaces, we randomly initialize the learning rate α_i and the margin γ_i within a certain range, and obtain the best parameter settings as follows:$\alpha_i \in [0.001, 0.005]$, $\gamma_i \in [1.5, 5.5]$ on WN18; $\alpha_i \in [0.0008, 0.005]$, $\gamma_i \in [3.5, 5.5]$ on WN18RR; $\alpha_i \in [0.005, 0.015]$, $\gamma_i \in [3.5, 5.5]$ on FB15K-237.

Table 2 presents the corresponding experimental results of three model settings: (1) MvTransE (Global-view) denotes

Table 2: Experimental results of link prediction on WN18.

Dataset	WN18			
Metric	Mean rank		Hits@10 (%)	
	Raw	Filt	Raw	Filt
RESCAL	1180	1163	37.2	52.8
SE	1011	985	68.5	80.5
SME (linear/bilinear)	545/526	533/509	65.1/54.7	74.1/61.3
LFM	469	456	71.4	81.6
TransE	263	251	75.4	89.2
TransH (unif/bern)	318/401	303/388	75.4/73.0	86.7/82.3
TransR (unif/bern)	232/238	219/225	78.3/79.8	91.7/92.0
CTransR (unif/bern)	243/231	230/218	78.9/79.4	92.3/92.3
puTransE	39	29	88.1	94.9
TransAt	169	157	81.4	95.1
MvTransE (global-view)	198	186	79.7	92.2
MvTransE (local-view)	46	42	84.1	93.9
MvTransE (Multiview)	**29**	**24**	**88.3**	**95.1**

Table 3: Results of link prediction on WN18RR and FB15K-237 under the Filt setting.

Dataset	WN18RR		FB15K-237	
Metric	Mean Rank(Filt)	Hits@10 (filt)	Mean Rank(Filt)	Hits@10 (filt)
DisMult	5110	0.49	254	0.41
ComplEx	5261	0.51	339	0.42
ConvE	5277	0.46	246	**0.49**
MvTransE	**1323**	**0.52**	**139**	**0.49**

the prediction results by using the global-view representation. (2) MvTransE (Local-view) denotes the prediction results by using all parallel local-view representation. (3) MvTransE (Multiview) denotes the prediction results of the integrated multiview representation derived from a multiview fusion strategy. It can be seen from Table 2, the multiview representation yields better results than global-view and local-view representations. The results prove that our idea of representing knowledge from multiple perspectives is effective. In detail, our method substantially outperforms state-of-the-art methods in the Mean Rank metric. In Hits@10, our method is also superior to all baseline methods and achieves the best performance under "Raw," and achieves the same best performance as TransAt under "Filt" settings.

Table 3 presents the results of MvTransE (Multiview) on WN18RR and FB15K-237 to further illustrate the merits of our method. Obviously, MvTransE (Multiview) markedly outperforms all methods on WN18RR, achieving state-of-the-art performance in two metrics. On FB15K-237, MvTransE (Multiview) has achieved the best performance over all of the methods on the Mean Rank metric, performing the same as ConvE on Hits@10 metric. Particularly, our model achieves excellent performance on all three different datasets in the Mean Rank metric, which evaluates the overall quality of the learned knowledge representations. This is because that MvTransE aims at learning knowledge representations from multiple perspectives and dynamically fusing these representations as an optimal combination.

To further explain the above observation, we present the prediction results of our method regarding all types of relational facts of WN18. Table 4 lists the type distribution of triplets in WN18 based on four relation categories. Table 5 presents the experimental results of three model settings on each relation category. Specifically, the global-view setting outperforms the local-view setting in predicting 1-to-1, 1-to-N head, and N-to-1 tail on both metrics, which actually fall into the simple relational facts learning category. On the contrary, the local-view setting exhibits superior performance in predicting complex relational facts, i.e., including N-to-N, N-to-1 head, and 1-to-N tail facts learning category. Clearly, by combining the advantages of global-view and local-view settings, the multiview method achieves the best performance in learning simple and complex relational facts.

Table 4: Type distribution of triplet.

Relation category	1-to-1	1-to-N	N-to-1	N-to-N
#Triplet (train)	1281	54655	54555	31014
#Triplet (valid)	44	1938	1897	1119
#Triplet (test)	42	1847	1981	1130

4.3. Triplet Classification. Triple classification task aims to determine whether a given triple (h, r, t) is correct or not. In this experiment, we adopt WN11 to verify the effectiveness of our method. Following the experiment setting of previous work [5], we set a classifying threshold δ_r for each relation r. To maximize the classification accuracy, we optimize δ_r on the validation set. Giving a test triple (h, r, t), if its score is lower than δ_r, it will be classified as a positive sample, otherwise a negative sample.

Table 5: Experimental results of WN18 by mapping properties of relations.

Tasks ($k = 35$)		Predicting head				Predicting tail			
		1-to-1	1-to-N	N-to-1	N-to-N	1-to-1	1-to-N	N-to-1	N-to-N
Global-view	Hits@10 (%)	90.5/90.5	91.9/92.7	56.0/87.6	85.7/89.7	92.9/93.2	54.5/86.9	94.1/94.2	84.6/88.6
(Raw/Filt)	Mean rank	12/12	158/158	228/198	217/216	35/35	258/228	132/132	227/225
Local-view	Hits@10 (%)	71.4/71.4	91.8/92.0	70.3/86.8	94.6/95.0	71.4/71.4	78.4/91.7	84.4/84.4	94.3/94.8
(Raw/Filt)	Mean rank	97/97	23/23	72/61	**22/21**	177/177	44/33	67/67	23/23
Multiview	Hits@10 (%)	**90.5/90.5**	**94.2/94.3**	**76.1/94.2**	**95.0/95.4**	**92.9/93.9**	**79.8/93.9**	**94.1/94.2**	**94.8/95.2**
(Raw/Filt)	Mean rank	**8/8**	**12/11**	**47/34**	39/38	**20/19**	**44/31**	**14/14**	**23/22**

The bold values stand for the best performance in the corresponding metric.

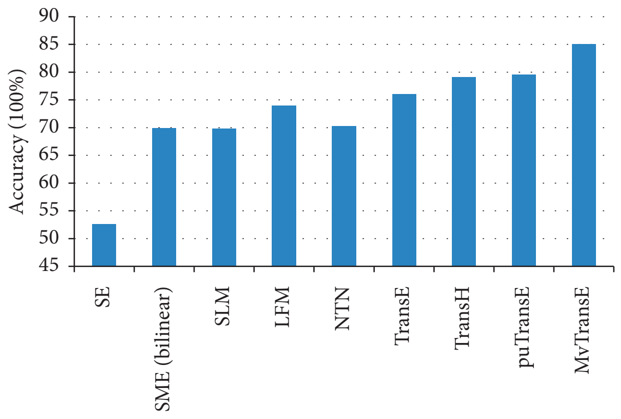


Figure 3: The experimental results of triplet classification.

We choose SE, SME, Single Layer Model (SLM) [17], LFM, NTN, TransE, TransH, and puTransE as baselines. We use the results reported in [7] directly since the data set is the same. For MvTransE, our experimental settings are as follows.

4.3.1. Subgraph Generation Setup. We set the number of subgraphs $n = 5000$, the length of random walk $n_s \in [50, 300]$, and the size of subgraphs $n_t \in [200, 1000]$.

4.3.2. Graph Embedding Setup. The dimension of vectors k is set among {15, 20, 25}, the training phase *epoch* among {200, 500, 1000}. In global-view vector space, we set the learning rate α_0 among {0.02, 0.01, 0.05}, the margin value γ_0 among {3, 4, 4.5, 5}, and the minibatch size B_0 among {100, 200, 500}. We choose the best configuration by the validation set, which is as follows: $k = 20$, epoch = 1000, $\alpha_0 = 0.01$, $\gamma_0 = 4.5$, $B_0 = 100$. In local-view vector spaces, we randomly initialize the learning rate $\alpha_i \in [0.005, 0.015]$, the margin value $\gamma_i \in [3.5, 5.5]$.

The experimental results of the triplet classification are shown in Figure 3. Clearly, MvTransE obtains the best performance among all of the baseline methods. Compared with the translation-based methods, i.e., TransE and TransH, our method is able to learn complex relational facts more subtly by constructing subgraphs from the original knowledge graph. On the other side, our method outperforms the recent competitor puTransE due to our knowledge fusion strategy which leverages an adaptive selection principle to integrate the global-view and the local-view knowledge representations reasonably. Therefore, MvTransE is more suitable for embedding large and complex knowledge graphs. MvTransE has great advantages in knowledge graph completion tasks due to its multiview knowledge learning and fusing methods.

5. Conclusion and Future Work

In this work, we propose a multiview translation learning model, named MvTransE, which aims at presenting graph relational facts from global-view and local-view, respectively. MvTransE achieves state-of-the-art performance by solving the entity spatial congestion problem and the relational fact impairment problem. Extensive experiments demonstrate that MvTransE outperforms state-of-the-art models on the link prediction task and triplet classification task. In the future, we will focus on the subgraph construction scheme to learn the local relational facts more efficiently.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Nos. U1811264, 61966009, and U1711263), the Natural Science Foundation of Guangxi Province (Nos. 2020GXNSFAA159055, 2019GXNSFBA245059, and 2019GXNSFBA245049), the Guangxi Innovation-Driven Development Project (No. AA17202024), and the Beihai Science and Technology Project Contract (No. 202082001).

References

[1] R. Ronald Bakker, “Knowledge graphs: representation and structuring of scientific knowledge,” 1987.
[2] G. A. Miller, “WordNet: a lexical database for English,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[3] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: a collaboratively created graph database for

structuring human knowledge," in *Proceedings of the Special Interest Group on Management of Data*, pp. 1247–1250, Vancouver, Canada, June 2008.

[4] X. Dong, E. Gabrilovich, G. Heitz et al., "Knowledge vault: a web-scale approach to probabilistic knowledge fusion," in *Proceedings of the Special Interest Group on Knowledge Discover and Data Mining*, pp. 601–610, New York, NY, USA, August 2014.

[5] B. Antoine, N. Usunier, A. Garc´ıa-Dur´an, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proceedings of Neural Information Processing Systems*, pp. 2787–2795, Lake Tahoe, NV, USA, December 2013.

[6] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of Association for the Advances of Artificial Intelligence*, pp. 1112–1119, Québec City, Canada, July 2014.

[7] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of Association for the Advances of Artificial Intelligence*, pp. 2181–2187, Austion, TX, USA, January 2015.

[8] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of the Association for Computer Linguistics*, pp. 687–696, Beijing, China, July 2015.

[9] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix," in *Proceedings of the Association for the Advances of Artificial Intelligence*, pp. 985–991, Phoenix, AZ, USA, February 2016.

[10] Yi Tay, A. T. Luu, and S. C. Hui, "Non-parametric estimation of multiple embedding for link prediction on dynamic knowledge graphs," in *Proceedings of the Association for the Advances of Artificial Intelligence*, pp. 1243–1249, San Francisco, CA, USA, February 2017.

[11] X. Tian Xia, D. Dacheng Tao, T. Tao Mei, and Y. Yongdong Zhang, "Multiview spectral embedding," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 6, pp. 1438–1446, 2010.

[12] D. Zhai, H. Chang, S. Shan, X. Chen, and W. Gao, "Multiview metric learning with global consistency and local smoothness," *ACM Transactions on Intelligent Systems and Technology*, vol. 3, no. 3, pp. 1–22, 2012.

[13] B. Antoine, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *Proceedings of the Association for the Advances of Artificial Intelligence*, pp. 301–306, San Francisco, CA, USA, February 2011.

[14] B. Antoine, X. Glorot, J. Weston, and Y. Bengio, "Joint learning of words and meaning representations for open-text semantic parsing," in *Proceedings of the Artificial Intelligence and Statistics*, pp. 127–135, Canary Islands, Spain, April 2012.

[15] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," *Machine Learning*, vol. 94, no. 2, pp. 233–259, 2014.

[16] R. Jenatton, N. Le Roux, B. Antoine, and G. Obozinski, "A latent factor model for highly multi-relational data," in *Proceedings of the Neural Information Processing Sysytems*, pp. 3176–3184, Lake Tahoe, NV, USA, December 2012.

[17] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, "Reasoning with neural tensor networks for knowledge base completion," 2013.

[18] J. Feng, M. Huang, M. Wang, M. Zhou, H. Yu, and X. Zhu, "Knowledge graph embedding by flexible translation," in *Proceedings of the Knowledge Representation and Reasoning*, pp. 557–560, Cape Town, South Africa, April 2016.

[19] L. Chang, M. Zhu, T. Gu, C. Bin, J. Qian, and J. Zhang, "Knowledge graph embedding by dynamic translation," *IEEE Access*, vol. 5, pp. 20898–20907, 2017.

[20] P.F. Wang, S. Li, and R. Pan, "Incorporating gan for negative sampling in knowledge representation learning," in *Proceedings of Association for the Advances of Artificial Intelligence*, pp. 2005–2012, New Orleans, LO, USA, February 2018.

[21] K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference," in *Workshop on Continuous Vector Space and their Compositionality*, pp. 57–66, Beijing, China, July 2015.

[22] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[23] Tim Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2D knowledge graph embeddings," in *Proceedings of Association for the Advances of Artificial Intelligence*, pp. 1811–1818, New Orleans, LO, USA, February 2018.

[24] M. Nickel, V. Tresp, and H. P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of International Conference on Machine Learning*, pp. 809–816, Bellevue, WA, USA, February 2011.

[25] B. Yang, W.-T. Yih, X. He, J. Gao, and Li Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proceedings of International Conference on Learning Representations*, pp. 1–13, San Diego, CA, USA, December 2015.

[26] Th´eo Trouillon, J. Welbl, S. Riedel, E. ´ric Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of International Conference on Machine Learning*, pp. 2071–2080, New York, NY, USA, June 2016.

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:

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

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:

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

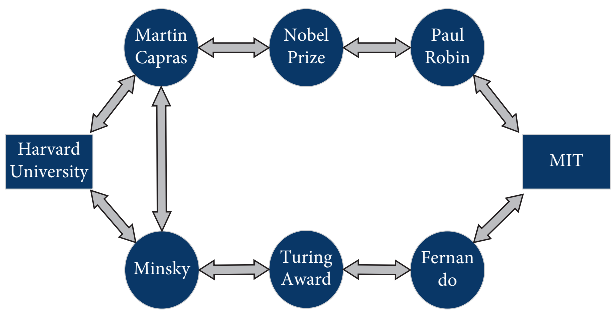


Figure 1: KSP query example.

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.

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, δ_{uv}), u is the any entity node and δ_{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 δ_{st}; $\text{QUERY}(s,t,L)$ is

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

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 entity ID. Add node distance from v_k to departure when the breadth-first-search is performed from node 1 at the kth 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 δ. If $\text{QUERY}(v_k, u, D_{k-1}) \le \delta$, then cut u and no longer put (v_k, δ) 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.

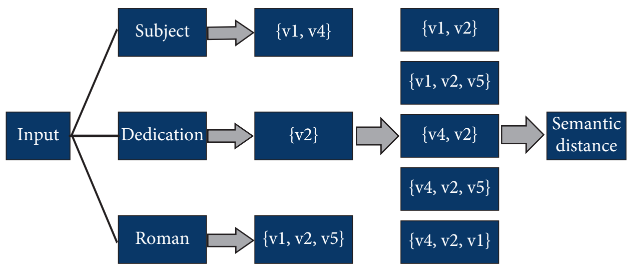


Figure 2: Problem model.

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

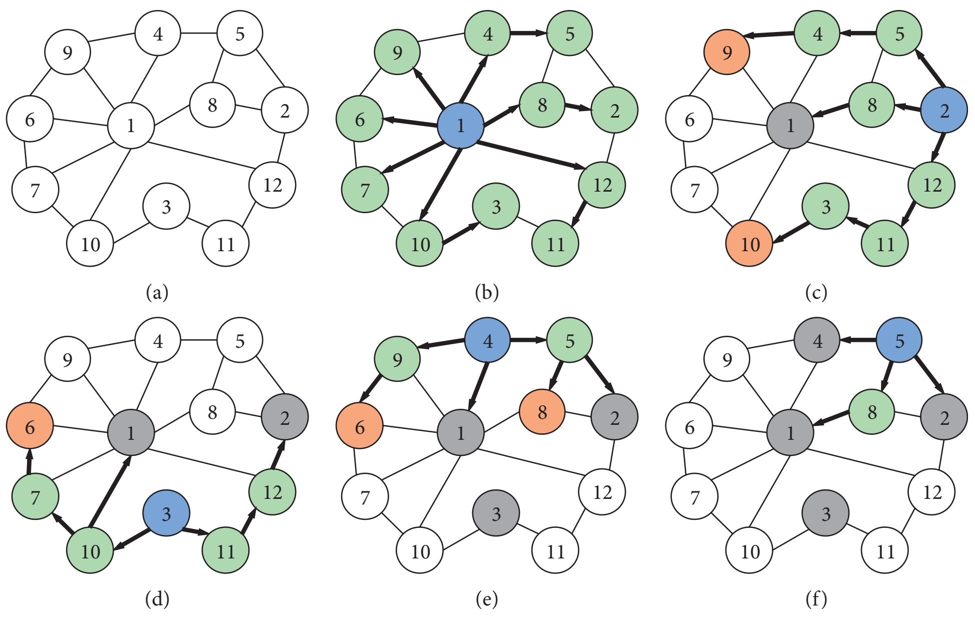


Figure 3: Semantic distance index construction.

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

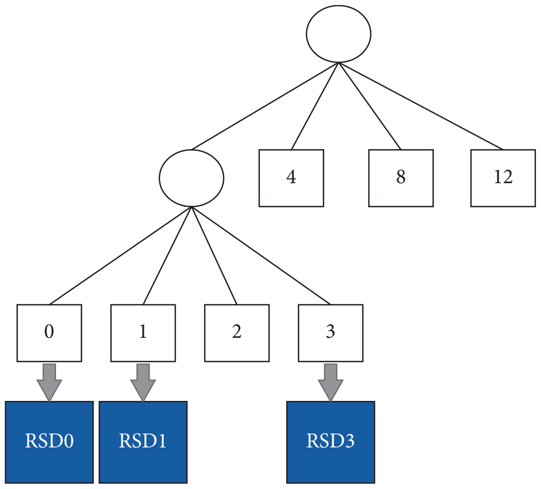


Figure 4: Spatial semantic hybrid index structure.

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
		Minsky	1
		Nobel Prize	2
	L (MIT)	Paul Robin	1
		Fernando	1
		Turing Award	2

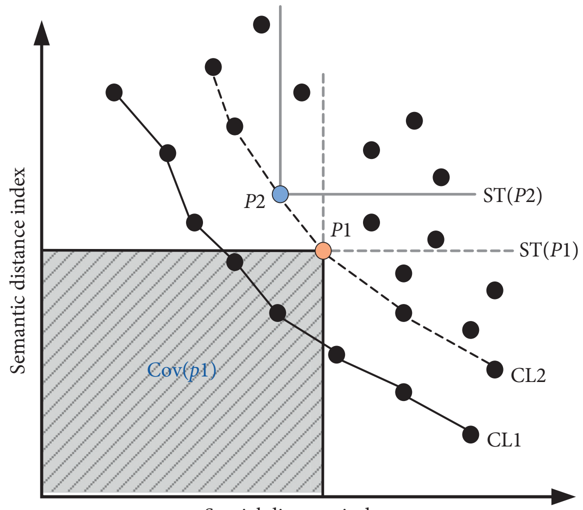


Figure 5: Skyline coordinate.

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 θ is the cost of the current kth 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 θ 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 $H_{top-k}$
 (1) Heap $H_{top-k} = \varnothing$,
 (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)             $H_{top-k}$.add$(T_p, f)$
(13)             Update $\theta$
(14) return $H_{top-k}$
```

Algorithm 1: Online query algorithm.

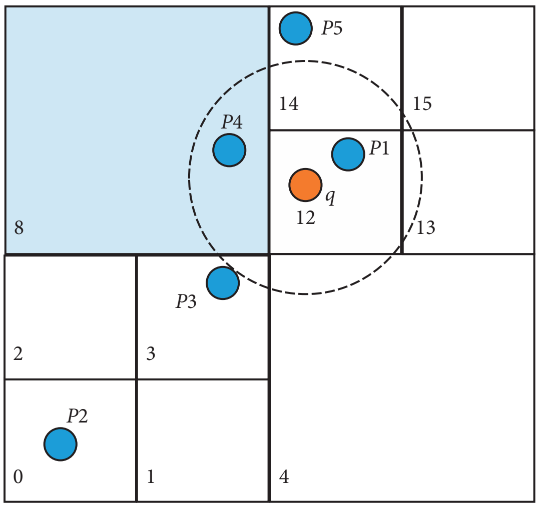


Figure 6: Pruning query example.

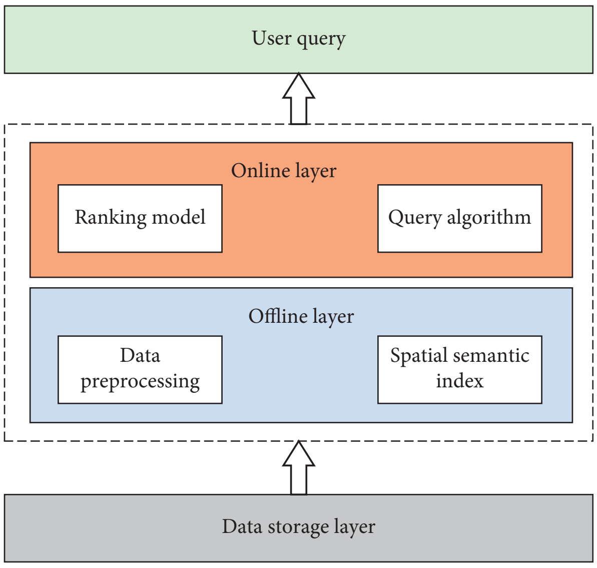


Figure 7: System module diagram.

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.

(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

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

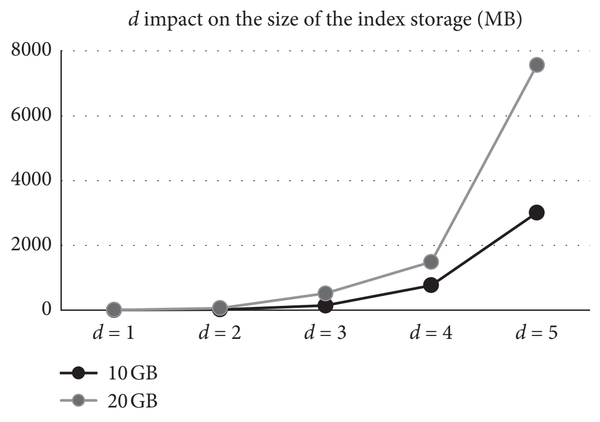


Figure 8: Experiment of determining the parameter *d*.

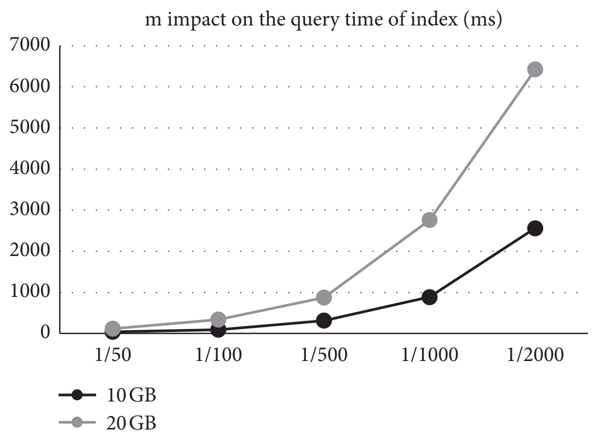


Figure 9: Experiment of determining the parameter *m*.

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 β of ranking model: β represents how sensitive the ranking model is to variables. In the experiment, β 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 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.

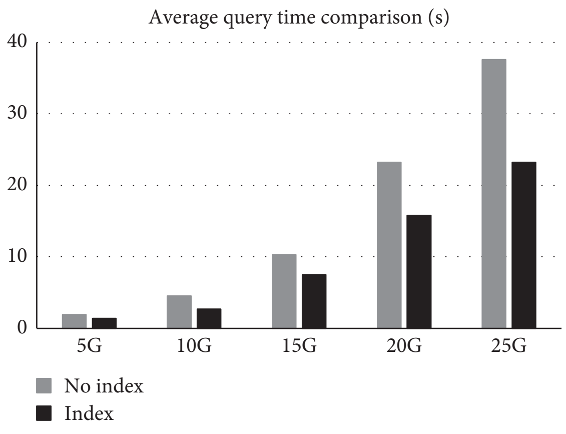


Figure 10: YAGO experimental results.

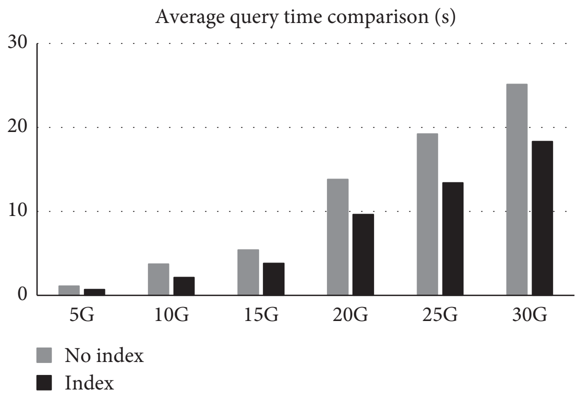


Figure 11: DBpedia experimental results.

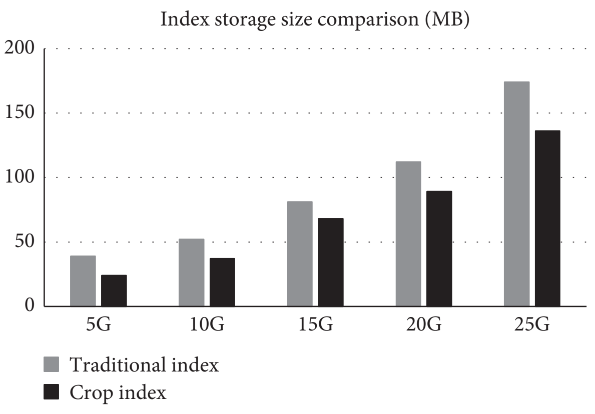


Figure 12: Index construction optimization experiment.

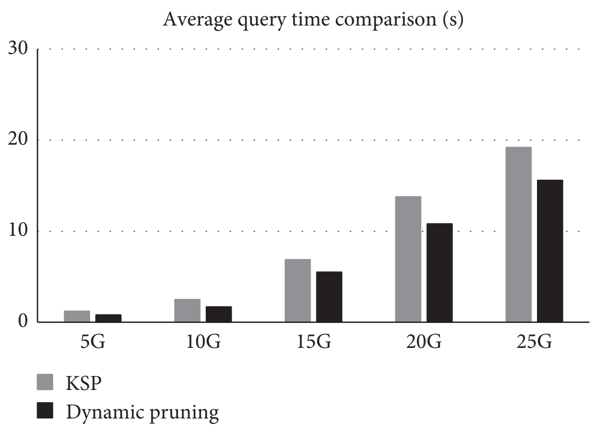


Figure 13: YAGO experimental results.

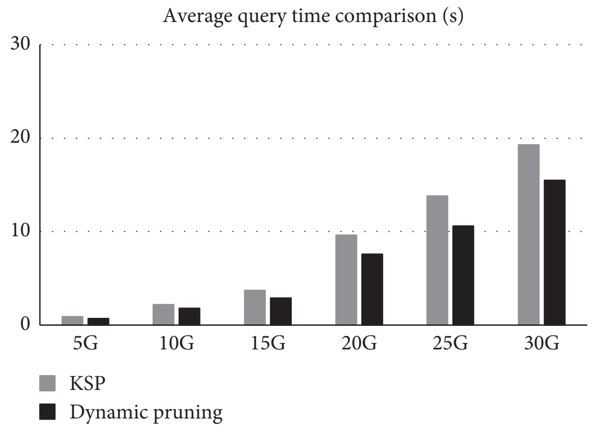


Figure 14: DBpedia experimental results.

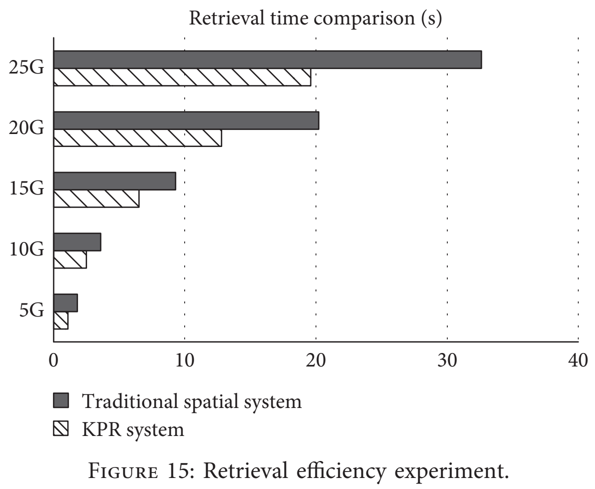


Figure 15: Retrieval efficiency experiment.

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.

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.

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

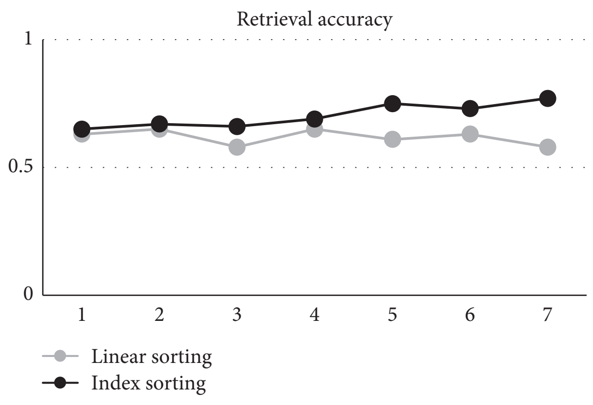


Figure 16: Retrieval accuracy experiment.

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.

Hindawi
Scientific Programming
Volume 2020, Article ID 1704258, 12 pages
https://doi.org/10.1155/2020/1704258

Research Article

An Adaptive Data Placement Architecture in Multicloud Environments

Pengwei Wang,[1] Caihui Zhao,[1] Yi Wei,[1] Dong Wang,[2] and Zhaohui Zhang[1]

[1]*School of Computer Science and Technology, Donghua University, Shanghai, China*
[2]*School of Computer Science & Information Engineering, Shanghai Institute of Technology, Shanghai, China*

Correspondence should be addressed to Pengwei Wang; wangpengwei@dhu.edu.cn and Dong Wang; dongwang@sit.edu.cn

Received 23 October 2019; Accepted 24 February 2020; Published 10 June 2020

Guest Editor: Aibo Song

Cloud service providers (CSPs) can offer infinite storage space with cheaper maintenance cost compared to the traditional storage mode. Users tend to store their data in geographical and diverse CSPs so as to avoid vendor lock-in. Static data placement has been widely studied in recent works. However, the data access pattern is often time-varying and users may pay more cost if static placement is adopted during the data lifetime. Therefore, it is a pending problem and challenge of how to dynamically store users' data under time-varying data access pattern. To this end, we propose ADPA, an adaptive data placement architecture that can adjust the data placement scheme based on the time-varying data access pattern and subject for minimizing the total cost and maximizing the data availability. The proposed architecture includes two main components: data retrieval frequency prediction module based on LSTM and data placement optimization module based on Q-learning. The performance of ADPA is evaluated through several experimental scenarios using NASA-HTTP workload and cloud providers information.

1. Introduction

With the development of cloud computing, more and more companies adapt the cloud to store their data for low maintenance costs and reliable SLAs (Service Level Agreements) comparing to the traditional data storage mode. Many mainstream cloud service providers (CSPs) offer a variety of data storage services to satisfy different users' demands. The pricing of services for the same functionality is diverse among cloud service providers (CSPs). And the pricing policy of the same CSP is various in different regions. The cost of data migration among data centers of the same CSP is cheaper than that in different CSPs.

In addition to the constraints of high migration costs, a single cloud faces the risk of vendor lock-in; i.e., major risks include the price of cloud services and the interruption of SLA. These situations may result in making users pay expensive migration costs. In our previous work, we have proposed an ant colony algorithm-based approach for cost-effective data hosting with high availability in multicloud environments [1]. In order to avoid vendor lock-in, we can divide the original data and store the data chunks into multiple CSPs. Furthermore, there is a more comprehensive solution to this problem in [2]. We aim at providing a cost-effective and high-available data placement in multicloud environments based on users' demands.

In the previous work [1–3], the workload of data object affects the choice of data placement scheme. The data workload is related to the data retrieving frequency (DAF) (i.e., Get access rate) during a fixed time period. In the lifecycle of data placement, the workload is time-varying. The data object with high data is read-intensive and in hot-spot status, which is more likely to be stored in CSPs with lower out-bandwidth costs additionally. Conversely, the data object with low DAF is storage-intensive and in cold-spot status, which is more likely to be stored in CSPs with lower storage costs [4]. If a user adapts a data placement scheme with a relatively low-accessible frequency for the entire lifecycle of data storage, it may generate more out-bandwidth costs when DAF increases. In another case, if a user employs the strategies that are more suitable for hot-spot status in the entire life-cycle of data storage, it may produce expensive storage costs when DAF reduces.

In order to reduce the overall cost and enhance availability during the data object lifetime, it is essential to develop a mechanism to dynamically adjust the data placement scheme based on data object workload. Due to the uncertainty of future data workload, the overall cost cannot get the best result. Therefore, predicting the future workload is a key part of the dynamic data placement mechanism. When the future workload is obtained, how to design a dynamic placement scheme based on future access frequency becomes another significant component.

ADPA, which can dynamically host the data object with cost-effective and high availability based on time-varying data workload, is an adaptive data placement architecture proposed in our study. The key contributions of our study are as follows.

Firstly, we propose a LSTM [5] algorithm based on workload prediction. It can use historical workload data to predict future workload.

Secondly, a dynamic data placement algorithm based on reinforcement learning is proposed. It can migrate data according to the change of workload to ensure cost optimization and availability.

Finally, we evaluate ADPA through several experiment scenarios. The results show that ADPA algorithm not only is superior to SOA algorithm, but also can save more time than ACO and DP algorithm to obtain the optimal data placement.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the motivation of the adaptive data placement. Section 4 describes the architecture of adaptive data placement. Section 5 introduces the proposed algorithm. The performance of our proposed algorithm is shown via extensive experiments by using real-world cloud information in Section 6. Finally, Section 7 summarizes this paper.

2. Related Work

There are many studies for data placement optimization in cloud computing. We can divide these studies into two categories based on whether the data storage scheme can be adjusted according to workload: static data placement and dynamic data placement. In static data placement, data placement scheme is determined in advance and does not adapt to variable data object workload. However, in dynamic data placement, data placement scheme can be obtained according to the change of DAF. In the following, we review and discuss them, respectively.

2.1. Static Data Placement. In [6], the authors present the fact that availability of service and data lock-in are two obstacles for cloud computing. In cloud storage, replication and erasure coding are widely used in increasing data availability [7]. To avoid data lock-in, multicloud plays an important role in data storage.

Mansouri et al. [8] propose an approach to replicate data objects in multiple data centers to enhance data availability and avoid data lock-in. In [9], the authors adapt erasure coding to ensure data security and minimize the total cost under many unreasonable assumptions. Hadji [10] focuses on minimizing data storage and presents two efficient algorithms to solve it. But the out-bandwidth cost is ignored. In these studies, each data item is at a fixed number in a static way; Qu et al. [11] present a resilient, fault-tolerance, and high-efficient global replication algorithm (RFH) to determine each data items' replication, migration, and suicide.

The above studies only optimize to minimize the monetary cost or enhance data availability under other QoS metrics. The trade-off between cost and availability is not considered. Wang et al. [1] propose an approach based on ant colony algorithm that minimizes total cost and enhances data availability through erasure coding. It is still a single objective optimization problem. In [12], Liu et al. use multiobjective particle swarm optimization algorithm to minimize storage space cost, data migration cost, and communication cost as well as enhancing the storage reliability. However, they cannot determine how to choose an optimal solution in the Pareto front.

All the above works only consider the static data placement and ignore the uncertainty of data object workload. Once the workload of data object is changed, the data placement scheme needs to be calculated again.

2.2. Dynamic Data Placement. The dynamic data placement in cloud storage becomes a hot direction in research. Zhang et al. [3] propose a data hosting scheme which contains a transition of storage modes based on changes of data access frequency. However, they only put forward a transfer condition without considering the global data placement sequence optimization problem. In [13], Su et al. present a systematic model, in order to formulate data placement optimization for complex requirements in multicloud environments. They also only discuss the data migration without making the final decision for developers.

Due to the unknown future data access pattern, the above studies do not propose the automatic data migration. Mansouri et al. [4] propose two online algorithms that make a trade-off residential and migration cost. Due to the absence of the future data object workload, the authors use the deterministic online algorithm to solve the problem.

To achieve a dynamic data replication strategy based on real data access pattern, Gill and Singh [14] propose a dynamic cost-aware rereplication and rebalancing strategy. This method firstly determines which file and when to replicate based on files popularity. The more the popular it is, the more possible the file is replicated. Then, it calculates the replica number to meet the availability requirements. Finally, a replica placement mechanism is proposed. Lyapunov optimization is an online control method to solve the problem which closes the optimal solution. In [15], Qiu et al. adapt Lyapunov for dynamic content distribution service deployment without the need for any future request arrival information. In other works of dynamically optimized storage, T. Sreenath Reddy and G. MURALI [16] use GA-based totally records dissemination alternate technique and dynamic request redirection to minimize the cost of grouped gets. The algorithm IMPSO used in [17] also inspires our work.

Except for the above methods, a method based on the historical workload pattern is also effective to solve the dynamic data placement. In [18], Papaioannou et al. propose a cloud storage brokerage solution that can periodically recompute the best provider set using data access statics of the last sampling periods. It can adjust data placement for dynamically changing data access pattern. In recent years, reinforcement learning (RL) has received extensive attention. RL is a powerful approach for a long time decision-making under uncertainties [19]. As far as we know, there are no studies on using RL to optimize the dynamic data placement.

In this paper, we propose an adaptive data placement method that mainly contains data retrieval frequency prediction and data placement optimization. In data retrieval frequency prediction module, we adapt LSTM to predict the future workload of data object. In data placement optimization module, an approach based on Q-learning is used to get a sequential data placement solution according to the prediction data object workload.

3. Motivation

3.1. Dynamic Data Object Workload. The DAF of data object is time-varying. We collect the trace of NASA-HTTP that depicts all HTTP requests to the NASA Kennedy Space Center WWW server in Florida [24]. We calculate the DAF in a cycle of 10 minutes. As shown in Figure 1, it depicts the DAF's change from 01/Jul/1995 : 0:0 : 0 to 07/Jul/1995 : 23 : 59 : 59. The difference between the maximum DAF value and the minimum DAF value is close to 2. DAF was relatively low at night but increased during the day. The data placement scheme needs to be dynamically adjusted according to the changing DAF, which will be discussed in Section 3.3.

3.2. Heterogeneous Cloud Market. Now there are many CSPs providing storage services, and we collect four most popular CSPs' pricing strategies: Amazon S3 [20], Microsoft Azure Cloud Storage [21], Alibaba Cloud Object Storage [22], and Google Cloud Storage [23] as shown in Table 1. From the point of users' cost, the selection of CSP is a research direction in cloud computing. In our previous work, we proposed some approaches for optimizing selection of cloud instance types [25–27]. Actually, the price of the same functional storage service provided by the same CSP in different regions is different. And the price of the same functional storage service across CSPs is also different. For example, Amazon S3 in New York, USA, has lower storage price than that in Tokyo. For instance, Amazon S3 in New York has lower storage cost than Microsoft Azure Cloud Storage in New York.

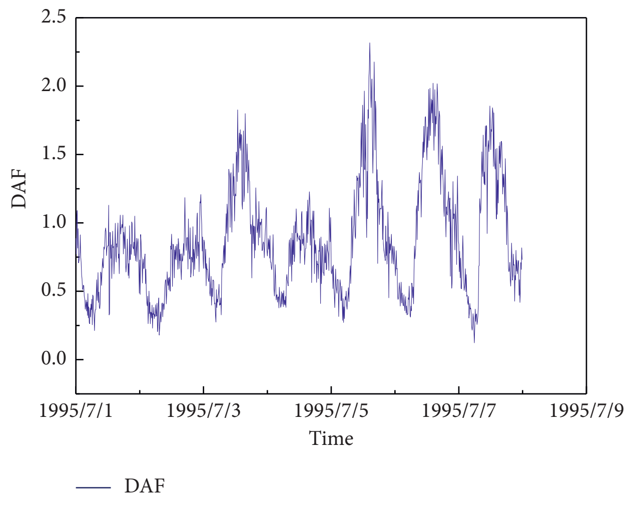


FIGURE 1: NASA-HTTP requests in ten-minute cycles from 01/Jul/1995 to 07/Jul/1995.

3.3. Discussions. The dynamic data object workload and heterogeneous cloud market inspired us to propose an adaptive data placement scheme. Then, we discuss this in detail through the NASA-HTTP example.

When the access frequency is low, the data is suitable for CSPs with low storage cost. But the bandwidth cost of these cloud service providers may be relatively high. We assume that Amazon S3 in New York, Tokyo, and London is chosen as the data placement scheme because of the lower storage cost. The out-bandwidth price of Amazon S3 in New York, Tokyo, and London is very expensive. As time goes by, DAF gradually becomes larger and the output bandwidth price will be very expensive. If users migrate their data to CSPs with lower out-bandwidth cost, they can save more money even if they need to pay for additional migration cost.

In this study, we propose an adaptive data placement scheme which dynamically adjusts the DAF-based data storage scheme to minimize the total cost.

4. An Adaptive Data Placement Architecture

In this section, we present an adaptive data placement architecture with high cost-effective multicloud availability. Then, we formulate the problem in the architecture.

4.1. Architecture Overview. Figure 2 depicts the architecture of dynamic data placement. There are four components: *Cloud Storage Information Collection*, *Optimization Module*, *Workload Statistics Module*, and *Prediction Module*.

Cloud Storage Information Collection is used to collect the information of CSPs including storage price, out-bandwidth price, and operation cost. Apart from this, it receives DAF prediction to adjust data placement scheme.

Optimization Module is used to optimize the data placement scheme according to users' demands which

Table 1: Pricing of storage (in \$/GB/month), Out-bandwidth (in \$/GB/month), and get requests (in \$/10 K/month) of each CSP.

CSP	Amazon S3 [20]			Microsoft Azure Cloud Storage [21]			Alibaba Cloud Object Storage [22]			Google Cloud Storage [23]		
	New York	Tokyo	London	New York	Dublin	Hong Kong	Beijing	San Francisco	Sydney	Atlanta	St. Ghislain	Changhua County
Storage price	0.0125	0.019	0.0131	0.0208	0.022	0.024	0.0226	0.02	0.0209	0.026	0.026	0.026
Out-bandwidth price	0.05	0.12	0.05	0.02	0.02	0.9	0.117	0.076	0.13	0.02	0.02	0.2
Get request price	0.004	0.0037	0.0042	0.004	0.004	0.004	0.001	0.001	0.002	0.004	0.004	0.004

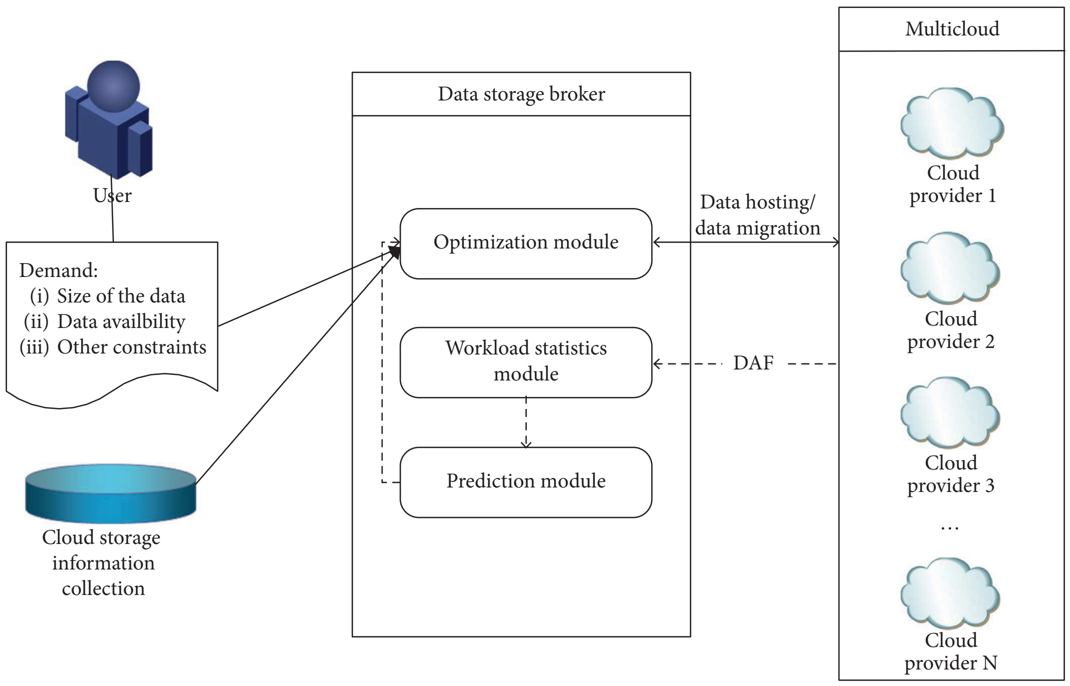


Figure 2: The architecture of ADPA.

contain the size of data object, required availability, initial DAF, and so on.

Workload Statistics and Prediction Module are responsible for collecting the historical data of workload and were based on these historical data to predict the DAF for the next period of time.

4.2. Problem Definition. To describe a dynamic data placement architecture in detail, we introduce the following definitions. The symbols used in this paper are listed in Table 2.

Definition 1 (data center specification). Assume that there are N data centers $\mathrm{DC} = \{d_1, d_2, \ldots, d_N\}$. Each data center d has tuple: P_d^s, P_d^b, P_d^o, a_d, where a_d defines the probability of data center d being available.

Definition 2 (data object specification). Assume that there is a data object with the size S and the DAF $r(t)$ where $t \in [1, T]$, the required availability is A_{req}, and the required data retrieval latency is L_{req}.

In this work, we also adapt erasure coding to avoid vendor lock-in, enhance data availability, and reduce storage and out-bandwidth costs compared to replication. It is worth noting that the following definitions for availability and cost are similar to those in [1, 3, 13], which is a universal way to define them for data hosting in erasure coding mode. We give the definition of erasure coding at first.

Definition 3 (erasure coding). An (m, n)-erasure coding denotes the data object which is divided into m chunks and encodes m chunks into n chunks. Users can retrieve their data through any m chunks.

Definition 4 (data availability). Since users can tolerate $0 \sim (n - m)$ shut-down of data centers at the same time, data availability is the sum probability of all cases that k DCs are simultaneously available, $k \in [m, n]$. Assume that $D(t)$ denotes the data centers chosen for data storage at time slot

Table 2: Symbol table.

Symbols	Descriptions
DC	List of data centers
N	Total number of data centers
P_d^s	Storage price of data center d
P_d^b	Out-bandwidth price of data center d
P_d^o	Operation price of data center d
L_d	Location of data center d, which contains latitude and longitude
T	Number of time slots
$r(t)$	The data access frequency at time slot t
A_{req}	The required availability
L_{req}	The required data retrieval latency
(m, n)	The parameters of erasure coding
S	Size of the data object
$D(t)$	Data placement solution at time slot t
$A(t)$	The availability of the solution of time slot t
$P_{\text{stor}}(t)$	Storage cost of a solution at time slot t
$P_{\text{net}}(t)$	Network cost of a solution at time slot t
$P_{\text{op}}(t)$	Operation cost of a solution at time slot t
$C_B(t)$	The sum of storage cost, network cost, and operation cost at time slot t
$C_M(t)$	The migration cost
$D_r(t)$	Data centers for data retrieval at time slot t
$l(t)$	Data retrieval latency at time slot t
D	The candidate data placement solutions
D^*	The optimal data placement during $[1, T]$

t, where $D(t) = \{d_1, d_2, \ldots, d_n\}$. We use $\Theta = \binom{|D(t)|}{k}$ to indicate the number of cases that k data centers are available. $D_j^{\Theta}(t)$ denotes the jth data centers collection in θ cases at time slot t. The data availability at time slot t can be defined as follows:

$$A(t) = \sum_{k=m}^{n} \sum_{j=1}^{\Theta} \left[\prod_{d \in D_j^{\Theta}(t)} a_d \prod_{d \in D(t) \backslash D_j^{\Theta}(t)} (1 - a_d) \right], \tag{1}$$

where $D(t) \backslash D_j^{\theta}(t)$ is the data centers that are not available.

Definition 5 (storage cost). the storage cost is the sum of storage cost of n data centers. It can be calculated as

$$P_{\text{stor}}(t) = \sum_{d \in D(t)} \frac{S}{m} P_d^s, \tag{2}$$

where S/m denotes the size of data stored in each data center.

Definition 6 (network cost). Users can retrieve their data through any m data chunks. We choose m data centers with the lowest out-bandwidth price to retrieve data. Since the DAF is time-varying, network cost is related to $r(t)$ of each time slot. It can be calculated as follows:

$$P_{\text{net}}(t) = \min_{j \in [1,\Theta]} \left(\sum_{d \in D_j^{\Theta}(t)} \frac{S}{m} r(t) P_{bi} \right). \tag{3}$$

Definition 7 (operation cost). The operation cost can be defined as follows:

$$P_{\text{op}}(t) = \min_{j \in [1,\Theta]} \left(\sum_{d \in D_j^{\Theta}(t)} r(t) P_{oi} \right). \tag{4}$$

It is worth noting that the value of j in (3) and (4) is equal.

Definition 8 (base cost). The base cost $C_b(t)$ is the sum of storage cost, network cost, and operation cost at time slot t. It can be defined as

$$C_B(t) = P_{\text{stor}}(t) + P_{\text{net}}(t) + P_{\text{op}}(t). \tag{5}$$

Definition 9 (migration cost). Different DAFs may correspond to different optimal data placement solutions. It generates expensive cost if users adapt the previous solution. Therefore, it can save more cost according to the DAF dynamically adjustment data placement. But the data placement adjustment means data migration which also requires cost. Data migration is not the moment when DAF changes but satisfies the conditions of the cost saved by the new solution (compared to the old one) which can cover the migration cost. We use $D' = D(t-1) \cap D(t)$ to indicate the intersection of data placement solutions of time slots t and $(t-1)$. The data centers that need data migration are $D(t-1)/D'$. The migration cost can be defined as follows:

$$C_M(t) = \sum_{d \in D(t-1) \backslash D(t)} \left\{ \frac{S}{m} P_d^b + P_d^o \right\}, \tag{6}$$

where the condition is as follows:

$$C_B(t-1) - C_B(t) \geq C_M(t). \tag{7}$$

Definition 10. (data retrieval latency). The data centers for data retrieval $D_r(t)$ at time slot t are m data centers with the lowest out-bandwidth cost. The data retrieval latency is the maximum latency of $D_r(t)$. Since the data retrieval latency is determined by network latency, we use round-trip time to indicate data retrieval latency [4, 28, 29]. We use the calculation method in [30], as follows:

$$l(t) = \max_{d \in D_r(t)} \{5 + 0.02\text{Distance}(d)\}, \tag{8}$$

where Distance(d) is the distance between users and data centers.

4.3. Optimization Problem. In ADPD, it aims to obtain the data placement solution $D(t)$ in each time slot so that the total cost during $t \in [1, T]$ is minimized. The optimization problem is defined as follows:

$$\min_{D(t)} \sum_{t \in [1,T]} (C_B(t) + C_M(t)), \tag{9}$$

subject to

(1) $|D(t)| = n, \forall t \in [1, T]$

(2) $A(t) \geq A_{\text{req}}, \forall t \in [1, T]$

(3) $l(t) \leq L_{\text{req}}, \forall t \in [1, T]$

Constraint (1) indicates that only n data chunks in each time slot. Constraints (2) and (3) ensure that the data placement solutions satisfy the user required availability and data retrieval latency in each time slot. Obviously, this optimization problem is NP-hard.

5. Solution

In this section, we describe the implementation of ADPD. Firstly, we adapt LSTM for data object DAF prediction. In order to make decision-making effectively, the method to solve the problem is mainly based on Q-learning that is an off-policy temporal difference (TD) control algorithm [31, 32].

5.1. DAF Prediction. Recently, there are many machine learning approaches including Support Vector Machines (SVMs), Linear Regression (LR), Random Forest (RF), and k-Nearest Neighbors (kNNs) adopted for prediction. In this study, we use LSTM to predict the future DAF. We can also use other more accurate algorithms to replace LSTM Algorithm 1.

Since our previous work is to predict the price of cloud spot instance whose problem is also for time series data prediction [33]. In this study, we also use sliding window to split the historical DAF data. For DAF prediction, its goal is to find a function f that used historical DAF to predict the future DAF.

5.2. Data Placement Optimization. In this section, the data placement optimization aims to obtain the optimal solution $D(t)$ according the future DAF. Assuming that $D = \{D_1, D_2, \ldots, D_{|D|}\}$ denotes the candidate data placement solutions at each time slot, where $|D| = \binom{|DC|}{n}$. Let D^* indicate the optimal data placement solution from $t = 1$ to $t = T$, where $|D^*| = T$. We describe the process of data placement during $[1, T]$ via Figure 3. For each $D_i \in D$, it has Markov property. The reason is that the solution choice for the next time slot is only related to the currently selected plan and is independent of the previous choices. It can be expressed as $P[D(t+1) \mid D(t)] = P[D(t+1) \mid D(1), D(2), \ldots, D(t)]$.

In data placement process, the cost directly determines the choice of the plan. And the data placement optimization is a sequential decision-making problem in our paper. So we can attribute it to Markov decision process (MDP) and solve it through reinforcement learning (RL). The MDP of data placement process can be defined as follows [34].

Definition 11 (MDP of data placement process). We can transfer data placement process into MDP, which is a 5-tuple one:

(1) A finite set of states is denoted by D

(2) A finite set of actions A are the choice of solution of next time slot

(3) $P^a_{DD'} = P[D(t+1) = D' \mid D(t) = D, A(t) = a]$ is the probability that data placement D at time slot t when taking choice a results in solution D' at time t+1

(4) A reward function $R(D, a, D')$ is the negative value of solution transitions total costs from D to D' when taking action a between two time slots

(5) The discount factor $\lambda \in [0, 1]$, which is the attention of future rewards

The goal of RL is to find the optimal policy based on MDP. The basic architecture of RL is shown in Figure 4 [32]. The agent not only interacts with the environment through making action according to the current state, but also gets an immediate reward and the state of observation [32]. RL aims to find an optimal policy to maximize the total rewards that are obtained from every step. In this optimization problem, the reward function is defined as follows:

$$R(t) = -C_B(t) - C_M(t). \tag{10}$$

So, the RL objective of this problem can be presented as

$$J_{D^*} = \max_{D} E_D \left[\sum_{t=1}^{T} \lambda^t r(t) \right], \tag{11}$$

where $\lambda = 1$ and D^* is the optimal data placement solution during time slot $[1, T]$.

For the above problem, Q-learning is widely used tabular RL algorithm [19, 31] which is an off-policy TD control. It is defined by [31]

$$Q(S, A) \longleftarrow Q(S, A) + \alpha\left[R + \lambda \max_a Q(S', a) - Q(S, A)\right], \tag{12}$$

where $R = R(t)$ that is defined in (10) and α is the learning rate.

As stated previously, we transfer the time series data placement optimization into an MDP and adapt the approach based on Q-learning to solve it. For that purpose, we proposed an MDP that comes from data placement optimization being the most important approach. Algorithm 2 represents the transfer methodology. In order to satisfy the constraints (1)–(3) in (9), we filter all candidate data placement solutions according to their availability and latency (Line 5). In the whole life-cycle of data storing, the DAF is time-varying, which results in the fluctuation of cost. Then, we calculate the state matrix that contains basic cost for all satisfied solutions (i.e., $D = \{D_1, D_2, \ldots, D_{|D|}\}$) at each time slot (Lines 6–9). Finally, the migration cost between two solutions at time slots t and $t-1$ needs to be calculated as the transfer matrix which is a $|D| \times |D|$ matrix (Lines 12–16). Since the migration cost between the same two solutions is free, the values of the diagonal on the matrix are 0.

Compared with general MDP, the reward function contains not only the value between states, but also the value of states which is defined in (10). Based on the above MDP, we present an approach based on Q-learning. The core of the method is calculation and update of Q tabular which is the basis for the agent to choose the next time slot's action. Since this paper solves a time-based sequential decision problem, the Q tabular also has a time dimension. The strategy of Q tabular is based on (12). The value function of next state is adapted to update the current state, which is called bootstrapping [32]. The approach based on Q-learning is an off-policy method; that is, the action strategy and the target strategy are different. In Q-learning, the action chosen strategy is ϵ-greedy and the target is greedy. Algorithm 3 represents the data placement methodology based on Q-learning.

6. Evaluation

In this section, we firstly describe the dataset used in experiments. Then, we present a basic algorithm to verify the necessity of future DAF prediction. Since ant colony optimization algorithm (ACO) and genetic algorithm (GA) are widely used to solve optimization problem [1], we compare ADPA with them through different experimental scenarios.

6.1. Settings. In this work, the setup for DAF, data centers' specification, and experiment parameters are as follows.

6.1.1. DAF. We use the real workload trace of NASA-HTTP from [24] which contains two months' worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida from 01/Jul/1995 to 31/Aug/1995. Each piece of data in NASA-HTTP represents a request, including host, timestamp, request, HTTP reply code, and bytes in the reply. In our paper, in order to obtain data access frequency, we count the number of *Get* operations in a specific time period, which is 600 seconds. The trace is separated as a train and test set for the LSTM at a ratio of 7: 3, respectively.

6.1.2. Data Center Specification. The real-world information of data center, including storage price, out-bandwidth price, *Get* operation price, and latitude and longitude, is collected from CSPs official website [20, 22, 23, 35]. We use 18 data centers from different CSPs, and each CSP's data center is located in a different city. We also simulate the availability values of each data center in the [95.0%, 99.9%] interval.

6.1.3. Experiment Parameters. In the experiments, we choose $(3, 5)$-erasure coding as the data splitting method. The size of sliding window for LSTM is 12. The size of data object is initially 200 GB. The required availability and data retrieval latency are initially 99.9% and 1000 ms, respectively.

6.2. Performance of ADPA. In the time-varying DAF data placement, we cannot obtain the global data placement solutions because of the absence of future DAF. In our method, we firstly predict future DAF for the global optimal data placement solutions during the life-cycle of data storage. In order to verify the performance of ADPA, we propose a step-by-step optimization algorithm called SOA as follows.

6.2.1. SOA Algorithm. This is shown in Algorithm 4. This method minimizes the total cost through Greedy. At the beginning of time slots, the method finds the data placement solution with the lowest basic cost in all candidate solutions. Then, the data objects are allowed to the solution with the lowest total cost at the current time slot (Line 3). In fact, the optimal solution of SOA is locally optimal.

Based on the DAF prediction, we can obtain the global optimal data placement solutions, but the solutions of Greedy are locally optimal. In order to verify this situation, we study the total costs of ADPA and SOA by varying data size and data retrieval latency, respectively, as shown in Table 3 and 4. Obviously, the results of ADPA can save \$13.566, \$26.693, \$38.78, \$52.433, and \$65.437 comparing to Greedy with data size from 100 GB to 500 GB, respectively. Due to the absence of future DAF, SOA only chooses the best solution at current time slot and ignores the importance of future total cost. Conversely, ADPA, based on Q-learning, chooses the current solution which is based not only on current costs, but also on future reward.

As shown in Table 4, we study the results of two methods with the data retrieval latency constraint from 200 ms to 500 ms, the number of time slots is 12 (i.e., 2 hours), and data size is 200 GB. As latency constraint increases from 200 ms to 500 ms, the cost saving of ADPA is 18.6%. When data retrieval latency is loose, comparing with strict latency, the method explores the solution with lower cost in a larger

```
**Input**: The required length max_step of the series to be forecast, a trained LSTM model $L_{\text{trained}}$.
**Output**: Predicted future DAF;
(1) cnt = 0;
(2) Feed the data in the last window into $L_{\text{trained}}$ and get a prediction $L(\text{SW}_{\text{cnt}})$;
(3) **while** cnt < max_step **do**
(4)     Slide the window forward by one step and put the newly predicted value $L(\text{SW}_{\text{cnt}})$ at the end of the slide window;
(5)     Feed the data in the new window into $L_{\text{trained}}$ and get another prediction
$L(\text{SW}_{\text{cnt}})$;
(6)     cnt = cnt + 1;
(7) **end while**
```

Algorithm 1: Algorithm LSTM: LSTM network with sliding window.

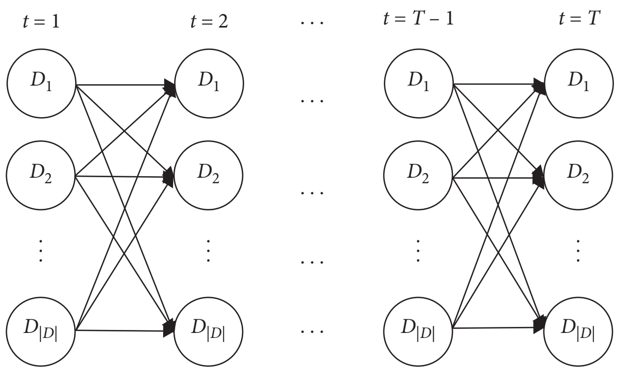


Figure 3: The data placement during the period $[1, T]$.

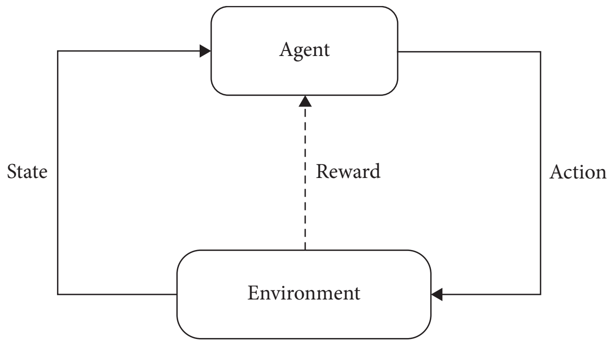


Figure 4: The basic architecture of RL.

range. The algorithm explores solutions in the area near users to satisfy the strict latency constraint.

In actual life, there are different levels of data access and these situations have different usage scenarios. In order to verify the universality of the ADPA algorithm, we extend the time range of NASA-HTTP request data to one year (24/Oct/1994–11/Oct/1995) and perform DAF statistics based on one-day cycles. Due to the wide time range of the second data set, it has a lower DAF than our first data set. The experimental environment is the same; we use the second dataset to conduct a comparison experiment as shown in Tables 5 and 6. Similarly, it is clear that ADPA can help users save more money.

6.3. Performance Comparison with Other Methods. We compare our method with ACO and DP through the following experimental scenarios. Assume that ACO and GA can obtain the DAF prediction which is predicted by ADPA.

6.3.1. Cost Saving with Varying Data Size. In this experimental scenario, we evaluate the cost performance of ADPA when data size is increased from 100 GB to 500 GB. From the experimental results, we can know that the total cost of three algorithms increases when the data size increases, and the costs of the three algorithms obtained under different data sizes are the same. Since the DP algorithm can find the optimal solution, it can prove the correctness of ADPA in solving the solution. To demonstrate the advantages of ADPA, the time required to find the optimal solution for the three algorithms is shown in Figure 5. It can be seen that the time required by ACO is greater than that of ADPA and DP algorithm. The algorithm ADPA is higher than DP when the data size is 200 GB, and the others are better than DP.

6.3.2. Cost Saving with Varying Time Slot Count. In this scenario, we explore the impact of time slot count (T) on total costs when T varies from 10 to 18. From these experimental results, we can know that the greater the T value, the longer the optimization time and the costs of the three algorithms obtained under different time slot count (T) are the same. Because of the same reason as above, this proves that ADPA has the ability of solving the solution. In order to reflect the advantages of ADPA, the time required to find the optimal solution for the three algorithms is shown in Figure 6. It can be seen that, for ADPA and DP, except for $T = 10$, the running time of the algorithm proposed in this paper is only 50% of the DP algorithm; that is, ADPA can solve the optimal solution in the least time.

6.3.3. Cost Saving with Varying Candidate Data Centers. We investigate the impact of the number of candidate data centers (N) on total cost. Figure 7 gives the results when N increases from 12 to 15. The change in candidate data centers count is adding a new data center in the previous set. For example, when $N = 10$, the index of data centers is $DC = \{3, 9, 16, 15, 14, 6, 7, 12, 8, 4, 0, 10\}$ and we add a new data center with index of 13 to this set when $N = 11$. When N

```
     Input: Data center specification, DC, DAF, $r(t)$, the required availability, $A_{req}$, the required data retrieval latency, $L_{req}$.
     Output: The state matrix, SM, the transfer matrix, TM.
(1)  $D \longleftarrow$ Calculate all $n$-combinations of data centers DC.
(2)  for all $c \in D$ do
(3)      Calculate availability of $c$ ava$(c)$ through (1);
(4)      Calculate latency of $c$ latency$(c)$ through (8);
(5)      if ava$(c) \geq A_{req}$ and latency$(c) \leq L_{req}$ then
(6)          for $t = 1$ to $T$ do
(7)              Calculate the base cost of $c$ $C_B^c(t)$ through (5);
(8)              SM$[c][t] = C_B^c(t) + C_M^c(t)$;
(9)          end for
(10)     end if
(11) end for
(12) for all $c1 \in D$ do
(13)     for all $c2 \in D$ do
(14)         TM$[c1][c2] \longleftarrow$ Calculate the migration cost between $c1$ and $c2$ through (6);
(15)     end for
(16) end for
(17) return SM, TM;
```

Algorithm 2: Algorithm TDM: transform data placement into an MDP.

```
     Input: The state matrix, SM, the transfer matrix, TM.
     Output: The optimal data placement solution during $t \in [1, T]$, $D^*$
(1)  Initialize parameters of algorithm including learning rate $\alpha$, discount
(2)  Initialize the tabular $Q$ with zero;
(3)  $s \longleftarrow$ Initialize the start state;
(4)  Initialize data placement sequences Seq;
(5)  for $e = 1$ to epochs do
(6)      for $t = 1$ to $T$ do
(7)          $f$Set $\longleftarrow$ Choose feasible data placement solutions through equation
(8)          Choose a data placement solution $D(t)$ from $f$Set through $\epsilon$-greedy function;
(9)          Append $D(t)$ in Seq$[e]$;
(10)         Obtain the next state $s_{nxt}$, reward $r(t)$, and the next state's
(11)         $Q(t, s, D(t)) = Q(t, s, D(t) + \alpha * (-TM[s, D(t)] - SM[D(t), t] + \gamma * \max(Q[t+1, s_{nxt}, A_{nxt}])) - Q[t, s, D(t)])$;
(12)         $s = s_{nxt}$;
(13)         Take $\varepsilon$ decay;
(14)     end for
(15) end for
(16) $D^* \longleftarrow$ Find the sequence with lowest cost in Seq;
(17) return $D^*$;
```

Algorithm 3: Algorithm DPQ: data placement algorithm based on Q-learning.

```
    Input: The state matrix, SM, the transfer matrix, TM.
    Output: The optimal data placement solution during $t \in [1, T]$,
(1) $D(0) \longleftarrow$ Calculate data placement solution with the lowest
(2) for $t=1$ to T-1 do
(3)     $D(t) \longleftarrow$ Find the solution with min(TM$[D(t-1), :]$ + SM$[:, t]$);
(4) end for
(5) $D^* = D[0: T-1]$
(6) return $D^*$;
```

Algorithm 4: Algorithm SOA: data placement optimization step by step.

Table 3: The total cost ($) of ADPA and SOA with varying data size in seven days (based on high DAF).

Data size (GB)	ADPA	SOA
100	1027.01	1042.33
200	2052.43	2082.52
300	3077.85	3122.71
400	4103.26	4162.90
500	5128.68	5203.09

Table 4: The total cost ($) of ADPA and SOA with varying data retrieval latency in seven days (based on high DAF).

Data retrieval latency (ms)	ADPA	SOA
200	6305.49	6370.93
300	5128.68	5203.95
400	5128.68	5203.95
500	5128.68	5203.95

Table 5: The total cost ($) of ADPA and SOA with varying data size within one year (based on low DAF).

Data size (GB)	ADPA	SOA
100	204.35	245.27
200	408.44	490.10
300	612.53	734.93
400	816.62	979.76
500	1020.71	1224.59

Table 6: The total cost ($) of ADPA and SOA with varying data retrieval latency within one year (based on low DAF).

Data retrieval latency (ms)	ADPA	SOA
200	204.35	245.27
300	164.65	188.16
400	126.42	157.22
500	126.42	157.21

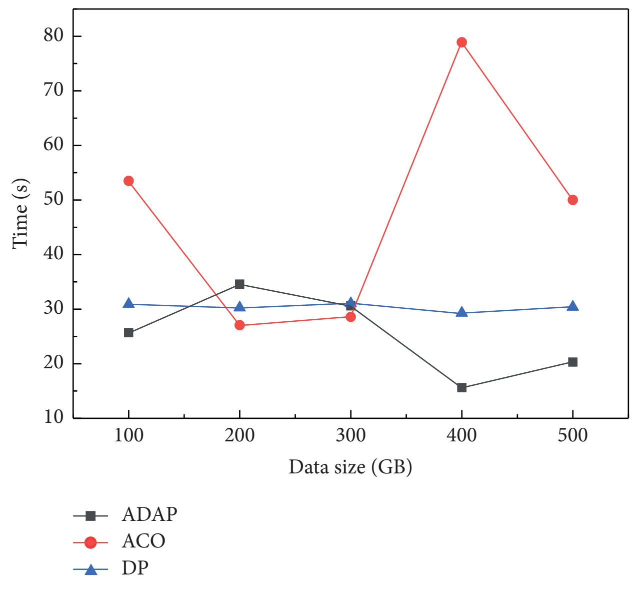


Figure 5: The comparison of run time.

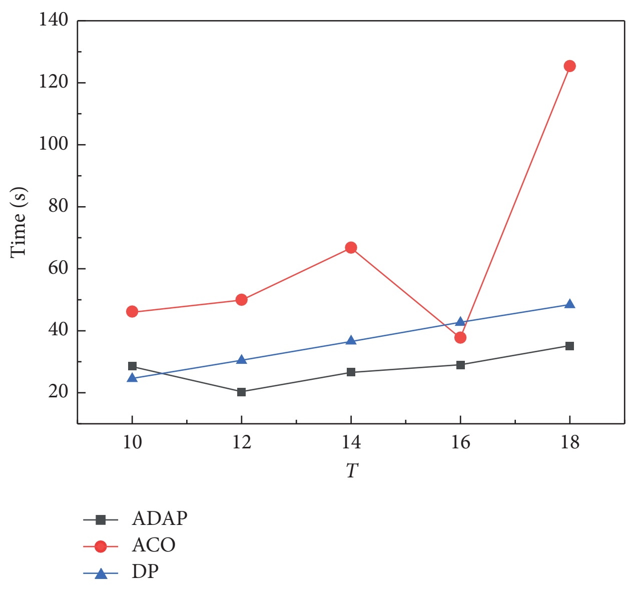


Figure 6: The comparison of run time.

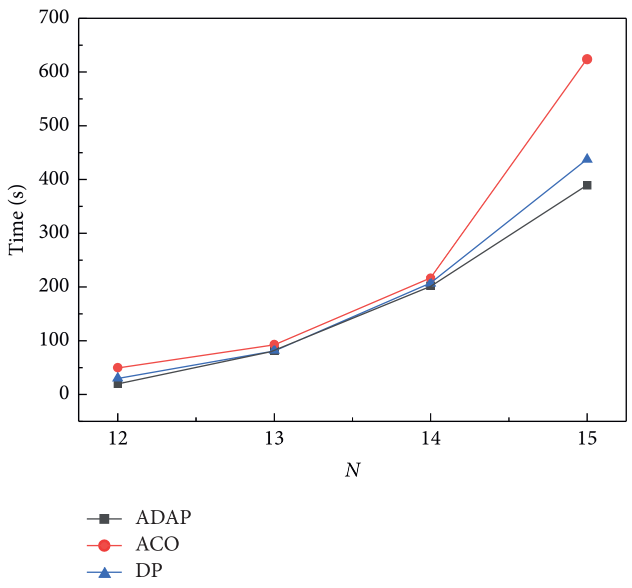


Figure 7: The comparison of run time.

Table 7: The total cost ($) comparison with varying data size within one year (based on low DAF).

Data size (GB)	ADPA	ACO	DP
100	204.35	238.51	238.51
200	408.44	476.73	476.73
300	612.53	714.94	714.94
400	816.62	953.16	953.16
500	1020.71	1191.38	1191.38

increases, the candidate solutions $D(t)$ have a clear growth trend. Assuming that data retrieval constraint is 500 ms, $T = 12$, data size is 500 GB and data availability is 0.999; $|D(t)|$ are 792, 1287, 2002, and 3003 when N varies from 12 to 15. When $N = 12$, 13, ADPA and ACO can solve the

Table 8: The total cost ($) with varying data retrieval latency within one year (based on low DAF).

Data retrieval latency (ms)	ADPA	ACO	DP
200	204.35	238.51	238.51
300	164.65	165.16	165.16
400	126.42	141.84	141.84
500	126.42	141.84	141.84

optimal solution, but when $N = 13, 14$, the cost of the ADPA and ACO is higher than the optimal scheme by $0.99 and $1.23, respectively. However, from Figure 7, as is manifested, the running time of ADPA is less than that of the other two algorithms in all N values; that is, the efficiency of solving the optimal solution is higher than that of ACO and DP. In this section, we also perform a comparative experiment on the dataset with lower DAF and achieve good results, as shown in Tables 7 and 8.

7. Conclusion

Users should adjust data placement solution based on DAF to minimize the storage, get operation, out-bandwidth, and migration costs in the whole life-cycle of data storage. In order to achieve this goal, we present an adaptive data placement architecture named *ADPA*. Because of the absence of the future DAF, the DAF prediction module based LSTM of ADPA can predict the future DAF through historical data. And then, the data placement optimization module, which is based on Q-learning of reinforcement learning, solves the optimal data placement solution sequence according to the prediction DAF. The experiments driven by a real workload of NASA-HTTP and cloud providers information indicate that ADPA not only is superior to the algorithm SOA but can save more time than ACO and DP to obtain the optimal data placement. In the future, we intend to present an architecture which can adjust data placement based on the change of cloud market.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

Pengwei Wang and Dong Wang are corresponding authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (NSFC) under Grant no. 61602109, DHU Distinguished Young Professor Program under Grant no. LZB2019003, Shanghai Science and Technology Innovation Action Plan under Grant no. 19511101802, Natural Science Foundation of Shanghai under Grant no. 19ZR1401900, Fundamental Research Funds for the Central Universities, and Alliance PlanSpecial Tenders For Difficult Problems under Grant no. LM201819.

References

[1] P. Wang, C. Zhao, and Z. Zhang, "An ant colony algorithm-based approach for cost-effective data hosting with high availability in multi-cloud environments," in *15th IEEE International Conference on Networking, Sensing and Control*, pp. 1–6, IEEE, Zhuhai, China, March 2018.

[2] P. Wang, C. Zhao, W. Liu, Z. Chen, and Z. Zhang, "Optimizing data placement for cost effective and high available multi-cloud storage," *Computing and Informatics*, vol. 39, no. 1, pp. 1001–1032. In press, 2020.

[3] Q. Zhang, S. Li, Z. Li, Y. Xing, Z. Yang, and Y. Dai, "Charm: a cost-efficient multi-cloud data hosting scheme with high availability," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 372–386, 2015.

[4] Y. Mansouri, A. N. Toosi, and R. Buyya, "Cost optimization for dynamic replication and migration of data in cloud data centers," *IEEE Transactions on Cloud Computing*, vol. 99, p. 1, 2017.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[7] Y. Mansouri, A. N. Toosi, and R. Buyya, "Data storage management in cloud environments: taxonomy, survey, and future directions," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, 2017.

[8] Y. Mansouri, A. N. Toosi, and R. Buyya, "Brokering algorithms for optimizing the availability and cost of cloud storage services," in *Proceedings of IEEE 5th International Conference on Cloud Computing Technology and Science*, pp. 581–589, Bristol, UK, December 2013.

[9] Y. Singh, F. Kandah, and W. Zhang, "A secured cost-effective multi-cloud storage in cloud computing," in *Proceedings of the 2011 IEEE Conference on Computer Communications Workshops*, pp. 619–624, Shanghai China, June 2011.

[10] M. Hadji, "Scalable and cost-efficient algorithms for reliable and distributed cloud storage," in *International Conference on Cloud Computing and Services Science*, pp. 15–37, Lisbon, Portugal, May 2015.

[11] Y. Qu and N. Xiong, "A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage," in *Proceedings of the 2012 41st International Conference on Parallel Processing (ICPP)*, pp. 520–529, Pittsburgh, PA, USA, September 2012.

[12] X. Liu, L. Fan, L. Wang, and S. Meng, "Multiobjective reliable cloud storage with its particle swarm optimization algorithm," *Mathematical Problems in Engineering*, vol. 2016, Article ID 9529526, 14 pages, 2016.

[13] M. Su, L. Zhang, Y. Wu, K. Chen, and K. LI, "Systematic data placement optimization in multi-cloud storage for complex requirements," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1964–1977, 2016.

[14] N. K. Gill and S. Singh, "A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers," *Future Generation Computer Systems*, vol. 65, pp. 10–32, 2016.

[15] X. Qiu, H. Li, C. Wu, Z. Li, and F. Lau, "Cost-minimizing dynamic migration of content distribution services into hybrid clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, 2015.

[16] T. S. Reddy and G. Murali, "Implementing the least-price cloud storage service in multiple cloud providers," in *Proceedings of the 2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, pp. 464–469, Coimbatore, India, October 2018.

[17] P. Wang, Y. Lei, P. R. Agbedanu, and Z. Zhang, "Makespan-driven workflow scheduling in clouds using immune-based PSO algorithm," *IEEE Access*, vol. 8, pp. 29281–29290, 2020.

[18] T. G. Papaioannou, N. Bonvin, and K. Aberer, "Scalia: an adaptive scheme for efficient multi-cloud storage," in *Proceedings of the 2012 International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–10, Salt Lake City, UT, USA, November 2012.

[19] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: a comprehensive overview," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 385–398, 2015.

[20] Amazon S3, 2018, https://aws.amazon.com/cn/s3/pricing/?nc=sn&loc=4.

[21] Microsoft Azure Cloud Storage, https://azure.microsoft.com/en-us/pricing/details/storage/.

[22] Alibaba Cloud Object Storage, https://www.aliyun.com/price/product/oss/detail.

[23] Google Cloud Storage, https://cloud.google.com/pricing/.

[24] 2018, http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html.

[25] P. Wang, W. Zhou, C. Zhao, and Y. Lei, "A dynamic programming-based approach for cloud instance types selection and optimization," *International Journal of Information Technology and Management*, vol. 19, no. 4, pp. 358–375, 2020.

[26] W. Liu, P. Wang, Y. Meng, Q. Zhao, C. Zhao, and Z. Zhang, "A novel model for optimizing selection of cloud instance types," *IEEE Access*, vol. 7, pp. 120508–120521, 2019.

[27] W. Liu, P. Wang, Y. Meng, G. Zou, and Z. Zhang, "A novel algorithm for optimizing selection of cloud instance types in multi-cloud environment," in *Proceedings of the 25th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2019)*, Tianjin, China, December 2019.

[28] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. Madhyastha, "Spanstore: cost-effective geo-replicated storage spanning multiple cloud services," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, (SOSP13), New York, USA*, p. 292308, 2013.

[29] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. Lau, "Scaling social media applications into geo-distributed clouds," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 8, pp. 689–702, 2015.

[30] A. Qureshi, *Power-demand routing in massive geo-distributed systems*, PhD thesis, MIT, Cambridge, MA, USA, 2010.

[31] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[32] R. Sutton and A. Barto, *Reinforcement Learning: An introduction*, MIT press, Cambridge, MA, USA, 1998.

[33] W. Liu, P. Wang, Y. Meng, C. Zhao, and Z. Zhang, "Amazon EC2 spot instance price prediction using kNN regression," in *Proceedings of the 2018 Asia-Pacific Services Computing Conference, IEEE, Zhuhai, China*, January 2018.

[34] R. Bellman, "A Markovian decision process," *Indiana University Mathematics Journal*, vol. 6, no. 4, pp. 679–684, 1957.

[35] IBM Cloud Object Storage Pricing, https://www.ibm.com/cloud-computing/bluemix/pricing-object-storage.

Hindawi
Scientific Programming
Volume 2020, Article ID 4690974, 10 pages
https://doi.org/10.1155/2020/4690974

Research Article

Big Data Management and Analytics in Scientific Programming: A Deep Learning-Based Method for Aspect Category Classification of Question-Answering-Style Reviews

Hanqian Wu,[1,2] Mumu Liu,[1,2] Shangbin Zhang,[1,2] Zhike Wang,[1,2] and Siliang Cheng[1,2]

[1]*School of Computer Science and Engineering, Southeast University, Nanjing, China*
[2]*Key Laboratory of Computer Network and Information Integration of Ministry of Education, Southeast University, Nanjing, China*

Correspondence should be addressed to Hanqian Wu; hanqian@seu.edu.cn

Received 23 October 2019; Revised 7 January 2020; Accepted 16 January 2020; Published 8 June 2020

Guest Editor: Zhiang Wu

Online product reviews are exploring on e-commerce platforms, and mining aspect-level product information contained in those reviews has great economic benefit. The aspect category classification task is a basic task for aspect-level sentiment analysis which has become a hot research topic in the natural language processing (NLP) field during the last decades. In various e-commerce platforms, there emerge various user-generated question-answering (QA) reviews which generally contain much aspect-related information of products. Although some researchers have devoted their efforts on the aspect category classification for traditional product reviews, the existing deep learning-based approaches cannot be well applied to represent the QA-style reviews. Thus, we propose a 4-dimension (4D) textual representation model based on QA interaction-level and hyperinteraction-level by modeling with different levels of the text representation, i.e., word-level, sentence-level, QA interaction-level, and hyperinteraction-level. In our experiments, the empirical studies on datasets from three domains demonstrate that our proposals perform better than traditional sentence-level representation approaches, especially in the *Digit* domain.

1. Introduction

Nowadays, Internet finance has evolved rapidly, and e-commerce platform occupies the central position in its development by providing virtual payment means. In the virtual trading environment created by the e-commerce platform, both parties can realize communication on time by online product reviews. To some extent, the content of those reviews can affect the business of the e-commerce platform, thus spreading to the stability of the whole Internet finance. On these grounds, it is very important to mine valuable information contained in those reviews, which can not only help consumers make purchase decisions but also help organizations know customer satisfaction and make adjustment strategies. However, the number of online product reviews is exploding. It is difficult for us to manually collect and manipulate these texts. Thus, sentiment analysis comes into being.

Sentiment analysis, also known as opinion mining, is a task which aims to analyze people's sentimental orientation of a given text. Researchers used to pay attention on document-level or sentence-level sentiment analysis; however, with the rapid development of e-commerce business, consumers tend to learn about products in detail. Thus, aspect-level sentiment analysis has become a research hotspot, which is proposed to have a better understanding of rapid-growing online reviews than traditional opinion mining by extracting fine-grained insights such as aspect terms, aspect categories, and sentiment polarities.

Recently, there appears a new question-answering (QA) style of reviews, namely, "customer questions and answers," in various popular e-commerce platforms at home and abroad, such as Taobao, Jingdong, and Amazon. Figure 1 shows an example of QA-style reviews. In this novel reviewing form, a potential customer can ask questions

Question1: *How long does it last when you are playing games?Is the quality of this phone good?*
Answer1: *The electricity is not very durable, but its appearance is good.*
→ *Battery*

Question2: *I want to buy it for design, how about its running?*
Answer2: *It has lots of running memory, but I sometimes feel a little* choppy
→ *Performance*

Figure 1: Translated examples of QA-style reviews on e-commerce websites. The coloured words are called aspect terms, and words in red are matched in a QA-style review while those in other colours are unmatched.

about a certain product, and others who have purchased the same item can answer these questions. Along with the popularity of this new reviewing form, the relevant research is really worthwhile due to its own characteristics. For one thing, consumers prefer QA-style reviews to traditional reviews. For another, this QA reviewing way can largely reduce fake information which makes product reviews more reliable and convincing. However, there is less existing research on the aspect category classification of QA-style reviews which aims to identify the aspect category of a given QA-style review.

And the existing deep learning-based approaches to aspect category classification on traditional product reviews cannot be directly applied to identify the aspect category of a given QA-style review. On the one hand, for a QA-style review, the aspect category referred in both question and answer texts is the valid aspect. Thus, the aspect-related matching information between the question and answer text is helpful for aspect category classification on QA-style reviews. We argue that the 2D textual representation is difficult to capture the semantic relationship between the question and its corresponding answer due to the possible long distance between them if we simply concatenate them into a sequence as the representation of a QA-style review for classification. On the other hand, the matching information between the question and answer sentence is more or less related to the annotated aspect category, but the 2D textual representation may not explore the correlation degree of the matching information between the question and answer sentence with the annotated aspect category.

Thus, we illustrate our 4-dimension (4D) textual representation model in Figure 2. In our 4D textual representation approach, the word-level representation is leveraged as the first dimension, and the sentence level is leveraged as the second dimension, which is similar to the representation of the traditional textual representation. Furthermore, different from traditional text representation approaches, another two dimensions, namely, QA interaction-level and hyperinteraction-level representations, are proposed as the third and fourth dimensions. Our empirical studies demonstrate the effectiveness of our proposals for the aspect category classification task on QA-style reviews.

The main contributions of our work are summarized as follows:

(i) We introduce the QA interaction-level dimension representation to capture the matching information between the question and answer sentence.

(ii) We propose the hyperinteraction-level dimension representation to explore the correlation degree of the matching information between the question and answer sentence with the annotated aspect category.

(iii) We conduct comparison experiments on our annotated datasets extracted from the domain of *Digit*, *Beauty*, and *Bag* in the famous website Taobao. We find that our proposed classification model, 4-dimension textual representation, tailored for QA-style reviews performs better than the 2D textual representation. Especially in the *Digit* domain, the accuracy and Macro-F1 of our proposed approach are, respectively, 7.5% and 10.7% higher than the 2D representation.

The rest of our paper is organized as follows. Section 2 discusses the related work on the aspect category classification. Section 3 presents data collection and annotation. Section 4 proposes our approach to aspect category classification on QA-style reviews. Section 5 reports the experimental setup, and Section 6 discusses and analyzes the results. Finally, Section 7 gives the conclusion and our future work.

2. Motivation

The existing learning approaches to the aspect category classification on traditional product reviews are deep learning-based approaches which first model each word as a real-valued vector, i.e., word-embedding phase. Then, the whole sentence or document is represented as a word-embedding sequence and trained with a sequence learning model, such as RNN [1] and LSTM [2]. In such approaches, the sentence or document text is represented by two dimensions, i.e., word-embedding dimension and word sequence dimension. In brief, we refer to this kind of representation as 2-dimension (2D) textual representation. Taking an example of traditional product reviews "I like beef very much!," the flow of aspect category classification based on the 2D representation is as shown in Figure 3.

QA-style aspect category classification task aims to identify the aspect category referred in both question and answer texts inside a given QA-style review. One straightforward way to deal with this task is to directly apply existing learning approaches based on the 2D representation to aspect category classification on other kinds of text styles. The part of results based on the 2D representation of QA-style reviews is illustrated in Table 1. We can find that if we adopt methods based on the 2D representation to deal with

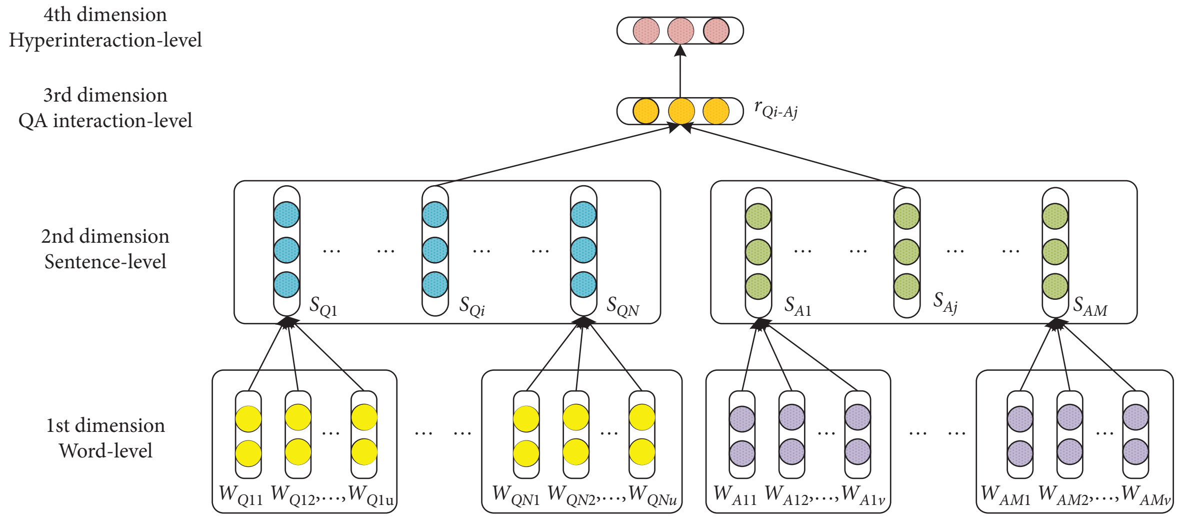


FIGURE 2: The overall architecture of the aspect category classification based on the 4D textual representation of QA-style reviews.

the aspect category classification task on QA-style reviews, the performance of classification is very bad. Thus, by analysis, the 2D representation is not the best choice for representing QA-style texts due to the following reasons.

First, in the QA-style review, it consists of the question text and answer text. And the question text and answer text are more likely to be two parallel units rather than a sequence form. For instance, in Figure 1, Answer 1 "The electricity is not very durable," is actually not following the last sentence in Question 1 "Is the quality of this phone good?" but is corresponding to the first sentence in Question 1 "How long does it last when you are playing games?" Therefore, when the question text and answer text are presented as two units in a sequence, it is rather difficult to capture the relationship between the question and its corresponding answer due to the possible long distance between them. A better way to handle this issue is to segment the question and answer text into some parallel sentences and capture the matching measurement between the question and answer sentence.

Second, in the question or answer text, there often exist different aspect categories in different sentences. And the matching information between the question and answer sentence is more or less related to the annotated aspect category. For instance, in Figure 1, the matching information between the sentence in Question 1 "How about its running?" and the sentence in Answer 2 "It has lots of running memory" is most related to the annotated aspect category "performance." A better choice to handle this issue is to explore the correlation degree of the matching information between the question and answer sentence with the annotated aspect category.

To summarize, we propose another two dimensions, i.e., QA interaction-level and hyperinteraction-level, to build a novel classification model based on the 4-dimension representation for the aspect category classification task on QA-style reviews. We leverage the QA interaction-level dimension to capture the matching measurement between the question and answer sentence and the hyperinteraction-level dimension to explore the matching information between the question and answer sentence with the annotated aspect category. Then, we will introduce our model in detail in the following section.

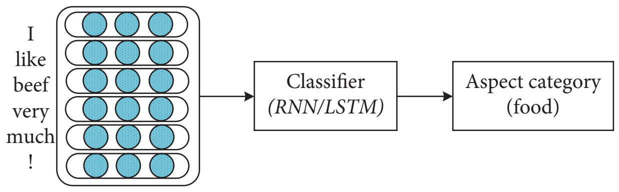


FIGURE 3: The flow of the aspect category classification on traditional product reviews.

3. Related Work

3.1. Aspect Category Classification. Aspect category classification, a basic task for aspect-level sentiment analysis [3], aims at identifying an aspect category referred in a given review, which is usually treated as a special case of the text classification task. Therefore, text classification approaches can be naturally applied to solve the aspect category classification task, such as SVM [4]. Alvarez-López et al. [5] used SVM to classify aspect categories based on restaurant review corpus in English and Spanish. Machacek [6] used the bigram model to solve the aspect category classification task. Kiritchenko et al. [7] used a set of binary SVMs with different types of n-grams and information from a specially designed lexicon. However, traditional machine learning approaches focus on sparse lexical features such as n-grams, one-hot vector representation, and term frequency-inverse document frequency (TF-IDF) to represent the text; these approaches highly depend on the quality of features, and feature engineering is labor-intensive.

Recently, with the spread of the word2vec model [8], neural network approaches based on CNNs [9, 10] or RNNs have shown promising results. Kim [9] pioneered the use of

Table 1: The part of results based on the 2D representation of QA-style reviews.

Question text	Answer text	Ground aspect	Predicted aspect	Yes/no
How about the quality of this phone? How long is the stand-by time?	I can make few phone calls and then it is off. The sound is not too loud.	Battery	Quality	No
Hello, are the products in this store authentic? How about the quality of products?	The quality is OK.	Quality	Certified product	No
Is it authentic?	Yes, it is. We can take clear pictures and it does not stutter.	Certified product	IO	No
How about its appearance?	It looks very good.	Appearance	Appearance	Yes

convolutional neural networks for text categorization and achieved better results than most traditional methods on related datasets. CNNs have strong feature expression ability and can be easily parallelized. However, it is not easy to determine the optimal size of the convolutional kernel. Tang et al. [11] proposed a method of text classification using RNNs. Compared to CNNs, RNNs are able to process variable-length sentences better and have a strong ability to relate context. Gated recurrent unit (GRU) networks and long short-term memory (LSTM) [2, 12] networks can mitigate the problem of gradient disappearance and gradient explosion in addition to preserving information for a longer time than plain RNNs.

Neural networks with attention mechanisms have achieved very good results in the field of machine reading comprehension, machine translation, etc., causing a lot of research around attention mechanisms [11, 13]. Yang et al. proposed a text classification approach that stratified documents according to their structure and then combined a bidirectional GRU network with an attention mechanism [14].

Considering methods used in the text classification, Toh and Su [15] adopted the sigmoidal feedforward network to train a binary classifier for the aspect category classification. Xue et al. [16] utilized the correlation between the aspect category classification task and the aspect term extraction task to perform joint learning. Wan et al. [17] proposed a representation learning algorithm for the aspect category classification. Those approaches are based on the 2D textual representation for the aspect category classification task. Besides, different from all of the above, our work devotes to the aspect category classification task on QA-style reviews in which little research focus on.

4. Model

In this section, we present our model to identify the aspect category of a given QA-style review. The basic idea of our approach is to learn a 4-dimension textual representation for the question text in a QA-style review. As introduced in Section 1, our model contains four dimensions for representing different levels' texts including word level, sentence level, QA interaction-level, and hyperinteraction level.

The 1st dimension and the 2nd dimension have been widely used in the previous studies in the text classification research area. In this paper, we adopt the word2vec model [18] to obtain the word-level dimension and apply the sequence-to-sequence neural network, i.e., bidirectional gated recurrent unit (Bi-GRU) [19], to obtain the sentence-level representation.

In the following sections, we propose the representation models for the 3rd dimension and the 4th dimension, i.e., QA interaction-level and hyperinteraction level.

4.1. The 3rd Dimension Representation.

4.1. The 3rd Dimension Representation. According to the characteristics of QA-style reviews, we can see the matching measurement between the question and answer text is important for the aspect category classification of QA-style reviews. In this section, we learn the matching representation between a sentence from the question text and a sentence from the answer text, i.e., learning sentence-sentence matching representation.

Figure 4 shows the architecture of the representation learning process for the sentence-sentence matching. Formally, we assume that the question text contains N sentences and is denoted as $[S_{Q_1}, S_{Q_2}, \ldots, S_{Q_N}]$, where S_{Q_i} is the sentence representation of the i-th sentence in the question text with words $[w_{Q_{i1}}, w_{Q_{i2}}, \ldots, w_{Q_{iu}}]$. Similarly, S_{A_j} is the sentence representation of the j-th sentence in the answer text with words $[w_{A_{j1}}, w_{A_{j2}}, \ldots, w_{A_{jv}}]$.

Formally, we use bidirectional GRU (Bi-GRU) layers, which can effectively utilize the forward and backward features, to get contextual representations of the question and answer sentences. Through the sequence-to-sequence Bi-GRU layers, the annotation of each word is produced by averaging the forward and backward hidden states. For a question sentence S_{Q_i}, we obtain the hidden state matrix H_{Q_i} by the following formulas. Similarly, we obtain the hidden state matrix of the answer sentence S_{A_j}.

$$\begin{aligned}\overrightarrow{R_{Q_i}} &= \overrightarrow{\mathrm{GRU}}\left(S_{Q_i}\right),\\ \overleftarrow{R_{Q_i}} &= \overleftarrow{\mathrm{GRU}}\left(S_{Q_i}\right),\\ R_Q &= \mathrm{AVE}\left(\overrightarrow{R_{Q_i}}; \overleftarrow{R_{Q_i}}\right).\end{aligned} \tag{1}$$

Then, we calculate the pairwise matching matrix, which represents the matching degree of the question sentence and the answer sentence. Given a question sentence S_{Q_i} and an answer sentence S_{A_j}, we can compute a matching matrix by using the following formula, i.e.,

$$\mathrm{Matching}\left(Q_i, A_j\right) = \tanh\left(W_{ij} \cdot \left(H_{Q_i}^T \cdot H_{A_j}\right) + b_{ij}\right), \tag{2}$$

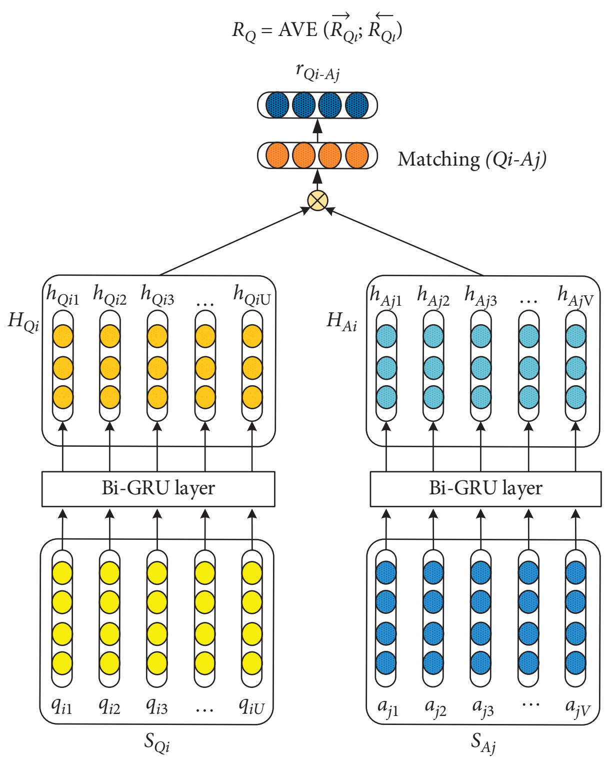

R_Q = AVE ($\overrightarrow{R}_{Qi}$; $\overleftarrow{R}_{Qi}$)
$r_{Qi\text{-}Aj}$
Matching (Qi-Aj)
H_{Qi}
h_{Qi1} h_{Qi2} h_{Qi3} … h_{QiU}
H_{Ai}
h_{Aj1} h_{Aj2} h_{Aj3} … h_{AjV}
Bi-GRU layer
Bi-GRU layer
q_{i1} q_{i2} q_{i3} … q_{iU}
a_{j1} a_{j2} a_{j3} … a_{jV}
S_{Qi}
S_{Aj}

Figure 4: Learning sentence-sentence matching representation.

where Matching(Q_i, A_j) denotes the matching matrix between the two sentences, i.e., S_{Q_i} and S_{A_j}.

On this basis, we obtain the matching representation vector of the question sentence S_{Q_i} by the following formula:

$$r_{Q_i-A_j} = \left(H_{Q_i}\right)^T \cdot \text{softmax}\left(W_r \cdot \text{Matching}\left(Q_i, A_j\right)\right). \quad (3)$$

4.2. The 4th Dimension Representation. Formally, after obtaining all sentence-sentence matching vectors, we concatenate them into a new matrix $M = [\mathbf{r}_{\mathbf{Q_1-A_1}}, \mathbf{r}_{\mathbf{Q_1-A_2}}, \ldots, \mathbf{r}_{\mathbf{Q_N-A_M}}]$. Furthermore, we obtain the correlation degree vector α, in which each value represents the correlation degree of the matching information between the question and answer sentence with the annotated aspect category as follows:

$$\begin{aligned} T &= \tanh\left(W_m \cdot M^T \cdot M + b_m\right), \\ \alpha &= \text{softmax}\left(W_t^T \cdot T\right). \end{aligned} \quad (4)$$

Finally, we can get the 4th dimension representation r, which finally transforms the QA interaction-level representation into the hyperinteraction-level representation, as shown in Figure 5.

$$r = s \cdot \alpha^T. \quad (5)$$

4.3. Classification Model. The hyperinteraction vector $\mathbf{r}$ is a high-level representation for the question text in a QA-style review, and it is concatenated into the sentence-level representation $\mathbf{h_q}$ of the unsplit question text as the final

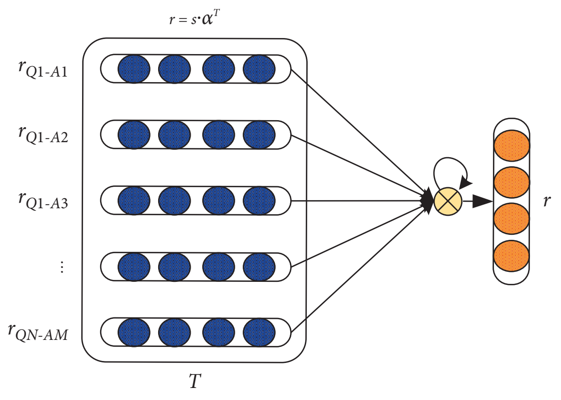


Figure 5: Learning hyperinteraction-level representation.

representation for the classification, where $\mathbf{h_q}$ is the last hidden vector obtained by the bi-GRU mode:

$$h_q^* = \tanh\left(W_p r + W_s h_q\right). \quad (6)$$

Finally, we put the final representation into the softmax layer to compute the conditional probability distribution. Then, the label with the highest probability is the predicted aspect category of a QA-style review:

$$y = \text{softmax}\left(W_l h_q + b_l\right). \quad (7)$$

To learn the whole model, we train our model end-to-end given the training data, and the goal of training is to minimize the cross-entropy loss function:

$$J(\theta) = -\sum_{t=1}^{K}\sum_{k=1}^{C_i} y_t^k \cdot \log\left(S_{Q_t}, S_{A_t}\right) + \frac{l}{2}\|\theta\|_2^2, \quad (8)$$

where K is the number of training data and C is the number of all aspect categories. Besides, $\varnothing(S_{Q_t}, S_{A_t})$ is a black-box function whose output represents a vector representing the probability of aspects, and l is a L_2-regularization term.

5. Experiments

In this section, we first introduce our models and baselines in Section 5.1. Then, we describe experimental settings, i.e., the datasets and evaluation metrics used in our experiment in Section 5.2. Finally, we describe details of the model setup in Section 5.3.

5.1. Model Summary. Since the word-level and sentence-level dimension representation have been widely studied in the natural language processing (NLP) area, we use approaches based on the two dimensions as our baseline methods.

For thorough comparison, we implement several different models for representing the question text in a QA-style review, which are illustrated in the following (the baselines and our proposed models are distinguished with the marker ∘ and ∗, respectively):

(i) A(2D) ∘: this approach employs the word-level and sentence-level dimension representations for answer texts

(ii) Q(2D) ∘: this approach employs the word-level and sentence-level dimension representations for question texts

(iii) Q + A(2D) ∘: this approach employs the word-level and sentence-level dimension representations for the concatenation of question and answer texts

(iv) Q(3D) ∗: this approach employs the word-level, sentence-level, and QA interaction-level dimensions for question texts

(v) Q(4D) ∗: this approach employs the word-level, sentence-level, QA interaction-level, and hyperinteraction-level dimensions for question texts

5.2. Experimental Settings

(i) Datasets: we conduct our experiments on the high-quality annotated corpus composed of QA-style reviews from *Digit*, *Beauty*, and *Bag* domains in Taobao, which conform to our annotation guidelines. Considering the imbalanced distribution of data, we eliminate aspect categories which contain less than 50 reviews. The statistics of the experimental datasets are summarized in Table 2.

(ii) Evaluation metrics: we use the standard accuracy and Macro-F1(F) to evaluate the overall QA aspect classification performance, and Macro-F1 is calculated by the formula $F = 2PR/(P + R)$, where the overall precision P and recall R are the average value of the precision/recall of all aspect categories.

5.3. Model Configuration

(i) Data processing: in order to construct the 4-dimension representation model, we present the sentence segmentation Algorithm 1 based on the Standford CoreNLP toolkit [20] to segment the question and answer text into sentences. Besides, we use word embedding to initialize the words in our datasets and pretrain the word embeddings with our crawled dataset containing 320 thousand QA-style reviews.

(ii) Model setup: in the experiment, we initialize all the out-of-vocabulary words by sampling from the uniform distribution $U(-0.01, 0.01)$. The word-embedding dimension and LSTM hidden state dimension are set to be 100, and the batch size is set to be 32. Considering that QA-style reviews are short texts, the minimum number of words in a sentence $N_{\min}$ is 5. The other parameters are tuned according to the development data. In the training process, we use the gradient descent approach to train our models, and the learning rate is 0.02, and the dropout rate is set to be 0.25 to avoid overfitting. Besides, the optimal number of sentences in the question or answer texts is tuned to be 2.

5.4. Research Question. As described in Section 4, our 4-dimension textual representation model for the QA aspect classification task is based on QA interaction-level and hyperinteraction-level mechanisms, which are built on sentence level. Thus, the impact on performance of the sentence number inside the question and answer texts is the key research question. Accordingly, we will discuss and analyze in the following section.

6. Results and Analysis

In Section 6.1, we compare the classification performance of our proposed approaches and other baselines on the datasets from three domains. Then, Section 6.2 analyzes the influence on the classification performance of the sentence number in the question and answer texts. Besides, the error analysis of misclassified QA-style reviews is illustrated in Section 6.3.

6.1. Performance Comparison. We adopt the holdout method to compare the performance of the approaches described in Section 4.1. In the holdout method, for each dataset from one domain, we set aside 10% from the training data as the development data by stratified sampling to tune learning algorithm parameters. Figures 6 and 7 give the experimental results of all discussed approaches.

From the results, we can see that

(1) In three domains, all Q(2D) approaches are fairly superior to A(2D) approaches, which demonstrates that question texts contain more aspect-related information than answer texts and contribute more in identifying the aspect category of a given QA-style review.

(2) Clearly, among all 2D approaches, when the question and answer texts are both employed in our task, Q + A(2D) approaches perform best. This means that we can utilize auxiliary information contained in answer texts to further improve the performance of the aspect category classification task on QA-style reviews.

(3) In the *Digit* domain, our proposed approach Q(3D) based on the QA interaction-level dimension achieves a definite improvement of 7.5% (accuracy) and 9.5% (Macro-F1) compared with Q(2D) approach, which indicates the importance of capturing the matching information between the sentence inside the question text and the sentence inside the answer text. Furthermore, in *Beauty* and *Bag* domains, the accuracy and Macro-F1 are both increased, but the increase rate is not significant.

(4) Note that our 4-dimension textual representation approach Q(4D) using both QA interaction-level and hyperinteraction-level dimensions achieves the best performance among all approaches. In the *Digit* domain, the accuracy and Macro-F1 of Q(4D) approach are 1.3% and 1.2% higher than Q(3D) approach which only employ the QA interaction-level dimension, respectively. In the *Beauty* domain,

Table 2: Statistics of experimental datasets (the QA-style reviews not in accordance with our annotation guidelines are excluded).

Domains	Digit	Beauty	Bag
Aspect categories	7	10	11
Total QA-style reviews	2,566	3,065	3,077
Maximal number of question sentences	6	5	4
Proportion of one question sentence and one answer sentence (%)	26.4	38.0	28.0
Maximal length of question texts	71	48	42
Aspect containing most QA-style reviews	IO	Effect	Quality
Number of most QA-style reviews in one aspect	1,044	911	868

```
     **Input**: Question or Answer text $S = \{w_i \mid w_i \text{is a word}\}$;
            $N_{\min}$: the minimum number of words in a sentence;
            $N_{\max}$: the maximum number of sentences in the answer text
     **Output**: All sentences (Stored in $C = \{c_i\}$) mined from $S$ that satisfy $N_{\min}$ and $N_{\max}$
 (1) $C = \varnothing$
 (2) $C_{\text{temp}}$ = null // the candidate sentence
 (3) Segment $S$ into $n$ sentences $\{\sigma_1, \sigma_2, \ldots, \sigma_v\}$ with Stanford CoreNLP toolkit;
 (4) **for** $i = 1; i \leq |S| - 1; i+ = 1$ **do**
 (5)   **if** $|C| \geq N_{\max}$ **then**
 (6)     **break**;
 (7)   **end**
 (8)   **if** $s_i$.length $> N_{\min}$ **then**
 (9)     $C_{\text{temp}} = s_i$;
(10)     $C = C \cup C_{\text{temp}}$;
(11)   **else**
(12)     $j = i + 1$;
(13)   **while** $j \leq |S_A| - 1$ **do**
(14)     $C_{\text{temp}} = s_i + s_j$;
(15)       **if** $C_{\text{temp}}$.length $\geq s_j$ **then**
(16)         $C = C \cup C_{\text{temp}}$;
(17)       **else**
(18)         $j += l$;
(19)       **end**
(20)     **end**
(21)     $i = i + j$;
(22)   **end**
(23) **End**
```

Algorithm 1: Sentence segmentation algorithm.

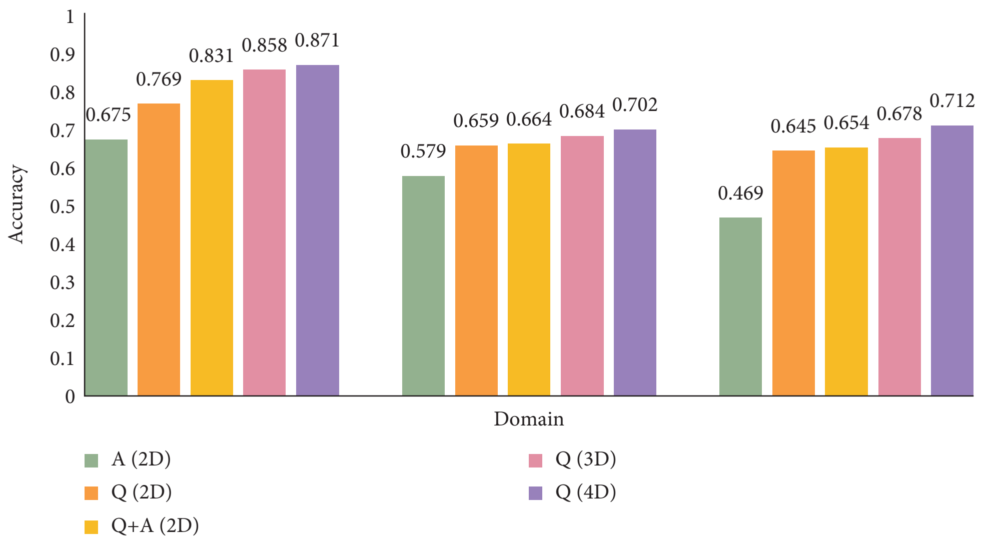


Figure 6: The comparison of accuracy in three domains.

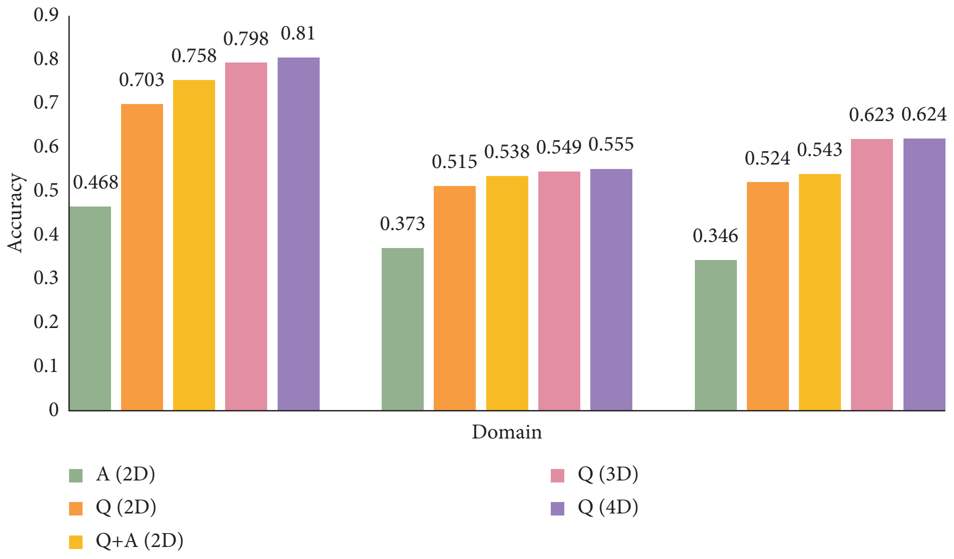


Figure 7: The comparison of F1 value in three domains.

Q(4D) approach achieves the improvement of 1.7% (accuracy) and 0.5% (Macro-F1), and in the *Bag* domain, the improvement of accuracy and Macro-F1 is 3.4% and 0.1%, respectively. This demonstrates that our proposed approach Q(4D) using the hyperinteraction-level dimension can effectively capture the importance degree of the question sentence and its aspect-related matching answer sentence for our task and can further improve the performance of the classification task. Furthermore, significance test, *t*-test, shows that this improvement is significant (p value <0.05).

(5) In our previous paper, the accuracy of the aspect category classification is 0.865 in the *Digit* domain. We can see that the performance of the multi-attention model is worse than our 4-dimension textual representation model.

On the basis of the above analysis, our proposals can be applied to QA-style reviews in three domains and improve the classification performance compared with the traditional sentence-level textual representation method, especially in the *Digit* domain.

6.2. Impact of Sentence Numbers on the Question or Answer Text. To answer the research question in Section 5.4, we examine the performance of our 4-dimension textual representation model with various sentence numbers of question and answer texts ranging from one to four according to the statistics of experimental data in each domain. We present results on classification performance in Tables 3–5 on QA-style reviews from *Digit*, *Beauty*, and *Bag* domains, in which the columns represent the number of question sentences, and the rows represent the number of answer sentences.

Clearly, for three datasets, we can find that, under the circumstances that the number of question sentences is not equal to the number of answer sentences, when the number of question sentences is fixed, the accuracy and Macro-F1 are both improved with the increase in the number of answer sentences. However, when the number of answer sentences is fixed, the classification performance becomes worse with the increase in the number of answer sentences, even worse than the traditional Q(2D) representation approaches which employs word-level and sentence-level representations. This would be that our classification task mainly depends on question texts, while the answer texts can assist question texts to further improve the performance of the aspect category classification task on QA-style reviews.

When they are equal in number, our model can achieve a tradeoff performance improvement. In particular, when the number of sentences in the question and answer texts is equal to 2, our proposals achieve the best performance in *Digit*, *Beauty*, and *Bag* domains, especially in the *Digit* domain. Besides, the number of QA-style reviews confirming to our annotation guidelines is not very large, and in *Beauty* and *Bag* domains, there are 10 and 11 aspect categories, respectively, which are both more than those in the *Digit* domain. Thus, the performance improvement in the *Beauty* or *Bag* domain is less significant than that in the *Digit* domain.

In addition, because QA-style reviews from the Taobao website are short texts, the number of sentences inside question or answer texts used in our 4-dimension textual representation model is not the more, the better. Particularly for question texts, as shown in Table 1, the maximal length of question texts in three domains is 71. If we segment them into more sentences, it could be counterproductive.

According to the above analysis, considering that QA-style reviews from other e-commerce platforms are similar to our corpus in format and expression, our proposals could be applied to the aspect category classification task on QA-style reviews on these QA-style reviews.

Table 3: Accuracy and Macro-F1 on the aspect category classification task on QA-style reviews with various sentence numbers in the *Digit* domain.

	Question							
Answer	1		2		3		4	
	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
1	—	—	0.783	0.680	0.752	0.641	0.742	0.611
2	0.792	0.727	**0.871**	**0.810**	0.781	0.676	0.744	0.623
3	0.816	0.753	0.785	0.698	**0.806**	**0.682**	0.767	0.655
4	0.818	0.755	0.808	0.704	0.800	0.692	**0.755**	**0.677**

Table 4: Accuracy and Macro-F1 on the aspect category classification task on QA-style reviews with various sentence numbers in the *Beauty* domain.

	Question							
Answer	1		2		3		4	
	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
1	—	—	0.636	0.504	0.634	0.498	0.612	0.492
2	0.677	0.535	**0.702**	**0.555**	0.650	0.513	0.629	0.511
3	0.681	0.544	0.645	0.515	**0.664**	**0.538**	0.648	0.516
4	0.683	0.542	0.662	0.553	0.652	0.517	**0.654**	**0.533**

Table 5: Accuracy and Macro-F1 on the aspect category classification task on QA-style reviews with various sentence numbers in the *Bag* domain.

	Question							
Answer	1		2		3		4	
	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
1	—	—	0.654	0.562	0.648	0.558	0.631	0.551
2	0.664	0.603	**0.712**	**0.624**	0.659	0.560	0.641	0.558
3	0.689	0.606	0.668	0.591	**0.666**	**0.598**	0.659	0.577
4	0.701	0.619	0.671	0.594	0.664	0.574	**0.661**	**0.583**

6.3. Error Analysis. According to the analysis of misclassified QA-style reviews in *Digit*, *Beauty* and *Bag* domains, we find some main reasons for misclassification as follows.

First, the imbalanced distribution of experiment data may lead to that these QA-style reviews tend to be predicted as the aspects which contain more QA-style reviews. For instance, in the *Digit* domain, the predicted aspect of 22.95% of misclassification QA-style reviews is IO. Similarly, in the *Beauty* domain is Effect and in *Bag* is Quality.

Second, the QA-style reviews are colloquial, so the existing word segmentation toolkits may not segment words well, which can influence the pretraining of word embeddings and the word-level textual representation.

Third, some annotated aspect terms are ambiguous in different contexts. We can consciously determine the meaning of these words according to the context and categorize them into the correct aspect category, which is still difficult for a well-trained machine at present.

Fourth, QA-style reviews used in our experiments are short texts, and our classification task mainly depends on question texts. However, as shown in Table 1, the maximal length of question texts is 71 in the *Digit* domain, while in the other two domains, its value is 42. After sentence segmentation, the inputs for the neural network are much shorter. This may lead to that our model could not be trained very well.

Last but not the least, sentence segmentation algorithm based on the Stanford CoreNLP toolkit is not good enough for QA-style reviews because it may ignore the syntax between the sentences in the question or answer text.

7. Conclusions

In this paper, we address a novel aspect category classification task against QA-style reviews, which aims at automatically classifying the aspect category of a given QA-style review and builds a high-quality annotated corpus. To solve this task, we propose a 4-dimension textual representation model based on QA interaction-level and hyperinteraction-level dimensions to capture the aspect-related matching information between the question and answer texts as much as possible.

Our experiment results on three manually annotated datasets, i.e., *Digit*, *Beauty*, and *Bag* datasets, demonstrate that our proposed approaches significantly outperform the baseline approaches, i.e., the textual representation based on sentence-level and word-level dimensions. For our proposed approaches Q(3D) and Q(4D), Q(4D) clearly performs better than Q(3D). In detail, Q(4D) presents an improvement ranging from 3.7% to 5.8% in terms of accuracy against the best baseline, i.e., Q + A(2D).

In our future work, we would like to solve some challenges in the aspect category classification task on QA-style reviews according to the error analysis, such as short question texts, imbalanced data distribution, and syntax parsing, to further improve the performance of this task. Furthermore, we would like to combine char embeddings with word embeddings to better represent the colloquial reviews.

Data Availability

Training data used in our experiment are mainly from "Asking All" in Taobao. The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

The funding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported in part by Industrial Prospective Project of Jiangsu Technology Department under Grant no. BE2017081 and the National Natural Science Foundation of China under Grant no. 61572129.

References

[1] T. Mikolov, M. Karafiat, L. Burget, J. Cemocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of the Eleventh Annual Conference of the International Speech Communication Association*, Chiba, Japan, 2010.

[2] Y. Wang, M. Huang, L. Zhao et al., "Attention-based Istm for aspect-level sentiment classification," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 606–615, Austin, TX, USA, 2016.

[3] B. Liu, "Sentiment analysis and opinion mining," *Synthesis Lectures on Human Language Technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[4] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent twitter sentiment classification," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics*, pp. 151–160, Portland, OR, USA, 2011.

[5] T. Alvarez-López, J. Juncal-Martinez, M. Fernández-Gavilanes et al., "Gti at semeval-2016 task 5: svm and crf for aspect detection and unsupervised aspect based sentiment analysis," in *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pp. 306–311, San Diego, CA, USA, 2016.

[6] J. Machacek, "BUTknot at SemEval-2016 Task 5: supervised machine learning with term substitution approach in aspect category detection," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pp. 301–305, San Diego, CA, USA, 2016.

[7] S. Kiritchenko, X. Zhu, C. Cherry, and S. M. Mohammad, "NRCCananda-2014: detecting aspects and sentiment in customer reviews," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pp. 437–442, Dublin, Ireland, 2014.

[8] Z. Su, H. Xu, D. Zhang, and Y. Xu, "Chinese sentiment classification using a neural network tool—word2vec," in *Proceedings of the 2014 International Conference on Multi-sensor Fusion and Information Integration for Intelligent Systems (MFI)*, IEEE, Beijing, China, pp. 1–6, 2014.

[9] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar, 2014.

[10] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," 2014, https://arxiv.org/abs/1404.2188.

[11] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, pp. 1422–1432, Lisbon, Portugal, September 2015.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] N. Vaswani, N. Shazeer, J. Parmar et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[14] Z. Yang, D. Yang, C. Dyer, X. He et al., "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, 2016.

[15] Z. Toh and J. Su, "NLANGP: supervised machine learning system for aspect category classification and opinion target extraction," in *Proceedings of the International Workshop on Semantic Evaluation*, pp. 496–501, Denver, CO, USA, 2015.

[16] W. Xue, W. Zhou, T. Li, and Q. Wang, "MTNA: a neural multi-task model for aspect category classification and aspect term extraction on restaurant reviews," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, Asian Federation of Natural Language Processing, Taipei, Taiwan, pp. 151–156, 2017.

[17] X. Zhou, X. Wan, and J. Xiao, "Representation learning for aspect category detection in online reviews," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, TX, USA, 2015.

[18] T. Mikolov, I. Sutskever, K. Chen et al., "Distributed representations of words andphrases and their compositionality," *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[19] B. Dhingra, H. Liu, Z. Yang, W. Cohen, and R. Salakhutdinov, "Gated-attention readers for text comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1832–1846, Vancouver, Canada, 2017.

[20] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, Baltimore, MD, USA, 2014.

Hindawi
Scientific Programming
Volume 2020, Article ID 3967847, 9 pages
https://doi.org/10.1155/2020/3967847

Research Article

A Task Scheduling Strategy in Edge-Cloud Collaborative Scenario Based on Deadline

Shudong Wang,[1] Yanqing Li,[1] Shanchen Pang,[1] Qinghua Lu,[2] Shuyu Wang,[1] and Jianli Zhao[3]

[1]*College of Computer Science and Technology, China University of Petroleum, Qingdao 266000, China*
[2]*Data61, Eveleigh, NSW, Australia*
[3]*College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266000, China*

Correspondence should be addressed to Shanchen Pang; pangsc@upc.edu.cn

Received 24 October 2019; Accepted 27 December 2019; Published 18 March 2020

Guest Editor: Jinghui Zhang

Task scheduling plays a critical role in the performance of the edge-cloud collaborative. Whether the task is executed in the cloud and how it is scheduled in the cloud is an important issue. On the basis of satisfying the delay, this paper will schedule tasks on edge devices or cloud and present a task scheduling algorithm for tasks that need to be transferred to the cloud based on the catastrophic genetic algorithm (CGA) to achieve global optimum. The algorithm quantifies the total task completion time and the penalty factor as a fitness function. By improving the roulette selection strategy, optimizing mutation and crossover operator, and introducing cataclysm strategy, the search scope is expanded. Furthermore, the premature problem of the evolutionary algorithm is effectively alleviated. The experimental results show that the algorithm can address the optimal local issue while significantly shortening the task completion time on the basis of satisfying tasks delays.

1. Introduction

With the rise of edge computing, the convergence of cloud computing and edge computing has become a major focus [1–3]. Especially when we make great strides towards the digital era of the Internet of Everything, edge-cloud collaboration has become an important application in many scenes such as CDN, industrial Internet, energy, intelligent transportation, and security monitoring. Cloud computing and edge computing need to work closely together to better match the various demand scenarios, thus maximizing the value of edge computing and cloud computing collaboration. Take the example of an IoT scenario. The devices in the Internet of Things generate a large amount of data, and the data are uploaded to the cloud for processing, which will cause great pressure on the cloud. To share the pressure of the central cloud node, the edge computing node can be responsible for data calculation and storage within its own scope [4–6]. Cloud computing excels in global, non-real-time, long-cycle big data processing and analysis and can play an advantage in long-term maintenance, business decision support, etc. Edge computing is more suitable for local, real-time, short-cycle data processing and analysis. Edge computing can better support real-time intelligent decision making and execution of local business. There are some high real-time performance applications, such as industrial system detection applications, control applications, executive applications, and emerging VR/AR applications. Some scenarios require real-time performance within 10 ms or even lower [7, 8]. If data analysis and processing are all implemented in the cloud, it is sometimes difficult to meet the real-time requirements of the service. It seriously affects the business experience of end customers. But, usually more studies usually consider the process of unloading, ignoring the assignment of tasks after unloading.

Tasks can be scheduled to the edge or the far cloud based on energy consumption and time delay. For the problem that needs to be processed in the cloud center, how to perform proper scheduling to achieve the goal is worthwhile research question.

Task scheduling methods in the cloud center can be divided into heuristic algorithms (such as RR and SJF), metaheuristic algorithms (based on biological incentives and swarm intelligence), and hybrid task scheduling algorithms [9]. In the scheduling process, various performance-based performance indicators such as system utilization, execution time, load balance, network communication cost, delay, and the like are used [10]. The heuristic task scheduling algorithm can easily schedule tasks and provide the best solution. However, it does not guarantee the best results and is easy to fall into partial selection. The metaheuristic algorithm is an improved algorithm based on a heuristic algorithm, which is a combination of random algorithms and local search algorithm[11–13]. It enables the exploration and development of search space and handles a large amount of search space information. In addition, it can use learning strategies to acquire and master information to effectively find approximate optimal solutions. Among them, genetic algorithm (GA), particle swarm optimization (PSO), and ant colony algorithm (ACO) are the most widely used evolutionary algorithms in the task scheduling in recent years [14]. However, these algorithms usually converge prematurely and are prone to finite optimally. When approaching the optimal solution, it may also swing left and right, making the convergence slower [15]. In genetic algorithms, the crossover operators become the main operators because of its global search ability and mutation operator is to become the auxiliary operator because of its local search ability. Genetic algorithms have the ability to balance the global search space with the local search space. Genetic algorithms always search for global and local spaces through crossover and mutation operators. They cooperate with each other and monitor each other. How to effectively cooperate with the intersection and mutation operations, make the convergence faster, and jump out of the local optimum in the solution process is a valuable research content of the current genetic algorithm.

This paper proposes a task scheduling strategy for edge-cloud collaborative computing based on disaster genetic algorithm. Considering the meaning of the cross operation, the individual optimal retention, and the magnitude of the mutation probability in the evolutionary process, the ability to optimize convergence and the three genetic operators of the genetic algorithm are improved. A penalty factor determines the execution time objective function based on the time delay. At the same time, a catastrophic strategy was introduced to simulate the phenomenon of disasters in biological evolution. During the first 1/2 iterations, premature aging may occur and the best chromosomes of successive generations will not develop at all. Therefore, we increase the probability of mutation, break the monopoly of the original gene, make the individual away from the current optimal solution into the group, increase the diversity of genes, and create new survival individuals. The algorithm we proposed can jump out of the local optimum and effectively alleviate the problem of premature convergence.

The rest of this article is organized as follows. Section 2 introduces the related work. Section 3 introduces the task classification strategy. Section 4 introduces the task scheduling model in the cloud center. Section 5 introduces the CGA algorithm. Section 6 introduces the experimental and comparison results. Finally, Section 7 summarizes this paper.

2. Relevant Work

Research on edge-cloud collaboration is still in the initial stage, but many domestic and foreign scholars have carried out related research and achieved research results on the task scheduling problem at the edge or cloud. Ke et al. [16] proposed classifying tasks according to whether they meet the delay and energy consumption. In the scheduling of tasks in the cloud, genetic algorithms are widely studied for their adaptability to various task scheduling problems. The genetic algorithm is appropriate for various task scheduling problems. The improvement of the genetic algorithm is mainly to improve the genetic operator and to achieve the purpose of improving the convergence speed and the performance of the classical genetic algorithm. At present, many corking algorithms have been proposed successively after experimentation and demonstration by scholars. Keshanchi et al. [17] proposed an improved heuristic-based genetic algorithm, called N-GA. The N-GA is used for the static task scheduling in the cloud. Akbari et al. [18] improved the performance of genetic algorithm by significantly changing genetic operators to ensure the sample diversity and reliable coverage of the entire space. In [19], a hybrid metaheuristic algorithm is offered, which uses the HEFT (Heterogeneous Earliest Completion Time) algorithm combined with PSO and GA to improve performance. Johnson proposed a rule-based genetic algorithm (JRGA) [20] for a two-stage task scheduling in data centers. In [11], the authors proposed a task scheduling scheme for heterogeneous computing systems built on a genetic algorithm, which maps each task to the processor according to the assigned priority to shorten the manufacturing time as much as possible. Goyal and Agrawal [21] proposed a model for scheduling a group of independent tasks on multiple machines and solved the question by combined the GA and the electoral heuristic algorithm. The goal of this model is also intended to minimize the maximum time. Kumar et al. [22] put forward a new task scheduling method, which integrated min-min algorithm and min-max algorithm in a genetic algorithm. The goal of the research is to shorten the generation time and execution time to the greatest extent [23].

However, the methods mentioned above may still fall into a local optimum when solving a multimode problem [10]. Therefore, the algorithm needs some strategies to avoid this limitation. Literature [24–28] mentioned an integer genetic algorithm using a “catastrophe” operator. It is designed to help to jump out of the local extreme points. The bionic significance of “catastrophe” operator and the improvement of disaster genetic algorithm in solving the above problems are emphatically introduced. These operations can mitigate the phenomenon of falling into a local optimum and premature convergence.

In addition, there are few studies that achieve the least total time based on the delay of meeting each task. Therefore, based on the research of genetic algorithms, this paper raises a task scheduling algorithm called CGA based on cataclysm strategy [29], which mainly considers the time delay to achieve the minimum total execution time. And the effectiveness of the proposed algorithm is checked by experiments.

3. Task Classification

In the system, we consider a set of tasks to be performed, each of which comes from an edge device which is denoted as $N = \{1, 2, 3, \ldots, N\}$. The tasks include interactive gaming, natural language processing, image location, etc [16]. Each task should be completed within the deadline. Each task with three attributes is defined as $\text{Task}_i = [\text{data}_i, d_i, \exp T_i], i \in N$. For Task_i, d_i is the size of the input data for the computation, which may include program codes, input files, etc [16].$\exp T_i$ is the deadline for completion of a task. data_i is the length of the task. Therefore, we must first classify the tasks that need to be processed to determine whether to execute in the cloud. According to the ratio of the delay of the task and the length of the task, the sensitivity of the task is determined. And finally, the tasks in the cloud will be scheduled to reduce total execution time.

Let f_i^c represent the computing power assigned to the Task_i by the edge device. Thus, we can get the time of the local execution of Task_i as

$$T_{\text{local}}^i = \frac{\text{data}_i}{f_l^c}. \tag{1}$$

The time transferred to the cloud is defined as

$$T_{\text{tran}}^i = \frac{d_i}{\text{Rate}}. \tag{2}$$

Rate is the upload rate of tasks transferred to the cloud; here the upload rate is a fixed value.

In order to facilitate subsequent task scheduling in the cloud, tasks need to be sorted according to sensitivity. The task sensitivity can be defined as

$$\text{sen}T_i = \frac{\text{data}_i}{\exp T_i}. \tag{3}$$

The complete task classification process is illustrated in Algorithm 1.

4. Task Scheduling Model in the Cloud Center

The task scheduling problem in the cloud is how to reasonably arrange each task to multiple virtual machines so that all tasks can be completed in a shorter execution time and meet the delay as much as possible [10]. Here, the following assumptions are made:

(1) There is no interdependence between tasks and tasks
(2) The size of the task and the computing speed of the virtual machine are known

Definition 1. Virtual machines on physical machines:

$$\text{PM}_i = [\text{VM}_1, \text{VM}_2, \ldots, \text{VM}_{\text{NVM}}], \tag{4}$$

where PM_i represents the host machine, Nvm represents the number of virtual machines, and VM_k represents the kth virtual machine resource in the cloud environment.

Definition 2. Virtual machine resources:

$$\text{VM}_K = [\text{IDV}_k, \text{MIPS}_k], \tag{5}$$

where IDV_k is the serial number of the virtual machine and MIPS_k represents the computing power of the kth virtual machine.

Definition 3. Task sequence:

$$T = [\text{TAsk}_1, \text{TAsk}_2, \ldots, \text{TAsk}_i, \ldots, \text{TAsk}_{\text{Ntsk}}], \quad i \in GC, \tag{6}$$

where Ntsk represents the number of tasks that need to be performed in the cloud and Task_i represents the ith task in the task sequence.

Definition 4. Task expected completion time.

The ECT matrix is used to represent the completion time of all tasks on each virtual machine resource.

$$\text{ECT} = \begin{bmatrix} \text{ECT}_{1,1} & \text{ECT}_{1,2} & \cdots & \text{ECT}_{1,\text{Nvm}} \\ \text{ECT}_{2,1} & \text{ECT}_{2,2} & \cdots & \text{ECT}_{2,\text{Nvm}} \\ \cdots & \cdots & \cdots & \cdots \\ \text{ECT}_{\text{Ntsk},1} & \text{ECT}_{\text{Ntsk},2} & \cdots & \text{ECT}_{\text{Ntsk},\text{Nvm}} \end{bmatrix}. \tag{7}$$

The execution time required for each task to run on a computing resource (virtual machine) is calculated as follows:

$$\text{ECT}_{i,k} = \frac{\text{data}_i}{\text{MIPS}_k}, \quad k = 1, 2, \ldots, \text{Nvm}; \; i = 1, 2, \ldots, \text{Ntsk}. \tag{8}$$

Let the task set be assigned to the kth virtual machine; then, the task completion time $\text{RT}(k)$ on the kth virtual machine is

$$RT(k) = \sum_{l \in N_l} \text{ECT}(l, k), \quad k = 1, 2, \ldots, \text{Nvm},$$
$$\forall l \in [1, \text{Ntsk}] \text{mapped to } k\text{th VM}, \quad k = 1, 2, 3, \ldots, \text{Nvm}. \tag{9}$$

AllNTime is the maximum completion time for each computing resource:

$$\text{AllNTime} = \max(\text{RT}(K)), \quad k = 1, 2, \ldots, \text{Nvm}. \tag{10}$$

Definition 5. Matching matrix A.

We can get the matrix A:

```
(1) Initialization
(2) Task set: N = $\{1, 2, 3, \ldots, N\}$;
(3) Categorized task sets: GC = GL = $\phi$;
(4) For each task $i \in$ N do
(5) Calculate sen$T_i$ by data$_i$/exp $T_i$, respectively;
(6) If ($T_{\text{local}} >$ exp $T_i$) then
(7)     $i \Longrightarrow$ GC;
(8) Else if ($T_{\text{local}} \le$ exp $T_i$) then
(9)     $i \Longrightarrow$ GL;
(10) End if;
(11) End for;
(12) Output: GC (Sort by sensitivity in ascending order), GL.
```

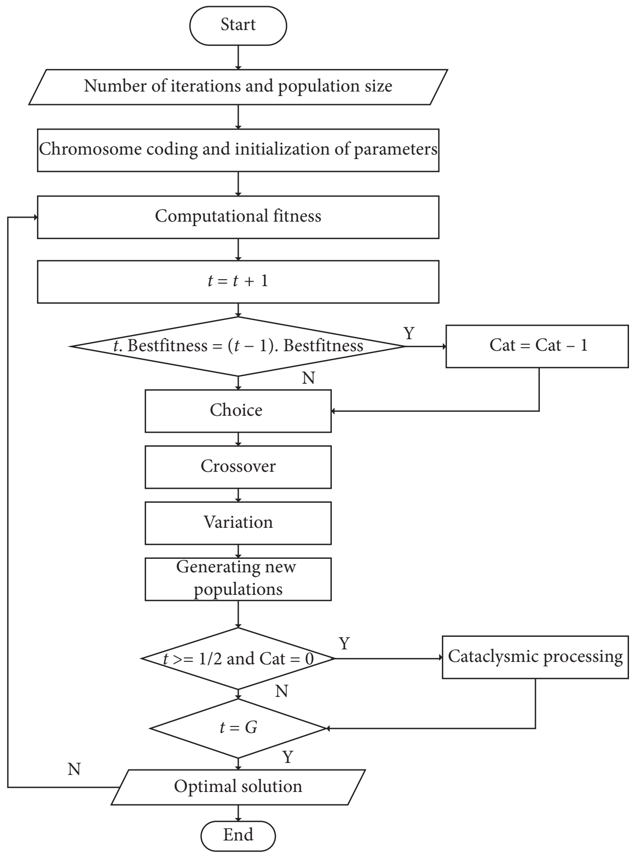
Algorithm 1: The algorithm for classifying the tasks.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,\text{Nvm}} \\ a_{2,1} & \cdots & \cdots & \cdots \\ \cdots & a_{i,j} & \cdots & \cdots \\ a_{\text{Ntsk},1} & \cdots & \cdots & a_{\text{Ntsk},\text{Nvm}} \end{bmatrix}. \tag{11}$$

Among them, $a_{i,j} = \{0, 1\}$. And the value of $a_{i,j}$ indicates whether the task numbered i is executed on the virtual machine numbered j, and if it is 1, it is executed.

5. CGA Algorithm

5.1. Algorithmic Thought. The three genetic operations of the genetic algorithm affect the convergence speed of the algorithm. This paper mainly considers satisfying the delay and minimizing the total execution time, and improves the selection operation and the crossover operation as well as the mutation operation of the genetic algorithm to generate a new generation of the population while simulating biological evolution in the iterative process. The catastrophic phenomenon in the process makes the algorithm increase individual diversity without expanding the population size, and it is easier to get rid of the optimal local trap. The algorithm flow chart is shown in Figure 1:



Start
Number of iterations and population size
Chromosome coding and initialization of parameters
Computational fitness
$t = t + 1$
t. Bestfitness = (t − 1). Bestfitness
Y
Cat = Cat − 1
N
Choice
Crossover
Variation
Generating new populations
t >= 1/2 and Cat = 0
Y
Cataclysmic processing
N
$t = G$
Y
N
Optimal solution
End

Figure 1: Algorithmic flow.

5.2. Basic Operations of the Algorithm

5.2.1. Encoding. In cloud computing scheduling problem, the encoding of solutions usually uses binary coded and real coded, where real coded is multi-to-one mapping pairing encoding. The task of this paper and the virtual machine are coded by the mapping pairing method [2]. For example, if there are M vms, that is, $\{v_1, v_2, v_3, \ldots, v_M\}$, and N tasks, that is, $\{\text{Tak}_1, \text{Task}_2, \text{Task}_3, \ldots, \text{Task}_n\}$, the length of the code will be N and the value of each gene will come from 1 to M, as shown in Figure 2:

Task	t_1	t_2	t_3	t_4	t_5	t_6
Vm	v_1	v_3	v_2	v_4	v_5	v_5
Code	1	3	2	4	5	5

Figure 2: Encoding.

5.2.2. Fitness Function. The fitness function represents the degree of an individual's fitness in the evolutionary process. The greater the fitness is, the easier it is to be retained in the evolutionary process. The fitness function will directly affect the performance of the algorithm and whether it can achieve the goal. In this paper, we need to consider the effect of time delay and execution time on individual fitness.

The difference between execution time and deadline for each task:

$$\sum_{i\in[1,i]} a_{i,j}\mathrm{ECT}_{i,j} + T^i_{\mathrm{tran}} - \exp T_i. \quad (12)$$

Penalty factor based on whether delay is satisfied:

$$\mathrm{punish} = \begin{cases} 0, & \text{if } \sum_{i\in[1,i]} a_{i,j}\mathrm{ECT}_{i,j} + T^i_{\mathrm{tran}} - \exp T_i \le 0, \\ \left|\sum_{i\in[1,i]} b a_{i,j}\mathrm{ECT}_{i,j} + T^i_{\mathrm{tran}} - \exp T_i\right|, & \text{if } \sum_{i\in[1,i]} a_{i,j}\mathrm{ECT}_{i,j} + T^i_{\mathrm{tran}} - \exp T_i > 0. \end{cases} \quad (13)$$

Because the goal is to minimize the total execution time of the task scheduling while meeting the deadline of tasks, the fitness function of this paper is designed as

$$\mathrm{fitness} = \frac{1}{\mathrm{AllNTime} + \sum_{i=1}^{\mathrm{Ntsk}} \mathrm{punish}}. \quad (14)$$

5.2.3. Improve Roulette Choice. The roulette selection method is also called the proportional selection method. The basic idea is that the larger the individual's adaptability is, the easier it is to be selected. The traditional roulette method can select the best individual, but it cannot guarantee that the best individual will remain to the next generation, and the subsequent crossover operation may destroy the best individual. Therefore, this paper combines roulette with the best individuals to save individuals with the greatest fitness in each generation directly to the next generation and does not participate in the crossover operation or mutation operation. The remaining individuals use traditional roulette to select the progeny population. The probability $Ps(j)$ of individual selection in traditional roulette is

$$Ps(j) = \frac{\mathrm{fitness}(j)}{\sum_{i=1}^{N} \mathrm{fitness}(i)}. \quad (15)$$

5.2.4. Crossover. The crossover operation of the traditional genetic algorithm is to select the number of individuals to cross according to the crossover rate, to generate a crossover operation for each of the intersecting individuals using the random function $\mathrm{rand}(1, n)$, and to map the two chromosomes to the segments after the location point are exchanged. Traditional crossover operations are prone to the situation of the high similarity of crossover fragments, at which time the crossover meaning becomes smaller. To this end, this paper sets a cross threshold, and only if the threshold is exceeded, the cross is considered meaningful. Otherwise, no crossover occurs. The threshold size represents the proportion of similar genes in the total gene. This operation is mainly based on the principle of preventing inbreeding and optimizing offspring in the process of human evolution. In this paper, we set the threshold to 0.8 and the crossover probability higher than 0.7 to avoid slowing down the speed of convergence rate caused by abandoning the cross operation because the similarity is too high. The specific crossover operation is shown in Figure 3:

5.2.5. Variation. A mutation operator is a very important operation. There are two purposes for introducing mutations into genetic algorithms: one is to make the genetic algorithm have local random search ability. When the genetic algorithm is close to the optimal solution neighborhood through the crossover operator, the local random search ability using the mutation operator can accelerate the convergence to the optimal solution [30]. In this case, the mutation probability should take a smaller value. The second is to enable the genetic algorithm to maintain group diversity to prevent immature convergence. At this time, the mutation probability should take a larger value. The probability of variation usually takes a small value and generally does not exceed 0.1.

In this paper, two variability values are set. When the number of iterations reaches 2/3, the mutation probability is reduced by 0.02. Determine the number of individuals that need to be mutated based on the probability of mutation, randomly select two locations on the chromosome, and exchange the values of the genes. The genic value may have not changed after the mutation operation was executed, which is equivalent to no mutation operation, and the variation operation is improved in order to ensure that the variation operation can be executed even if it is already a small probability event. If two genic values of mutation are the same, add the first random number to 1 and let it perform mutation operation with another gene point. If the first random number is still the same, increment the value by one until the value is different to ensure the mutation operation (see Figure 4).

5.2.6. Catastrophe. After many generations of evolution, the group may obtain a locally optimal solution. At this time, the group implies a large amount of information related to the local optimum, tending to premature convergence and the possibility of jumping out by operators such as crossover operation and mutation operation. It is possible to introduce "catastrophe" strategy, obtain some useful global information, and obtain a solution far away from the original locality with a large probability so that a larger diversity can be obtained at smaller group size. It can provide more opportunities to get rid of the original local optimal solution. However, the catastrophe cannot go through evolution all the time. We should consider avoiding the problem of destroying the optimal solution and reoptimizing in the later stage.

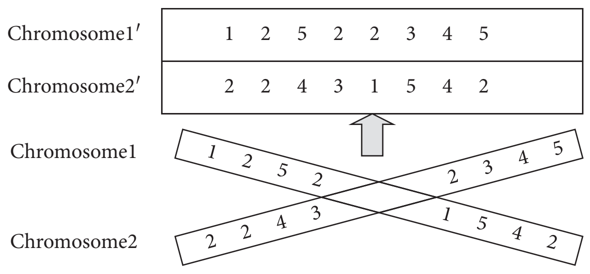


Figure 3: Crossover.

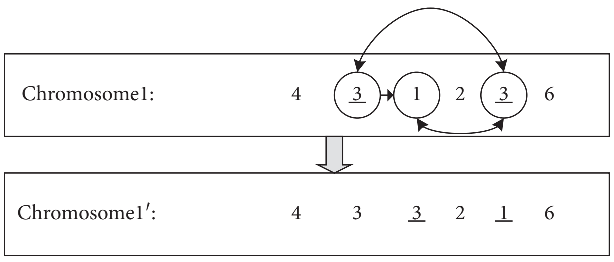


Figure 4: Variation.

The genetic algorithm has the disadvantages of easy to fall into local optimum and premature convergence [31]. Once it falls into local optimum, it will be difficult to jump out. For this reason, we add the catastrophic strategy mentioned in the literature [28] to this paper. By increasing the mutation probability to stay away from the current optimal, the solution that is far from the current optimal solution is included in the population to jump out of the optimal local solution. Catastrophic operation is shown in Algorithm 2.

5.3. Task Classification and Scheduling Description

Step 1: classify all tasks from different devices according to Algorithm 1.

Step 2: for tasks that need to be uninstalled to the cloud, sort by sensitivity. The initial coding is optimized according to the computing power of virtual machine.

Step 3: chromosome coding and initialization of parameters.

Step 4: calculate fitness.

Step 5: superposition algebras plus one.

Step 6: judge whether the optimal individual fitness of $(t-1)$th generation is equal to that of the tth generation, and if so, the catastrophe threshold is reduced by one; otherwise, it will continue.

Step 7: perform selection operation, cross operation, and mutation operation.

Step 8: generate the descendant population and determine whether the catastrophe threshold cat is equal to 0 (before t/2 iterations). If equal to 0, carry on the catastrophe operation.

Step 9: if the number of iterations reaches the maximum, output; otherwise, turn to step 4.

```
(1)  Input: Catastrophe threshold cat;
(2)    cat = $a$;
(3)    For ($t = 0$; $t < G/2$; $t$++)
(4)      {
(5)        If (t. Bestfitness = ($t$–1). Bestfitness)
(6)          {
(7)            cat = cat−−;
(8)          }
(9)        If (cat = 0)
(10)           The first third variation;
(11)       Else
(12)           Continue circulation;
(13)         }
```

Algorithm 2: Catastrophic operation.

6. Evaluation

In this experiment, for tasks that need to be processed in the cloud, we used CloudSim 3.0 to implement the algorithms, by adding the bindCloudletToVM method in the DAtacenterBroker class; the CGA algorithm based on the catastrophe genetic algorithm is added to carry out the simulation experiment. Data such as resource computing power and task calculations are derived from data randomly generated in MATLAB. We choose the different number of tasks, and the experimental data of different iteration times are analyzed and compared with the time-based differential evolution algorithm (TDE) and simple genetic algorithm under the same data conditions. The TDE algorithm is based on differential evolution (DE) task scheduling algorithm that minimizes the completion time. The differential evolution algorithm is also a population-based heuristic search algorithm. There is a great similarity between differential evolution algorithm and genetic algorithm. They all include mutation, crossover, and selection operations, but the specific definition of these operations is different from the genetic algorithm. The experimental results are shown in Figures 5–9.

Parameter setting: crossover probability crossover = 0.8, maximum evolution algebra = 200, and mutation probability is 0.03, and in order to avoid errors as much as possible, this paper will perform ten times for each group of experiments and finally get the total task completion time. The experimental values are taken as the average of ten experiments.

When the number of tasks is small, the optimal effect is not obvious. However, the optimization of the algorithm is more obvious when the number of tasks is large. But the more tasks there are, the fewer tasks that are unloaded into the cloud, because as the number of tasks increases, the task takes longer to execute. With the increase of evolutionary algebra, the proposed algorithm can converge more quickly and save more time. Figures 5 and 6 show the changes of total task execution time and adaptive value of CGA algorithm, classical genetic algorithm, and TDE algorithm under different iterations. It can be seen that the effect of the classical genetic algorithm is the worst. The CGA algorithm uses less evolutionary algebra than other algorithms to get

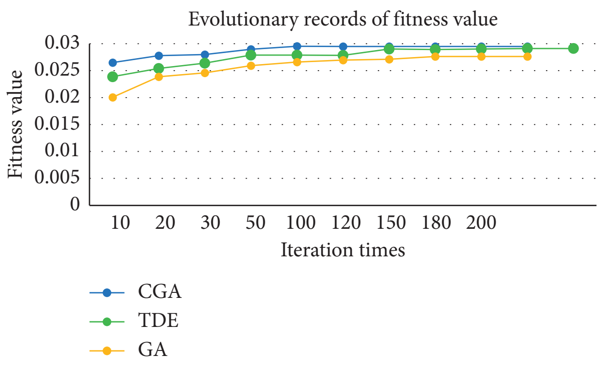


Figure 5: Evolution—200 iterations.

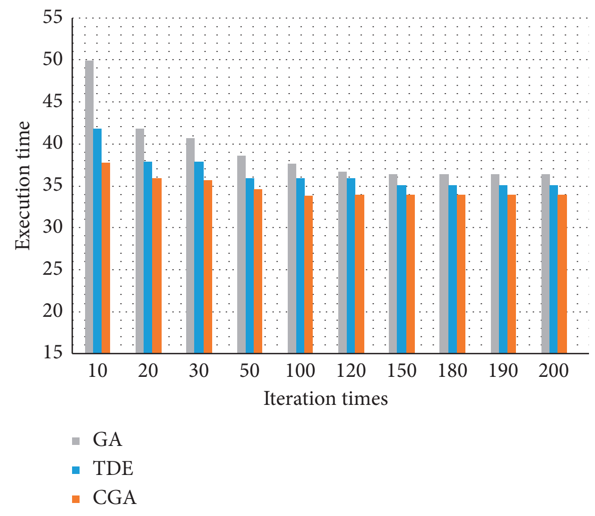


Figure 6: Execution time—50 tasks.

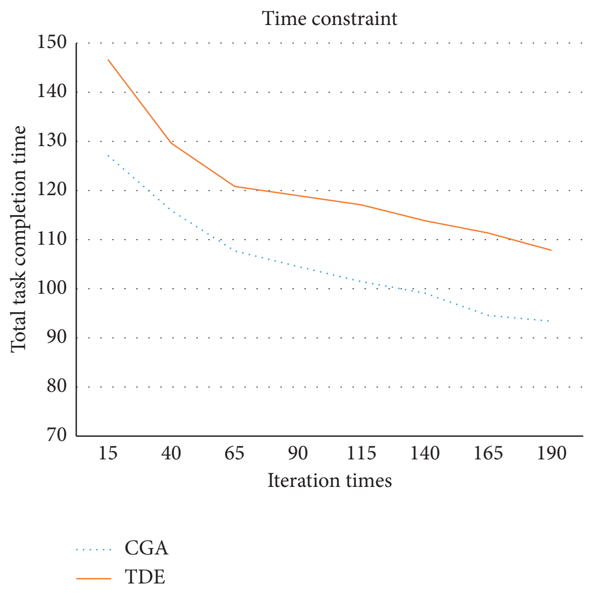


Figure 7: Execution time—200 iterations.

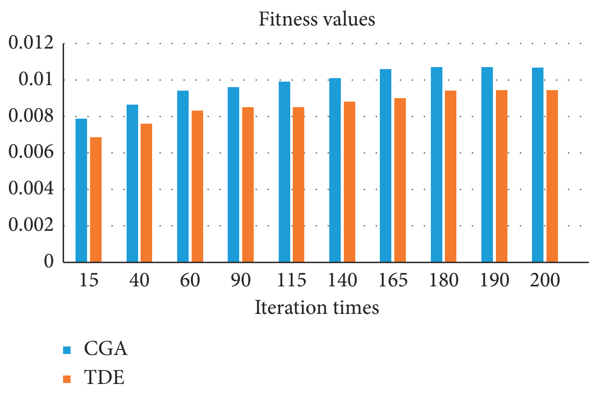


Figure 8: Evolution—200 iterations.

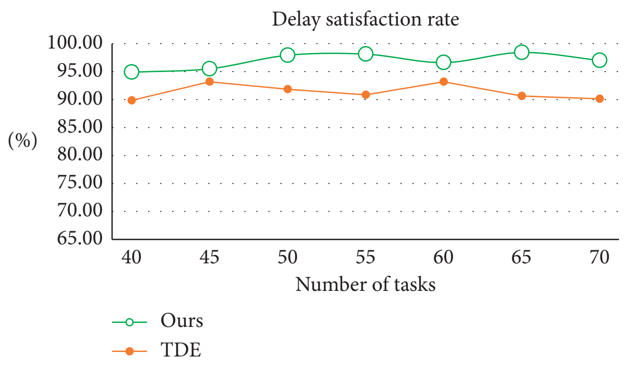


Figure 9: Delay satisfaction rate.

better average fitness. Among them, this paper also optimizes the initial population, and CGA algorithm can find the optimal solution faster. As we all know, the solution found by genetic algorithm may not be optimal, but the experimental results show that the CGA algorithm is better than the other two algorithms, and it is easier to jump out of the local optimum and find the optimal solution. Figures 7 and 8 are comparisons between CGA algorithm and TDE algorithm. We can see that CGA algorithm can achieve the goal of this paper better.

According to the experimental results, it can be seen from Figure 9 that the delay satisfaction rate of the experiment is above 95%, which can meet the demand. And the performance of CGA algorithm is better than the TDE algorithm. In addition, the CGA algorithm is superior to the TDE algorithm in the task completion time and convergence speed of the evolutionary process, and its convergence speed is significantly better than the TDE algorithm and the traditional genetic algorithm. As the number of iterations increases, the CGA algorithm can find the optimal solution better and make the convergence rate faster. The mutation strategy called cataclysm policy is designed to help the population jump out of the local extreme points [27]. It can be seen that the catastrophic strategy in this paper does not slow down the convergence rate and destroy the optimal direction. Instead, it can help the operation to continuously optimize the population and is not easy to fall into the local optimum.

7. Conclusion and Future Work

The task scheduling in edge-cloud collaborative scenario is considered to be one of the critical challenges. Whether the task is executed in the cloud and how it is scheduled in the cloud is an important issue. In the past, many heuristics and metaheuristic task scheduling strategies have been used in cloud computing or edge computing. Genetic algorithms have unique advantages that traditional methods do not have in solving complex problems such as big space, nonlinearity, and global optimization. They have been widely used in more and more fields. In this paper, we proposed a task scheduling strategy under deadline constraint, where tasks on edge devices could select the execution place including cloud and local devices. And the goal is to minimize the execution time of all tasks. The CGA algorithm as an alternative method to solve the task scheduling problem; this algorithm adds cataclysm strategy to it. We have considered the constraint of time [5] and optimized the task scheduling. The algorithm CGA was inspired by the behavior of the extinction in the Ice Age, and it is used as a global optimization algorithm [10].

The CGA algorithm we proposed was simulated in the CloudSim environment, and the main objective was to minimize the execution time and meet delay. The results are compared with the results of existing heuristic methods such as the traditional genetic algorithm (GA) and the time-based differential evolution algorithm (TDE). From the experimental results, we can also get the conclusion that the proposed CGA can efficiently schedule the tasks to the VM and achieve our goals.

In the future, we will consider improving the algorithm under conditions that are closer to the actual environment so that the algorithm can be applied to dynamic and real-time task scheduling in edge-cloud collaboration. Besides, we want to build a multi-objective version of CGA for optimizing the task scheduling problem in the cloud. Study of workflow scheduling using CGA is another future investigation. And we can also mine or forecast its potential relationships [32–34]. In addition, the method of task scheduling can consider many other parameters, such as the use of memory, peak of the demand, and overloads [10]. Besides, we can combine the Markov chain with the parallel computing framework and apply it in our model [35, 36].

Data Availability

Because this paper only deals with time and static tasks, we used randomly generated data to export it as a dataset for the length of tasks.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank International Networks Service and Bio-Computing Innovation Team from the college of Computer Science and Technology in China University of Petroleum (East China) for their discussion and technical support. This study was supported by the National Natural Science Foundation of China (nos. 61572522, 61873281, and 61572523).

References

[1] M. Satyanarayanan, "Edge computing," *Computer*, vol. 50, no. 10, pp. 36–38, 2017.

[2] Y. Mao, C. You, J. Zhang et al., "Mobile edge computing: survey and research outlook," 2017, http://arxiv.org/abs/1701.01090.

[3] Y. Yu, "Mobile edge computing towards 5G: vision, recent progress, and open challenges," *China Communications*, vol. 13, no. 2, pp. 89–99, 2017.

[4] G. Ananthanarayanan, P. Bahl, P. Bodik et al., "Real-time video analytics: the killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[5] J. Liu, Y. Mao, J. Zhang et al., "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT)*, Barcelona, Spain, July 2016.

[6] Z. Ke, Y. Mao, S. Leng et al., "Optimal delay constrained offloading for vehicular edge computing networks," in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017.

[7] Y. Yin, W. Xu, Y. Xu, Li He, and L. Yu, "Collaborative qos prediction for mobile service with data filtering and slopeone model," *Mobile Information Systems*, vol. 2017, Article ID 7356213, 14 pages, 2017.

[8] Y. Yin, F. Yu, Y. Xu, L. Yu, and J. Mu, "Network location-aware service recommendation with random walk in cyber-physical systems," *Sensors*, vol. 17, no. 9, p. 2059, 2017.

[9] T. Jena and J. R. Mohanty, "Disaster recovery services in intercloud using genetic algorithm load balancer," *International Journal of Electrical and Computer Engineering*, vol. 6, no. 4, p. 1828, 2016.

[10] M. A. Elaziz, S. Xiong, K. P. N. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowledge-Based Systems*, vol. 169, pp. 39–52, 2019.

[11] Y. Xu, J. K. Li, and K. Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues," *Information Sciences*, vol. 270, pp. 255–287, 2014.

[12] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[13] J.-F. Li and J. Peng, "Task scheduling algorithm based on improved genetic algorithm in cloud computing environment," *Jisuanji Yingyong/Journal of Computer Applications*, vol. 31, no. 1, p. 184186, 2011.

[14] J. Xue, L. Li, S. Zhao, and L. Jiao, "A study of task scheduling based on differential evolution algorithm in cloud computing," in *Proceedings of the International Conference on Computational Intelligence Communication Networks*, Bhopal, India, November 2014.

[15] C. Zhu and J. Ni, "Cloud model-based differential evolution algorithm for optimization problems," in *Proceedings of the Sixth International Conference on Internet Computing for Science Engineering*, Henan, China, April 2012.

[16] Z. Ke, Y. Mao, S. Leng et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, no. 99, pp. 5896–5907, 2016.

[17] B. Keshanchi, A. Souri, and N. J. Navimipour, “An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing,” *Journal of Systems and Software*, vol. 124, pp. 1–21, 2017.

[18] M. Akbari, R. Hassan, and S. H. Alizadeh, “An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems,” *Engineering Applications of Artificial Intelligence*, vol. 61, pp. 35–46, 2017.

[19] K. Amin and A. Ghaffari, “Hybrid task scheduling method for cloud computing by genetic and de algorithms,” *Wireless Personal Communications*, vol. 97, no. 4, pp. 6301–6323, 2017.

[20] Y. Xiong, S. Huang, M. Wu, J. She, and K. Jiang, “A johnson’-rule- based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing,” *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 597–610, 2017.

[21] T. Goyal and A. Agrawal, “Host scheduling algorithm using genetic algorithm in cloud computing environment,” *International Journal of Advances in Engineering & Technology*, vol. 1, no. 1, pp. 7–12, 2013.

[22] P. Kumar and A. Verma, “Independent task scheduling in cloud computing by improved genetic algorithm,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 5, pp. 111–114, 2012.

[23] H. Aziza and S. Krichen, “Bi-objective decision support system for task-scheduling based on genetic algorithm in cloud computing,” *Computing*, vol. 100, no. 2, pp. 65–91, 2018.

[24] L. Xin-Rong and G. Yang, “Application of catastrophic adaptive genetic algorithm to reactive power optimization of power system,” in *Proceedings of the 2010 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 2, pp. 450–454, IEEE, Sanya, China, October 2010.

[25] M. Wang, B. Li, Z. Wang, and X. Xie, “An optimization strategy for evolutionary testing based on cataclysm,” in *Proceedings of the Computer Software Applications Coirference Workshops*, Seoul, South Korea, July 2010.

[26] Z. X. Cai, “Application of grey theory in forecasting the diaphania pyloalis cataclysm,” *Science of Sericulture*, vol. 35, pp. 869–871, 2009.

[27] S. X. Lv, Yu R. Zeng, and L. Wang, “An effective fruit fly optimization algorithm with hybrid information exchange and its applications,” *International Journal of Machine Learning Cybernetics*, vol. 9, no. 10, p. 16231648, 2018.

[28] S. C. Xiao, S. D. Sun, and H. Guo, “Cataclysm genetic algorithm for solving vechicle scheduling problem with time windows,” *Application Research of Computers*, vol. 31, no. 12, 2014.

[29] Z. Wang and Y. Chen, “Binary decision diagram variable ordering based on catastrophe genetic algorithm,” *Computer Engineering Applications*, vol. 51, no. 3, pp. 55–60, 2015.

[30] A. W. Mohamed, H. Z. Sabry, and T. Abd-Elaziz, “Real parameter optimization by an effective differential evolution algorithm,” *Egyptian Informatics Journal*, vol. 14, no. 1, pp. 37–53, 2013.

[31] K. Chandrasekaran and U. Divakarla, *Load Balancing of Virtual Machine Resources in Cloud Using Genetic Algorithm*, National Institute of Technology Karnataka, Surath Ned, Mangalore, Karnataka, 2013.

[32] H. Gao, W. Huang, X. Yang, Y. Duan, and Y. Yin, “Toward service selection for workflow reconfiguration: an interface-based computing solution,” *Future Generation Computer Systems*, vol. 87, pp. 298–311, 2018.

[33] H. Gao, D. Chu, Y. Duan, and Y. Yin, “Probabilistic model checking-based service selection method for business process modeling,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 06, pp. 897–923, 2017.

[34] Y. Yin, L. Chen, y. xu, and J. Wan, “Location-aware service recommendation with enhanced probabilistic matrix factorization,” *IEEE Access*, vol. 6, pp. 62815–62825, 2018.

[35] S. Pang, T. Ding, A. Rodrfguez-Patdn, T. Song, and Z. Phen, “A parallel bioinspired framework for numerical calculations using enzymatic p system with an enzymatic environment,” *IEEE Access*, vol. 6, pp. 65548–65556, 2018.

[36] H. Gao, S. Mao, W. Huang, and X. Yang, “applying probabilistic model checking to financial production risk evaluation and control: a case study of Alibaba’s Yu’e Bao,” *IEEE Transactions on Computational Social Systems*, vol. 5, no. 3, pp. 785–795, 2018.

Hindawi
Scientific Programming
Volume 2020, Article ID 2548351, 13 pages
https://doi.org/10.1155/2020/2548351

Research Article

A Peak Prediction Method for Subflow in Hybrid Data Flow

Zhaohui Zhang,[1,2,3] Qiuwen Liu,[1] Ligong Chen,[1] and Pengwei Wang[1]

[1]*School of Computer Science and Technology, Donghua University, Shanghai, China*
[2]*The Key Laboratory of Embeded System and Service Computing, Ministry of Education, Tongji University, Shanghai, China*
[3]*Shanghai Engineering Research Center of Network Information Services, Shanghai, China*

Correspondence should be addressed to Zhaohui Zhang; zhzhang@dhu.edu.cn

Received 24 October 2019; Accepted 10 January 2020; Published 14 February 2020

Guest Editor: Aibo Song

Subflow prediction is required in resource active elastic scaling, but the existing single flow prediction methods cannot accurately predict the peak variation of subflow in hybrid data flow. These do not consider the correlation between subflows. The difficulty is that it is hard to calculate the correlation between different data flows in hybrid data flow. In order to solve this problem, this paper proposes a new method DCCSPP (subflow peak prediction of hybrid data flow based on delay correlation coefficients) to predict the peak value of hybrid data flow. Firstly, we establish a delay correlation coefficient model based on the sliding time window to determine the delay time and delay correlation coefficient. Next, based on the model, a hybrid data flow subflow peak prediction model and algorithm are established to achieve accurate peak prediction of subflow. Experiments show that our prediction model has achieved better results. Compared with LSTM, our method has decreased the MAE about 18.36% and RMSE 13.50%. Compared with linear regression, MAE and RMSE are decreased by 27.12% and 25.58%, respectively.

1. Introduction

The hybrid data flows are widely used in practical applications. For example, Alibaba's e-commerce platform uses a large-scale hybrid technology. This technology mixes online services with offline tasks. Hybrid data flow consists of online services and offline tasks. They enter the cluster at the same time and save the cost without affecting service quality.

The flow peak prediction is important in the active elastic expansion of the system [1]. Lombardi et al. [2] propose a novel elastic scaling approach, named ELYSIUM which contains the "predictionInputLoad" method to predict the maximum load. Bauer et al. [3] describe a new hybrid autoscaling mechanism, called Chameleon. Chameleon employs on-demand, automated time series-based forecasting methods to predict the arriving load intensity in combination. Hirashima et al. [4] give a new autoscaling mechanism which changes the scale of the target system based on the predicted workload.

In the active elastic scaling of the flow processing system, there are some studies on peak flow prediction. The authors regard network flow as a whole in the existing prediction methods. There are some traditional methods for network flow prediction, such as the ARIMA linear model and wireless network flow prediction model based on combinatorial optimization theory. Meanwhile, with the development in the neural network, the support vector machine (SVM) and other prediction model based on machine learning algorithm appears. Some authors use neural network models such as RNN [5], NARX recursive neural network model, LSTM [6], and GRU for predicting network peak flow. These prediction models can well explain the randomness and periodicity of flow.

However, the above methods are based on single flow prediction, without considering the possible correlation between individual flows in a hybrid data flow. Therefore, aiming at considering the influence of data correlation on peak flow prediction, this paper proposes a flow prediction method, named DCCSPP (subflow peak prediction of hybrid data flow based on delay correlation coefficients). We establish a delay correlation coefficient model to solve the correlation uncertainty of different subflows and consider the correlation influence between subflows based on the predicting results of the single flow. The more accurate the prediction of flow peaks, the more reliable the system flow information will be obtained, and this will provide better indexing parameters for the system's elastic scaling.

2. Related Work

In recent years, flow predictions based on time series have always been an attractive research area. Developing predictive models plays an important role in interpreting complex real-world elements [7].

Many of the traditional learning methods are used for time series prediction. Zhang et al. [1] propose an agile perception method to predict abnormal behavior. Yu et al. [8] describe an ARIMA linear model to predict network flow sequence. Aiming at solving the problem that a single model cannot fully describe change characteristics, a wireless network flow prediction model based on combinatorial optimization theory is proposed by Chen and Liu [9]. Liu et al. [10] give online learning algorithms for estimating ARIMA models under relaxed assumptions on the noise terms. Adebiyi et al. [11] examine the forecasting performance of ARIMA and artificial neural networks model. Wu and Wang [12] investigate time series prediction algorithms by using a combination of nonlinear filtering approaches and the feedforward neural network (FNN). Joo and Kim [13] propose a forecasting method based on wavelet filtering. Han et al. [14] introduce a multioutput least square support vector regressor. Chandra and Al-Deek [15] discuss a vector autoregressive model for prediction at short-term flow prediction on freeways. Conventional techniques for time series prediction are limited in their ability to process big data with high dimensionality, as well as efficiently represent complex functions. If the amount of linear data are not too large, the statistical method is reliable enough to be used for prediction. At the same time, the generated model is very complex and difficult to be implemented by nonlinear data types, so the prediction results are not very accurate when there are massive data.

Deep learning-based models have been successfully applied in many fields to time series prediction. There are many prediction models, which based on machine learning have been proposed. Haviluddin and Alfred [16] introduce a NARX recursive neural network model to predict network flow. Nie et al. [17] propose a novel network flow prediction method based on deep belief network (DBN) and logistic regression model for network flow prediction. In [18], network flow prediction of neural network models such as RNN [5], LSTM [6], and GRU is used. Hoermann et al. [19] report a deep CNN model for dynamic occupancy grid prediction with data from multiple sensors. The advantage of a Gaussian processes lies in its ability of modeling the uncertainty hidden in data, which is provided by predicting distributions [20]. Deep learning-based models are good at discovering intricate structure in large data sets [7]. These prediction models can well explain the randomness and periodicity of flow.

As mentioned above, the above methods are all for single flow prediction, without considering the possible correlation between data flows in hybrid flow. However, in the hybrid data flow, there is a lack of research on such flow prediction. Therefore, this paper mainly studies the correlation between different subflows in the hybrid flow and the peak prediction of each subflow.

3. Delay Correlation Coefficient Model Based on Sliding Time Window

In hybrid data flows, there are different degrees of correlation between different subflows. Considering the correlation between subflows and the pseudocorrelation caused by time analysis, this paper proposes a delay correlation coefficient model, which adds sliding time window according to Pearson correlation coefficient and time difference analysis [21]. This model is to calculate the delay correlation coefficient and delay time difference between different subflows. Based on the delay coefficient, the data flow that has an influence on the target subflow prediction is filtered out.

Correlation analysis [21] refers to the measure the closeness of the variables between two or more related variable elements. Correlation elements need to have a certain connection or probability to conduct correlation analysis.

The Pearson correlation coefficient, also known as Pearson product-moment correlation coefficient, represents the linear correlation between the two sets of variables X and Y. The formula is shown as follows:

$$\operatorname{cov}(X, Y) = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{n - 1}. \tag{1}$$

Equation (1) is the covariance formula. The covariance is divided by the standard deviation of the two related variables to obtain the Pearson correlation coefficient, which is described in formula (2). It is to compensate for the weak representation of the covariance value in the degree of random variable correlation:

$$\rho(X, Y) = \frac{\operatorname{cov}(X, Y)}{\sigma_X \sigma_Y}. \tag{2}$$

The Pearson correlation coefficient can always be between $[-1, 1]$. The closer the coefficients are to the extremes at both ends, the greater the linear relationship between the two random variables. If the coefficient is close to 0, it means that the two variables are not linearly related. If the coefficient approaches 1, it means that X and Y can be well described by the straight line equation, all data points fall well on a straight line, and X increases as Y increases. The coefficient approaching -1 means that all data points fall on a straight line, and X decreases as Y increases.

In the flow processing system, the input of data is generally composed of multiple subflows, which we call it a hybrid data flow. This article defines the hybrid data flow as follows

Definition 1. The hybrid data flow in the k period is $S_k = [(t_i, a_j) \mid 1 \le j \le n, 1 \le i \le k, n \le k]$, where n indicates that there are n kinds of data flows and (t_i, a_j) indicates that data belonged to the jth data flow arrives system at the time of t_i.

Definition 2. The data set constituting a business is $U = \{a_1, a_2, \ldots, a_z\}$, where z indicates that the data set of the service consists of z kinds of data flows. Thus, service correlation exists in these data. For example, a hybrid data flow consisting of device login information and user behavior information. The flow of user behavior information is affected by the flow of device login information, and the two have a partial-order relationship. Since different service data flows require different processing operations and computing resources, it is necessary to perform shunt operations on the data of the hybrid data flow, as shown in Figure 1.

Through the statistics of discrete hybrid data, the observation sequence of each subflow is obtained. A set of hybrid data flow observation sequences composed of subflow observation sequences are defined.

Definition 3. The hybrid data flow observation sequence set is $M = \{m_1, m_2, \ldots, m_n\}$, where n represents M contains n data flows. m_i represents the observed sequence of the ith data flow in M, that is, $m_i = \{m_i^1, m_i^2, \ldots, m_i^t, \ldots, m_i^l\}$, where m_i^t represents the observed value of data flow m_i at t time and l represents the l observation values of the data flow m_i. m_j represents the observation sequence of the jth data flow in M, that is, $m_j = \{m_j^1, m_j^2, \ldots, m_j^t, \ldots, m_j^l\}$, where m_j^t represents the observed value of data flow m_j at time t and l represents l observations in the data flow m_j. And $i \neq j$.

Definition 4. The ith subflow in hybrid data flow m_i.

Definition 5. The delay time e is shown in Figure 2. It means that the change of m_j in time $t-e$ has an effect on m_i at time t.

Definition 6. The size of the sliding time window is h, as shown in Figure 3.

Let $X = m_i$, where $X = \{x_1, x_2, \ldots, x_t, \ldots, x_l\}$, so $x_t = m_i^t$. Let $Y = m_j$, where $Y = \{y_1, y_2, \ldots, y_t, \ldots, y_l\}$, so $y_t = m_j^t$.

Definition 7. The correlation coefficient of m_i and m_j when the delay time is $d\rho(m_i, m_j)_e$. The calculation formula of $d\rho(m_i, m_j)_e$ is described in the following formula:

$$d\rho\left(m_i, m_j\right)_e = \frac{1}{l-h-e} \sum_{t=0}^{l-h-e} \left|\rho\left(X_{t-1}, Y_{t-e-1}\right)\right|, \tag{3}$$

where $X_{t-1} = \{x_{t-h}, x_{t-h+1}, \ldots, x_{t-1}\}$ and $X_{t-1} \subseteq X$. $Y_{t-e-1} = \{y_{t-h-e}, y_{t-h-e+1}, \ldots, y_{t-e-1}\}$ and $Y_{t-e-1} \subseteq Y$. X_{t-1} and Y_{t-e-1} are shown in Figure 4.

Definition 8. The maximum delay correlation coefficient between m_i and m_j is $\max[d\rho(m_i, m_j)]$. Its calculation formula is as follows:

$$\max\left[d\rho\left(m_i, m_j\right)\right] = \max\left[d\rho\left(m_i, m_j\right)\right]_e, \quad e \in [1, l-2h] \text{ and } e \in N. \tag{4}$$

When predicting m_i, it is necessary to select the data flow m_k $(1 \leq k \leq n \text{ and } k \neq i)$ with the highest delay correlation for the auxiliary prediction. The selection formula of m_k is as follows:

$$\max\left[d\rho\left(m_i, m_k\right)\right] = \max\left\{\max\left[d\rho\left(m_i, m_j\right)\right]\right\}, \quad j \in [1, n],\ j \in N,\ j \neq i. \tag{5}$$

Algorithm 1 gives the pseudocode for selecting the auxiliary data flow algorithm as follows.

4. Hybrid Data Flow Subflow Peaking Prediction Model

The selected data flow m_i (i.e., X) is separately predicted by a single flow prediction method, and an initial prediction result set $X' = \{x_1', x_2', \ldots, x_t', \ldots, x_l'\}$ of X is obtained, where x_t' represents an initial prediction result for the value x_t at time t in X.

Definition 9. The variation in x at time t is Δx_t. Δx_t represents the difference between the single prediction result at time t and time $t-1$. The calculation formula is as follows:

$$\Delta x_t = x_t' - x_{t-1}. \tag{6}$$

Definition 10. The amount of change in y at time t is Δy_t. Δy_t represents the difference between the observed value at time $t-e$ and $t-e-1$. The calculation formula is as follows:

$$\Delta y_t = y_{t-e} - y_{t-e-1}. \tag{7}$$

Definition 11. To scale the range of y to the range of x in a same level, we defined pro_{t-1}, which is described as follows:

$$\text{pro}_{t-1} = \frac{\max\left(X_{t-1}\right) - \min\left(X_{t-1}\right)}{\max\left(Y_{t-1-e}\right) - \min\left(Y_{t-1-e}\right)}. \tag{8}$$

Definition 12. At the time t, the final prediction result of x_t is x_t''. The calculation formula is as follows:

$$x_t'' = \Delta x_t * \alpha + \frac{\rho\left(X_{t-1}, Y_{t-1-e}\right)}{\left|\rho\left(X_{t-1}, Y_{t-1-e}\right)\right|} * \Delta y_t * (1-\alpha) * \text{pro}_{t-1}, \tag{9}$$

where α represents the weight of the correlation coefficient, and the calculation formula is as follows:

$$\alpha = \frac{1}{1 + \left|\rho\left(X_{t-1}, Y_{t-1-e}\right)\right|}. \tag{10}$$

Algorithm 2 gives the pseudocode for the hybrid data flow correlation prediction algorithm as follows.

The evaluation indexes in this paper are root mean square error (RMSE) and mean absolute error (MAE). The calculation formulas are as follows:

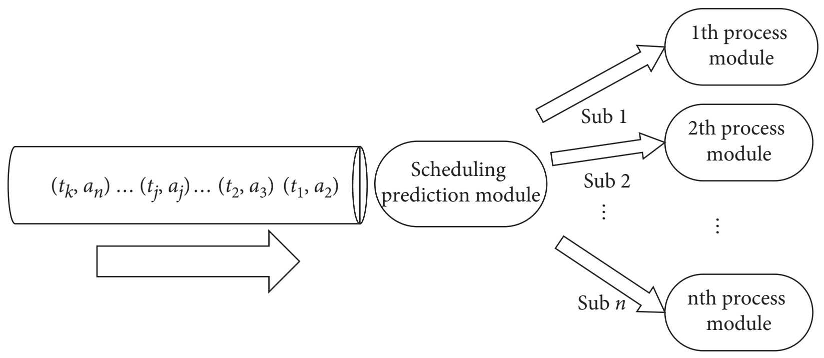

$(t_k, a_n) \dots (t_j, a_j) \dots (t_2, a_3)\ (t_1, a_2)$
Scheduling prediction module
Sub 1
Sub 2
Sub n
1th process module
2th process module
nth process module

Figure 1: Split flow diagram.

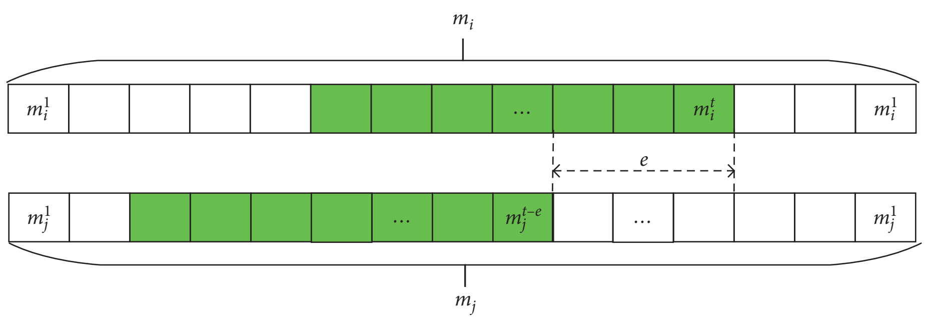


Figure 2: Delay time.

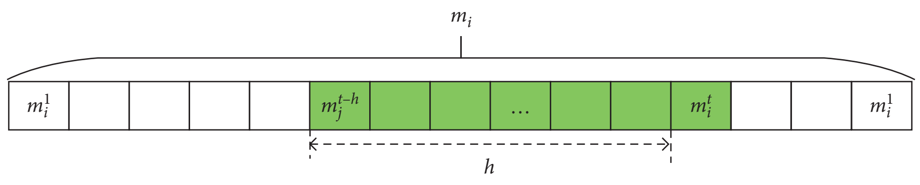


Figure 3: Sliding time window h.

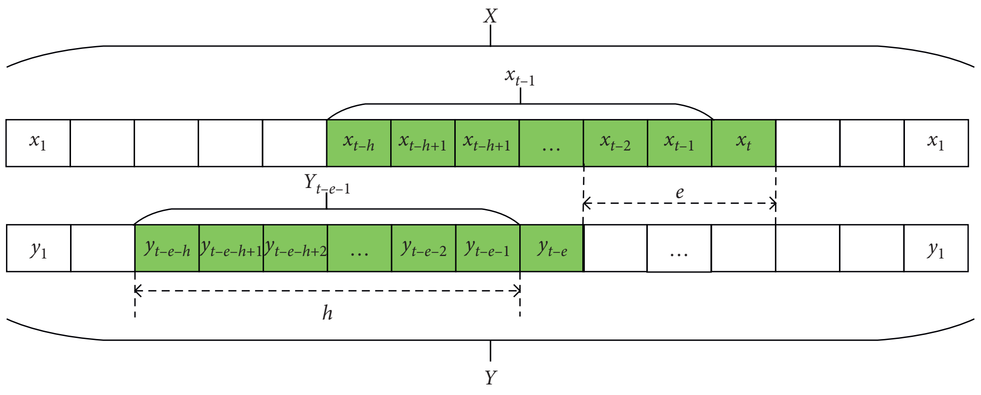


Figure 4: X_{t-1} and Y_{t-e-1} schematic.

```
Input: the list of steams; the size of window; the number of predicted steam
Output: the number of subsidiary steam; the number of delay time
 (1) procedure chooseSteam ()
 (2) for iterate through the list of steams do
 (3) for iterate through all of the number of delay time do
 (4) for iterate through all of the size of window do
 (5) Calculate and get the correlation coefficient between the delay time and window
 (6) Summing up the correlation coefficients
 (7) Calculate and get the mean of the correlation coefficient
 (8) Update the maximum delay correlation coefficient and the delay time
 (9) Update the maximum delay correlation coefficient and the delay time
(10) Update the number of the auxiliary data flow
(11) return the number of the auxiliary data flow and the delay time
```

Algorithm 1: Choose flow.

```
Input: the list of predicted steam; the list of first prediction flow; the list of subsidiary steam; the number of delay time; the size of
window; the number of time
Output: the number of the final predicted value at time t
(1) procedure prediction ()
(4) Calculate and get $\Delta x_t$ based on formula (6)
(5) Calculate and get $\Delta y_t$ based on formula (7)
(6) Calculate and get $\text{pro}_{t-1}$ based on formula (8)
(7) Calculate and get $\alpha$ based on formula (10)
(8) Calculate and get the final prediction result based on formula (9)
(9) return the final prediction result
```

Algorithm 2: Confounding flow correlation prediction.

$$\text{RMSE} = \sqrt{\frac{1}{l}\sum_{t=1}^{l}\left(x_t'' - x_t\right)^2},$$
$$\text{MAE} = \frac{1}{l}\sum_{t=1}^{l}\left|x_t'' - x_t\right|. \tag{11}$$

The smaller the mean absolute error index value is, the more accurate the prediction result is. The smaller the root mean square error value is, the fewer the abnormal discrete points are, and the higher the prediction accuracy is.

5. Experimental Verification

5.1. Data Set. In order to analyze the prediction performance of the prediction method proposed in this paper, the device login data and behavior acquisition data provided by the mobile phone APP of a credit company in three periods of three months are selected. We collect 13,567 pieces of equipment login data and 282,685 pieces of behavioral data in a certain period of June, as data set 1, as shown in Figures 5 and 6. There are 27,381 device login records and 344,109 behavior data selected in a certain period of July, as data set 2, as shown in Figures 7 and 8. And data set 3 selects 17550 device login records and 755693 behavior data in a certain period of November, as shown in Figures 9 and 10.

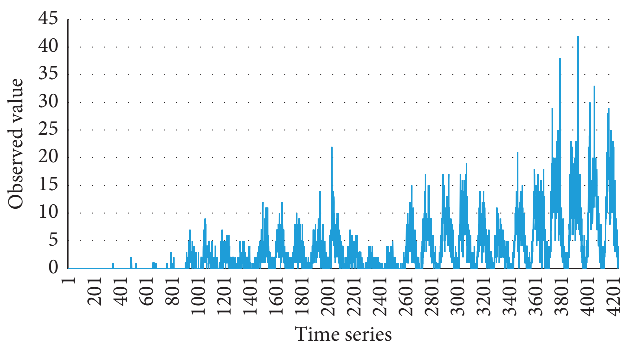


Figure 5: Data set 1 device login statistics.

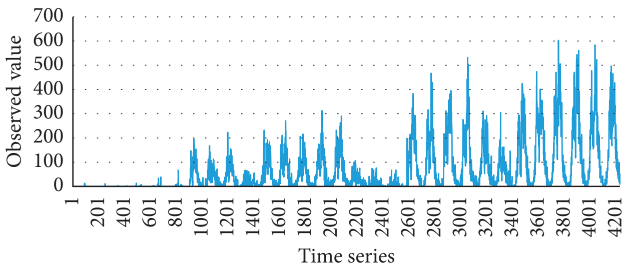


Figure 6: Data set 1 behavior collection statistics.

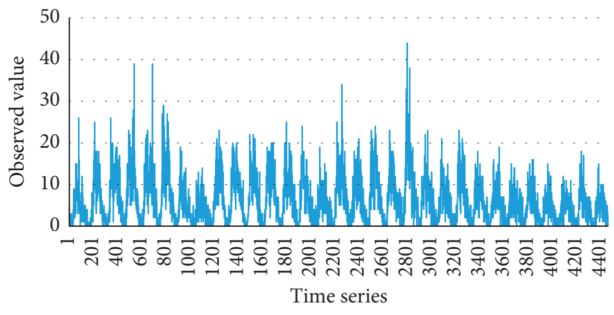


Figure 7: Data set 2 device login statistics.

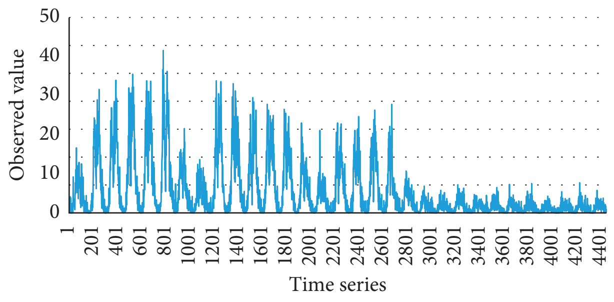


Figure 8: Data set 2 behavior collection statistics.

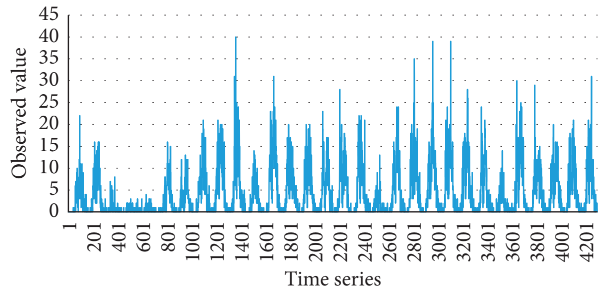


Figure 9: Data set 3 device login statistics.

Each subset selects 4465 observations. From Figures 5–10, we can see that the change trend of device login statistics and behavior collection statistics is close, and there is a correlation between them. In the experiment, firstly, the results predicted by LSTM and unary linear regression model are as the control group. Then, the results by our model are as the experimental group. In the end, compare their prediction indicators and error indicators of peak prediction.

5.2. Compared with LSTM Prediction Method. In this paper, the first 90% observed values of each data set is selected as training sets to train the LSTM learning model, and the last 10% is used as the test set to analyze the predictive ability of the model. The overall prediction results of the test sets of data set 1, data set 2, and data set 3 are obtained, as shown in Figures 11–13. And the prediction results for a period with high observed values in data set 1, data set 2, and data set 3 are shown in Figures 14–16. In the DCCSPP, it is necessary

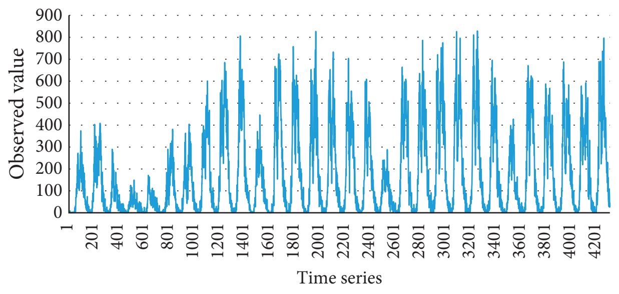


Figure 10: Data set 3 is behavior collection statistics.

to intercept the observation value of time window size for calculation, so before 90, the prediction method cannot give the prediction result, and the value is 0.

In this paper, we need to discuss the influence of time window, and the results of experiment on data set 2 are shown in Figure 17.

Compared with the LSTM model, it can be seen from Figures 14–16 that the results changes in DCCSPP are closer to the real-observed values.

It can be seen from Figure 17 that the selection of time window has certain influence on the prediction results. Too small or too large time window has a bad influence on the prediction results. Therefore, in addition to data set 3, this article selects 90 as the size of the time window. On data set 3, the prediction method can get better prediction results when the time window size is 240.

The errors of prediction results for data set 1, data set 2, and data set 3 in this paper are shown in Table 1. The prediction method has the most obvious improvement in data set 1. MAE and RMSE decreased by 13.46% and 17.80%, respectively. And we found that the smaller values of the test set of data set 2 lead that the MAE and RMSE of data set 2 are smaller than the others. In the end, the overall results show that the accuracy of the prediction results can be improved by using the correlation coefficient algorithm based on the prediction results of the LSTM model.

This paper compares the calculation indexes of prediction results of multiple maximum peak points in data set 1, data set 2, and data set 3, and the results are shown in Table 2. It illustrates that the peak prediction in the test set is not accurate due to the unfavorable data in the training set of data set 1. The method proposed in this paper can significantly improve the index of peak prediction, with MAE and RMSE increasing by 41.46% and 33.79%, respectively. In data set 2 and data set 3, MAE is decreased about 12.83% averagely. However, the improvement in the RMSE index was limited, with an average increase of 3.3%. In conclusion, the method proposed in this paper can improve the final peak prediction results.

5.3. Compared with Simple Linear Regression. In this paper, the unary linear regression model is used to predict the test sets in data set 1, data set 2, and data set 3. Through experiments, the prediction results of data set 1, data set 2, and data set 3 are shown in Figures 18–20, respectively. The

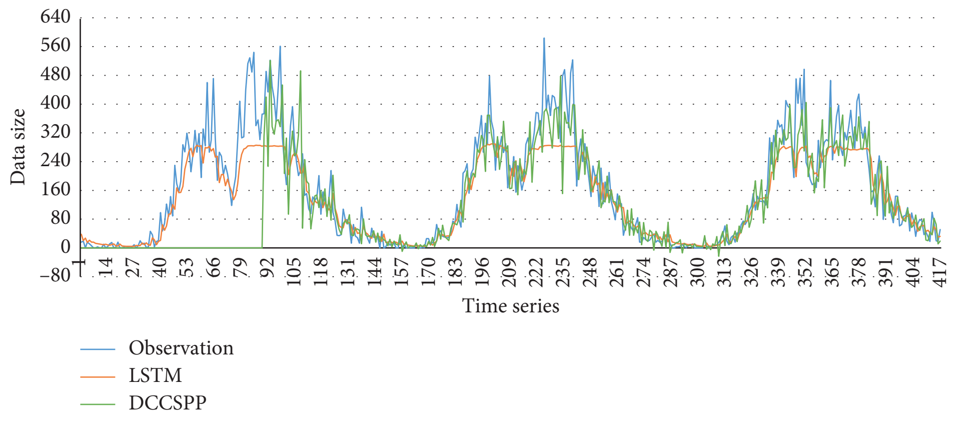


Figure 11: Comparison graph of predicted results on data set 1.

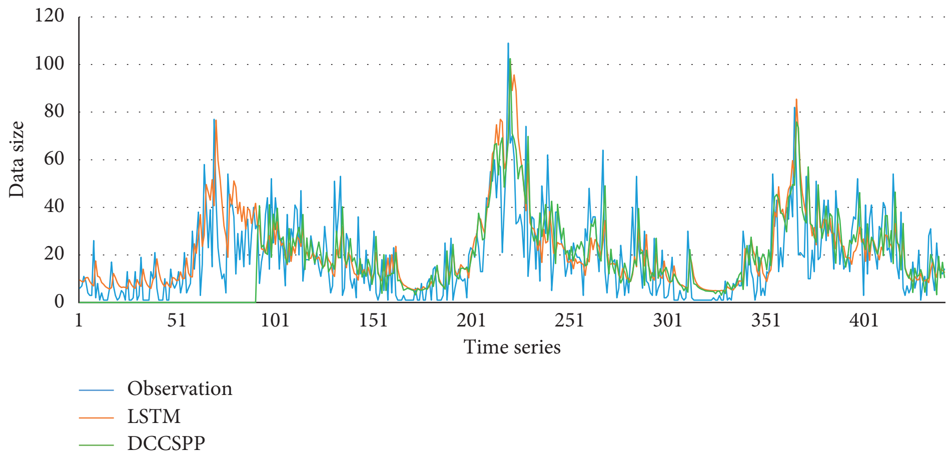


Figure 12: Comparison graph of predicted results on data set 2.

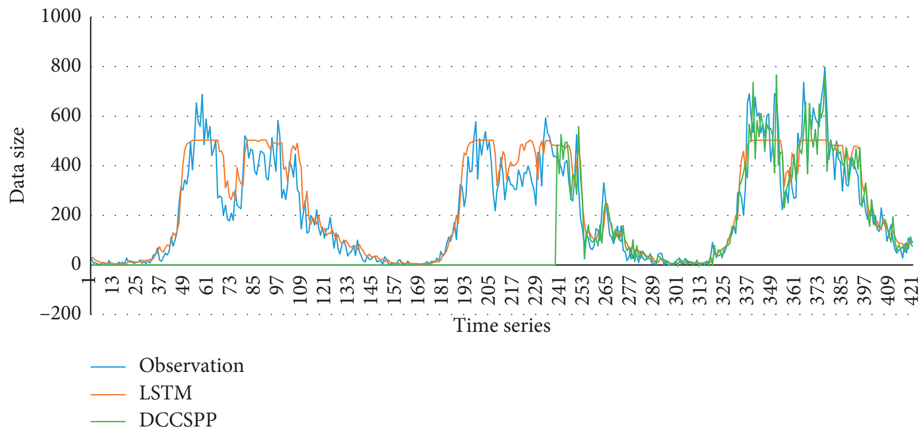


Figure 13: Comparison graph of predicted results on data set 3.

prediction results for a period with high observation values are shown in Figures 21–23, respectively.

It can be concluded from Figures 21–23 that the results of the prediction model in this paper are closer to the actual changes in the observations compared to the unitary linear regression model.

In this paper, the error comparison of prediction results for data set 1, data set 2, and data set 3 is shown in Table 3. It

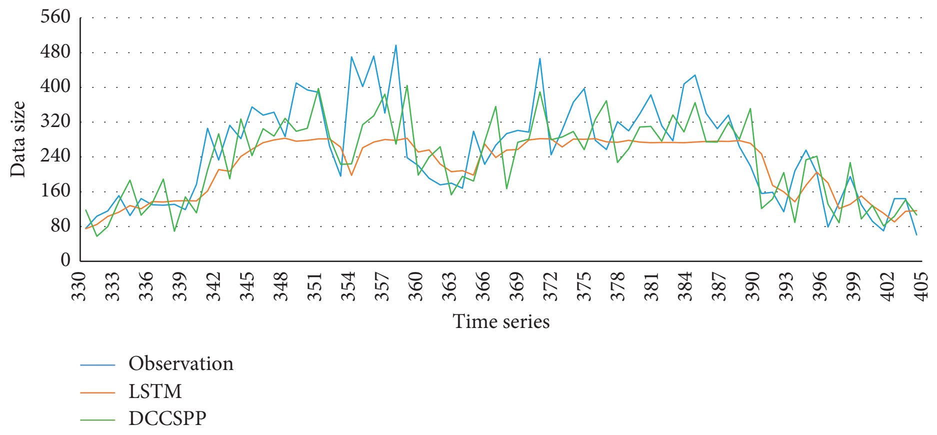


Figure 14: Comparison graph of prediction results for a time period in data set 1.

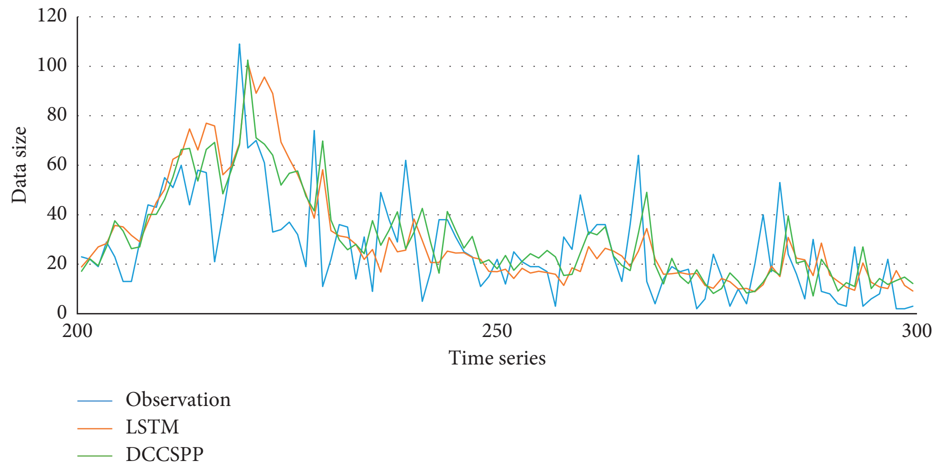


Figure 15: Comparison graph of prediction results for a time period in data set 2.

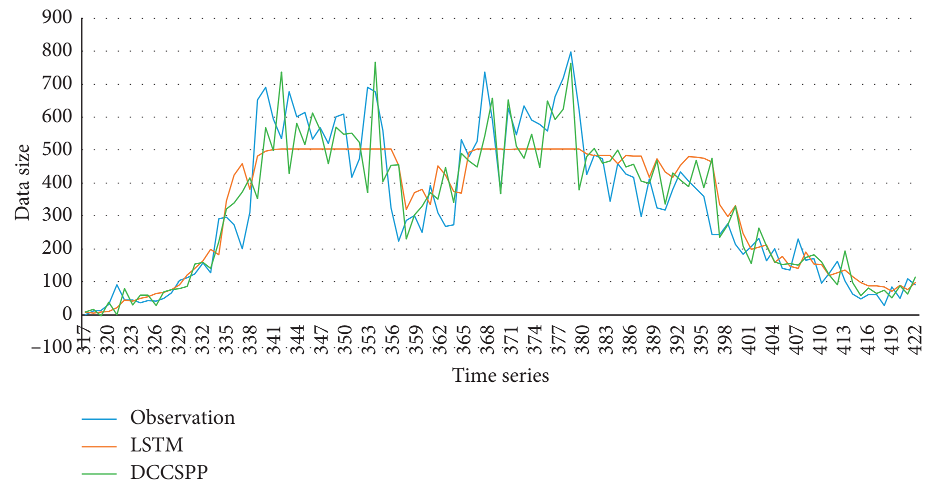


Figure 16: Comparison graph of prediction results for a time period in data set 3.

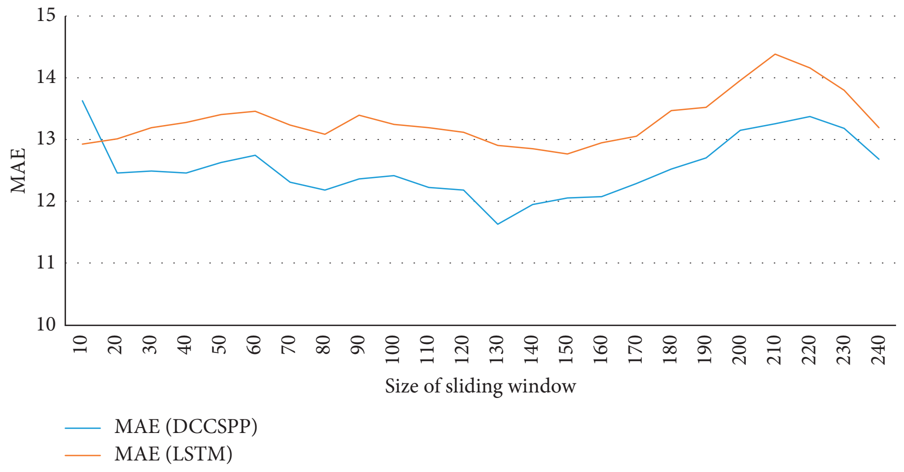


Figure 17: Comparison graph of average absolute errors of different time windows.

Table 1: Error comparison of prediction results.

Data set	Rate	LSTM	New	Improved (%)
Data set 1	MAE	68.85	59.58	13.46
	RMSE	98.32	80.83	17.80
Data set 2	MAE	13.39	12.36	7.73
	RMSE	17.34	16.13	7.00
Data set 3	MAE	59.57	53.86	9.59
	RMSE	90.98	87.01	4.36

Table 2: Peak prediction index decreased.

Data set	Number of peak points	Improved MAE (%)	Improved RMSE (%)
Data set 1	13	41.46	33.79
Data set 2	8	12.63	6.54
Data set 3	11	13.03	0.16

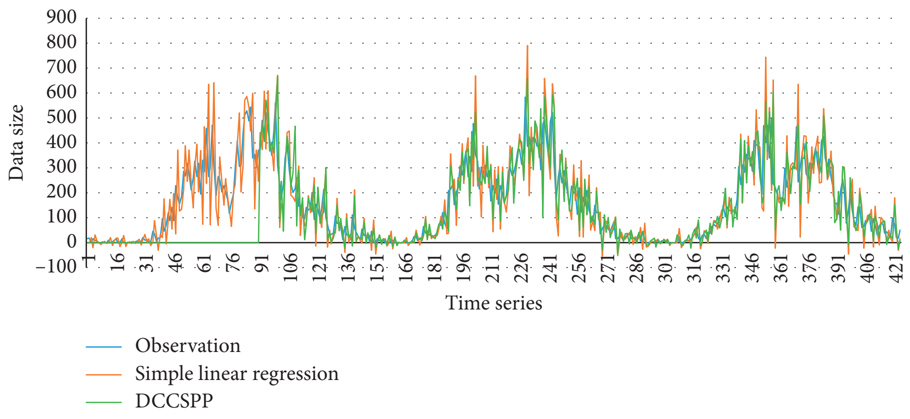


Figure 18: Comparison graph of predicted results on data set 1.

can be seen from the chart that the prediction results of the unary linear regression model are not as good as the LSTM model in MAE and RMSE indexes. Through the method proposed in this paper, the prediction result index on data set 1 is better than LSTM. The method proposed in this paper is used in the unary linear regression prediction model. The

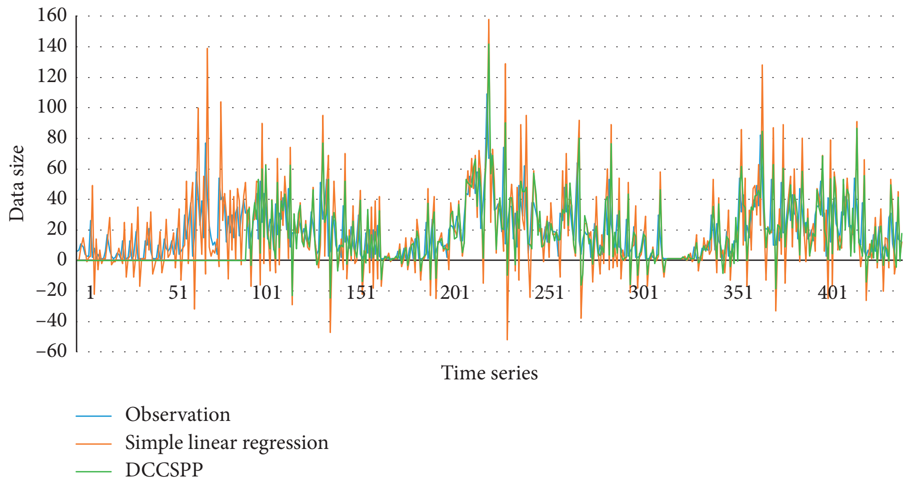


Figure 19: Comparison graph of predicted results on data set 2.

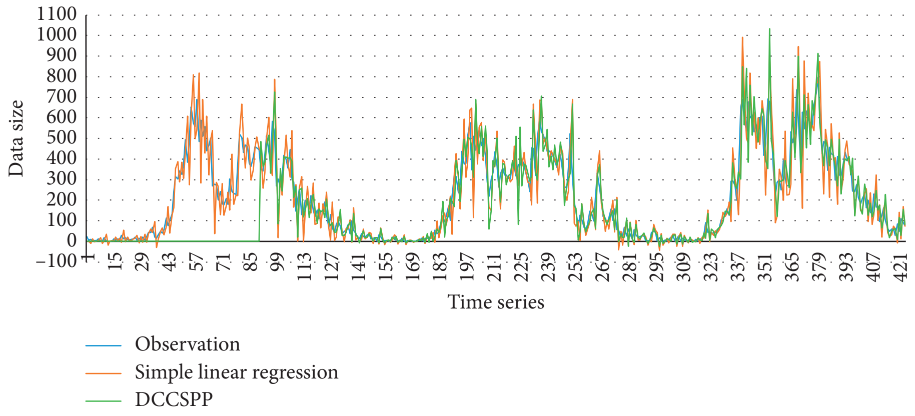


Figure 20: Comparison graph of predicted results on data set 3.

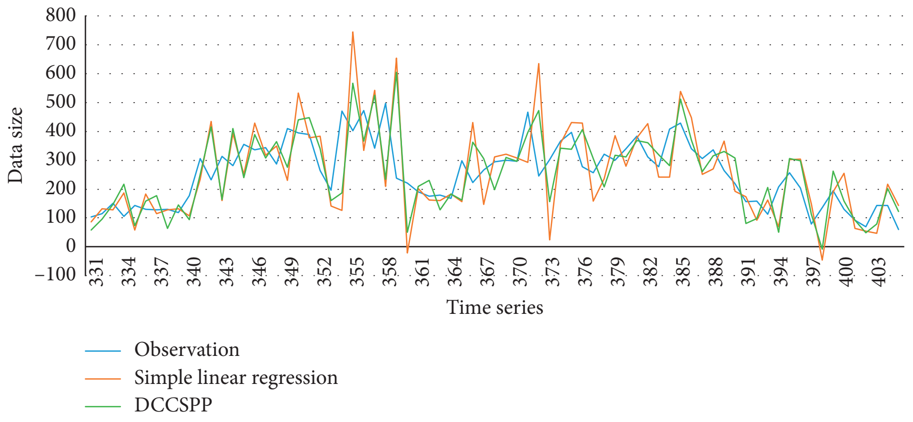


Figure 21: Comparison graph of prediction results for a time period in data set 1.

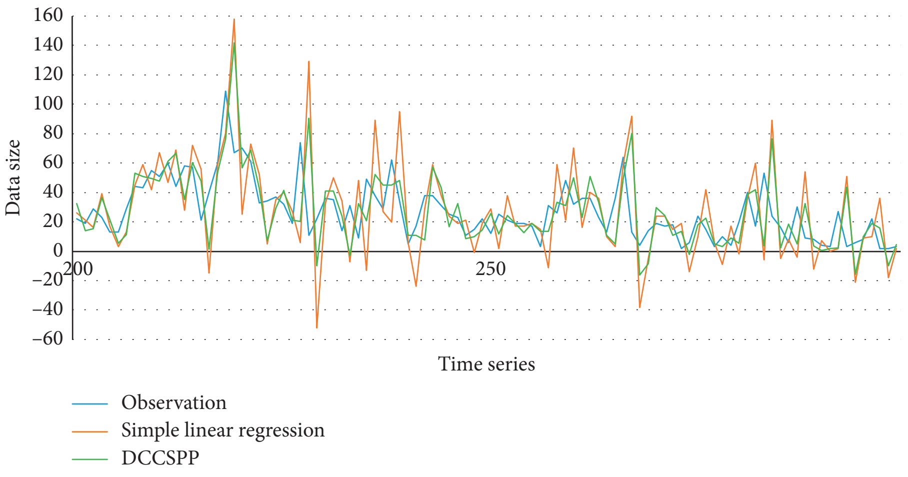


Figure 22: Comparison graph of prediction results for a time period in data set 2.

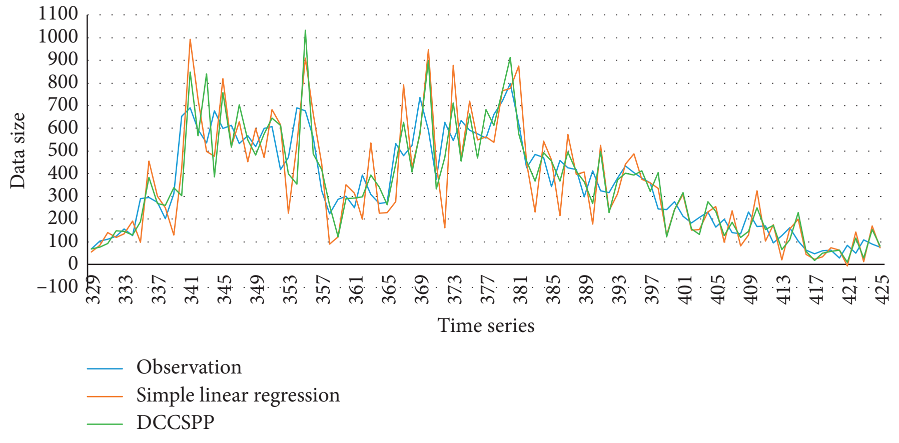


Figure 23: Comparison graph of prediction results for a time period in data set 3.

Table 3: Error comparison of prediction results.

Data set	Rate	Simple linear regression	New	Improved (%)
Data set 1	MAE	75.21	63.60	15.44
	RMSE	102.58	87.01	16.90
Data set 2	MAE	21.17	15.67	26.00
	RMSE	27.19	20.11	26.03
Data set 3	MAE	80.73	67.09	16.90
	RMSE	121.77	95.17	21.84

Table 4: Peak prediction index decreased.

Data set	Number of peak points	Improved MAE (%)	Improved RMSE (%)
Data set 1	13	33.45	28.73
Data set 2	8	32.40	29.49
Data set 3	11	15.50	18.52

experimental results show that the MAE value and the RMSE value are decreased by 15% to 26%. In conclusion, the method proposed in this paper used in the unary regression model can greatly improve the accuracy of the prediction results.

This paper compares the prediction results of multiple maximum peak points in data set 1, data set 2, and data set 3, and the results are shown in Table 4. As can be seen from the chart, 13 peak points with the highest observed values are selected in data set 1 to calculate the improvement of MAE and RMSE. They increase 33.45% and 28.73%, respectively. And 8 peak points with the highest observed values are selected in data set 2 to calculate the improvement of MAE and RMSE. They improve 32.40% and 29.49%, respectively. In data set 3, the 11 peak points with the highest observed values are selected to calculate the MAE and RMSE, which increase 15.50% and 18.52%, respectively. In conclusion, the method proposed in this paper can improve the final peak prediction results in the single-variable linear regression model's peak prediction results.

The chart information of experiment 1 and experiment 2 can be obtained. The method proposed in this paper can improve the prediction results in both overall prediction and peak prediction. Compared with the LSTM method, MAE and RMSE decreased by 18.36% and 13.50%, respectively. Compared with the unary linear regression method, MAE and RMSE decreased by 27.12% and 25.58%, respectively. In the overall forecast, MAE and RMSE rose about 14.85% and 15.66%, respectively. In the peak forecast, MAE and RMSE decreased by about 24.75% and 19.54%, respectively. Therefore, the peak prediction method of hybrid data subflow proposed in this paper can effectively improve the result based on the prediction result.

6. Conclusions

For the hybrid data flow, there are related uncertainties in each subflow at different times. This paper establishes the delay correlation coefficient model. Through this model, the delay correlation coefficient and delay time are calculated. The prediction results of the respective flows are calculated by using the peak prediction method in the hybrid data flow. Experiments show that the DCCSPP model has good prediction results when there is uncertainty between the subflows in the hybrid flow.

In future work, we will introduce the correlation between subflows into the machine learning model. Using machine learning methods improves the accuracy of delay correlation coefficient calculations and the prediction results. At the same time, the model can also be applied to dynamic hybrid data flows. Design a dynamic allocation scheme based on the predicted peak results of each subflow, dynamically allocating resources to systems that require elastic scaling.

Data Availability

The data used in the paper came from an insurance company of China. Subject to the confidentiality agreement, the experimental data set cannot be disclosed to the public, and the name of the company cannot be mentioned in the paper. However, we guarantee that the data set used is authentic with the company.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Natural Science Foundation of Shanghai (no. 19ZR1401900), Shanghai Science and Technology Innovation Action Plan Project (no. 19511101802), and National Natural Science Foundation of China (nos. 61472004 and 61602109).

References

[1] Z. Zhang and J. Cui, "An agile perception method for behavior abnormality in large-scale network service systems," *Chinese Journal of Computers*, vol. 40, no. 2, pp. 503–519, 2017, in Chinese.

[2] F. Lombardi, L. Aniello, S. Bonomi, and L. Querzoni, "Elastic symbiotic scaling of operators and resources in stream processing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 3, pp. 572–585, 2017.

[3] A. Bauer, N. Herbst, S. Spinner, A. Ali-Eldin, and S. Kounev, "Chameleon: a hybrid, proactive auto-scaling mechanism on a level-playing field," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 4, pp. 800–813, 2018.

[4] Y. Hirashima, K. Yamasaki, and M. Nagura, "Proactive-reactive auto-scaling mechanism for unpredictable load change," in *Proceedings of the 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp. 861–866, IEEE, Kumamoto, Japan, July 2016.

[5] R. Madan and P. SarathiMangipudi, "Predicting computer network traffic: a time series forecasting approach using DWT, ARIMA and RNN," in *Proceedings of the Eleventh International Conference on Contemporary Computing (IC3)*, pp. 1–5, IEEE, Noida, India, August 2018.

[6] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in *Proceedings of the IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pp. 153–158, IEEE, Chengdu, China, December 2015.

[7] Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, "A review of deep learning models for time series prediction," in *IEEE Sensors Journal*, pp. 1–1.

[8] Y. Yu, J. Wang, M. Song et al., "Network traffic prediction and result analysis based on seasonal ARIMA and correlation coefficient," in *Proceedings of the International Conference on Intelligent System Design and Engineering Application*, vol. 1, pp. 980–983, IEEE, Changsha, China, October 2010.

[9] H. Chen and J. Liu, "Modeling and forecast of wireless network traffic based on combinatorial optimization theory," *Modern Electronics Technique*, vol. 39, no. 23, pp. 43–47, 2016.

[10] C. Liu, S. C. H. Hoi, P. Zhao et al., "Online arima algorithms for time series prediction," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, AZ, USA, February 2016.

[11] A. A. Adebiyi, A. O. Adewumi, and C. K. Ayo, "Comparison of ARIMA and artificial neural networks models for stock price prediction[J]," *Journal of Applied Mathematics*, vol. 2014, Article ID 614342, 7 pages, 2014.

[12] X. Wu and Y. Wang, "Extended and Unscented Kalman filtering based feedforward neural networks for time series prediction," *Applied Mathematical Modelling*, vol. 36, no. 3, pp. 1123–1131, 2012.

[13] T. W. Joo and S. B. Kim, "Time series forecasting based on wavelet filtering," *Expert Systems with Applications*, vol. 42, no. 8, pp. 3868–3874, 2015.

[14] Z. Han, Y. Liu, J. Zhao, and W. Wang, "Real time prediction for converter gas tank levels based on multi-output least square support vector regressor," *Control Engineering Practice*, vol. 20, no. 12, pp. 1400–1409, 2012.

[15] S. R. Chandra and H. Al-Deek, "Predictions of freeway traffic speeds and volumes using vector autoregressive models," *Journal of Intelligent Transportation Systems*, vol. 13, no. 2, pp. 53–72, 2009.

[16] Haviluddin and R. Alfred, "Performance of modeling time series using nonlinear autoregressive with eXogenous input (NARX) in the network traffic forecasting," in *Proceedings of the International Conference on Science in Information Technology*, IEEE, Yogyakarta, Indonesia, October 2016.

[17] L. Nie, D. Jiang, L. Guo, S. Yu, and H. Song, "Traffic matrix prediction and estimation based on deep learning for data center networks," in *Proceedings of the Globecom Workshops*, IEEE, Washington, DC, USA, December 2017.

[18] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying deep learning approaches for network traffic prediction," in *Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2353–2358, IEEE, Udupi, India, September 2017.

[19] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic occupancy grid prediction for urban autonomous driving: a deep learning approach with fully automatic labeling," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2056–2063, IEEE, Brisbane, Australia, May 2018.

[20] C. K. I. Williams and C. E. Rasmussen, *Gaussian Processes for Machine learning*, MIT press, Cambridge, MA, USA, 2006.

[21] H. Wang, Z. Zhang, and P. Wang, "A situation analysis method for specific domain based on multi-source data fusion," in *Proceedings of the International Conference on Intelligent Computing*, pp. 160–171, Springer, Wuhan, China, August 2018.

Hindawi
Scientific Programming
Volume 2020, Article ID 6364752, 12 pages
https://doi.org/10.1155/2020/6364752

Research Article

Query Execution Optimization in Spark SQL

Xuechun Ji,[1,2] **Maoxian Zhao**,[3] **Mingyu Zhai**,[2] **and Qingxi Wu**[2]

[1]*School of Computer Science and Engineering, Southeast University, Nanjing, China*
[2]*NARI Research Institute NARI Technology, Nanjing, China*
[3]*College of Mathematics and Systems Science, Shandong University of Science and Technology, Qingdao, China*

Correspondence should be addressed to Maoxian Zhao; sdzmx66@163.com

Received 23 October 2019; Accepted 11 January 2020; Published 7 February 2020

Guest Editor: Jianming Bian

Spark SQL is a big data processing tool for structured data query and analysis. However, due to the execution of Spark SQL, there are multiple times to write intermediate data to the disk, which reduces the execution efficiency of Spark SQL. Targeting on the existing issues, we design and implement an intermediate data cache layer between the underlying file system and the upper Spark core to reduce the cost of random disk I/O. By using the query pre-analysis module, we can dynamically adjust the capacity of cache layer for different queries. And the allocation module can allocate proper memory for each node in cluster. According to the sharing of the intermediate data in the Spark SQL workflow, this paper proposes a cost-based correlation merging algorithm, which can effectively reduce the cost of reading and writing redundant data. This paper develops the SSO (Spark SQL Optimizer) module and integrates it into the original Spark system to achieve the above functions. This paper compares the query performance with the existing Spark SQL by experiment data generated by TPC-H tool. The experimental results show that the SSO module can effectively improve the query efficiency, reduce the disk I/O cost and make full use of the cluster memory resources.

1. Introduction

With the increasing popularity of e-commerce, social network, artificial intelligence and other new Internet applications, the amount of data being stored and processed by governments, enterprises and research institutions has increased dramatically. A substation in the power system generates 100000 alarm data per minute, and the Facebook generates more than 400 TB of logs every day. It is an urgent problem for enterprises and research institutions to store such large-scale data on hard disk persistently and retrieve the information required by users in a short time. Data storage and processing system in big data environment has received more and more attention in recent years.

The early big data processing systems mainly revolved around Hadoop platform. In order to solve the problem that Hadoop platform frequently read and wrote intermediate data in HDFS, a method to cache Hadoop's Shuffle data in memory was proposed by Shi et al. [1] Although this method could effectively reduce the large amount of random disk I/O cost caused by reading and writing intermediate data, it was inflexible as the cache size created by this method was fixed for different applications. A certain amount of memory would be wasted for some applications with small amount of Shuffle data. To solve the problems of Hadoop platform, the memory-based distributed computing framework Apache Spark emerged.

Spark is a high-speed and versatile big data processing engine. It is based on the implementation of RDD [2] (Resilient Distributed Datasets) and implements data distribution and fault tolerance. As shown in Figure 1, the current Spark ecosystem consists of three layers: the bottom layer, the middle layer, and the top layer. The bottom layer can read input data from HDFS [3], Amazon S3, HyperTable and HBase [4]. The middle layer uses resource scheduling management platforms such as Standalone, Yarn [5] and Mesos [6] to complete the analysis and processing of applications. The top layer consists of a series of advanced tools, including Spark Streaming [7], GraphX [8], BlinkDB [9], MLlib [10] and Spark SQL [11]. Spark SQL is developed from Shark [12], it provides functions like Hive [13] which allows users to process structured data directly by entering

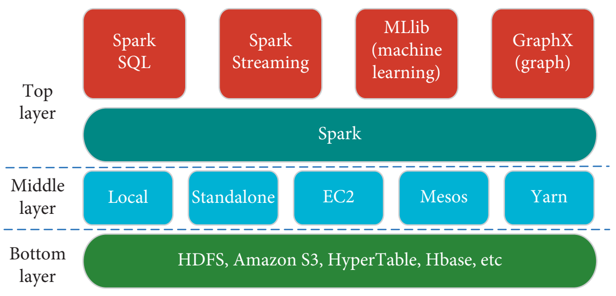


Figure 1: Spark ecosphere.

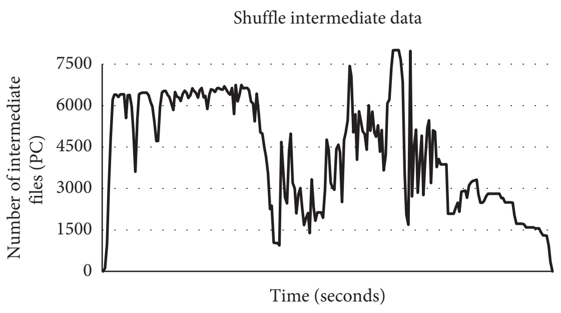


Figure 2: Wave pattern of spark shuffle intermediate data.

SQL statements. The Catalyst which generates and optimizes execution plan of Spark SQL will perform algebraic optimization for SQL query statements submitted by users and generate Spark workflow and submit them for execution.

However, the Spark SQL system currently faces two problems. One problem is that the frequent reading and writing of intermediate data in the data interaction process between Spark tasks lead to serious random disk I/O cost. The number of intermediate files produced by a simple program on Spark is shown in Figure 2.

The other problem is that there are no suitable optimization rules for Spark workflow. Spark jobs with intermediate data correlation need to read the same input data from disk repeatedly, resulting in redundant disk I/O cost.

Under the above background, this paper aims to improve the execution efficiency of Spark SQL. The main contributions are as follows.

(1) Reduce the random disk I/O cost in the Shuffle phase by adding an intermediate data cache layer between the Spark core layer and the underlying distributed file system.

(2) Reduce the read/write cost of the same intermediate data between Spark jobs by using a cost-based correlation merging algorithm, then further improving the performance of the data analysis system.

2. System Architecture

The SSO (Spark SQL Optimizer) prototype system is designed and developed. Its system architecture is shown in Figure 3.

The existing Spark system runs in a master-slave mode. The Driver process on the main node receives query requests submitted from Client. The Worker process on the slave node executes specific query tasks. SSO system optimizes and improves the original Spark, adding a dynamic Shuffle optimizer module named Dynamic Shuffle Optimizer on the main node. Spark's original Catalyst framework is also modified to include a cost model module renamed SQL workflow optimizer. Reading and writing interfaces for distributed memory file system are added to the slave node for the Shuffle Map Task and Shuffle Reduce Task. The process of optimizing the users' queries by SSO system is described below.

Queries submitted by users are first parsed by the parser in the SQL workflow optimizer module to form a logic execution plan. Then, the optimized execution plan is submitted to Dynamic Shuffle Optimizer and DAG scheduler. Dynamic Shuffle Optimizer calculates the size of intermediate data generated by the optimized SQL queries using the query pre-analysis module. Then the allocation module at the cache layer performs buffer allocation on the distributed memory file system. Finally, the workflow generated by the execution plan will be submitted to the DAG scheduler for task assignment at the slave node.

3. Spark Shuffle Intermediate Data Cache Layer

In the existing Spark system, data interaction between Stages will frequently generate disk I/O overhead. Therefore, a strategy for caching immediate data is proposed. The dynamic Shuffle optimizer is used to create buffers with different sizes on the distributed memory file system dynamically for different Spark SQL workflow. The random disk I/O cost of the Shuffle phase will be reduced by caching Shuffle intermediate data in memory.

3.1. Query Pre-Analysis Module. The following query execution process is analyzed to show when Spark SQL writes intermediate data to disk. The query statement is:

select

l_partkey, l_quantity, l_extendedprice

from

lineitem, part

where

p_partkey = l_partkey
and l_quantity <40

Analysis shows that the query includes only one join operation. The execution process of the query under Spark SQL is shown in Figure 4.

The above execution process of Spark is divided into three Stage. Stage 0 reads the part table and runs projection operations on the attribute p_partkey. Stage 1 reads the lineitem table and performs projection operations on the three attributes of l_partkey, l_quantity and l_extendedprice as well as selection operations of l_quantity <40. Stage 2 reads the intermediate results of Stage 0 and Stage 1,

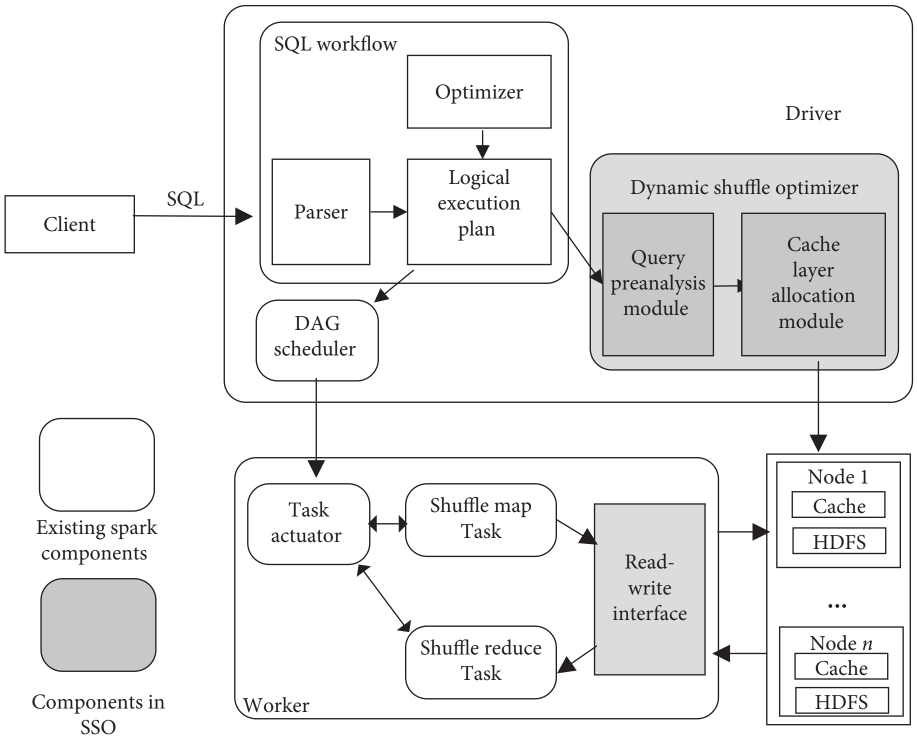


Figure 3: System architecture of SSO.

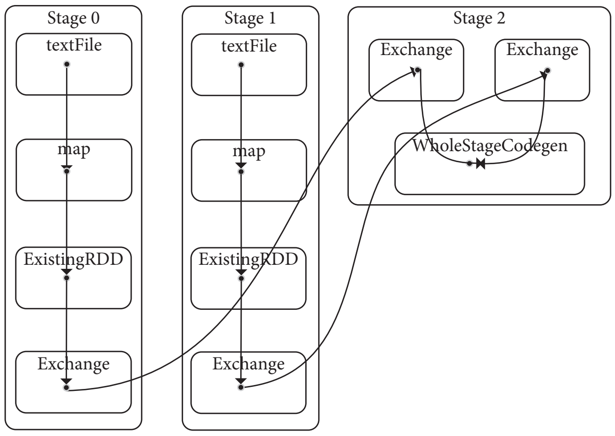


Figure 4: The execution process of spark.

performs join operations of p_partkey = l_partkey and finally writes the query results to disk.

Analysis of this query show that the query execution workflow in Spark can be divided into two stages. One is S-Stage for projection, selection and aggregation operations, and the other is J-Stage for join operations.

For S-Stage, the following rules can be used to calculate the size of output data at this stage.

(1) Projection

$$\left|D_{\text{out}}(\text{pro})\right| = \beta_{\text{pro}}\left|D_{\text{in}}\right|,$$
$$\beta_{\text{pro}} = \frac{L_{\text{pro}}}{L_D}. \tag{1}$$

L_{pro} represents the length of projection attribute, and L_D represents the total length of all attributes.

(2) Selection

$$\left|D_{\text{out}}(\text{fil})\right| = \beta_{\text{fil}}\left|D_{\text{in}}\right|. \tag{2}$$

The value of β_{fil} is related to specific selection conditions and data distribution of the original table.

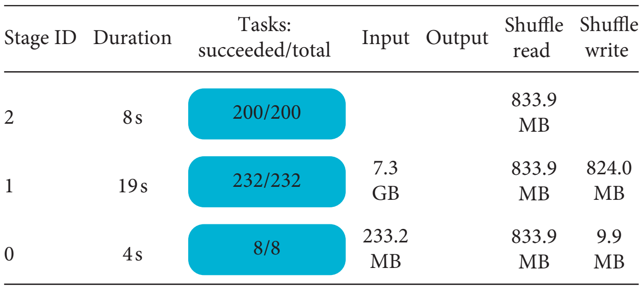
(3) Aggregation

$$\left|D_{\text{out}}(\text{agg})\right| \approx 0. \tag{3}$$

Aggregation operation returns the sum or average value of an attribute, so the output size is negligible.

For example, Figure 5 is the running state diagram of this SQL query on Spark. The input data size read by Stage 0 is 233.2 MB, known Lpartkey = 10, $L_D = 194$. Through the above analysis, the output of this Stage can be calculated $|D_{\text{out}}| = \beta_{\text{pro}}|D_{\text{in}}| = 10/194 * 233 \approx 13$ MB. It approximates the actual output of Stage 0 of 10 MB. Similarly, the input data size read by Stage 1 is 7.3 GB, known Lpartkey = 10, Lquantity = Lextendedprice = 15, $L_D = 231$. Sampling shows that the tuples with l_quantity >40 account for about 60% of the total tuples. Therefore the output of this Stage is $|D_{\text{out}}| = \beta_{\text{pro}} * \beta_{\text{fil}}|D_{\text{in}}| = (10 + 15 + 15)/231 * 0.6 * 7.3\ \text{GB} \approx$ 776 MB, which approximates the actual output of 824 MB generated by Stage 1.

3.2. Cost Analysis of Join Operation. For the J-Stage, the following rules are used to calculate the output data size of this Stage.



Stage ID	Duration	Tasks: succeeded/total	Input	Output	Shuffle read	Shuffle write
2	8 s	200/200			833.9 MB	
1	19 s	232/232	7.3 GB		833.9 MB	824.0 MB
0	4 s	8/8	233.2 MB		833.9 MB	9.9 MB

Figure 5: The result of query execution.

$$\left|D_{\text{out}}(\text{join})\right| = \gamma\left(\left|D_1^{\text{in}}\right| \times_C \left|D_2^{\text{in}}\right|\right). \quad (4)$$

If there is no join condition C, it becomes Cartesian product and $\gamma = 1$. If no tuple satisfies join condition C, γ equals 0. Normally $0 \le \gamma \le 1$. If the join condition is $D_1 \cdot A = D_2 \cdot B$, there are three special cases.

(1) If A is the primary key of D_1, each tuple in D_2 matches at most one tuple in D_1, that is $|D_{\text{out}}(\text{join})| \le |D_{\text{in}}\ 2|$. So, $\gamma \le 1/|D_{\text{in}}\ 1|$. If B is the primary key of D_2, $\gamma \le 1/|D_{\text{in}}\ 2|$.

(2) If A is not the primary key of D_1 and B is not the primary key of D_2, and the attributes A and B obey uniform distribution on the same domain M, then $\gamma = 1/|M|$.

(3) If A is not the primary key of D_1 and B is not the primary key of D_2, and the attributes A and B do not obey uniform distribution, γ needs to be sampled according to the specific data set, table structure and data size. As for the cost analysis method of join operation under non-uniform distribution, the idea of histogram method mentioned in [14] is referred to develop a method suitable for Spark SQL. Assuming that R is a relationship, field C is an attribute of R, and the value range of C is [min, …, max], where min and max are the minimum and maximum values of field C in relation R respectively. The [min, …, max] is divided into several intervals, called straight or bucket. Then the number of tuples whose C attribute values are in these intervals is counted. In order to analyze the cost of join operation by histogram method, the first step is to construct the variable-width distribution histogram. The flow of the algorithm is shown in Algorithm 1. The input of the algorithm is frequency distribution histogram which records the value of each attribute and its occurrence times. The attribute values are sorted from small to large. The output is variable-width distribution histogram which consists of histogram buckets. Each histogram bucket records the starting and ending values of the attribute, and the frequency sum of attribute within the range. There is no intersection between buckets. In the algorithm, a variable max is declared to record the maximum frequency in the current histogram bucket. For each of the following attributes and frequency pairs, if the difference between the frequency value and max only accounts for 5% or less of the frequency value, it can be considered that this attribute obeys the same distribution as each attribute in the current histogram bucket. Therefore, this attribute is also included in the current histogram bucket. If the above conditions are not met, a new histogram bucket is created whose starting value is the current attribute value and max is the frequency value of the corresponding attribute value. Continue the above steps until all attributes and frequency pairs have been traversed.

After constructing the variable-width distribution histogram, the algorithm to calculate the total tuple number after joining is shown in Algorithm 2.

The input of the algorithm is the variable-width distribution histogram of relation R and relation S. The output is the total number of tuples after the join operation of the two relations. The algorithm seeks overlaps between two histogram buckets from the first histogram bucket of relation R and relation S histogram. As the histogram bucket is uniformly distributed, templeft and tempright can be obtained by dividing the size of the overlaps by the width of the histogram bucket, and then multiplying the result by the total frequency of the histogram bucket. Templeft and tempright represent the frequency of the equivalent attribute in the histogram bucket of relation R and relation S respectively. Multiply templeft by tempright and divide by the size of the overlaps, then the total tuple number generated by the join operation in the first histogram bucket with overlaps is obtained. If there is no overlap between two histogram buckets and the range of bucket $h_i r$ is smaller than the range of bucket $h_j s$, the range of bucket $h_{i+1} r$ and bucket $h_j s$ need to be compared. Continue the above steps until all the histogram buckets in relation R or relation S are traversed.

3.3. Cache Layer Allocation Module

3.3. Cache Layer Allocation Module. After calculating the total size of the cache layer, the next question is how much memory is allocated for each node in the cluster. Due to the principle of Spark Data Locality, when reading HDFS files, Spark will assign the nearest Executor to the data storage. For nodes that store more input data, more memory should be allocated. The input data in HDFS is organized as blocks, and the default size of a block is 128 MB. So, after parsing users' query statements to obtain the tables to be required, it is necessary to analyze the distribution of blocks in the cluster for each table. Since the size of each block in HDFS is the same, the memory size M_i allocated to each node can be obtained by calculating the ratio of the block number of each node B_i to the total blocks B_{total}.

$$M_i = \frac{B_i}{B_{\text{total}}} M_{\text{total}}. \quad (5)$$

For the Q1 query mentioned before, the size of table Lineitem is 7.2 GB including 60 blocks. The size of table Part is 233 MB including 2 blocks. The distribution of these blocks in the cluster is shown in Figure 6.

```
Construct variable-width distribution histogram by frequency distribution histogram
input: $H_{fre}$ = {<attr1, freq1>, <attr2, freq2>, . . ., <attrn, freqn>}
output: $H_{width}$ = {<$start_1$, $end_1$, $times_1$>,<$start_2$, $end_2$, $times_2$>, . . ., <$start_m$, $end_m$, $times_m$>}
procedure
  $i \longleftarrow 1$; $H_{width} \longleftarrow$ {}
  start $\longleftarrow$ $attr_1$; end $\longleftarrow$ $attr_1$;
  max $\longleftarrow$ $freq_1$; $T \longleftarrow freq_1$
  while $i \leq n$ do
    $i \longleftarrow i + 1$
    if $|max\text{-}freq_i|/freq_i < 0.05$ then
      end $\longleftarrow$ $attr_i$
      $T \longleftarrow T + freq_i$
      if $freq_i > max$ then
        max $\longleftarrow$ $freq_i$
      end if
    else
      $H_{width} \longleftarrow H_{width}$ + <start, end, $T$>
      start $\longleftarrow$ $attr_i$; end $\longleftarrow$ $attr_i$
      max $\longleftarrow$ $freq_i$; $T \longleftarrow freq_i$
    end if
  end while
end procedure
```

Algorithm 1: Algorithm to construct variable-width distribution histogram.

```
Estimate the size of join operation by histogram method
input: $H_R = \{h_1r, h_2r, \ldots, h_nr\}$, HS = $\{h_1s, h_2s, \ldots, h_ms\}$
Output: Total tuples Sum after join;
procedure
  $i \longleftarrow 1$; $j \longleftarrow 1$; Sum $\longleftarrow 0$;
  while $i \leq n$ and $j \leq m$ do;
    if $h_i$ and $h_j$ have overlap then;
      Overlap $\longleftarrow$ Overlap of two histogram buckets;
      templeft $\longleftarrow$ $h_i$.times $*$ Overlap/($h_i$.end-$h_i$.start)
      tempright $\longleftarrow$ $h_j$.times $*$ Overlap/($h_j$.end-$h_j$.start)
      Sum $\longleftarrow$ Sum + templeft $*$ tempright/Overlap
      if $h_i$.end < $h_j$.end then
        $i \longleftarrow i + 1$
      else
        $j \longleftarrow j + 1$
      end if
    else
      if $h_i$.end < $h_j$.start then
        $i \longleftarrow i + 1$
      else
        $j \longleftarrow j + 1$
      end if
    end if
  end while
end procedure
```

Algorithm 2: Calculation of the tuple number of join operation by histogram method.

After obtaining the distribution of the input data in the cluster, according to the total size of the cache layer calculated by query pre-analysis module and the formulas in the cache layer allocation module, the memory size required for each node is $790 * 2/62 \approx 25.48$ MB, $790 * 21/62 \approx 267.58$ MB, $790 * 3/62 \approx 38.22$ MB, $790 * 20/62 \approx 254.83$ MB, $790 * 4/62 \approx 50.96$ MB, $790 * 5/62 \approx 63.71$ MB. The results approximate to the actual Shuffle data in Table 1.

Since Spark is a framework based on memory computing, the operations on Resilient Distributed Datasets are

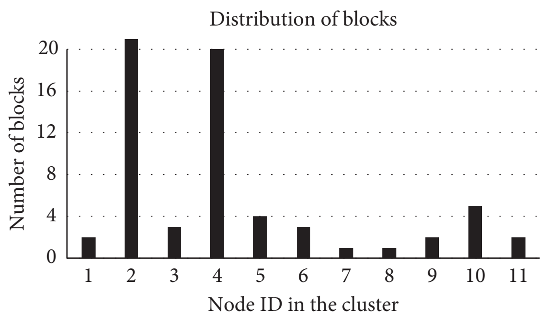


Figure 6: Distribution of blocks in the cluster.

Table 1: Comparison between calculated and actual shuffle sizes.

ID	1	2	3	4	5	6	7	8	9	10	11
General task	9	114	14	119	7	14	9	22	14	7	7
Input (MB)	256	2688	384	2560	512	384	128	128	256	640	256
Calculated shuffle size	27	282.4	40.3	269	53.81	40.35	13.45	15.05	28.03	67.25	29.06
Actual shuffle size	25	267.58	38.2	254.8	50.96	38.22	12.76	12.76	25.48	63.71	25.48

all carried out in memory before or after Shuffle operations. If the cluster has enough memory, even if a certain size of memory space has been allocated for intermediate data cache layer, the remaining memory is enough for Spark task to perform calculation. If the memory resources of the cluster are limited, the query pre-analysis module will calculate the cache size of Shuffle before the query runs, and allocate the memory through the cache layer allocation module. This will lead to occupying the memory before the Shuffle operation is carried out. This is not a reasonable approach obviously. The cache layer allocation module adopts a delay allocation scheme to solve this problem. The Spark program allocates as much memory as possible to Spark works in the non-Shuffle phase. In the Shuffle phase, Spark works do not need memory resources for calculation. Then part of the memory resources in Spark works and the remaining memory resources in the cluster are called for the intermediate data cache layer.

4. Cost-Based Correlation Merging Algorithm

The optimization process of SQL query by the Catalyst framework in the traditional Spark system is shown in Figure 7. First, the query statements entered by users are parsed by SQL parser to form a logical execution plan tree. The plan tree is then algebraically optimized to some extent by the SQL optimizer. In the current Spark SQL workflow, there is a case of repeatedly reading and writing the same intermediate data. There are multiple tasks at the nodes of the logical execution plan tree output the same intermediate data, resulting in additional disk I/O cost. Therefore, an optimization rule can be added to the original SQL optimizer to merge the same intermediate data. Although merging will reduce the cost of writing output data on disk, subsequent tasks will read the intermediate data they do not need which also brings extra disk reading cost. So, it needs cost calculation whether to merge tasks with intermediate data correlation. A cost model module is introduced based on the original Catalyst framework. When merging tasks with intermediate data correlation, SQL optimizer will determine whether to execute the optimization rule by cost calculation. The optimized Catalyst framework is named as SQL workflow optimizer and its framework is shown in Figure 7.

4.1. Cost Model.

4.1. Cost Model. The cost model for Spark tasks execution should be established in order to merge Spark tasks with intermediate data correlation. Then the benefits and extra costs of merging should be calculated based on model to decide whether to merge by comparison.

For establishing the task execution cost model in Spark, we improve the method proposed by Singhal and Singh [15] and add the cost generated by sorting operation. When calculating the Stage cost, reading input data, merging and sorting intermediate data, and writing output data are considered, that is

$$C(\text{Stage}) = C_{\text{read}}(\text{Stage}) + C_{\text{sort}}(\text{Stage}) + C_{\text{write}}(\text{Stage}). \tag{6}$$

Since both $C_{\text{read}}(\text{Stage})$ and $C_{\text{write}}(\text{Stage})$ are I/O cost, the cost calculation formula is $C_{I/O} = C_0\mathrm{T} + C_1 x$. The definition of each parameter is shown in Table 2.

The cost of Stage reading phase $C_{\text{read}}(\text{Stage})$ can be calculated by

$$C_1 x = |D_{\text{in}}| t_r + \alpha |D_{\text{in}}| t_b = (t_r + \alpha t_b)|D_{\text{in}}|. \tag{7}$$

$|D_{\text{in}}|$ is determined by the size of the source input data or the output data of other Stage. The value of α is 0.3. It is determined by the default three-copy storage strategy of HDFS, one locally, one on the same rack and one on the remote rack. The number of I/O occurrence depends on the specific Stage. For S-Stage, if the source input data is read, T equals 1 as the source data is stored continuously. If the

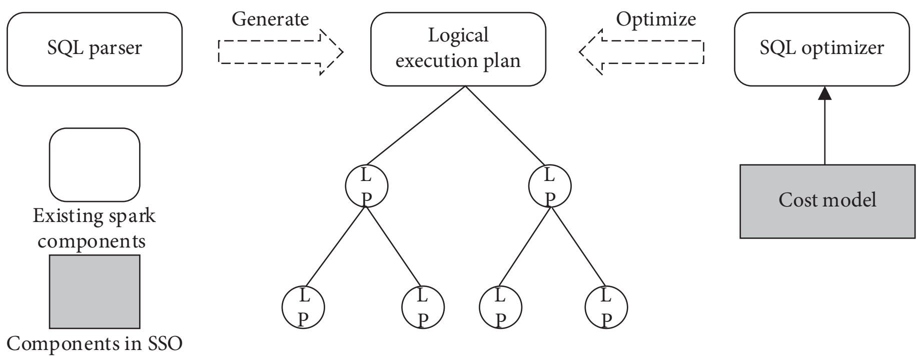


Figure 7: The framework of SQL workflow optimizer.

Table 2: Parameters in the cost model.

Parameter	Meaning
X	Size of read/written data
C_0	Seeking time and rotation delay time
C_1	Time required to transmit 1 MB data
A	Proportion of non-local data to total data
T	Number of I/O occurrences
$\|D_{\text{in}}\|$	Size of stage input data
$\|D_{\text{out}}\|$	Size of stage output data t_r time to read 1 MB data locally
t_w	Time to write 1 MB data locally
t_b	Time to transfer 1 MB data over network
B	Buffer size of spark task m task number in stage

output of other Stages is read, the value of T is determined by the file number of intermediate data generated at the previous Stage. Spark tasks write the intermediate data to the buffer first, and then overwrite the disk to form a file when the buffer is full. Suppose that the previous Stage generates data of size $|D_{\text{out}}|$, then $|D_{\text{out}}|/B$ intermediate files will be generated, where B is the size of the Spark task buffer. So, the number of I/O occurrence is $T = |D_{\text{out}}|/B$. For J-Stage, the input of this Stage must be the output of the other two stages due to the join operation of two tables. Assuming that the previous two stages produce a total of $|D_{\text{out}}|$ size output data, if the two-way merge sort join algorithm is used, $\lceil \log 2(|D_{\text{out}}|/B) \rceil$ times of scanning are needed, and the number of I/O occurrence is $|D_{\text{out}}|/B\lceil \log 2(|D_{\text{out}}|/B) \rceil$. In summary, the calculation formula for the cost in the reading phase of Stage is as follows.

$$C_{\text{read}}(\text{Stage}) = C_0 T + (t_r + \alpha t_b)|D_{\text{in}}|. \tag{8}$$

For the cost of the writing phase of Stage C_{write} (Stage), since the intermediate data is overwritten to the local disk, this process does not involve the network transmission cost. The calculation formula is

$$C_1 x = |D_{\text{out}}| t_w, \tag{9}$$

where $|D_{\text{out}}|$ can be calculated by the method described in Chapter 2. The number of I/O occurrence T is determined by the number of intermediate files. The number of intermediate files is calculated by $|D_{\text{out}}|/B$. Then the calculation formula of the writing cost in Stage is given by

$$C_{\text{write}}(\text{Stage}) = \frac{C_0|D_{\text{out}}|}{B} + t_w|D_{\text{out}}|. \tag{10}$$

Each task in Stage needs to sort and merge all the intermediate data files generated by itself. If a total of $|D_{\text{out}}|/B$ intermediate files are generated, it can be considered that each task generates $|D_{\text{out}}|/B_m$ intermediate files, where m is the number of tasks in each Stage. So, the cost of the sorting phase $C_{\text{sort}}(\text{Stage})$ is calculated by

$$C_{\text{sort}}(\text{Stage}) = \frac{C_0|D_{\text{out}}|}{B_m}\left\lceil \log 2\left(\frac{|D_{\text{out}}|}{B_m}\right)\right\rceil + t_r|D_{\text{out}}|. \tag{11}$$

Assuming that the number of sorting P equals $|D_{\text{out}}|/B_m\lceil \log 2(|D_{\text{out}}|/B_m) \rceil$, the execution cost of a Stage in Spark is given by

$$C(\text{Stage}) = (t_r + \alpha t_b)|D_{\text{in}}| + (t_r + t_w)|D_{\text{out}}| + C_0\left(T + P + \left(\frac{|D_{\text{out}}|}{B}\right)\right). \tag{12}$$

4.2. Format of Shuffle Intermediate Data. In order to merge the intermediate data, the format of the Shuffle intermediate data is first introduced and then the shared field merging algorithm is proposed.

The MapTask in the Spark Shuffle phase extracts the required data by scaning each line of the source data file. Then it stores the data in the form of key and value to the <key, value> pairs and outputs to the local disk. Take the work Stage 1 in Figure 8 as an example, the corresponding query is *select l_partkey, l_quantity, l_extendedprice from lineitem, part where p_partkey = l_partkey.*

The parsed workflows are all joined at the Reduce Task in Spark SQL. So, in the MapTask phase, the data is read from the lineitem table by row, then l_partkey is saved as key, and l_quantity and l_extendedprice are saved as values. To facilitate the join operation at Reduce, the actual <key, value> pairs are of the form < l_partkey, l_quantity|l_extendprice|>.

Similarly, For Stage 3 in Figure 8, the query is select l_partkey, l_quantity from lineitem, supplier where s_suppkey = l_suppkey and s_nationkey = 2. The <key, value> pairs generated in the Shuffle phase is <l_suppkey, l_partkey|l_quantity>.

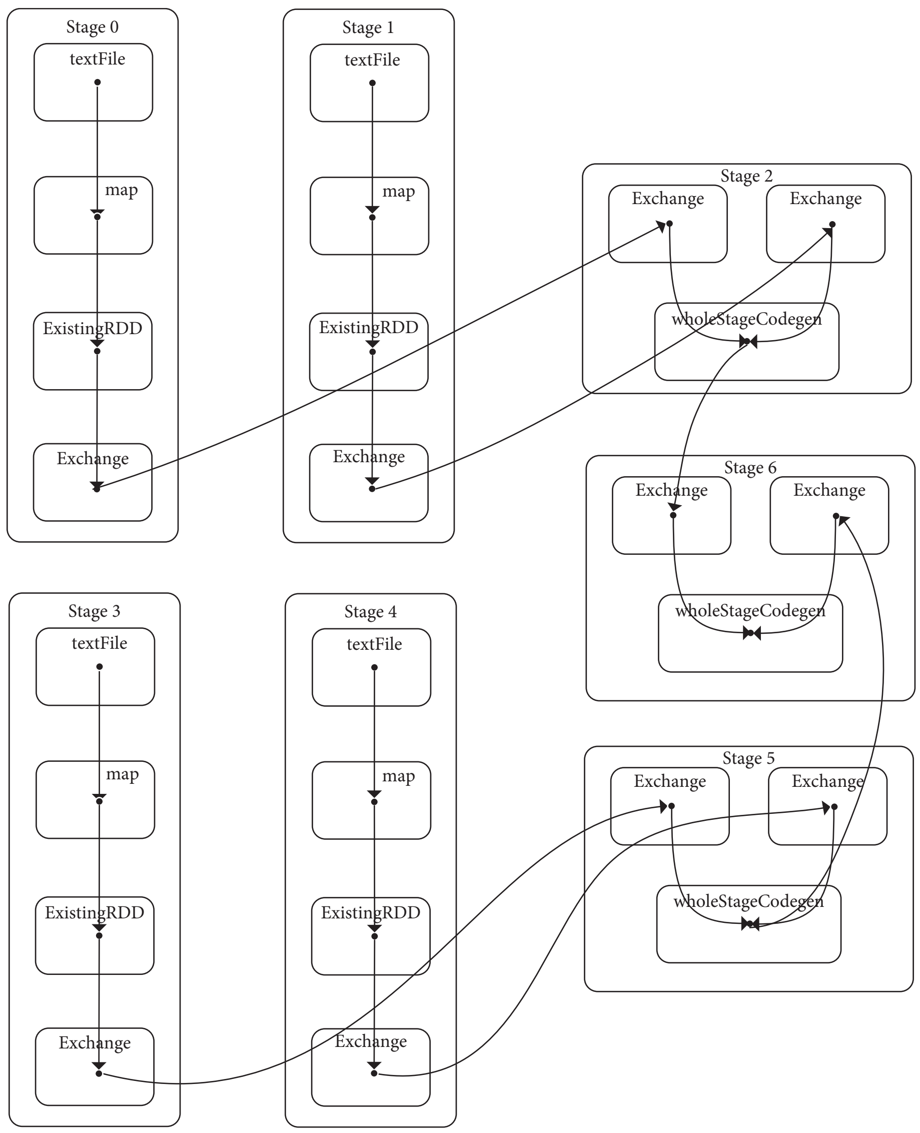


Figure 8: The query execution flow of TPC-H Q17.

The field l_partkey appears in the intermediate data output of both Stage 1 and Stage 3, one as the Key and the other as the Value. Meanwhile, l_quantity also appears in the Value of both stages. Since both sub-queries involve only join and projection operations and no selection operations, the output data from MapTask of Stage 1 and Stage 3 contains all the rows in the lineitem table. It can be considered that the two types of pairs contain all the l_partkey and l_quantity in the lineitem table, so they can be merged. Although merging reduces the writing cost of the current Stage, it also increases the reading cost of the subsequent Stages. For example, Stage 2 joins the part table and lineitem table, but reads the unwanted l_suppkey field after merging. Similarly, Stage 5 joins the supplier table and lineitem table, but reads the unwanted l_extendedprice field after merging. Therefore, the writing cost of saved field and the reading cost of increased field should be weighed to determine whether to merge the Shuffle intermediate data or not. The problem can be solved by the shared field merging algorithm below.

4.3. Shared Field Merging Algorithm. Assuming that the SQL statement of Stage$_i$ is *select* $i_1, i_2, \ldots, i_n$ *from table*$_i$, the format of <key, value> pairs generated is <i_1, $i_2|i_3|\ldots|i_n$>. The format of <key, value> pairs generated by Stage$_j$ is <j_1, $j_2|j_3\ldots|j_m$>. Let $S_i = \{i_2, i_3, \ldots, i_n\}$ and $S_j = \{j_2, j_3, \ldots, j_m\}$, then the format of <key, value> pairs after merging is <$i_1 \cup j_1$, $S_i \cup S_{j\text{-}i1} \cup j_1$>. The keys of two pairs are first compared. Unify them into one field if the field names are the same, otherwise, join the two fields together separated by "|." Next, merge the value sets of the two pairs and remove the fields that have already appeared in the Key. In summary, the data format after merging the intermediate data of Stage 1 and Stage 3 is <l_partkey |l_suppkey, l_quantity |l_extendedprice>.

To measure the writing benefit of saved fields and the reading cost of increased fields, the definition of cost model is

$$\text{Savings} = \frac{C_0 |D_{\text{write}}|}{B} + t_w |D_{\text{write}}|,$$

$$\text{Costs} = C_0 \frac{|D_{\text{read}}|}{B} \left\lceil \log_2 \left(\frac{|D_{\text{read}}|}{B} \right) \right\rceil + (t_r + \alpha t_b) |D_{\text{read}}|, \tag{13}$$

where $|D_{\text{write}}| = |D_{\text{out}}| \quad L_{\text{save\_cols}}/L_{\text{total}}$, $|D_{\text{read}}| = |D_{\text{out}}| \quad L_{\text{add\_cols}}/L_{\text{total}}$. $L_{\text{save\_cols}}$ is the length of saved fields of writing. $L_{\text{add\_cols}}$ is the length of increased fields of reading. L_{total} is the length of the total fields. $|D_{\text{out}}|$ can be calculated by the method described in Chapter 2. For example, after merging into the above pairs, the repeated writing of l_parkey and l_quantity can be reduced, and $L_{\text{save\_cols}} = L_{\text{l\_partkey}} + L_{\text{l\_quantity}}$. Similarly, the redundant reading of l_extendedprice and l_suppkey is increased, and $L_{\text{add\_cols}} = L_{\text{l\_extendedprice}} + L_{\text{l\_suppkey}}$. Let $E = \text{Savings–Costs}$. If $E > 0$, it means that the reduced cost of sharing redundant fields is greater than the cost of reading additional fields. The public fields in the intermediate data can be merged. Otherwise, if Earn <0, these fields will not be merged. Whether to merge the intermediate data needs comparison between the value of Savings and the value of Costs.

5. Experiments

We verified the performance of SSO system in this section. The test data was generated by the TPC-H benchmarking tool. Part of the query statements provided by TPC-H was selected to test the performance of this system in two parts: intermediate data caching test and intermediate data correlation merging algorithm test.

5.1. Experimental Environment and Data Set. The hardware environment was as follows. The experimental cluster included a main node and 10 sub-nodes. The nodes were connected through gigabit ethernet. Each node was configured with 2.7 GHz CPU(8-core 16 threads, 20M Cache, Turbo frequency), 64 GB DDR3 1600 and 500 GB SAS mechanical hard disk.

As for the software environment, the operating system used in the header node and the slave node was Centos6.7. Each node was installed with JDK 1.8.0, Hadoop 2.6.0, Spark 2.0.1 and Scala 2.1.0. The programming language used in the experimental development were Java and Scala. IDEA IDE was used in the development environment.

The experimental data set used here was generated by TPC-H benchmarking tool. TPC-H was a standard for DBMS performance testing developed by TPC (American Transaction Processing Performance Council), a non-profit organization. It could be used to simulate the business application environment in real life and was widely used to evaluate the comprehensive performance of decision support system. The test data was generated using the command *dbgen -s x* provided by TPC-H where *x* represented the data size in GB to be generated. In addition to the test data set, TPC-H also provided 22 query statements Q1-Q22. Users utilized command *qgen* provided by TPC-H to generate these queries for performance testing.

5.2. Spark Shuffle Intermediate Data Caching Test. After completing the memory allocation of Spark Shuffle cache layer in each node, several representative queries in TPC-H were chosen which were Q1, Q5, Q9, Q18 and Q19. Q1 and Q19 had larger input data and fewer intermediate data. Q5, Q9 and Q18 had fewer input data and larger intermediate data. The queries were programmed to be submitted to SSO system and the original Spark system, and the query execution time was recorded. The experimental results are shown in Figure 9.

Results shows that for Q5, Q9 and Q18 whose intermediate data was much larger than the input data, the intermediate data cache layer could solve the problem of high random disk *I/O* cost. The optimization effect was obvious. But for Q1 and Q19 whose input data was larger than the intermediate data, the reading of the input data accounted for the cast majority of disk *I/O*. The optimization effect of the intermediate data cache layer for those queries was relatively limited.

The change of disk *I/O* rate over the query process was shown in Figure 10. As can be seen from the figure, since the source data was stored sequentially, the process of reading input data produced centralized disk *I/O*. In both systems, the disk read rate was around 80 MB/S which is the peak performance of stable mechanical hard disk. During the running process, the program entered the Shuffle phase of reading and writing intermediate data. At this time, the original Spark system generated more random disk *I/O* and the disk read rate had a great fluctuation. It took a long time to complete the Shuffle process in the original Spark system. The SSO system used the intermediate data cache layer, and the intermediate data were read and written in memory. So, the Shuffle phase of SSO system did not generate disk I/O cost. The SSO system completed the query task faster than the original Spark system. Figure 11 showed that the memory usage of the two systems was similar in the first half of query process and both were steadily increasing. When entering the Shuffle phase, SSO system calculated the required memory size by query pre-analysis module and allocated the memory at each node by cache layer allocation module. Therefore, the memory usage of SSO system would increase instantaneously at this time. This part of memory would be released after the end of query. In the whole process of query, the main node had enough memory (64 GB), while the size of intermediate data generated by query Q9 was only 14.1 GB. It was possible to create a piece of memory space as intermediate data buffer. SSO system had a higher memory usage rate, which was one of the main reasons for the higher execution efficiency of SSO system.

5.3. Intermediate Data Correlation Merging Algorithm Test. By merging the intermediate data with the same fields, the size of intermediate data generated could be further reduced,

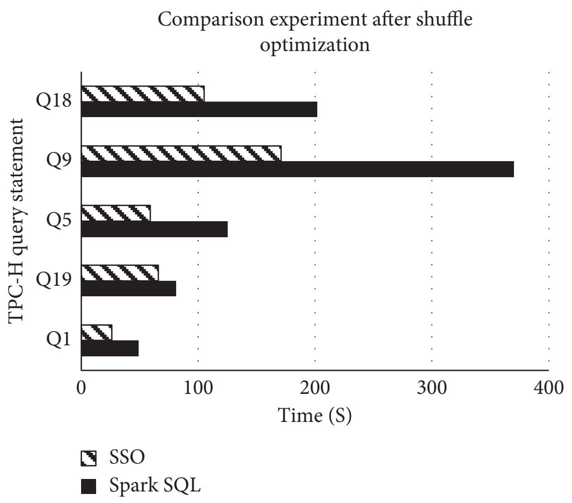


Figure 9: Comparison experiment before and after intermediate data caching.

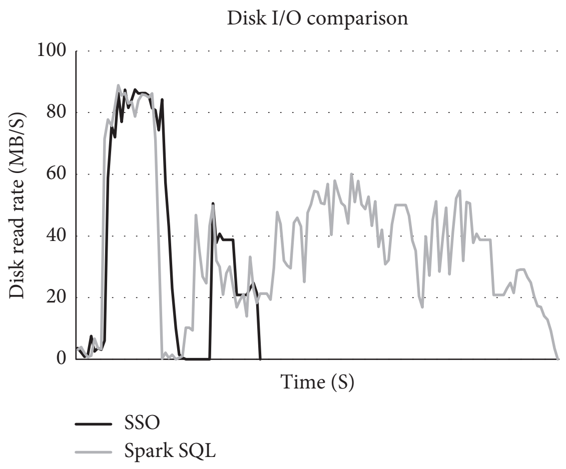


Figure 10: Disk I/O comparison.

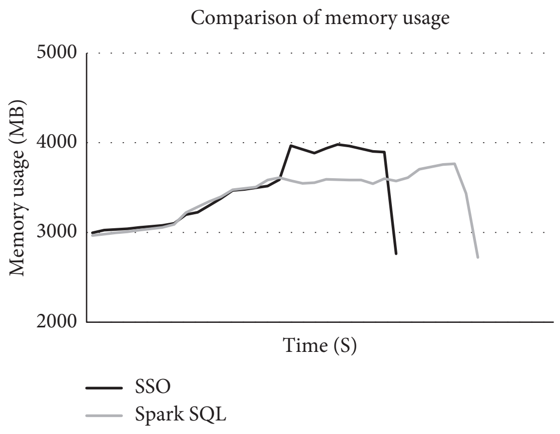


Figure 11: Comparison of memory usage.

thus reducing the amount of memory used by the intermediate data cache layer. To verify the effectiveness of the algorithm, in this section, query Q17 of TPC-H was selected to be submitted to SSO system and the original Spark SQL system respectively. Meanwhile, without using the intermediate data correlation merging algorithm, the original Spark SQL system was tested in two cases: using the intermediate data cache layer or not. The query execution time was recorded and shown in Figure 12.

Results showed that the execution time of SSO system and Spark SQL system with intermediate data cache layer was both shorter than that of Spark SQL system without any optimization, while the execution time of the former two systems tended to be the same. In order to compare the two systems, the monitoring results of the cache size used by each node were shown in Table 3.

The cache usage of SSO system with intermediate data correlation merging algorithm was less than that of Spark SQL system without optimization algorithm in each node. This was because by sharing the same fields l_partkey and l_quantity, intermediate data of 1.5 GB could be reduced and writing the data caused disk cost of 19.2 seconds. Redundant reads of field l_extendedprice and l_suppkey were also generated. Reading the 1.26 GB data caused disk I/O of 16.4 seconds. Comparison showed that the saved cost was greater

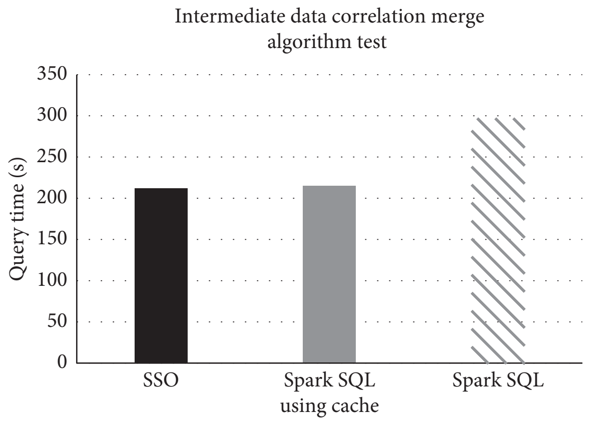


Figure 12: Intermediate data correlation merging algorithm test.

Table 3: Comparison of cache size with and without intermediate data correlation merging algorithm.

Id	1	2	3	4	5	6	7	8	9	10	11
Input data (MB)	256	896	384	1024	1792	1664	640	384	256	640	256
Spark buffer (MB)	227	792.2	340	908.8	1590	1476	568	340.8	227.2	568	227.2
SSO buffer (MB)	178	625.8	268	715.2	1251	1162	447	268.2	178.8	447	178.8

than the generated cost. So, in this case, the algorithm proposed would merge the same intermediate data. However, the intermediate data were read and written in memory after adopting the cache layer and the memory read/write rate was more than dozens of GB per second. On the premise that the cache layer cached all the intermediate data, the amount of cache usage had little effect on task execution efficiency. That was the main reason why the query time of the first two experiments tended to be consistent. But in the case of tight cluster memory resources, the intermediate data correlation merging algorithm could reduce the cache usage, thus saving memory resources effectively.

6. Conclusion

With the maturity of memory computing framework, Spark, as the representative of memory computing framework, has attracted more and more attention from enterprises and research groups. As a bridge between data analysts and Spark systems, Spark SQL plays an important role. To optimize the performance of Spark SQL query, the existing Spark SQL was improved and the SSO prototype system was developed. By adding the Spark Shuffle intermediate data cache layer, the high disk *I*/*O* cost caused by random reading and writing of intermediate data in Shuffle phase was reduced. In order to solve the problem of redundant read-write for intermediate data of Spark SQL, a cost-based correlation merging algorithm was proposed. It was used to determine whether to merge tasks with correlation by weighing the benefits and the extra costs of merging, thus improving the query execution efficiency. The experimental platform was built and the SSO system was developed. The benchmarking tool TPC-H was used to generate test data. The performance comparison with the existing Spark SQL system was carried out to verify the effectiveness of the work in this paper.

Data Availability

All data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the SGCC Science and Technology Program called "Research and Application of Key Technologies of Common Components of New Generation Power Grid Dispatch Control System Platform," and partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization and Collaborative Innovation Center of Wireless Communications Technology.

References

[1] X. Shi, M. Chen, L. He et al., "Mammoth: gearing hadoop towards memory-intensive mapreduce applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 8, pp. 2300–2315, 2015.

[2] M. Zaharia, M. Chowdhury, T. Das et al., "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, USENIX Association, San Jose, CA, USA, April 2012.

[3] K. Shvachko, H. Kuang, S. Radia et al., "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–10, IEEE, Incline Village, NV, USA, May 2010.

[4] Apache HBase, http://hbase.apache.org/.

[5] V. K. Vavilapalli, A. C. Murthy, C. Douglas et al., "Apache hadoop yarn: yet another resource negotiator," in *Proceedings of the 4th Annual Symposium on Cloud Computing*, ACM, Santa Clara, CA, USA, October 2013.

[6] B. Hindman, A. Konwinski, M. Zaharia et al., "Mesos: a platform for fine-grained resource sharing in the data center," *NSDI*, vol. 11, p. 22, 2011.

[7] M. Zaharia, T. Das, H. Li et al., "Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters," *HotCloud*, vol. 12, p. 10, 2012.

[8] R. S. Xin, J. E. Gonzalez, M. J. Franklin et al., "Graphx: a resilient distributed graph system on spark," in *Proceedings of the 1st International Workshop on Graph Data Management Experiences and Systems*, ACM, New York, NY, USA, 2013.

[9] S. Agarwal, B. Mozafari, A. Panda et al., "BlinkDB: queries with bounded errors and bounded response times on very large data," in *Proceedings of the 8th ACM European Conference on Computer Systems*, pp. 29–42, ACM, Prague, Czech Republic, April 2013.

[10] X. Meng, J. Bradley, B. Yavuz et al., "Mllib: machine learning in apache spark," 2015, https://arxiv.org/abs/1505.06807.

[11] M. Armbrust, R. S. Xin, C. Lian et al., "Spark sql: relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1383–1394, ACM, Melbourne, Australia, June 2015.

[12] R. S. Xin, J. Rosen, M. Zaharia et al., "Shark: SQL and rich analytics at scale," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 13–24, ACM, 2013.

[13] A. Thusoo, J. S. Sarma, N. Jain et al., "Hive: a warehousing solution over a map-reduce framework," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1626–1629, 2009.

[14] D. Vengerov, A. C. Menck, M. Zait, and S. P. Chakkappen, "Join size estimation subject to filter conditions," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1530–1541, 2015.

[15] R. Singhal and P. Singh, "Performance assurance model for applications on SPARK platform," in *Proceedings of the Technology Conference on Performance Evaluation and Benchmarking*, pp. 131–214, Springer, Munich, Germany, September 2017.

Hindawi
Scientific Programming
Volume 2020, Article ID 5190138, 13 pages
https://doi.org/10.1155/2020/5190138

Research Article

Android Malware Detection Using Fine-Grained Features

Xu Jiang,[1] **Baolei Mao**,[2] **Jun Guan**,[1] **and Xingli Huang**[3]

[1]*School of Automation, Northwestern Polytechnical University, Xi'an, China*
[2]*Cooperative Innovation Center of Internet Healthcare, Zhengzhou University, Zhengzhou, China*
[3]*College of Computer Science, Wenzhou University, Wenzhou, China*

Correspondence should be addressed to Baolei Mao; maobaolei524@gmail.com

Received 16 October 2019; Revised 25 November 2019; Accepted 18 December 2019; Published 25 January 2020

Guest Editor: Zhiang Wu

Nowadays, Android applications declare as many permissions as possible to provide more function for the users, which also poses severe security threat to them. Although many Android malware detection methods based on permissions have been developed, they are ineffective when malicious applications declare few dangerous permissions or when the dangerous permissions declared by malicious applications are similar with those declared by benign applications. This limitation is attributed to the use of too few information for classification. We propose a new method named fine-grained dangerous permission (FDP) method for detecting Android malicious applications, which gathers features that better represent the difference between malicious applications and benign applications. Among these features, the fine-grained feature of dangerous permissions applied in components is proposed for the first time. We evaluate 1700 benign applications and 1600 malicious applications and demonstrate that FDP achieves a TP rate of 94.5%. Furthermore, compared with other related detection approaches, FDP can detect more malware families and only requires 15.205 s to analyze one application on average, which demonstrates its applicability for practical implementation.

1. Introduction

Smartphones have become an integral part of our day-to-day life. New data for December 2018 shows that Android remains the most popular mobile operating system, with a worldwide market share of 75.16% [1]. With over one million Android applications in major app stores, applications such as WeChat, TikTok, and mobile banking applications are used in our daily life and continue to play an increasingly important role. Most of these applications have access to users' private information such as their location, credit card, and contact information. Almost all applications access the users' private data, although this provides users with better personalized services [2]. It may also result in information leakage of private data and economic loss [3]. Further, Android malicious applications keep emerging endlessly, and this security issue has gained increasing attention in the industry and academic fields [4].

A large body of research against Android malware has been proposed. Currently, static analysis and dynamic analysis are the two main types of detection methods. Each approach has its merits and shortcomings. The static analysis methods such as Kirin [5], PApriori [6], and DREBIN [7] analyze applications without executing the program requiring low overheads. However, the methods cannot defend against antidecompiling and obfuscation. On the contrary, dynamic analysis methods such as TaintDroid [8] and VetDroid [9] execute the application in real time to detect malware, but it is difficult to acquire all the execution paths. With malware being rapidly evolving, the machine learning method is used to perform Android malware detection. Consequently, gathering features that better represent malicious behavior as the features of machine learning is beneficial to improve the performance of malware detection.

By itself, Android has several security mechanisms in its different layers. The permission mechanism applied in the application layer is an important defence mechanism to protect sensitive resources on the Android platform. Applications must declare dangerous permissions to access sensitive data [10, 11]. Several studies have investigated Android malicious applications based on the declared permissions, using permission-based methods [5–7, 12–14]. Although these methods avoid high overhead, they consider the declared permissions as the features of machine learning, which cannot truly reflect the difference between benign applications and malicious applications. Thus, they cannot detect malicious applications that declare only a few, or dangerous permissions, which are also always declared by benign applications.

Compared with permissions as features, application programming interfaces (APIs) represent the entire picture of application behavior provided by the Android system [15]. DroidAPIMiner [16] exploits data flow analysis to extract the numbers of APIs used in malicious applications and benign applications to analyze the difference between them, which is similar to these methods in [17, 18]. In general, the API feature set contains a very large number of features, and therefore, detection methods need extra time to extract the API features and to train detection models. It is worth noting that there is a corresponding relation between permissions and APIs.

In this paper, we present FDP, a lightweight Android malware detection method to mine hidden patterns of malware. According to previous studies, there is considerable difference between malicious applications and benign applications in terms of the declared permissions [19–22]. In addition, some dangerous permissions declared for different components reflect the purpose of the developers [23, 24]. It can be used to distinguish different purposes of the same dangerous permission. Therefore, FDP exploits static analysis to collect more fine-grained and representative features from AndroidManifest.xml and decompiled code; these features include fine-grained permissions, intent filter, and the code features of malicious applications. These features can represent the difference between benign applications and malicious applications.

Experiments with 1700 benign applications from Xiaomi markets and 1600 malicious applications demonstrate the effectiveness of FDP, which achieves a TP rate of 94.5% and only requires 15.052 s to analyze an application on average. The main contributions of our work can be summarized as follows:

(1) We propose a new method to perform Android malware detection based on fine-grained permission mechanism, which represents the difference between malicious applications and benign applications as the features of machine learning, including the information of the dangerous permissions used in the components for the first time.

(2) We present a thorough study on the permissions frequently used by malicious applications and evaluate the importance of permission features used to classify malicious applications and benign applications.

(3) In terms of efficiency, FDP uses static methods to gather all features and analyzes an application in a reasonable time.

(4) The experiments demonstrate that FDP is more effective for detecting more malware families and malicious applications that declare few dangerous permissions or dangerous permissions as those declared by benign applications.

The remainder of this paper is organized as follows. Section 2 introduces the features of malicious applications. Section 3 covers the detection framework, including feature selection, feature extraction, and the fine-grained handle on the features. Section 4 introduces the key points for backtracking. Section 5 discusses the experiments and the results. Section 6 describes related work, and we conclude the paper in Section 7.

2. The Features of Malicious Applications

According to Google's definition, permissions are divided into several protection levels: normal, signature, dangerous, and special permissions. Normal permissions have very little risk to the user's privacy. Signature permissions are granted by Android during installation. Dangerous permissions refer to resources that involve the user's private information, which are shown in Table 1. Special permissions do not work like normal and dangerous permissions, which are particularly sensitive; therefore, most applications do not use them [25]. Malicious applications must exploit dangerous permissions to execute malicious behavior. According to the previous work, the research studies about dangerous permissions play an important role for the detection of malicious applications.

Moreover, malicious applications have some code features. AndroMalShare [26] holds 86,798 malicious applications and presents statistical information of the samples, which includes dynamic loading, native code, and reflection, as shown in Figure 1.

The reasons why malicious applications have the code features are provided as follows:

Dynamic loading: Android supports applications to load additional binaries at run time by dynamic loading. In order to evade static detection, malicious application developers separate the core functionality of the application into independent libraries and load them dynamically [27]. Therefore, malicious applications always employ dynamic loading to hide malicious behavior.

Native code: many developers use the native code as part of their applications to improve the execution efficiency of the code and to increase the difficulty of decompilation. Afonso et al. [28] estimate that 37% of all Android applications have the native code. Most malware is restricted to the detection of bytecode; therefore, malware developers use the native code to implement malicious behavior lest the code of the application is reversed, which means that static analysis cannot analyze the applications including the native code completely [29].

Reflection: since Android applications are composed of the Java code and native code, they have the ability to leverage the reflection mechanism. Malicious applications leverage reflection to invoke APIs corresponding

Table 1: Dangerous permissions.

Permission group	Permission
CALENDAR	READ_CALENDAR WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE CALL_PHONE READ_CALL_LOG WRITE_CALL_LOG ADD_VOICEMAIL USE_SIP PROCESS_OUTGOING_CALLS
SENSORS	BODY_SENSORS
SMS	SEND_SMS RECEIVE_SMS READ_SMS RECEIVE_WAP_PUSH
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

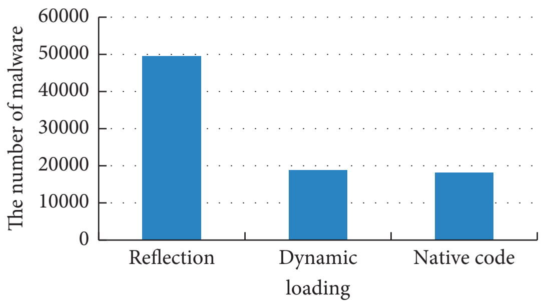


Figure 1: Code feature of malicious applications.

to dangerous permissions to perform malicious behavior, in an attempt to evade static detection [30].

3. Detection Methodology

3.1. Overall Framework. In order to extract features that can better represent application behavior, FDP exploits static analysis to extract key features by decompiling applications and analyzing them using the optimal algorithm of machine learning. The process is illustrated in Figure 2 and the details are provided as follows:

(a) Static analysis: FDP employs the apktool to decompile the application and acquires different feature sets from the AndroidManifest file and the decompiled source code.

(b) Embedding in vector space: all extracted features are mapped to a vector space that can be analyzed as features in machine learning. If the application has the features, the corresponding dimensions are set to 1. Instead, the corresponding dimensions of the other features are set to 0. In order to prevent the problem of overfitting, we take the action of "RemoveDuplicates" to remove the same instances.

(c) Optimal classifier: we use J48 (decision tree algorithm), K-Nearest Neighbor (KNN), Naïve Beyesian (NB), and Support Vector Machines (SVM) to analyze the same training set and choose the classifier with the best indicators as the optimal classifier for FDP.

3.2. Fine-Grained Permissions. The permissions mechanism restricts access to a series of critical APIs [31]. This paper refers to the APIs corresponding to dangerous permissions as sensitive APIs. Several studies choose APIs as the features of machine learning. The classification information is relatively rich, provided that all sensitive APIs are selected as features. However, it is time consuming to process with huge resource consumption and there must exist redundant and meaningless features. Nevertheless, the premise of invoking sensitive APIs is that the applications must declare permissions in the AndroidManifest file. For example, applications must declare the SEND_SMS permission to invoke sendTextMessage() and sendMultipartTextMessage() to send messages. Therefore, FDP selects permissions instead of APIs as the features of machine learning to ensure it is lightweight.

In terms of how to select permissions, it is not suitable to select permissions that are commonly used by both malicious applications and benign applications. If all permissions are used, it will inevitably lead to a "curse of dimensionality." As shown in the previous work [19], malicious applications tend to declare dangerous permissions more often than benign applications. Thus, our research method considers the permissions that are frequently used by malware as the object of our study and extracts the features from the following aspects.

Declared permissions and intent: applications invoking sensitive APIs must declare the corresponding permissions in the AndroidManifest file. We extract these declared permissions after decompiling the application. Activities, Service, and Broadcast Receivers are activated by intent and they register their type of intent using intent-filters in the AndroidManifest file. Malicious applications can exploit the announcement from the Android system to trigger malicious behavior. Therefore, our method extracts the intent that is always used in malicious applications as features, e.g., BATTERY_CHANGE_ACTION and SMS_RECEIVED.

Unused permissions and root privilege: some applications declare dangerous permissions; however, there are no mapped APIs in the Smali code of the application because dangerous permissions are used in dynamic loading or abused by the developers.

Our method first extracts dangerous permissions listed in the AndroidManifest file. Then, we traverse the decompiled source codes based on the relation between the permissions and the APIs provided by PScout [32], to look for declared permissions for which the mapped APIs are not invoked. Such dangerous permissions are defined as unused

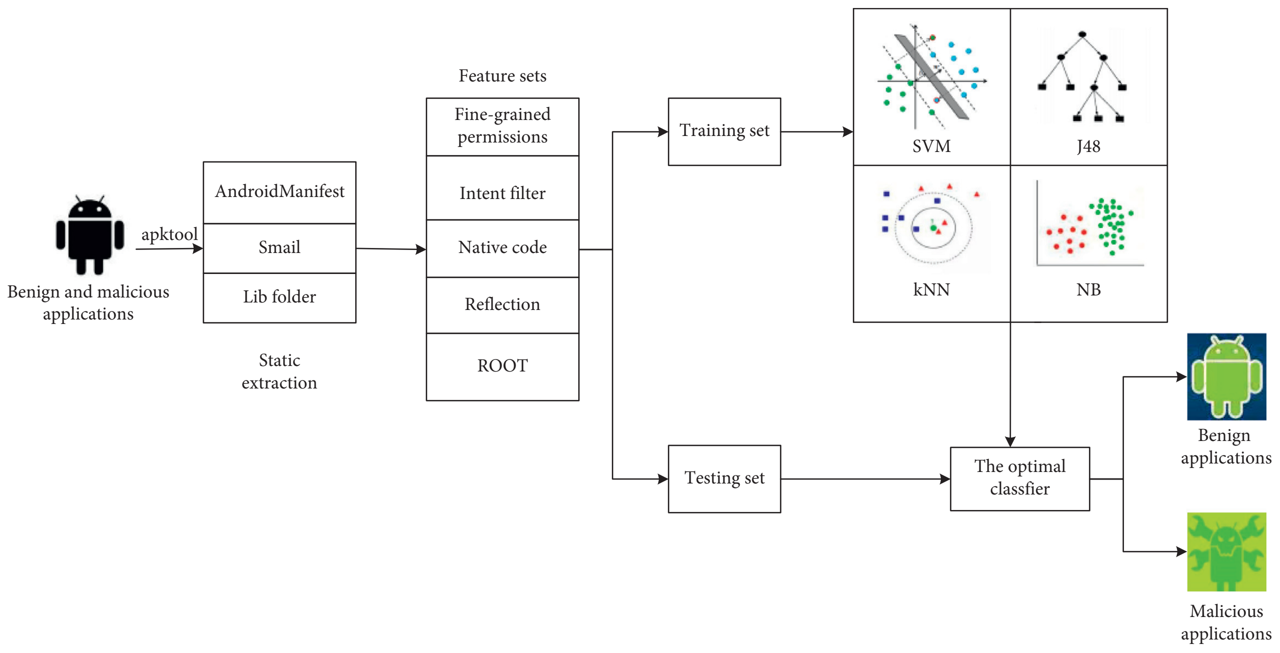


Figure 2: Framework of FDP.

permissions. If there exists sensitive APIs without mapped dangerous permissions in the AndroidManifest file, we suspect that the application is probably attempting to gain root privileges. Because the root privileges give the application absolute control over the device, it can execute any malicious behavior without permissions [33].

Permissions used in components: permissions used in components are the bright spot of our method. In this study, the sensitive APIs are used as the starting point to backtrack the generation of the call graph. According to the relation between the permissions and the APIs provided by PScout, our method scans the Smali code and looks for the mapped APIs. Then, with these APIs as a starting point, the process for the backtracking of parent function is iterated until the component can be identified through the files where a parent function is in.

The specific steps of the generation process of backtracking are as follows:

(1) Traverse all the Smali files in the folder after decompiling, and find out the sensitive API, such as *Landroid/telephony/TelephonyManager;>getDeviceId()Ljava/lang/String*. Then, look for the function that contains the sensitive API, which is the nearest function from the bottom to the top. Taking Figure 3 as an example, the function containing the sensitive API is *getDid()*.

(2) Generate a new parent function according to the sensitive API and the class it is in. We take the new parent function as the starting point, traverse all the Smali files in the folder again and find out the function which calls the newly generated parent function. Taking Figure 3 as an example, it means that we should find out where *getDid()* is called, i.e., *Lcom/qq/e/v2/managers/status/DeviceStatus;->getDid()Ljava/lang/String*.

```
.class public Lcom/qq/e/v2/managers/status/DeviceStatus;
.super Ljava/lang/Object;
......
.method public getDid()Ljava/lang/String;
    .local 2
    ......
    move-result-object v0
    check-cast v0, Landroid/telephony/TelephonyManager;
    invoke-virtual {v0},Landroid/telephony/TelephonyMan-
    ager;>getDeviceId()Ljava/lang/String;
```

Figure 3: Code snippet of Lcom/qq/e/v2/managers/status/DeviceStatus.

(3) Iterate the methods of steps 1 and 2 until we get the component information where the parent function is located. The iteration result of *getDid()* is shown in Figure 4. The statement of "*.super*" displays the component information. The "*.super Landroid/app/Activity*" in the Smali file indicates that the parent function called *Landroid/telephony/TelephonyManager;->getDeviceId() Ljava/Lang/String* belongs to the Activity component.

3.3. Native Code and Reflection. After decompilation, our method checks whether the application has the lib folder that stores the libraries. If the lib folder exists, the feature of the native code is set to 1. In order to alleviate the absence of reflection, FDP extracts APIs related to reflection, such as *Class.forName()*. The feature enables us to know that the hidden code is executed. Therefore, the feature of reflection is set to 1 or 0 depending on the code related to the reflection.

4. The Key of Backtracking

The feature of permissions used in components contributes a lot for FDP to improve the detection performance. Therefore,

```
.class public Lcom/qq/e/ads/AdActivity;
.super Landroid/app/Activity;
# instance fields
.field private a:Lcom/qq/e/v2/plugininterfaces/ActivityDelegate
#direct methods
.method public construction<init>()v
    .locals 0
    invoke-direct {p0}, Landroid/app/Activity;-><init>()v
    Return-void
end method
```

Figure 4: Code snippet of Lcom/qq/e/ads/AdActivity.

our method must backtrack the parent function calling the sensitive API to get the component information. Moreover, the key to the whole backtracking process is that the building process of the call graph cannot be interrupted. However, owing to the communication between components, callback, and other reasons, Android applications have implicit calls, which cause the interruption of a call graph depending on the conventional method. Some situations and solutions are presented as follows.

4.1. Annotations. Annotations are a language feature of Java widely used in the development of Android applications [34]. There are two types of annotations: the dalvik.Annotation package is not open to the public, which is only used for the core library and code testing, and it is under the *Dalvik\src\main\Java\dalvik\annotation* directory; the other one is android.Annotation and the corresponding annotation declaration is under the framework *\base\core\Java\android\annotation* directory. Annotation often appears in the process of backtracking.

Taking the app “net.maxicom.android.snake” as an example, the code snippet is shown in Figure 5. When the detection backtracks the parent function, FDP encounters the annotations code, in which “EnclosingMethod” specifies the scope of its own class. “Method” shows that the annotation acts on a function, and the value tells us that it is located in onCreate () of SnakeService. Therefore, FDP opens the SnakeService file, where the second line is “*.Super Landroid/app/Service.*” Thus, it can be inferred that the sensitive API is in the Service component.

4.2. Multithread Communications. Android is message-driven, and the essence of the message-handling mechanism is that one thread opens a loop to continuously listen to and process messages sent by other threads in turn. If a new thread is created to operate on the main thread, the system will throw an exception. Therefore, an asynchronous callback mechanism Handler is provided in the Android system for communication between threads [35].

As shown in Figure 6, the UI thread is the main thread. The system initializes a Looper object and also creates a MessageQueue associated with it. The Message is the object that the Handler receives and processes. The MessageQueue is the Message queue. Each UI thread can only have one Looper, continually pulling the Message out of the MessageQueue and distributing it to the corresponding Handler.

When Looper gets a new message, handleMessage() is called to process the new message. This Android message-

```
# annotation
.annotation system Ldalvik/annotation/EnclosingMethod;
  value=Lnet/maxicom/android/snake/SnakeService;->onCreate
   ()V
.end annotation
```

Figure 5: Code snippet of “net.maxicom.android.snake.”

handling mechanism causes the call graph to break, which leads to the failure of backtracking. Therefore, our method constructs an instance initialization method that has the special name “*<init>*,” which is supplied by a compiler. Then, we use the method as the new starting point of backtracking.

Taking “ServiceCommunication1.apk” as an example, we backtrack the parent function calling sensitive API until the parent function is “*handleMessage*,” as shown in Figure 7(a). Then, we find that the class inherits the *“Landroid/os/Handler”* based on the codes. If one method constructs “*Ledu/mit/icc_service_message/MessengerService$IncomingHandler;->HandleMessage(Landroid/os/Message;)V,*” the backtracking fails. Our method constructs “*Ledu/mit/icc_service_messages/MessengerService$IncomingHandler;-><init>(Ledu/mit/icc_service_messages/MessengerService;)V”* as the new start point and looks for its location. As shown in Figure 7(b), the second line “*.super Landroid/app/Service;*” shows that the dangerous permission is used in the service component.

4.3. Process and Thread. A process may contain several threads, which can be used as the basic unit of independent operation and independent scheduling. Because threads are smaller than processes and do not own system resources, their scheduling costs are low, which effectively improves the concurrent execution of programs between multiple programs of the system.

There are two ways to implement threads in an Android system. One is to extend the *“java.lang.threads,”* rewrite “*run(),*” and use multiple threads to complete their tasks separately. The other is to implement the Runnable interface and instantiate the thread class. The difference is that the threads created with the Runnable interface can share resources when multiple threads have access to the same resource, whereas the threads created by inheriting the thread class have their own resources.

When the parent function of the sensitive API is “*run(),*” it is necessary to check the keywords “*.Super*” and “*.Implement*” in the Smali file. If there exists “*.super Ljava/lang/Thread*” or “*.Implement Ljava/lang/Runnable*,” we need to traverse all the Smali files again, instantiate the thread and use the function that starts the thread as the starting point for backtracking, and search upward for the parent function. Note that the thread class has the methods of “*run()*” and “*start(),*” whereas the implementation of the Runnable interface only has the “*run()*” method.

Taking “net.maxicom.android.snake” as an example, the application invokes location/LocationManager;->requestLocationUpdates(Ljava/lang/String; JFLandroid/location/LocationListener)V corresponding to the ACCESS_FINE_LOCATION permission, and its parent

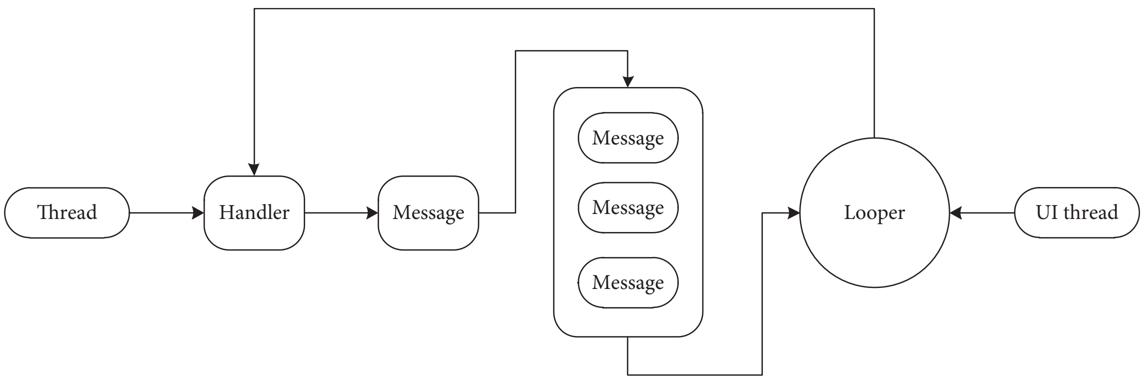


Figure 6: Sketch map of communication with UI thread.

```
.class Ledu/mit/icc_service_message/MessengerService$Incoming
Handler
.super Landroid/os/Handler
.source "MessengerService.java"
..... .
# virtual methods
.method public handleMessage(Landroid/os/Message;)V
```

(a)

```
.class public Ledu/mit/icc_service_messages/MessengerService;
.super Landroid/app/Service;
.source "MessengerService.java"
......
# direct methods
.method public constructor <init>()V
    new-instance v0, Landroid/os/Messenger;
    new-instance v1, Ledu/mit/icc_service_messages/MessengerService$IncomingHandler;
    invoke-direct {v1, p0}, Ledu/mit/icc_service_messages/MessengerService$IncomingHandler;->
    <init>(Ledu/mit/icc_service_messages/MessengerService;)V
```

(b)

Figure 7: Code snippet of SeviceCommunication1.apk.

function is Lnet/max-icom/android/snake/SnakeService$1;->run(V), where the second line of SnakeService$1 shows "*.super Ljava/lang/Thread*" in the "*net/maxicom/android/snake/SnakeService$1*" file. Provided that the method constructs its parent class function by class, method, and parameter as previously addressed, the backtrack fails. We take *Lnet/maxicom/android/snake/SnakeService$1;-> start (V)* as the new starting point for backtracking. Eventually, after scanning and matching, the parent function is *Lnet/maxicm/android/snake/SnakeService;->onCreate()*, and the file is located in *net/maxicom/android/snake/snakeService* where the second line shows that the sensitive API is called in the Service component, as shown in Figure 8.

5. Experimental Results and Discussion

5.1. Experiment Setup

Step 1: build a sample dataset. The malicious application samples come from MalGenome and AndroMalShare [26]. MalGenome is widely used by many researchers which help us compare the results with related approaches. As a remedy, AndroMalShare provides the latest malware samples which can be classified by malware families. We collect 1600 malicious applications as samples, which are classified according to Android malware families. The advantage of the dataset is that we can select specific Android malware families to build the detection model and examine the ability of the detection model detecting unknown applications from the other Android malware families. In order to ensure the balance of the data sets, we employ the crawler program to download 1700 benign applications from the Xiaomi App Store. To further check benign applications and malicious applications, we validate these applications with Virustotal, which is the website that analyzes suspicious files and URLs to detect types of malware, automatically sharing them with the security community.

We select 20 malware families shown in Table 2, including 673 malicious applications and 700 benign applications for building the detection model. The remaining applications are used as the testing set to examine the detection model.

```
.class public Lnet/maxicom/android/snake/SnakeService;
.super Landroid/app/Service;
.source "SnakeService.java"
```

Figure 8: Code snippet of SnakeService.

Step 2: decompile the application using the *apktool* tool to get the Smali code and AndroidManifest file. Then, we extract all features from the Smali code and AndroidManifest file. After further analysis, the fine-grained features are acquired.

Step 3: use the program written in Python to unify the features extracted in Step 2, and unify it into the arff format, which is convenient for machine learning by WEKA tools.

Step 4: compare the classification results of two different sets of permission features to select the features that can better represent the malicious behavior and exploit different machine learning algorithms to learn the data and select the optimal classifier according to the TP Rate, *F*-measure, and Receiver Operating Characteristic curve (ROC) area.

All experiments are carried out on a machine with a memory of 16 GB and Intel (R) Core(TM) i7-4720hq 2.60 GHz processor.

5.2. Evaluation Metrics. In this work, TP Rate, *F*-measure, and ROC area are employed to evaluate the performance of the detection model. As malicious applications are positive samples and benign applications are negative samples in our evaluation, we present four types of values. t_{-}pos is the number of malicious applications correctly identified as malicious applications; f_{-}neg is the number of malicious applications incorrectly identified as benign applications; t_{-}neg is the number of benign applications correctly identified as benign applications; and f_{-}pos is the number of benign applications incorrectly identified as malicious applications:

$$\text{TP Rate} = \text{Recall} = \frac{t_{-}\text{pos}}{t_{-}\text{pos} + f_{-}\text{neg}},$$

$$\text{FP Rate} = \frac{f_{-}\text{pos}}{t_{-}\text{neg} + f_{-}\text{pos}},$$

$$\text{Precision} = \frac{t_{-}\text{pos}}{t_{-}\text{pos} + f_{-}\text{pos}}, \qquad (1)$$

$$F\text{-measure} = \frac{2 * \text{recall} * \text{precision}}{\text{recall} + \text{precision}},$$

$$\text{Accuracy} = \frac{t_{\text{pos}} + t_{\text{neg}}}{t_{\text{pos}} + t_{\text{neg}} + f_{\text{pos}} + f_{\text{neg}}}.$$

In these metrics, *F*-measure is an indicator referring to precision and recall. The ROC area is one of the most important evaluation metrics for checking any classification model's performance. The higher the ROC area, the better the classification.

Table 2: Malware families used in the detection model.

Family	Quantity
BaseBridge	60
BaseBird	39
BeanBot	6
KMin	40
GoneSixty	6
Fakeinst	16
DroidDream	45
Lotoor	8
Pjapps	44
SendPay	34
Geinimi	48
Bgserv	8
DDLight	40
HippoSMS	4
GoldDream	10
GingerMaster	7
DroidKungFu1	30
DroidKungFu2	20
DroidKungFu3	152
DroidKungFu4	56

5.3. Selection of Features. In order to achieve a better detection performance, choosing informative, discriminating, and independent features is a crucial step for classification. We design two experiments to compare the permission sets and use the testing set to validate. Experiment I: select all dangerous permissions as features of machine learning; Experiment II: select the top 20 used permissions of 86798 malware samples collected by AndroMalShare as the features of machine learning, as shown in Figure 9.

Experiment I chooses 927 malicious applications and 1000 benign applications that do not contribute to the detection model as a testing set. The result of dangerous permissions as features of machine learning is shown in Table 3. KNN and SVM are relatively better than other classifiers, and they have the highest TP Rate in malware detection. However, the performances of the four classifiers are not very satisfactory.

In Experiment II, we use the same training set to build the detection model and analyze the same testing set by the same algorithms. The only difference is that the features are the top 20 used permissions of 86,798 malicious applications collected by AndroMalShare. The experimental result is shown in Table 4.

It can be seen from the results that the SVM algorithm is the best in the TP Rate, reaching 90.9%. To compare the results in Tables 3 and 4, similar performances are observed for the same testing set and classifier. It is difficult to evaluate which permission set is better; however, we can identify the features which are more important for classification depending on the information gain of the features. The two sets of results are shown in Tables 5 and 6, respectively.

According to the ranked features and the actual usage status of permissions, we choose the 24 permissions as features from the two sets, as shown in Table 7. And we find that the permissions, that is, the requirement of executing malicious behavior may not be the significant like

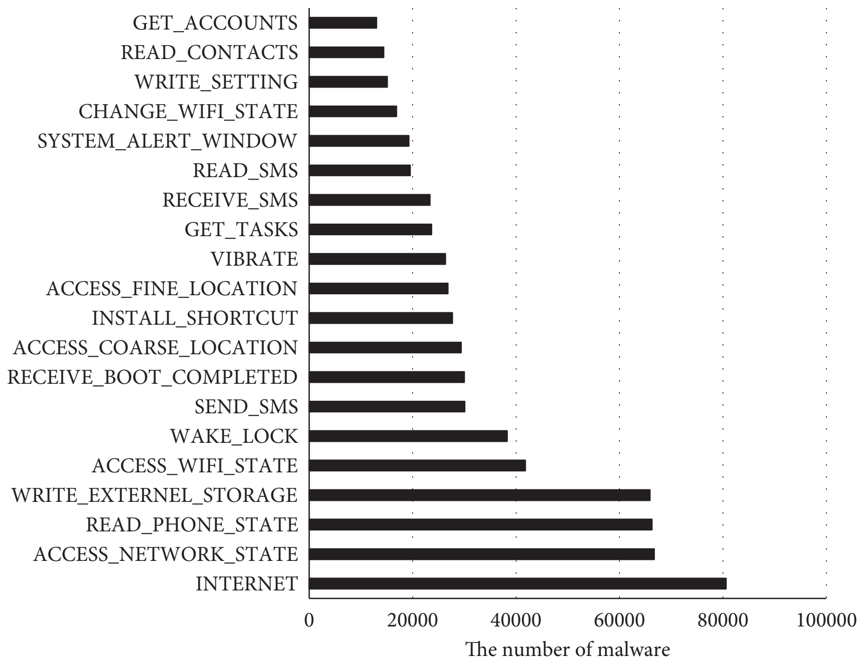


Figure 9: The top 20 used permissions of 86798 malicious applications.

Table 3: Detection rates with dangerous permissions of features.

	TP rate	FP rate	*F*-measure	ROC area
J48	0.877	0.149	0.877	0.813
KNN	0.899	0.091	0.901	0.937
SVM	0.899	0.099	0.900	0.900
NB	0.888	0.105	0.890	0.946

Table 4: Detection rates with the top 20 permissions of malware samples of features.

	TP rate	FP rate	*F*-measure	ROC area
J48	0.867	0.185	0.847	0.892
KNN	0.895	0.144	0.886	0.946
SVM	0.909	0.124	0.909	0.948
NB	0.888	0.103	0.891	0.942

INTERNET permission, which is used frequently by malware and benign applications.

Then, we employ the selected permission features as features of machine learning to learn the training set and analyze the same applications from the test set with the same classifiers. The result is shown in Table 8, with all indicators being superior to those in the above experiments. In these algorithms, the highest TP Rate achieved is 93.8% and the *F*-measure is the best with the J48 classifier.

5.4. FDP. Through further analysis on classifier errors, we find that most of these applications are wrongly classified as they declare dangerous permissions that are frequently declared by malware applications and benign applications, especially, READ_PHONE_STATE, ACCESS_COARSE_-LOCATION, ACCESS_FINE_LOCATION, SEND_SMS, and READ_SMS. Although these dangerous permissions are exactly same, the usage of these permissions is different between benign applications and malicious application. For example, the information gain of READ_PHONE_STATE is relatively low, but malicious applications often employ it in service component in order to avoid any users' attention which is different from benign applications.

Based on the previous work [23, 24], we further refine the above permissions and extract the information of

Table 5: Information gain of dangerous permissions.

Information gain	Permission
0.45761	CAMERA
0.38322	READ_EXTERNAL_STORAGE
0.23817	RECORD_AUDIO
0.18864	ACCESS_COARSE_LOCATION
0.16828	ACCESS_FINE_LOCATION
0.14783	GET_ACCOUNTS
0.12662	WRITE_EXTERNAL_STORAGE
0.09001	READ_CALENDAR
0.07381	READ_CALL_LOG
0.05988	WRITE_CALENDAR
0.02238	READ_PHONE_STATE
0.01669	READ_CONTACTS
0.01604	PROCESS_OUTGOING_CALLS
0.01390	WRITE_CALL_LOG
0.01322	CALL_PHONE
0.01233	READ_SMS
0.00565	SEND_SMS
0.00553	RECEIVE_WAP_PUSH
0.00343	BODY_SENSORS
0.00343	USE_SIP
0.00343	ADD_VOICEMAIL
0.00329	RECEIVE_SMS
0.00001	WRITE_CONTACTS

Table 6: Information gain of the top 20 permissions of 86,798 malware samples.

Information gain	Permission
0.33442	GET_TASKS
0.32002	SYSTEM_ALERT_WINDOW
0.29962	WRITE_SETTING
0.21801	WAKE_LOCK
0.1655	ACCESS_WIFI_STATE
0.1588	CHANGE_WIFI_STATE
0.14685	ACCESS_COARSE_LOCATION
0.12524	ACCESS_FINE_LOCATION
0.11073	GET_ACCOUNTS
0.09772	ACCESS_NETWORK_STATE
0.07362	VIBRATE
0.06159	WRITE_EXTERNAL_STORAGE
0.03961	READ_SMS
0.02997	INSTALL_SHORTCUT
0.00644	READ_CONTACTS
0.00546	READ_PHONE_STATE
0.00531	SEND_SMS
0.00399	RECEIVE_BOOT_COMPLETED
0.00314	RECEIVE_SMS
0.00154	INTERNET

Table 7: The 24 selected permissions.

CAMERA	GET_TASKS
WAKE_LOCK	READ_SMS
VIBRATE	WRITE_CALENDAR
READ_PHONE_STATE	READ_CONTACTS
PROCESS_OUTGOING_CALLS	WRITE_CALL_LOG
SYSTEM_ALERT_WINDOWS	WRITE_SETTING
READ_CALENDAR	READ_EXTERNAL_STORAGE
ACCESS_WIFI_STATE	ACCESS_COARSE_LOCATION
RECORD_AUDIO	WRITE_EXTERNAL_STORAGE
GET_ACCOUNTS	READ_CALL_LOG
CALL_PHONE	ACCESS_NETWORK_STATE
ACCESS_FINE_LOCATION	CHANGE_WIFI_STATE

components where the dangerous permissions used as the feature for the first time. And taking the code feature of malicious applications, intent that malicious applications are frequently used, and whether the application exploits native code, reflection mechanism, and root into consideration, the features of FDP are shown in Table 9, in which we add SEND_SMS and merge ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION into LOCATION, SEND_SMS and READ_SMS into SMS in view of curse of dimensionality.

FDP also uses the same data sets and classifiers for analysis. The result is shown in Table 10. Every algorithm has better performance than before. In these classifiers, J48 is the best and its TP Rate and *F*-measure are 0.945. It demonstrates that the features of FDP can better represent the malicious behavior.

Besides, the results of FDP are also shown as a ROC area in Figure 10. The area below the ROC curve is the value of the area under the curve (AUC). The larger the area, the better the classification. The results are obvious that these four classifiers have good performance in detecting malicious applications and all ROC area are over 0.9.

Table 8: Detection rates of the 24 selected permissions.

	TP rate	FP rate	*F*-measure	ROC area
J48	0.938	0.062	0.938	0.968
KNN	0.921	0.101	0.920	0.970
SVM	0.916	0.074	0.916	0.921
NB	0.918	0.107	0.917	0.949

5.5. The Evaluation of Dangerous Permissions Used in Component. In order to evaluate the importance of the feature of dangerous permissions used in components, we collect malicious applications from Android malware families, such as FakePlayer, DroidCoupon, TapSnake, and Plankton. These applications only declare few dangerous permissions such as READ_PHONE_STATE, ACCESS_-COARSE_LOCATION, and READ_SMS which are frequently used by benign applications. This means there is not enough available information for classification if the detection methods are based on the declared permissions.

For comparison, we employ two sets of the permission features. The features in Table 7 are taken as one of two sets, and the other is that the features of dangerous permissions applied in components replace the declared permissions based on the features of Table 7, which involve READ_PHONE_STATE, ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, and READ_SMS. For example, READ_PHONE_STATE is replaced by A-READ_PHONE_STATE, B-READ_PHONE_STATE, and S-READ_PHONE_STATE.

By testing 63 malicious applications and 70 benign applications that declare similar permissions, we perform 10-fold cross-validation using SVM algorithm. The accuracy of detection is 77.8% with the declared permissions, while the accuracy of detection reaches 92.5% with the fine-grained permissions. The scale of two data sets is relatively small because there are too few eligible benign applications.

In addition, we use information gain to evaluate the importance of features including the fine-grained permissions. As shown in Table 11, there are three features of dangerous permissions used in components ranked in the top fourteen of permissions. The result demonstrates the feature of permissions used in components contributes to the classification.

5.6. Detection of Unknown Malware Families. In order to examine the prediction ability of FDP, we collect the malware families as the unknown malware families shown in Table 12. These applications are not in the training set for constructing the detection model, which include the malware families that PApriori and DREBIN cannot detect.

We use three detection models to identify the same applications from unknown malicious families. The three detection models employ dangerous permissions as the features of machine learning (i.e., Experiment I), the top 20 used permissions of malicious applications declared as the features of machine learning (i.e., Experiment II), and FDP, respectively. The experimental results show that FDP can accurately detect more malicious families, especially with few dangerous permissions, which are shown in Figure 11. For instance, all the malicious

Table 9: The features of FDP.

CAMERA	D-CAMERA	WAKE_LOCK	REFLECTION
RECORD_AUDIO	D-RECORD_AUDIO	GET_TASKS	NATIVE CODE
READ_CONTACTS	D-READ_CONTACTS	VIBRATE	ROOT
READ_CALL_LOG	D-READ_CALL_LOG	PROCESS_OUTGOING_CALLS	I-BOOT_COMPLETED
CALL_PHONE	D-CALL_PHONE	SYSTEM_ALERT_WINDOWS	I-SMS_RECEIVED
WRITE_CALL_LOG	D-WRITE_CALL_LOG	WRITE_SETTING	I-BATTERY_CHANGE_ACTION
CHANGE_WIFI_STATE	D-CHANGE_WIFI_STATE	D- SMS	A-LOCATION
READ_CALENDAR	D-READ_CALENDAR	A- SMS	D-LOCATION
ACCESS_WIFI_STATE	D-ACCESS_WIFI_STATE	B- SMS	B-LOCATION
GET_ACCOUNTS	D-GET_ACCOUNTS	S- SMS	S-LOCATION
WRITE_EXTERNAL_STORAGE	D-WRITE_EXTERNAL_ STORAGE	D-READ_PHONE_STATE	S-READ_PHONE_STATE
READ_EXTERNAL_STORAGE	D-READ_EXTERNAL_ STORAGE	A-READ_PHONE_STATE	B-READ_PHONE_STATE
ACCESS_NETWORK_STATE	D-ACCESS_NETWORK_ STATE	D- WRITE_CALENDAR	WRITE_CALENDAR

A: activity, B:broadcast receiver, D: unused or dynamical loading, I: intent.

Table 10: Detection rates of FDP.

	TP rate	FP rate	*F*-measure	ROC area
J48	0.945	0.061	0.945	0.939
KNN	0.937	0.078	0.936	0.963
SVM	0.929	0.088	0.928	0.920
NB	0.920	0.095	0.919	0.953

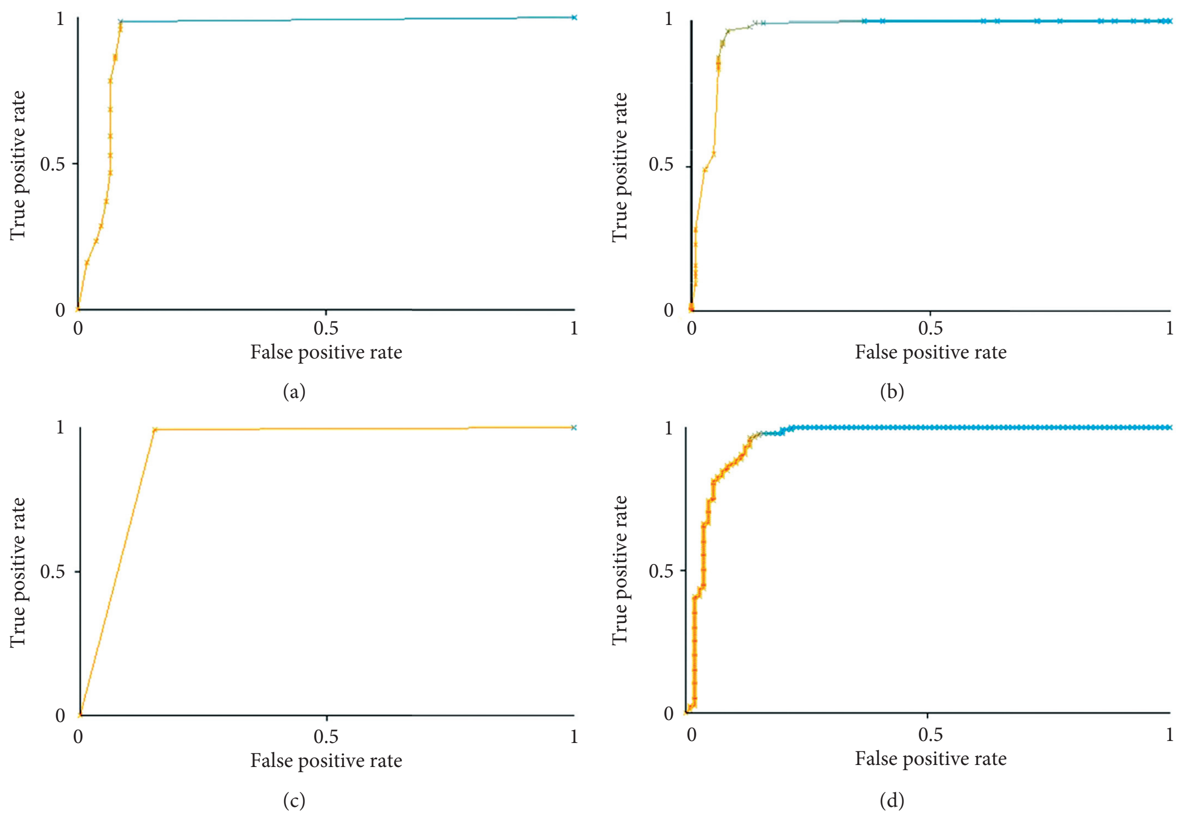


Figure 10: Detection performance as ROC area: (a) J48, (b) KNN, (c) SVM, and (d) NB.

Table 11: The top fourteen features of FDP according to information gain.

Permission	Information gain
S-READ_PHONE_STATE	0.55678
READ_EXTERNAL_STORAGE	0.41997
GET_ACCOUNTS	0.41997
GET_TASKS	0.41997
RECORD_AUDIO	0.41997
B-READ_PHONE_STATE	0.28129
CALL_PHONE	0.28129
CHANGE_WIFI_STATE	0.28129
VIBRATE	0.28129
READ_CONTACTS	0.28129
WRITE_SETTING	0.25643
S-READ_SMS	0.17095
PROCESS_OUTGOING_CALLS	0.17095
READ_CALENDAR	0.17095

Table 12: Malware families of the testing set.

Id	Family	Quantity
A	Asroot	6
B	FakePlayer	31
C	DroidCoupon	4
D	DroidDeluxe	1
E	Tapsnake	4
F	Gappusin	21
G	JsmsHider	23
H	RogueLemon	9
I	Plankton	24
J	Zsone	15
K	YZHC	8
L	JiFake	20

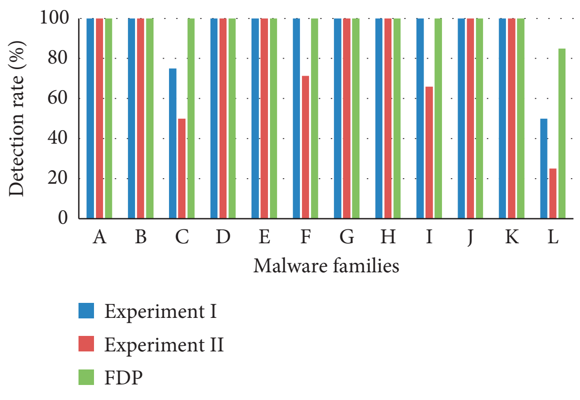


Figure 11: Detection of unknown malicious families.

applications from DroidCoupon family only declare READ_-PHONE_STATE and WRITE_EXTERNAL_STORAGE; all the malicious applications from the Plankton family only declare READ_PHONE_STATE. It demonstrates that FDP can identify more malicious applications owing to the fine-grained permission features.

5.7. Comparison with Related Approaches. PApriori uses frequent pattern mining to get the maximum frequent permissions of 49 malware families and constructs the permission characteristics library to detect malicious applications. For 1260 malware applications from MalGenome, PApriori detects 87% of the malware samples. However, when malicious applications declare less dangerous permissions as benign applications, PApriori fails. PApriori lists malware families from MalGenome undetected by itself, such as Asroot, FakePlayer, DroidCoupon, DroidDeluxe, Tapsnake, and Plankton. And that is why PApriori has obvious limitations in detecting benign applications. FDP can detect all the above malicious families because of the fine-grained selected dangerous permissions, which represents the difference between benign applications and malicious applications.

DREBIN detects malware with the accuracy of 94%, but it cannot detect malicious applications from the Gappusin family because there are too few malicious features to identify the sample. We employ applications from Gappusin family as unknown malicious applications and the TP Rate gets 100%. Meanwhile, there is the problem of unbalanced data in the data sets of DREBIN in which it contains 123,453 benign applications and 5,560 malicious applications. If DREBIN chooses the balanced data sets, the accuracy may improve.

DroidEnsemble employs string features such as permissions and intent and structural features to detect malware applications. The size of the dataset and the composing proportion of benign applications and malicious applications are similar to FDP, and the accuracy reaches 98.4%. It is worth noting that structural features may perform poorly in detecting unknown malicious families because it relies heavily on previous identified malicious families. Therefore, it has the same problem as PApriori and DREBIN that the detection cannot take effect if there are not obvious differences between benign applications and malicious applications for the declared permissions.

5.8. Run-Time Performance. In order to evaluate the applicability of practical implementation of our method, we calculate the time consumption using 50 malicious applications and 50 benign applications, and the size of the applications is from 12 KB to 61 MB. On average, FDP can analyze one application within 15.205 s, which demonstrates that our method is efficient in Android malware detection, though there are many fine-grained information to extract.

6. Related Work

There are a lot of research studies related to FDP, which exploits static analysis to extract features such as permissions, API calls, and intent for detection.

According to the differences between malicious apps and benign apps, DroidRanger [36] summarized the rule of permission characteristics to detect unknown malicious apps. Enck et al. developed Kirin [5], which created nine security rules for security. Flet et al. proposed stowaway [37] to detect whether the app declares excessive permission based on the analysis of the mapping relationship between permissions and APIs. DREBIN [7] employed a static method to gather as many features of an application as possible and use SVM to analyze them. Feizollah et al. [38] proposed the method to detect the malicious applications based on intents and permissions. DroidEnsemble [39] employed string features and structural features for identifying Android malicious applications and optimized the results with ensemble methods.

In this article, FDP differs in three aspects from the previous work. First, we choose the features depending on the characteristic of malicious applications in order to better represent the malicious behavior. Second, the permission features are further subdivided into fine-grained permission features, which enable FDP to detect more malware families, especially, when malicious applications declare few permissions or when the dangerous permissions declared by malicious applications are similar with those declared by benign applications. Third, FDP uses static analysis to extract fine-grained dangerous permission and other features, and it uses the J48 as the optimal classifier to detect malicious applications with TP Rate and *F*-measure reaching 94.5% and analyze an application in a reasonable time.

7. Conclusion and Future Work

In this paper, we propose a new method based on the fine-grained permissions for detecting Android malicious applications, which gathers the features that better represent the differences between malicious applications and benign applications. The experimental results demonstrate the effectiveness of our FDP method, which shows that FDP can detect more Android malicious families than existed methods. Moreover, the FDP method is efficient enough for practical implementation of Android malware detection.

Although FDP makes a breakthrough compared with the previous work, it still cannot extract more information from dynamic loading, reflection mechanism, and encryption because of inherent limitations of static analysis. Provided that FDP considers these limitations, it must use dynamic analysis method to extract related features, which increases the detection overhead significantly. In addition, more and more android applications try to protect themselves from decompiling, which also increases the difficulty for using our method.

Therefore, in the premise of efficiency, how to extract the valuable features from dynamic analysis and overcoming the antidisassemble problem is the direction of our future work. Besides, we only use the classic machine learning algorithms in this article, we will try to optimize these machine learning algorithms to construct a better detection model in our future work. Meanwhile, the training data is one important factor to affect the machine learning algorithms to recognize the features pattern. We try to collect more malicious applications and malware families in order to solve the problem of overfitting and underfitting in the future work.

Data Availability

The data used to support the findings of this study are available from AndroMalShare [26] and MalGenome. The applications from MalGenome are available from corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

Xu Jiang proposed ideas and implemented and experimented with the system. Baolei Mao was responsible for the elaboration of ideas and comments on research. Jun Guan and Xingli Huang were responsible for a basic simulation to validate the proposed idea.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No. 61672433).

References

[1] World's Most Popular Mobile Operating Systems (Android VS IOS: Market Share 2012–2018), June 2019, https://ceoworld.b-iz/2019/01/18/worlds-most-popular-mobile-operating-syste-ms-android-vs-ios-market-share-2012-2018.

[2] X. Liu, J. Liu, S. Zhu, W. Wang, and X. Zhang, "Privacy risk analysis and mitigation of analytics libraries in the android ecosystem," *IEEE Transactions on Mobile Computing*, p. 1, 2019.

[3] Research and Analysis Report on Online Privacy and Online Fraud in 2018, June 2019, https://m.qq.com/security_lab/news_deta-il_473.html.

[4] İ. Doğru and Ö. Kiraz, "Web-based android malicious software detection and classification system," *Applied Sciences*, vol. 8, no. 9, p. 1622, 2018.

[5] W. Enck, M. Ongtang, and P. Mcdaniel, "On lightweight mobile phone application certification," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pp. 235–245, Chicago, USA, November 2009.

[6] H. Yang, Y. Zhang, Y. Hu et al., "Android malware detection method based on permission sequential pattern mining algorithm," *Journal on Communications*, vol. 34, pp. 106–115, 2013.

[7] D. Arp, M. Spreitzenbarth, M. Hubner et al., "Drebin: effective and explainable detection of android malware in your pocket," in *Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS)*, pp. 23–26, San Diego, USA, February 2014.

[8] W. Enck, P. Gilbert, S. Han et al., "TaintDroid: an information-flow tracking system for real time privacy monitoring on smartphones," *ACM Transactions on Computer Systems*, vol. 32, no. 2, pp. 1–29, 2014.

[9] Y. Zhang, M. Yang, B. Xu et al., "Vetting undesirable behaviors in android apps with permission use analysis," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS)*, pp. 611–622, Berlin, Germany, November, 2013.

[10] S. Kumar, R. Shanker, and S. Verma, "Context aware dynamic permission model: a retrospect of privacy and security in android system," in *Proceedings of the International Conference on Intelligent Circuits and Systems (ICICS)*, pp. 324–329, Phagwara, India, April 2018.

[11] S. Rosen, Z. Qian, and Z. M. Mao, "AppProfiler: a flexible method of exposing privacy-related behavior in android applications to end users," in *Proceedings of the third ACM Conference*

on Data and Application Security and Privacy—CODASPY '13, pp. 221–232, San Antonio, USA, February 2013.

[12] O. Yildiz and I. A. Doğru, "Permission-based android malware detection system using feature selection with genetic algorithm," *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 2, pp. 245–262, 2019.

[13] R. S. Arslan, I. A. Doğru, N. Barişçi, and N. Barişçi, "Permission-based malware detection system for android using machine learning techniques," *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 1, pp. 43–61, 2019.

[14] W. Wang, X. Wang, F. Dawai, J. Liu, Z. Han, and X. Zhang, "Exploring permission-induced risk in android applications for malicious application detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 11, pp. 1869–1882, 2014.

[15] Z. Ma, H. Ge, Y. Liu et al., "A combination method for android malware detection based on control flow graphs and machine learning algorithms," *IEEE Access*, vol. 7, pp. 21235–21245, 2019.

[16] Y. Zhao, W. Du, and H. Yin, "DroidAPIMiner: mining API-level features for robust malware detection in android," in *Proceedings of the International Conference on Security and Privacy in Communication Systems*, pp. 86–103, Sydney, Australia, September 2013.

[17] A. K. Singh, C. D. Jaidhar, and M. A. A. Kumara, "Experimental analysis of android malware detection based on combinations of permissions and API-calls," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 4, pp. 1–10, 2019.

[18] V. Avdiienko, K. Kuznetsov, A. Gorla et al., "Mining apps for abnormal usage of sensitive data," in *Proceedings of the International Conference on Software Engineering (ICSE)*, pp. 426–436, Florence, Italy, May 2015.

[19] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine-learning-based android malware detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.

[20] H. Rathore, S. K. Sahay, P. Chaturvedi et al., "Android malicious application classification using clustering," in *Proceedings International Conference on Intelligent Systems Design and Applications*, pp. 659–667, Vellore, India, December 2018.

[21] P. Rovelli and V. Ýmir, "PMDS: permission-based malware detection system," in *Proceedings International Conference on Information Systems Security*, pp. 338–357, Hyderabad, India, December, 2014.

[22] V. Moonsamy, J. Rong, and S. Liu, "Mining permission patterns for contrasting clean and malicious android applications," *Future Generation Computer Systems*, vol. 36, pp. 122–132, 2014.

[23] D. J. Wu, C. H. Mao, H. M. Lee et al., "DroidMat: android malware detection through manifest and API calls tracing," in *Proceedings International Conference the Seventh Asia Joint Conference on Information Security*, pp. 62–69, Tokyo, Japan, August 2012.

[24] G. Tao, Z. Zheng, Z. Guo, and M. R. Lyu, "MalPat: mining patterns of malicious and benign android apps via permission-related APIs," *IEEE Transactions on Reliability*, vol. 67, no. 1, pp. 355–369, 2018.

[25] Permission Overview, June 2019, https://developer.android.com/guide/topics/permissions/overview.

[26] AndroMalShare, June 2019, http://andromalshare.org:8080/#overview.

[27] Z. Qu, S. Alam, C. Yan et al., "DyDroid: measuring dynamic code loading and its security implications in android applications," in *Proceedings of the 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 415–426, Denver, USA, June 2017.

[28] V. M. Afonso, M. F. de Amorim, A. R. A. Grégio, G. B. Junquera, and P. L. de Geus, "Identifying android malware using dynamically obtained features," *Journal of Computer Virology and Hacking Techniques*, vol. 11, no. 1, pp. 9–17, 2015.

[29] F. Wei, X. Lin, X. Ou et al., "JN-SAF: precise and efficient NDK/JNI-aware inter-language static analysis framework for security vetting of android applications with native code," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1137–1150, Toronto, Canada, October 2018.

[30] L. Li, T. F. Bissyandé, D. Octeau et al., "DroidRA: taming reflection to support whole-program analysis of android apps," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, pp. 318–329, Saarbrücken, Germany, July 2016.

[31] Y. Liu, K. Guo, X. Huang, Z. Zhou, and Y. Zhang, "Detecting android malwares with high-efficient hybrid analyzing methods," *Mobile Information Systems*, vol. 2018, Article ID 1649703, 12 pages, 2018.

[32] K. W. Y. Au, Y. F. Zhou, Z. Huang et al., "PScout: analyzing the android permission specification," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pp. 217–228, Raleigh, USA, October 2012.

[33] X. Hu, Q. Xi, and Z. Wang, "Monitoring of root privilege escalation in android kernel," in *Proceedings of the Cloud Computing and Security*, pp. 491–503, Haikou, China, June 2018.

[34] Q. Yang, G. Peng, P. Gasti et al., "MEG: memory and energy efficient garbled circuit evaluation on smartphones," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 913–922, 2019.

[35] S. Han, Y. Yun, and Y. H. Kim, "Profiling-based task graph extraction on multiprocessor system-on-chip," in *Proceedings of the 2016 IEEE Asia Pacific Conference on Circuits and Systems*, pp. 510–513, Jeju, South Korea, October 2016.

[36] Y. Zhou, Z. Wang, W. Zhou et al., "Hey, you, get off of my market: detecting malicious apps in official and alternative android markets," in *Proceedings of the Network & Distributed System Security Symposium*, pp. 50–52, San Diego, USA, February 2012.

[37] A. P. Felt, E. Chin, S. Hanna et al., "Android permissions demystified," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pp. 627–638, Chicago, USA, October 2011.

[38] A. Feizollah, N. B. Anuar, R. Salleh, G. Suarez-Tangil, and S. Furnell, "AndroDialysis: analysis of android intent effectiveness in malware detection," *Computers & Security*, vol. 65, pp. 121–134, 2017.

[39] W. Wei, Z. Gao, M. Zhao et al., "DroidEnsemble: detecting android malicious applications with ensemble of string and structural static features," *IEEE Access*, vol. 6, pp. 31798–31807, 2018.

Hindawi
Scientific Programming
Volume 2020, Article ID 2424381, 11 pages
https://doi.org/10.1155/2020/2424381

Research Article

AVBH: Asymmetric Learning to Hash with Variable Bit Encoding

Yanduo Ren, Jiangbo Qian, Yihong Dong, Yu Xin, and Huahui Chen

Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo, Zhejiang 315211, China

Correspondence should be addressed to Jiangbo Qian; qianjiangbo@nbu.edu.cn

Received 16 October 2019; Accepted 10 December 2019; Published 21 January 2020

Guest Editor: Aibo Song

Nearest neighbour search (NNS) is the core of large data retrieval. Learning to hash is an effective way to solve the problems by representing high-dimensional data into a compact binary code. However, existing learning to hash methods needs long bit encoding to ensure the accuracy of query, and long bit encoding brings large cost of storage, which severely restricts the long bit encoding in the application of big data. An asymmetric learning to hash with variable bit encoding algorithm (AVBH) is proposed to solve the problem. The AVBH hash algorithm uses two types of hash mapping functions to encode the dataset and the query set into different length bits. For datasets, the hash code frequencies of datasets after random Fourier feature encoding are statistically analysed. The hash code with high frequency is compressed into a longer coding representation, and the hash code with low frequency is compressed into a shorter coding representation. The query point is quantized to a long bit hash code and compared with the same length cascade concatenated data point. Experiments on public datasets show that the proposed algorithm effectively reduces the cost of storage and improves the accuracy of query.

1. Introduction

Given a query object/point $\mathbf{q}$ and a dataset $\mathbf{S}$, the nearest neighbour search (NNS) [1–3] is to return the nearest neighbours in $\mathbf{S}$ to $\mathbf{q}$. Nowadays, the NNS is widely used in many applications such as image retrieval, text classification, and recommendation systems. However, with the exponential growth of data scale and the disaster of the high data dimensionality, the NNS problem is now much more difficult to solve than before. Therefore, new efficient index structures and query algorithms for similarity searches have increasingly become the focus of research for the problem.

The hashing-based NNS methods [3–5] have attracted much attention. Generally, the hashing methods can project the original data with locality preserved to a low-dimensional Hamming space, i.e., binary codes [4–6]. The complexity of those methods is always in sublinear time. In addition, the hashing methods only need a simple bit operation to compute the similarity from Hamming encoding, which is very fast. As the high performance in large-scale data retrieval, hashing techniques have gained increasing interests in facilitating cross-view retrieval tasks [7, 8], online retrieval tasks [9], and metric learning tasks [10].

For large-scale data retrievals, the time and space costs are the two important issues. As we know, the accuracy of existing hash methods is limited by the length of hash encoding and usually requires a longer coding to get better accuracy. However, a long coding will increase the space cost, network communication overhead, and response time.

In order to solve this problem, a coding quantization mechanism [11] based on asymmetric hashing algorithm [12] was proposed. Different from the direct hash code comparison, by cascade concatenating the coding of the data point to the same encoding length of the query point, the coding storage cost of the dataset is reduced effectively and the accuracy of the result is ensured. However, this algorithm uses a unified compression method for all data, ignoring the effect of data distribution. Actually, the distribution of large-scale data is generally uneven. Hence, for most hashing algorithms, the frequency of quantization is also different. As we know, longer encoding can preserve most of the original information; however, it will bring high cost and vice versa. A careful trade-off among accuracy, computing overhead, and space-saving needs to be studied. Intuitively, high-density data require longer length encoding to ensure that the original information is preserved as much as possible, while low-density data can use shorter length encoding and still preserve most of the original information. That is the idea behind our algorithm.

In this paper, an asymmetric learning to hash with variable bit encoding algorithm (AVBH) is proposed. The AVBH uses two types of hash mapping functions to quantify the dataset and the query set separately to encode the hash codes with different length bits. In particular, the frequency of dataset is calculated by random Fourier encoding, and then the random Fourier coding with high frequency is compressed into a longer hash code representation, and the random Fourier coding with low frequency is compressed into a shorter hash code representation.

The main contributions of this paper are as follows: (1) a variable bit encoding mechanism (named AVBH) based on hash code frequency compression is proposed, which makes the encoding space effectively used, and (2) the experiment shows that the AVBH can effectively reduce the storage cost and improve the query accuracy.

2. Preliminaries and Description

In this section, we review some basic knowledge of LSH (locality-sensitive hashing) [13–15], vector quantization [16], and product quantization [17] that is essential to our proposed technique.

2.1. Vector Quantization. Vector quantization (VQ) is a classical data compression technique, which compresses the original data into discrete vectors. For a vector $\mathbf{x}$ of n dimensions, formally, a VQ function f can specified as $f(\mathbf{x}) \in \mathbf{C} = \{\mathbf{c}_i,\ i = 1, 2, \ldots, k\}$, where $\mathbf{x}$ (with n dimensions) is an original data/vector, $\mathbf{C}$ is a pretrained code set, and $\mathbf{c}_i$ is a codeword in the codebook $\mathbf{C}$. The objective of a VQ function is to quantify the original real number vector to the nearest codeword with the lowest VQ loss. Here, the VQ loss of vector $\mathbf{x}$ is given by

$$E_{vq}(\mathbf{x}) = \min_{\mathbf{c}\in\mathbf{C}} \|\mathbf{x} - \mathbf{c}\|^2. \quad (1)$$

2.2. Product Quantization. Product quantization (PQ) is an optimization of vector quantization. Firstly, the feature space is divided into m mutually exclusive subspaces, and each subspace is then quantized separately using VQ. That is, the coding of each subspace forms a small codebook $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m$, and small codebooks form a large codebook $\mathbf{C}$ by the Cartesian product. In this method, a high-dimensional data can be decomposed into m low-dimensional spaces and can be processed in parallel. Suppose an object $\mathbf{x}$ is represented as a combination of m codewords $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_m$, the loss of the product quantization of vector $\mathbf{x}$ is given by

$$E_{pq} = \min_{\mathbf{c}\in\mathbf{C}} \|\mathbf{x} - \mathbf{c}\|^2 = \min_{\mathbf{c}\in\mathbf{C}} \left\| \mathbf{x} - \begin{bmatrix} \mathbf{c}_1^T \\ \mathbf{c}_2^T \\ \vdots \\ \mathbf{c}_m^T \end{bmatrix}^T \right\|^2. \quad (2)$$

2.3. Random Fourier Feature. Traditional dimensionality reduction methods, such as PCA, map the data to the independent feature space and compute the main independent features. This method ignores the nonlinear information of the sample distribution and cannot apply to the actual data well. Based on the feature mapping method of random Fourier feature (RFF), data are mapped to the characteristic space under the approximate kernel function, and the inner product of any two points under the feature space is approximated by their kernel function values. Compared with the PCA method, RFF can maximize the data distribution information and obtain the dimensional characteristic by reducing the dimension or raising the dimension. This kind of characteristic is suitable for the characteristic compression processing. SKLSH [18] is a kind of classical hashing algorithm based on RFF, which has a good experimental result under the long bit digit coding.

The length coding hash learning algorithm firstly maps the sample points from the original n dimension real space to the n dimension of the approximate kernel function feature space by RFF. Because of the convergence of RFF consistency, the kernel function similarity between any two sample points can be maintained.

Specifically, for two points $\mathbf{x}, \mathbf{y}$, the translation invariant kernel function [12] $K(\mathbf{x}, \mathbf{y}) = E(\Phi_{\mathbf{w},b}(\mathbf{x}), \Phi_{\mathbf{w},b}(\mathbf{y}))$ satisfies the following equation:

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}) &= K(\mathbf{x} + \eta, \mathbf{y} + \eta) = K(\mathbf{x} - \mathbf{y}), \\ K(\mathbf{x}, \mathbf{y}) &\le K(\mathbf{x} - \mathbf{x}) = K(0) = 1, \end{aligned} \quad (3)$$

where $\Phi_{\mathbf{w},b} = \sqrt{2}\cos(\mathbf{w}^T\mathbf{x} + b)$, b satisfies the uniform distribution between $[0, 2\pi]$, $\mathbf{w}$ obeys the probability distribution $\mathbf{P}_K$ induced by the translation invariant kernel function, and η is a constant parameter.

Thus, the mapping from the n dimensional space to the feature space of the d dimensional approximation kernel function can be obtained by the following equation:

$$\Phi^d(\cdot) = \frac{1}{\sqrt{d}}\left(\Phi_{\mathbf{w}_1 b_1}(\cdot), \Phi_{\mathbf{w}_2 b_2}(\cdot), \ldots, \Phi_{\mathbf{w}_d b_d}(\cdot)\right), \quad (4)$$

where $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d$ is for the same-sense sampling subject to the probability distribution $\mathbf{P}_K$ and $b_1, b_2, \ldots, b_d$ obeys the same-distribution sampling which is uniformly distributed between the obedience $[0, 2\pi]$. When the translation invariant kernel function is a Gaussian kernel function, $K(\mathbf{x} - \mathbf{y}) = e^{-(\mathrm{y}/2)\|\mathbf{x}-\mathbf{y}\|^2}$, $\mathbf{P}_K$ is a Gaussian distribution, i.e., $\mathbf{P}_K \sim \mathrm{Normal}(0, \gamma\mathbf{I}_{n\times n})$.

2.4. Orthogonal Procrustes Problem. An orthogonal Procrustes problem is to solve an orthogonal transforming matrix $\mathbf{O}$, so that $\mathbf{PO}$ is as close to $\mathbf{Q}$ as possible, i.e.,

$$\Gamma(\mathbf{O}) = \min_{\mathbf{O}} \|\mathbf{PO} - \mathbf{Q}\|_F, \tag{5}$$

where $\mathbf{O}^T\mathbf{O} = \mathbf{I}$. This formula is not easy to be solved directly, and it can be optimized by alternating optimization. Namely, the matrix $\mathbf{P}$ is first to be fixed, and the matrix $\mathbf{Q}$ is optimized to make the target function value reduced. Then the matrix $\mathbf{Q}$ is fixed, and the orthogonal transforming matrix $\mathbf{O}$ is optimized to make the target function value reduced.

3. Asymmetric Learning to Hash with Variable Bit Encoding

3.1. Algorithm Framework. For a general hash learning algorithm, the length of the hash code by learning is always fixed. AVBH uses the idea of asymmetric hashing algorithm, that is, the hash code for the dataset is short and unfixed, and the query point of the code is long and fixed. The steps of the AVBH hashing algorithm are shown in Figure 1, which mainly includes the dataset encoding steps ①–③ and the query point encoding step ④.

The dataset encoding section consists of two phases: random Fourier feature encoding (RFF encoding) and variable bit encoding (AVBH encoding). First, step ① uses the random Fourier feature (RFF) to map the dataset and get RFF encoding. After RFF coding, considering the difference of RFF coding frequency, the RFF coding frequency is sorted in step ②. According to the requirement, the original dataset can be divided into the subset by the RFF code as the length of $k_1, k_2, \ldots, k_L$ shown in the figure. As shown in step ③, the AVBH subset encoding of the length of $k_1, k_2, \ldots, k_L$ can be reproduced by duplicating $(n/k_1), (n/k_2), \ldots, (n/k_L)$ times sequentially and then the Hamming code of n dimension is formed.

In the query point encoding section, the query point quantization is encoded into RFF encoding of length n by step ④.

3.2. Objective Function. The target of the AVBH method is to get L groups of hash encoding with the length of $k_1, k_2, \ldots, k_L$ through the hash function $G(\mathbf{x})$, namely,

$$\mathbf{B}^{(l)} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N_l} \\ b_{21} & b_{22} & \cdots & b_{2N_l} \\ \vdots & \vdots & \ddots & \vdots \\ b_{k_l1} & b_{k_l2} & \cdots & b_{k_lN_l} \end{bmatrix}, \quad b_{ij} \in \{+1, -1\},\ i \in [1, 2, \ldots, k_l],\ j \in [1, 2, \ldots, N_l],\ l \in [1, 2, \ldots, L], \tag{6}$$

where $N = \sum_{l=1}^{L} N_l$, $n = \sum_{l=1}^{L} k_l$.

This divides the dataset $\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(L)}$ into subset according to the RFF encoding frequency. By cascading $(n/k_1), (n/k_2), \ldots, (n/k_L)$ times, respectively, we can get L group n bit hash code. For example,

$$\widetilde{\mathbf{B}}^{(l)} = \left[\left(\mathbf{B}^{(l)}\right)^T, \left(\mathbf{B}^{(l)}\right)^T, \ldots, \left(\mathbf{B}^{(l)}\right)^T\right]^T, \quad l \in [1, 2, \ldots, L]. \tag{7}$$

Then combine $\widetilde{\mathbf{B}}^{(1)}, \widetilde{\mathbf{B}}^{(2)}, \ldots, \widetilde{\mathbf{B}}^{(L)}$ to get a n bit long hash code of the entire dataset $\mathbf{B} \in \{+1, -1\}^{n\times N}$. The AVBH method calculates the similarity by calculating the Hamming distance between the hash code of the query point and the concatenated dataset during the query process. Therefore, for the dataset, we need to construct the hash mapping function, so that the L group hash code obtained with the length of $k_1, k_2, \ldots, k_L$, respectively, can preserve the original information as much as possible. Therefore, the AVBH method obtains the hash mapping function by the reconstruction error (8) between the minimum cascading encoding $\mathbf{B}$ and n dimension sample vector $\mathbf{Y}$:

$$\text{Loss}(\mathbf{B}, \mathbf{R}) = \min\|\mathbf{RB} - \mathbf{Y}\|_F^2, \tag{8}$$

where $\mathbf{R}$ is an orthogonal rotation $n \times n$ matrix, namely, $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}$.

Combining the properties of associative matrices and the definition of F-norm of matrices, we can get the following equation:

$$\begin{aligned} \text{Loss}(\mathbf{B}, \mathbf{R}) &= \min\|\mathbf{RB} - \mathbf{Y}\|_F^2 \\ &= \min\left\{\text{tr}\left[(\mathbf{RB} - \mathbf{Y})^T(\mathbf{RB} - \mathbf{Y})\right]\right\} \\ &= \min\left\{\text{tr}\left[(\mathbf{RB})^T\mathbf{RB}\right] + \text{tr}\left(\mathbf{Y}^T\mathbf{Y}\right) - 2\text{tr}\left(\mathbf{RBY}^T\right)\right\}. \end{aligned} \tag{9}$$

As the unknown variable $\mathbf{B}, \mathbf{R}$ in formula (8) is the product relation, the expanded formula (9) contains two items of unknown variables, so it is difficult to solve. After further simplification, we can get the following formula:

$$\begin{aligned} \text{Loss}(\mathbf{B}, \mathbf{R}) &= \min\|\mathbf{RB} - \mathbf{Y}\|_F^2 \\ &= \min\left\{\text{tr}\left[(\mathbf{RB})^T\mathbf{RB}\right] + \text{tr}\left(\mathbf{Y}^T\mathbf{Y}\right) - 2\text{tr}\left(\mathbf{RBY}^T\right)\right\} \\ &= \min\left[\text{tr}\left(\mathbf{B}^T\mathbf{R}^T\mathbf{RB}\right) + \text{tr}\left(\mathbf{Y}^T\mathbf{Y}\right) - 2\text{tr}\left(\mathbf{RBY}^T\right)\right] \\ &= \min\left[\text{tr}\left(\mathbf{B}^T\mathbf{B}\right) + \text{tr}\left(\mathbf{Y}^T\mathbf{Y}\right) - 2\text{tr}\left(\mathbf{RBY}^T\right)\right]. \end{aligned} \tag{10}$$

As $\mathbf{B} \in \{+1, -1\}^{n\times N}$, it is easy to get $\text{tr}(\mathbf{B}^T\mathbf{B}) = nN$. As $\mathbf{R}^T\mathbf{R} = \mathbf{I}$, we can get that $\mathbf{Y}$ is unrelated to $\mathbf{B}$ and $\mathbf{R}$. As a result, $\text{tr}(\mathbf{Y}^T\mathbf{Y}) = \mathbf{c}$, where $\mathbf{c}$ is unrelated to $\mathbf{B}$. So formula (10) is simplified as follows:

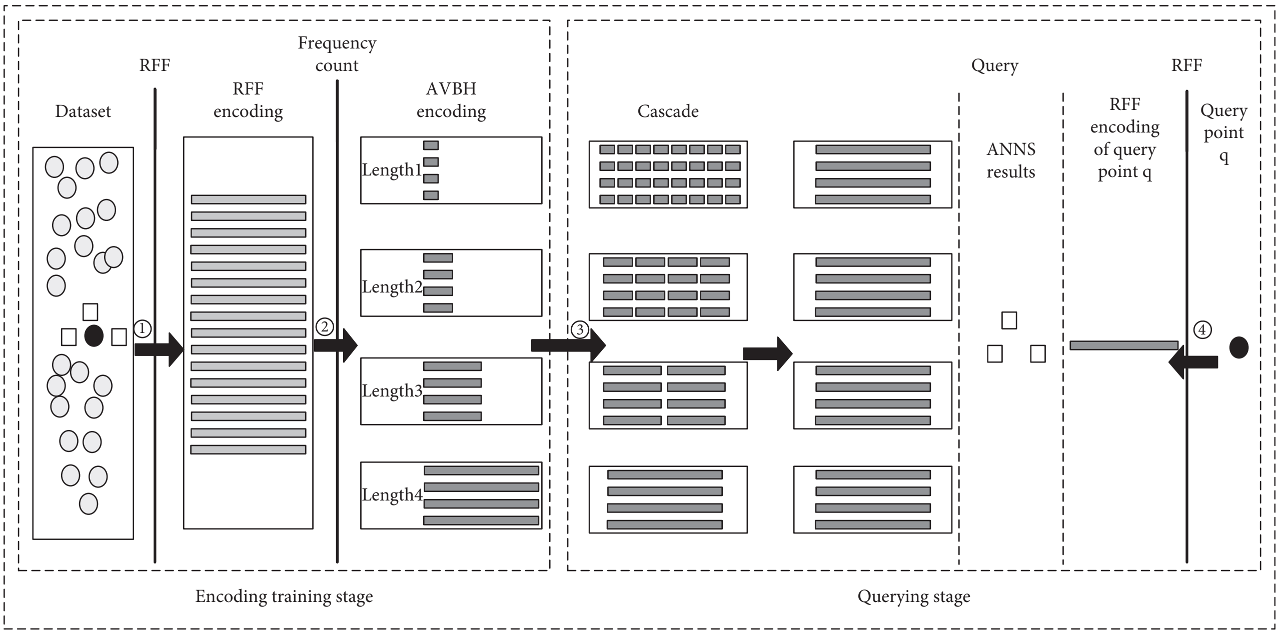


Figure 1: Algorithm framework of AVBH.

$$\begin{aligned}\text{Loss}(\mathbf{B},\mathbf{R}) &= \min\|\mathbf{RB}-\mathbf{Y}\|_F^2\\ &= \min\left[\operatorname{tr}\left(\mathbf{B}^T\mathbf{B}\right)+\operatorname{tr}\left(\mathbf{Y}^T\mathbf{Y}\right)-2\operatorname{tr}\left(\mathbf{RBY}^T\right)\right]\\ &= \min\left[\operatorname{tr}\left(\mathbf{B}^T\mathbf{B}\right)+\operatorname{tr}\left(\mathbf{Y}^T\mathbf{RR}^T\mathbf{Y}\right)-2\operatorname{tr}\left(\mathbf{RBY}^T\right)\right]\\ &= \min\left[nN+\mathbf{c}-2\operatorname{tr}\left(\mathbf{B}^T\mathbf{R}^T\mathbf{Y}\right)\right]\\ &= \min\left\{\operatorname{tr}\left(\mathbf{B}^T\mathbf{B}\right)+\operatorname{tr}\left[\left(\mathbf{R}^T\mathbf{Y}\right)^T\mathbf{R}^T\mathbf{Y}\right]-2\operatorname{tr}\left(\mathbf{B}^T\mathbf{R}^T\mathbf{Y}\right)\right\}\\ &= \min\left\{\operatorname{tr}\left[\left(\mathbf{B}-\mathbf{R}^T\mathbf{Y}\right)^T\left(\mathbf{B}-\mathbf{R}^T\mathbf{Y}\right)\right]\right\}\\ &= \min\left\|\mathbf{B}-\mathbf{R}^T\mathbf{Y}\right\|_F^2.\end{aligned} \tag{11}$$

Thus, minimizing the reconfiguration error (8) equals minimizing the quantization error (11).

The objective function of AVBH to encode the dataset is to minimize the reconstruction error of the concatenated encoding of the n bit by finding the orthogonal rotation matrix $\mathbf{R}$. In extreme cases of the dataset which is uniformly distributed, there is no significant difference in the frequency of the hash encoding of the dataset, and the AVBH method then degenerates into the ACH algorithm [16]. Compared with the ACH hashing algorithm, the AVBH hashing algorithm is more adaptable to the real data because it can adapt to the data of various distributions and the generalization ability is stronger.

3.3. Optimization Algorithm. The objective function (11) can be optimized by alternating optimization. Namely, the rotation matrix $\mathbf{R}$ is first to be fixed, and the encoding matrix $\mathbf{B}$ is optimized to make the target function value reduced. Then the encoding matrix $\mathbf{B}$ is fixed and the rotation matrix $\mathbf{R}$ is optimized to make the target function value reduced. In this way, the value of the target function decreases until it converges. The following is a discussion of how to tune and optimize the value of the target function.

(1) Fix the rotation matrix $\mathbf{R}$, and optimize the encoding matrix $\mathbf{B}$.

Given $\mathbf{V}=\mathbf{R}^T\mathbf{Y}$, $\mathbf{V}^{lm}$ is a matrix consisted of $(m-1)\times k+1$ row to $m\times k$ row, $(l-1)\times k+1$ column to $l\times k$ column of $\mathbf{V}$. From formula (11), we can get the following equation:

$$\begin{aligned}\text{Loss}_1(\mathbf{B}) &= \min\left[nN + \mathbf{c} - 2\text{tr}\left(\mathbf{B}^T\mathbf{R}^T\mathbf{Y}\right)\right] \\ &= \min\left\{nN + \mathbf{c} - 2\sum_{l=1}^{L}\text{tr}\left[\left(\widetilde{\mathbf{B}}^{(l)}\right)^T\mathbf{V}^l\right]\right\} \\ &= \min\left\{nN + \mathbf{c} - 2\sum_{l=1}^{L}\text{tr}\left\{\left[\left(\mathbf{B}^{(l)}\right)^T, \left(\mathbf{B}^{(l)}\right)^T, \ldots, \left(\mathbf{B}^{(l)}\right)^T\right]\begin{bmatrix}\mathbf{V}^{l1} \\ \mathbf{V}^{l2} \\ \vdots \\ \mathbf{V}^{l(n/k_l)}\end{bmatrix}\right\}\right\} \\ &= \min\left\{nN + \mathbf{c} - 2\sum_{l=1}^{L}\text{tr}\left[\left(\mathbf{B}^{(l)}\right)^T\left(\sum_{m=1}^{(n/k_l)}\mathbf{V}^{lm}\right)\right]\right\}.\end{aligned} \tag{12}$$

As $n, N, \mathbf{c}$ are unrelated to $\mathbf{B}$, for a fixed $\mathbf{R}$, the problem of minimizing (12) is equal to the problem of maximizing the following formula:

$$\begin{aligned}\text{Loss}_1(\mathbf{B}) &= 2\max\sum_{l=1}^{L}\text{tr}\left[\left(\mathbf{B}^{(l)}\right)^T\left(\sum_{m=1}^{(n/k_l)}\mathbf{V}^{lm}\right)\right] \\ &= \max\sum_{l=1}^{L}\left[\sum_{i=1}^{(n/k_l)}\sum_{j=1}^{n_l}\mathbf{B}_{ij}^{(l)}\left(\sum_{m=1}^{(n/k_l)}\mathbf{V}_{ij}^{lm}\right)\right],\end{aligned} \tag{13}$$

As $\mathbf{B}_{ij}^{(l)} \in \{+1, -1\}$, optimal analytic solution of formula (13) is given by

$$\mathbf{B}_{ij}^{(l)} = \text{sign}\left(\sum_{m=1}^{(n/k_l)}\mathbf{V}_{ij}^{lm}\right), \quad i \in [1, 2, \ldots, k_l],\ j \in [1, 2, \ldots, N_l],\ l \in [1, L]. \tag{14}$$

(2) Fix the encoding matrix $\mathbf{B}$, and optimize the rotation matrix $\mathbf{R}$.

Under $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}$, the problem of minimizing formula (11) is equal to an Orthogonal Procrustes problem [9]. The optimal solution of such problems with $\mathbf{R}$ is as follows:

$$\begin{aligned}\text{Loss}_2(\mathbf{R}) &= \min\left[nN + \mathbf{c} - 2\text{tr}\left(\mathbf{B}^T\mathbf{R}^T\mathbf{Y}\right)\right] \\ &= \min\left[nN + \mathbf{c} - 2\text{tr}\left(\mathbf{Y}\mathbf{B}^T\mathbf{R}^T\right)\right].\end{aligned} \tag{15}$$

Hence, the problem of optmizing $\mathbf{R}$ to get the minimum value of formula (15) is equal to the problem of maximizing the following formula:

$$\text{Loss}_2(\mathbf{R}) = \max\left[\text{tr}\left(\mathbf{Y}\mathbf{B}^T\mathbf{R}^T\right)\right]. \tag{16}$$

By calculating the SVD of $\mathbf{Y}\mathbf{B}^T$, we can get the following formula:

$$\mathbf{Y}\mathbf{B}^T = \mathbf{U}\boldsymbol{\Omega}\mathbf{C}^T, \tag{17}$$

where $\mathbf{U}$ is a matrix which consists of left singular value vector, $\mathbf{C}$ is a matrix which consists of right singular value vector, $\boldsymbol{\Omega}$ is a diagonal matrix which consists of corresponding singular value vectors, and the diagonal elements of which is $\boldsymbol{\Omega}_{ii} \geq 0$, $i \in [1, n]$.

By combining formulas (16) and (17), we can get the following equation:

$$\begin{aligned}\text{Loss}_2(\mathbf{R}) &= \max\left[\text{tr}\left(\mathbf{U}\boldsymbol{\Omega}\mathbf{C}^T\mathbf{R}^T\right)\right] \\ &= \max\left[\text{tr}\left(\mathbf{U}\boldsymbol{\Omega}(\mathbf{R}\mathbf{C})^T\right)\right] \\ &= \max\left[\text{tr}\left((\mathbf{R}\mathbf{C})^T\mathbf{U}\boldsymbol{\Omega}\right)\right].\end{aligned} \tag{18}$$

Given $\mathbf{A} = (\mathbf{R}\mathbf{C})^T\mathbf{U}$, $\widetilde{\mathbf{R}} = \mathbf{R}\mathbf{C}$, $\mathbf{A}_{ii}$ is the diagonal element of $\mathbf{A}$, and $\widetilde{\mathbf{R}}_i$, $\mathbf{U}_i$, respectively, represent the $i-$ th row of $\widetilde{\mathbf{R}}$, $\mathbf{U}$. By Cauchy–Schwarz inequalities [11], the following equation is obtained:

$$\mathbf{A}_{ii} \leq \left\|\widetilde{\mathbf{R}}_i\right\|\left\|\mathbf{U}_i\right\| = 1, \quad \text{when } \widetilde{\mathbf{R}}_i = \mathbf{U}_i. \tag{19}$$

So formula (18) can be written as formula (20):

$$\text{Loss}_2(\mathbf{R}) = \max[\text{tr}(\mathbf{A}\boldsymbol{\Omega})] = \max\sum_{i=1}^{n}\mathbf{A}_{ii}\boldsymbol{\Omega}_{ii}. \tag{20}$$

Combining formula (19), when $\widetilde{\mathbf{R}}_i = \mathbf{U}_i$, formula (20) takes the maximum value. For $\widetilde{\mathbf{R}}_i = \mathbf{U}_i$, we can get the following formula:

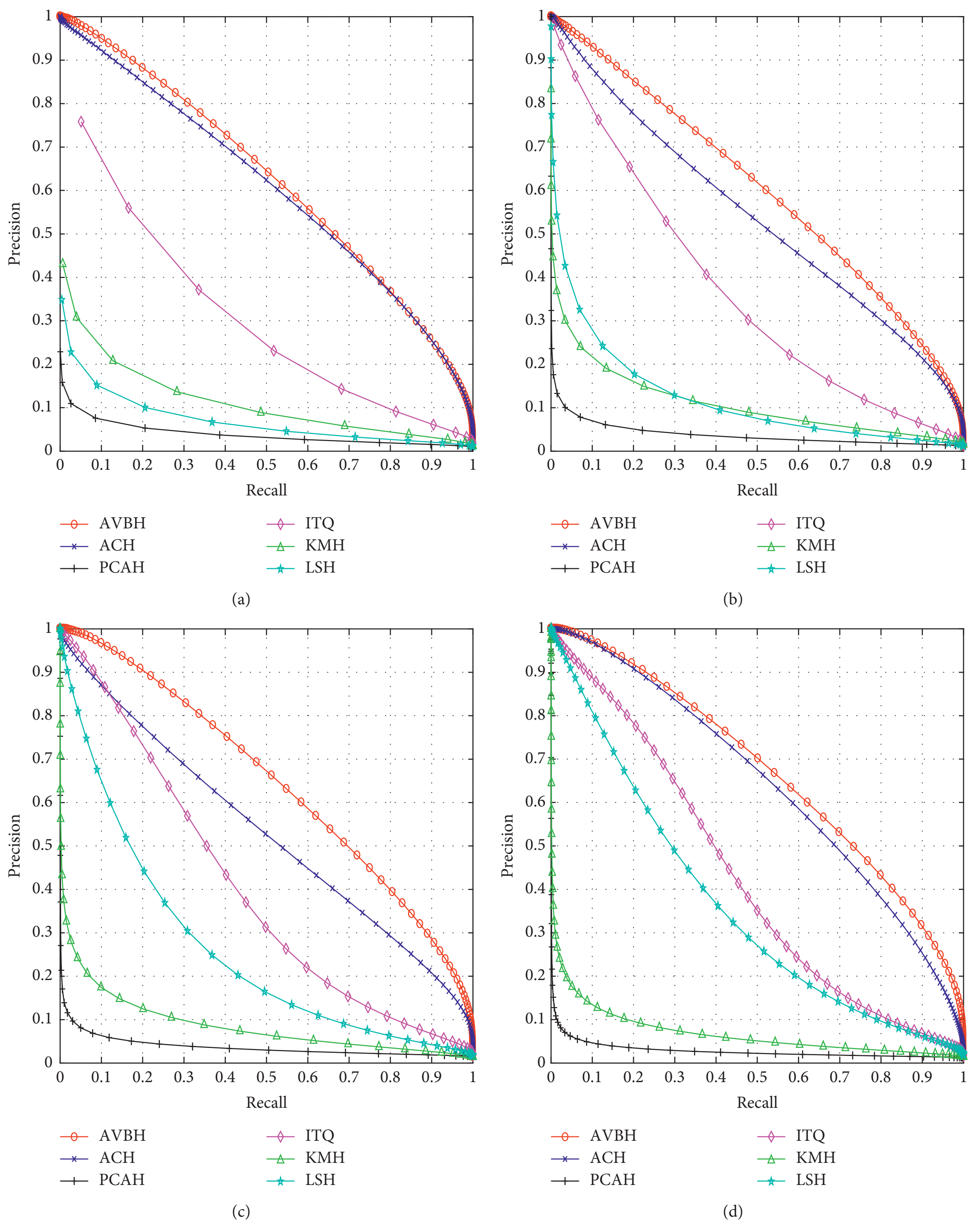


FIGURE 2: Comparison of the precision-recall experiment of different bits encoding under Cifar10: (a) Cifar10, 16 bit; (b) Cifar10, 32 bit; (c) Cifar10, 64 bit; (d) Cifar10, 128 bit.

$$\begin{aligned}
\widetilde{\mathbf{R}}_i &= \mathbf{U}_i \\
\Longleftrightarrow \widetilde{\mathbf{R}} &= \mathbf{U} \\
\Longleftrightarrow \mathbf{RC} &= \mathbf{U} \\
\Longleftrightarrow \mathbf{RCC}^T &= \mathbf{UC}^T \\
\Longleftrightarrow \mathbf{R} &= \mathbf{UC}^T,
\end{aligned} \tag{21}$$

when $\mathbf{R} = \mathbf{UC}^T$, formula (16) takes the maximum value, and formula (15) takes the minimum value. As a result, we can get the optimal result by formula (21).

3.4. Encoding Functions

3.4.1. Dataset Encoding. When the objective function value converges, we can get the mapping function $G(\mathbf{y}_1)$ of AVBH to the dataset according to formula (14), where $\mathbf{y}$ is the random Fourier feature (RFF) obtained by the mapping stage of the sample point $\mathbf{x}$:

$$\begin{aligned}
G(\mathbf{y}) &= \left[g_1(\mathbf{y}), g_2(\mathbf{y}), \ldots, g_{k_L}(\mathbf{y})\right] \\
&= \operatorname{sign}\left(\sum_{m=1}^{(n/k_l)} \left(\mathbf{R}^T\right)^{lm} \mathbf{y}\right), \quad l \in [1, L].
\end{aligned} \tag{22}$$

3.4.2. Query Point Encoding. The optimal rotation matrix $\mathbf{R}$ can be obtained from the training process of dataset coding. For the data $\mathbf{q}$ in the query set, the main goal of encoding is to keep as much accurate information as possible, so the query set encoding does not need to be compressed and mapped to the hash code of the length bit. Combining formula (14), we can get the mapping function of AVBH to the query set:

$$F(\mathbf{q}) = \left[f_1(\mathbf{q}), f_2(\mathbf{q}), \ldots, f_n(\mathbf{q})\right] = \operatorname{sign}\left(\mathbf{R}^T\mathbf{q}\right). \tag{23}$$

3.5. Convergence Analysis of AVBH. According to the objective function (8), we can get the following formula:

$$\begin{aligned}
\operatorname{Loss}(\mathbf{B}, \mathbf{R}) &= \min\left\|\mathbf{B} - \mathbf{R}^T\mathbf{Y}\right\|_F^2 \\
&= \min\left\|\mathbf{B} - \mathbf{D} - \left(\mathbf{R}^T\mathbf{Y} - \mathbf{D}\right)\right\|_F^2 \\
&\le \min\left(\|\mathbf{B} - \mathbf{D}\|_F^2 + \left\|\mathbf{R}^T\mathbf{Y} - \mathbf{D}\right\|_F^2\right) \\
&= \min\|\mathbf{B} - \mathbf{D}\|_F^2 + \min\left\|\mathbf{D} - \mathbf{R}^T\mathbf{Y}\right\|_F^2 \\
&= L_1(\mathbf{B}) + L_2(\mathbf{R}),
\end{aligned} \tag{24}$$

where $\mathbf{D}$ is a $n \times N$ constant matrix and satisfies the following two conditions: (1) signs of each element, i.e., positive or negative, in $\mathbf{D}$ are the same as that in $\mathbf{R}^T\mathbf{Y}$, and (2) each element in $\mathbf{D}$ is not greater than the corresponding position element in $\mathbf{R}^T\mathbf{Y}$.

Therefore, the optimization goal for $\operatorname{Loss}(\mathbf{B}, \mathbf{R})$ is transformed into the two suboptimization problems, i.e., $\operatorname{Loss}_1(\mathbf{B})$ and $\operatorname{Loss}_2(\mathbf{R})$. Specifically, for the subproblem $\operatorname{Loss}_1(\mathbf{B})$, formula (14) gives the optimal solution. Therefore, it can be guaranteed that the updated value of formula (14) is less than or equal to the value obtained before. For the subproblem $\operatorname{Loss}_2(\mathbf{R})$, formula (21) gives the optimal solution. Therefore, it also can be guaranteed that the updated value of formula (21) is less than or equal to the value obtained before.

Combining the two parts, the combination of (14) and (21) can guarantee that the updated value is less than equal to the value obtained before the update. We can conclude the AVBH algorithm is convergence.

4. Experimental Results

The experimental computer has the Intel Core i5-2410M CPU and 8 GB DDR3 memory. We compared the performance of AVBH with that of several typical hashing methods: ACH [12], ITQ [19], KMH [20], PCAH [21], and LSH [13].

4.1. Experimental Datasets

4.1.1. CIFAR-10[1]. It is a set of 60,000-32 × 32 colour images in 10 categories with each category containing 6,000 images. In this experiment, 320-D gist features were extracted for each image in the dataset. We randomly selected 1,000 images as the test data and the remaining 59,000 as the training data. In the training data, the closest 50 data points (based on the Euclidean distance) from a test data point were regarded as its nearest neighbours.

4.1.2. SIFT[2]. It is a local SIFT feature set containing 1,000,000 128-D images. 100,000 of these sample images were randomly selected as the training data and 10,000 of other sample images as the test data.

4.1.3. GIST[3]. It is a global GIST feature set containing 1,000,000 960-D images. 500,000 of these sample images were randomly selected as the training data and 1,000 sample images as the test data.

4.2. Performance Evaluation. The performance of AVBH was evaluated mainly by the relationship between the accuracy of the query (precision) and the recall rate (recall). We define

$$\begin{aligned}
\text{precision} &= \frac{\text{true positive results}}{\text{true positive results} + \text{false positive results}}, \\
\text{recall} &= \frac{\text{true positive results}}{\text{true positive results} + \text{false negative results}}.
\end{aligned} \tag{25}$$

For the sake of fairness, the average encoding length of AVBH was set to be the encoding length of other methods under a given dataset.

Figures 2–4 show precision-recall curves for Euclidean neighbour retrieval for several methods on CIFAR-10, SIFT, and GIST with Euclidean neighbour ground truth.

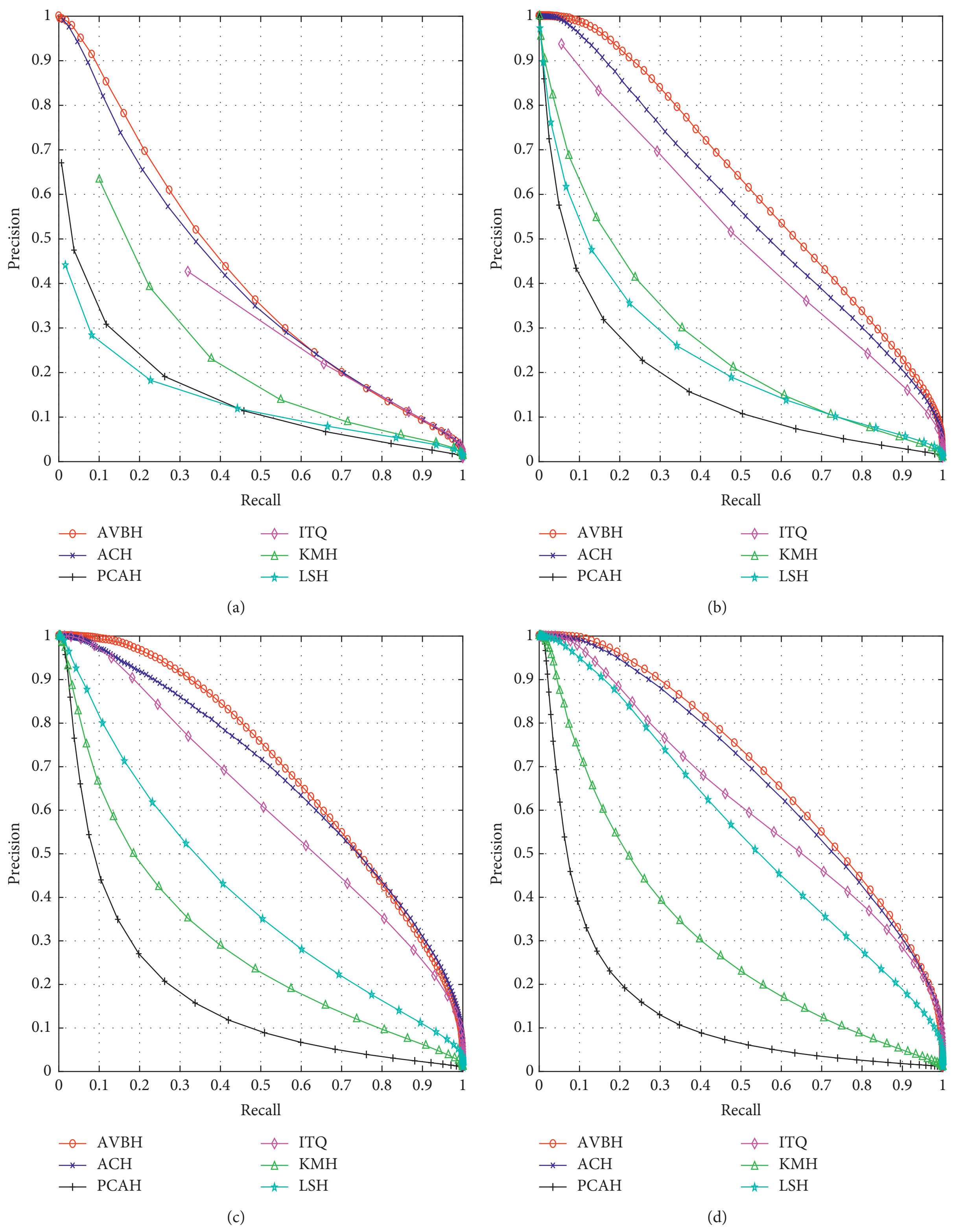
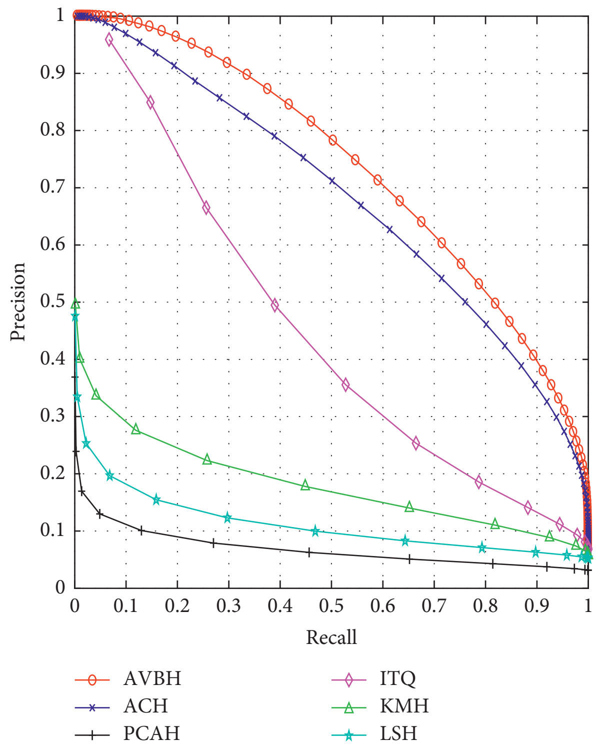
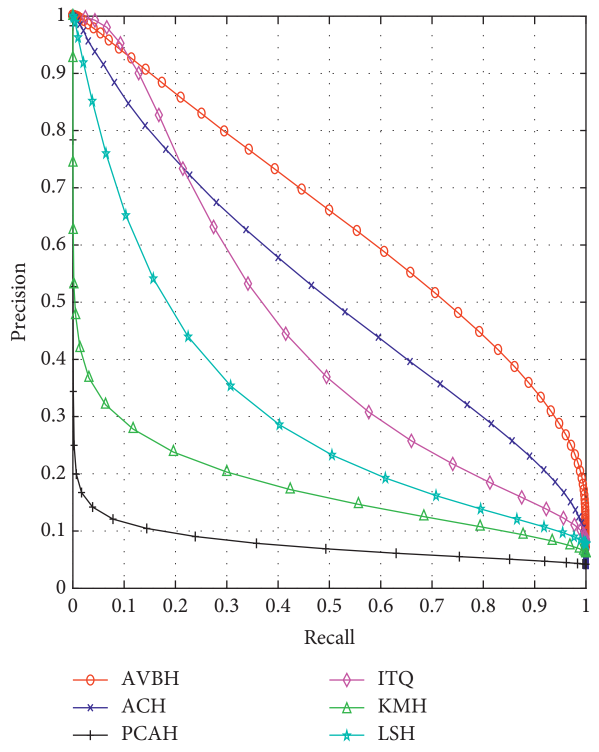


Figure 3: Comparison of the precision-recall experiment of different bits encoding under SIFT: (a) SIFT, 16 bit; (b) SIFT, 32 bit; (c) SIFT, 64 bit; (d) SIFT, 128 bit.

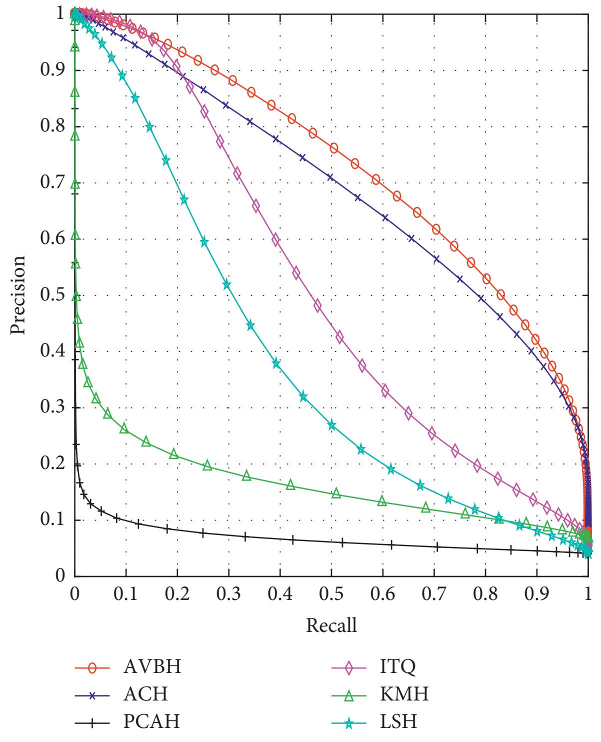
Our method, AVBH, can get a better precision performance on the four datasets. As the AVBH algorithm uses the variable bit code, the total code length is less than other algorithms. As a result, our method effectively reduces the cost of storage and improves the accuracy of query.



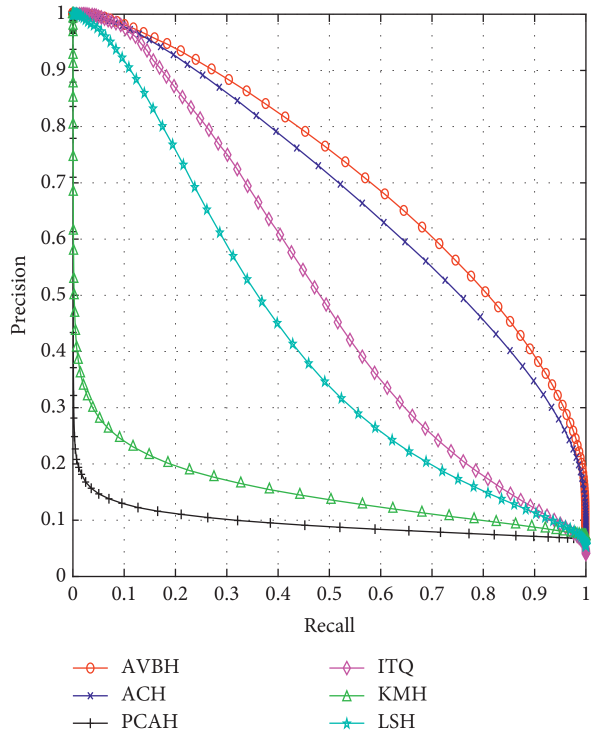
(a)



(b)



(c)



(d)

FIGURE 4: Comparison of the precision-recall experiment of different bits encoding under GIST: (a) GIST, 16 bit; (b) GIST, 32 bit; (c) GIST, 64 bit; (d) GIST, 128 bit.

5. Conclusion

In this paper, an asymmetric learning to hash with variable bit encoding algorithm was proposed. By the frequency statistics of the random Fourier feature encoding for the dataset, we compress high-frequency hash codes into longer encoding representations and low-frequency hash codes into shorter encoding representations. For a query data, we quantize to a long bit hash code and compare with the same length cascade concatenated data point to retrieve the nearest neighbours. This ensures that the original data information can be preserved as much as possible while the data are compressed, which maximizes the balance between coding compression and query performance. Experiments on open datasets show that the proposed algorithm effectively reduces the cost of storage and improves the accuracy of the query. As we use a two-stage algorithm framework for the hash codes generating, the training stage costs a lot of time. In future work, we will work on simplifying the training process.

Data Availability

The datasets for the experiment of this paper are as follows. (1) CIFAR-10 (available at http://www.cs.toronto.edu/~kriz/cifar.html): it is a set of 60,000-32×32 colour images in 10 categories with each category containing 6,000 images. In this experiment, 320-D gist features were extracted for each image in the dataset. We randomly selected 1,000 images as the test data and the remaining 59,000 as the training data. In the training data, the closest 50 data points (based on the Euclidean distance) from a test data point were regarded as its nearest neighbours. (2) SIFT (available at http://corpus-texmex.irisa.fr): it is a local SIFT feature set containing 1,000,000 128-D images. 100,000 of these sample images were randomly selected as the training data and 10,000 of other sample images as the test data. (3) GIST (available at http://corpus-texmex.irisa.fr): it is a global GIST feature set containing 1,000,000 960-D images. 500,000 of these sample images were randomly selected as the training data and 1,000 sample images as the test data.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by Zhejiang NSF (Grant nos. LZ20F020001 and LY20F020009) and China NSF (Grant nos. 61472194 and 61572266) as well as programs sponsored by K. C. Wong Magna Fund in Ningbo University.

References

[1] C. S. Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, Anchorage, AK, USA, June 2008.

[2] F. Shen, X. Yan, L. Li, Y. Yang, Z. Huang, and H. T. Shen, "Unsupervised deep hashing with similarity-adaptive and discrete optimization," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 40, no. 12, pp. 3034–3044, 2018.

[3] X. Qu, W. Yi, T. Wang, S. Wang, L. Xiao, and Z. Liu, "Mixed-integer linear programming models for teaching assistant assignment and extensions," *Scientific Programming*, vol. 2017, Article ID 9057947, 7 pages, 2017.

[4] Z. Zhang, Q. Zou, Q. Wang, Y. Lin, and Q. Li, "Instance similarity deep hashing for multi-label image retrieval," 2018, https://arxiv.org/abs/1803.02987v1.

[5] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 2, pp. 437–451, 2018.

[6] S. Berchtold, D. A. Keim, and H. P. Kriegel, "The X-tree: an index structure for high- dimensional data," in *Proceedings of the 22nd International Conference on Very Large Data Bases*, pp. 28–39, Mumbai, India, 1996.

[7] X. Shen, W. Liu, I. W. Tsang, Q.-S. Sun, and Y.-S. Ong, "Multilabel prediction via cross-view search," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 4324–4338, 2018.

[8] X. Shen, F. Shen, Q.-S. Sun, Y. Yang, Y.-H. Yuan, and H. T. Shen, "Semi-paired discrete hashing: learning latent hash codes for semi-paired cross-view retrieval," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4275–4288, 2016.

[9] L. K. Huang, Q. Yang, and W. S. Zheng, "Online hashing," *IEEE Transactions on Neural Networks & Learning Systems*, vol. 29, no. 6, pp. 2309–2322, 2017.

[10] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios, "Data fusion through cross-modality metric learning using similarity-sensitive hashing," in *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010*, IEEE, San Francisco, CA, USA, June 2010.

[11] A. Gordo, F. Perronnin, Y. Gong, and S. Lazebnik, "Asymmetric distances for binary embeddings," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 36, no. 1, pp. 33–47, 2014.

[12] Y. Lv, W. W. Y. Ng, Z. Zeng, D. S. Yeung, and P. P. K. Chan, "Asymmetric cyclical hashing for large scale image retrieval," *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1225–1235, 2015.

[13] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, vol. 8, no. 2, pp. 518–529, Edinburgh, Scotland, 1999.

[14] J. Qian, Q. Zhu, and H. Chen, "Multi-granularity locality-sensitive bloom filter," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3500–3514, 2015.

[15] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao, "Triplet-based deep hashing network for cross-modal retrieval," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3893–3903, 2018.

[16] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[17] W. Kong and W.-J. Li, "Isotropic hashing," in *Advances in Neural Information Processing Systems*, pp. 1655–1663, Curran Associates Inc., New York, NY, USA, 2012.

[18] M. Raginsky, "Locality-sensitive binary codes from shift-invariant kernels," *Advances in Neural Information Processing*

Systems, pp. 1509–1517, Curran Associates Inc., New York, NY, USA, 2009.

[19] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

[20] K. He, F. Wen, and J. Sun, "K-means hashing: an affinity-preserving quantization method for learning binary compact codes," in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, Portland, OR, USA, June 2013.

[21] X. Yu, X. Zhang, B. Liu, L. Zhong, and D. N. Metaxas, "Large scale medical image search via unsupervised PCA hashing," in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 393–398, IEEE, Portland, OR, USA, June 2013.

Hindawi
Scientific Programming
Volume 2019, Article ID 2753152, 12 pages
https://doi.org/10.1155/2019/2753152

Research Article

Intelligent Behavior Data Analysis for Internet Addiction

Wei Peng,[1] Xinlei Zhang ,[2] and Xin Li [2]

[1]*Information Technology Support, East China Normal University, Shanghai 200062, China*
[2]*Shanghai Key Laboratory of Trustworthy Computing, MOE International Joint Lab of Trustworthy Software, East China Normal University, Shanghai 200062, China*

Correspondence should be addressed to Xinlei Zhang; xinleizhang1997@gmail.com

Received 26 August 2019; Revised 21 October 2019; Accepted 7 November 2019; Published 29 November 2019

Guest Editor: Aibo Song

Internet addiction refers to excessive internet use that interferes with daily life. Due to its negative impact on college students' study and life, discovering students' internet addiction tendencies and making correct guidance for them timely is necessary. However, at present, the research methods used in analyzing students' internet addiction are mainly questionnaires and statistical analysis, which relies on the domain experts heavily. Fortunately, with the development of the smart campus, students' behavior data such as consumption and trajectory information in the campus are stored. With this information, we can analyze students' internet addiction levels quantitatively. In this paper, we provide an approach to estimate college students' internet addiction levels using their behavior data in the campus. In detail, we consider students' addiction towards the internet is a hidden variable which affects students' daily time online together with other behavior. By predicting students' daily time online, we will find students' internet addiction levels. Along this line, we develop a linear internet addiction (LIA) model, a neural network internet addiction (NIA) model, and a clustering-based internet addiction (CIA) model to calculate students' internet addiction levels, respectively. These three models take the regularity of students' behavior and the similarity among students' behavior into consideration. Finally, extensive experiments are conducted on a real-world dataset. The experimental results show the effectiveness of our method, and it is also consistent with some psychological findings.

1. Introduction

Internet addiction disorder refers to excessive internet use that interferes with daily life [1]. Some research shows that the addiction towards the internet has a negative impact on college students, such as the backwardness of study, health, and social relationship [1–3]. Therefore, it is necessary to discover students' addiction tendencies towards the internet and make correct guidance for them.

At present, related works of internet addiction are concentrated on psychological fields. Such works focus on the causes, the influence of internet addiction, and internal mechanisms leading to internet addiction, together with methods to eliminate internet addiction. There are few works on calculating internet addiction levels quantitatively. Besides, the methods used for analyzing are mainly questionnaires and statistical analysis, which are cumbersome and relies on the domain experts heavily. Therefore, it is necessary to develop an approach to explore students' internet addiction level quantitatively and automatically.

Fortunately, with the development of the smart campus, students' behavior data are collected, such as the access data and consuming data. With these data, it is possible to analyze students' internet addiction levels quantitatively.

To this end, in this paper, we propose an approach to estimate students' internet addiction levels using their behavior data. Currently, there is no method to evaluate students' addiction level precisely, so we are unable to study it with supervised methods explicitly. Instead, we can calculate students' internet addiction level through another task. In detail, based on the definition of internet addiction, we consider that the student's internet addiction level is a hidden variable, which will affect students' daily time online. Besides, student's behavior data such as consuming data and the internet access gap reflect student's daily activities, which may also influence the time they spend online. Then, we can predict students' online time with their behavior data and internet addiction level. Through such a task, the internet addiction value can be inferred. Along this line, we propose a linear internet addiction (LIA) model, a neural network internet addiction (NIA) model, and a clustering-based internet addiction (CIA) model to capture the relationship between students' behavior data, internet addiction, and the time they spend online every day.

Furthermore, students have fixed disciplines every week, which leads to the regularity of time they spend online every week. LIA and NIA models take the regularity of students' behavior into consideration, and the CIA model mainly uses the relationship among students' behavior to learn their internet addiction level. Finally, we conduct extensive experiments on a real-world dataset from a Chinese college, including internet addiction calculation, internet addiction verification, and internet addiction analysis experiments. Particularly, to verify the internet addiction value we calculate is credible, we compare our results with the results evaluated from the psychological scale. The experimental results demonstrate the correctness and effectiveness of the model we propose. And the results are also consistent with some psychological findings.

2. Related Work

The main related work of this paper can be divided into two parts: internet addiction analysis and campus data mining.

2.1. Internet Addiction Analysis. Internet addiction analysis is a research direction in the psychological field. Some works focus on the causes of internet addiction. Researchers found that interpersonal difficulties, psychological factors, social skills, etc., are all reasons for internet addiction [1, 4, 5]. Other works aim at finding the influence of internet addiction. Upadhayay et al. claimed that excessive use of the internet would lead to the drawback of the study [2]. He et al. explored internet addiction's influence on the sensitivity towards punishment and award [6]. Their result shows that people with serious internet addiction are more sensitive to risk. There are also some works about the inner mechanism of forming internet addiction. Zhang et al. focused on the inner reason of family function's negative influence on internet addiction. They revealed that the stability and development of family might affect users' mental situations such as dignity and loneliness, and then such mental situations will have an influence on internet addiction [7]. Zhao et al. noticed that stressful life events make users feel depressed, which causes the user addicted to the internet [8].

2.2. Campus Data Mining. Data are produced everywhere in our daily life activities, for example, the consumption records, chatting records, web browsing records, and so on. Using such data, we are able to make some interesting applications, such as tag recommendation, which suggests a list of tags when a user wants to annotate an item. Wang et al. proposed the TAPITF model to combine both time awareness and personalization aspects into tag recommendation task [9]. Campus data mining refers to solving problems on campus with data mining methods. Some works mainly analyze students' daily behavior in life. Guan et al. predicted students' financial hardship through their smart card usage, internet usage, and students' trajectories on campus (Dis-HARD model) so that the school can offer those students with stipend portfolios [10]. Based on this work, Ye et al. proposed a CS^3G model [11], which predicted stipend portfolios with multimodal data. Their work has higher accuracy compared to the Dis-HARD model and protects students' privacy. The Bayesian method is widely used in many fields. Wang et al. proposed a Bayesian probabilistic multitopic matrix factorization model for rating prediction [12]. And similarly Zhu et al. proposed an unsupervised method under the framework of empirical Bayes to calculate students' procrastination value with their borrow info in the library [13]. Peng et al. proposed a deep topical correlation analysis approach to track students' thoughts and serve the development of smart campus using multimodal data [14]. There are also some works aiming at analyzing students' studying process and improving their performance in class, which is called educational data mining (EDM). For example, Burlak et al. identified if a student is cheating in an exam by analyzing their interactive data with online course systems such as start time, end time, IP address, and access frequency [15]. Abdi et al. predicted students' grades based on their answers to usual work and duration of stay on a question [16].

Above all, to the best of our knowledge, there is no work on analyzing internet addiction using students' daily behavior. And we are the first to analyze internet addiction based on their behavior data with data mining methods.

3. Preliminaries

Internet addiction is an abstract concept in the psychological field, so it is hard to give a measurable definition of internet addiction. To solve this problem, we first make a reasonable assumption about internet addiction. Then, based on this assumption, we calculate the internet addiction value using students' behavior data.

3.1. Internet Addiction Assumption. Psychological research shows that most college students are addicted to the internet [17]. And we mentioned that internet addiction refers to excessive use of internet interfering with daily life. Therefore, students with different internet addiction levels are very likely to spend different time online. Besides, different behaviors show the different activities in school, which in turn also leads to different online time. And students of different genders or departments will also have some differences in the internet use.

Based on such fact, we assume that internet addiction is a hidden factor, which may influence students' daily time online together with their behavior and profile information. Therefore, we will learn such factors by modelling how students' internet addiction and behavior influence daily online time. To simplify the problem, we also assume students' internet addiction level will not change in a semester.

3.2. Problem Formulation. Since we do not have any label about internet addiction level, we cannot use supervised methods to study students' internet addiction value. Thus, we need to estimate it through some known data. Based on our assumption that the internet addiction value is a hidden variable, which may affect the time students spend online, the value can be learned by predicting students' daily online time.

Formally, we define a_u as the internet addiction level of student u. Daily time online sequence of student u during a period T is represented as $\{T_u(t)\}$. And the daily behavior sequence of u during the same period is represented as $\{B_u(t)\}$. We also define the personal profile information of student u as $\{p_u\}$. Our task is to model the relationship $\{a_u, p_u, B_u(t)\} \longrightarrow \{T_u(t)\}$, which is how students' behavior and internet addiction influence their daily time online. Then the internet addiction level a_u can be calculated from this model. Note that t above is in the set T.

4. Internet Addiction Calculation Model

To calculate students' internet addiction level, we propose three internet addiction calculation models: the linear internet addiction (LIA) model, the neural network internet addiction (NIA) model, and the clustering-based internet addiction (CIA) model. For the LIA model, we mainly consider the linear relation between students' behavior, internet addiction level, and their daily online time. Furthermore, since the neural network is powerful to capture the higher order relation among features, we explore the NIA model to find that nonlinear relation between students' behavior, internet addiction level, and their daily online time.

As for the CIA model, instead of directly studying the relation between students' behavior, internet addiction level, and their daily online time, we think that students who spend more time online than the normal online time are more likely to be addicted to the internet. So we devise a clustering-based method to find the normal online time and then regard the difference between students' actual online time and the normal online time as their internet addiction level.

In this chapter, we first describe these three models in detail, and then we will discuss the advantages and disadvantages of each model.

4.1. Linear Internet Addiction (LIA) Model.

In this section, we first introduce how we use a linear model to reveal the relationship of $\{a_u, p_u, B_u(t)\} \longrightarrow \{T_u(t)\}$. Then to strengthen the model, we take the regularity of students' behaviors into consideration.

4.1.1. Naive LIA. Based on the internet addiction assumption, the behavior is a factor which will influence students' online time. However, different kinds of behavior may have a different effect. Therefore, a weight vector is necessary to represent the different effects of each kind of behavior. The impact of behavior on online time is not different in individuals, so every student shares this weight vector. We deal with different kinds of personal attributes in the same way. Besides, even two students have the same behavior and personal attributes, and they may still spend different time online because of the difference in their addiction level towards the internet. We suppose that different internet addiction level is the only reason which causes different time online with the same behavior and personal attributes. Here comes our naive linear internet addiction model:

$$y_u(t) = wx_u(t) + a_u, \tag{1}$$

where $y_u(t)$ represents the duration student u spend online at time t. $x_u(t)$ refers to the combination of behavior vector and personal attributes of student u at time t, and w is the weight vector of that combined vector. a_u here is the internet addiction level of student u. Our task is to find the value of a_u and w that minimize the loss function, that is,

$$\arg\min_{w,a_u} \sum_{u\in U}\sum_{t\in T}\left(y_u(t) - w^T x_u(t) - a_u\right)^2 + \lambda\|w\|^2 + \mu\sum_{u\in U} a_u^2. \tag{2}$$

The item $\lambda\|w\|^2$ is used to prevent the model from overfitting. $\mu\sum_{u\in U} a_u^2$ can be used to adjust the weight between the behavior and internet addiction.

4.1.2. LIA with Regular Behavior. College students usually have a fixed curriculum. Therefore, their behavior has some regularity every week, which will also lead to the regularity of the time they spend online. Take student u as an example; courses on Monday are kind of boring, so he spends a lot of time surfing the internet. However, courses on Tuesday are hard, which means he must pay attention to the class, so he may not surf the internet in class. Based on such facts, it is necessary to take the regular online time into consideration.

So, we modify our linear internet addiction model by adding an item $d_u(\pi(t))$ to represent the regular online time of student u at time t. Due to the characteristics of the college study, they perform similar online habits every week. So here $\pi(t)$ means which day of time t is of the week it belongs to, and $d_u(x)$ means the regular online time of the day x of the week. Here comes our new model:

$$y_u(t) = wx_u(t) + a_u + d_u(\pi(t)). \tag{3}$$

For the convenience of calculation, we define $x_{2u}(t)$ as an 8-dimensional vector with the first item one standing for the internet addiction and others being a one-hot representation of the week. The formula above is equal to

$$y_u(t) = w^T x_u(t) + w_u^T x_{2u}(t), \tag{4}$$

with x_{2u} being equal to

$$\left(1, \pi_1(t), \pi_2(t), \pi_3(t), \pi_4(t), \pi_5(t), \pi_6(t), \pi_7(t)\right), \tag{5}$$

$$\pi_i(t) = \begin{cases} 1, & \text{if } \pi(t) = i; \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Our task is to find a suitable w and w_u that will minimize the loss function, the first item of w_u is the internet addiction level of student u:

$$\arg\min_{w,w_u} \sum_{u\in U}\sum_{t\in T}\left(y_u(t) - w^T x_u(t) - w_u^T x_{2u}(t)\right)^2 + \lambda\|w\|^2 + \mu\sum_{u\in U}\left\|w_u\right\|^2. \quad (7)$$

Similarly, we add $\lambda\|w\|^2$ to prevent the formula from overfitting, and we use the formula $\mu\|w_u\|^2$ to adjust the weights between behavior, personal attributes, internet addiction level, and regular habits.

4.2. Neural Network Internet Addiction (NIA) Model. The neural network is able to model the high-level relationship among features. It is powerful in a variety of application scenarios [18–20]. For example, in the tag recommendation task, Yuan et al. utilized the multilayer perceptron to model the nonlinearities of interactions among users, items, and tags [21]. In this section, we develop a neural network internet addiction (NIA) model to represent the nonlinear influence of students' behaviors, personal attributes, internet addiction, and their regular behavior on their daily online time.

4.2.1. Network Structure. The neural network consists of two parts: the public part and the private part. We use the public part to represent that the effect of the behavior and personal attributes on daily online time is not different in individuals, which means the input of the public part is the combination of the behavior vector of student u on time t and his personal attributes vector $x_u(t)$. The weight matrix V_c and the threshold vector λ_c of this part will update every iteration.

Because the internet addiction level and regular behavior are different in individuals, we use a private part to depict such characteristics. Every student has his own weight matrix V_u and threshold vector λ_u, and the parameters will only be updated when the corresponding student's data are used as the input. The private input x_2 of student u on time t is the same as vector (5). To ignore the influence of regular behavior, we can also only keep the first item of vector (5).

The target output of the model is the actual online time of student u on time t: $\hat{y}_u(t)$.

The structure of the network is shown as Figure 1.

Using the symbol we mentioned, the output of the public hidden layer is

$$b_c = f_c\left(V_c x_u(t) - \lambda_c\right). \quad (8)$$

The output of the private hidden layer is

$$b_u = f_u\left(V_u x_2 - \lambda_u\right), \quad (9)$$

and the output of the network is

$$y_u(t) = f_o\left(W\left(\operatorname{concat}\left(b_c, b_u\right)\right) - \theta\right), \quad (10)$$

where f_c, f_u, and f_o are the activation functions of the public hidden layer, private hidden layer, and the output layer and θ is the threshold of the output layer. The network will update for every input, and the loss function we use is the mean square error:

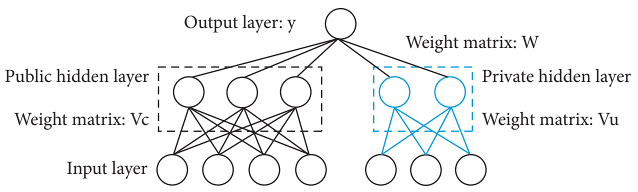


FIGURE 1: Neural network internet addiction model.

$$E = \frac{1}{2}\left(\hat{y}_u(t) - y_u(t)\right)^2, \quad (11)$$

where $\hat{y}_u(t)$ represents the actual online time of some student u on time t and $y_u(t)$ is the output of the whole model.

4.2.2. Internet Addiction Calculation. After the neural network training is completed, the sum of the contribution that internet addiction gives to the private hidden units is the value of students' internet addiction levels. We will calculate the internet addiction value as below:

$$a_u = \sum_{i=1}^{q_u} V_{uij}, \quad (12)$$

where q_u stands for the number of private hidden layer units. j is the corresponding index of internet addiction in the private part input vector, and here the index is one. V_u is the matrix, which connects the input layer and hidden layer of the private part. V_{uij} represents the i-th row and the j-th column value of the matrix V_u.

4.3. Clustering-Based Internet Addiction (CIA) Model. In this section, we develop a clustering-based method to calculate students' internet addiction value, which takes the similarity among students' behavior into consideration.

4.3.1. Internet Addiction Calculation. As the smartphone becomes an indispensable part of students' daily life, even the one not addicted to the internet will spend some time online, maybe for fun or just for killing time. However, those who are addicted to the internet heavily will spend much more time online than those who are not addicted to the internet. So, we believe that there is a normal online time corresponding to students' behavior, and those who spend more time online tend to be internet addicts. And the more time online compared with normal online time, the heavier the internet addiction level is. Therefore, here comes our online time prediction formula:

$$y_u(t) = n_u(t) + a_u, \quad (13)$$

where $y_u(t)$ represents the duration student u spends online at time t. $n_u(t)$ refers to the normal online time for student u at time t. a_u here is the internet addiction level of student u. Our task is to find the value of a_u that will minimize the loss function, that is,

$$\arg\min_{a_u} \sum_{u\in U}\sum_{t\in T}\left(y_u(t)-n_u(t)-a_u\right)^2+\mu\sum_{u\in U}a_u^2. \quad (14)$$

The item $\mu\sum_{u\in U}a_u^2$ is used to adjust the weight between the normal online time and internet addiction.

4.3.2. Normal Online Time. Due to this, the students have different activities every day, and normal online time differs from behavior to behavior. To find the normal online time $n_u(t)$ of student u at time t, we first need to find those who behave similarly with student u at time t. The average online time of those who behave similarly with student u at time t is approximately equal to the normal online time. That is,

$$n_u(t)=\frac{\sum_{(u',t')\in S} y_{u'}(t')\mathrm{sim}\left(x_u(t),x_{u'}(t')\right)}{\sum_{(u',t')\in S}\mathrm{sim}\left(x_u(t),x_{u'}(t')\right)}, \quad (15)$$

where $x_u(t)$ represents the behavior vector of student u at time t. S stands for the similar behavior set, and $x_{u'}(t')$ is one similar behavior vector of $x_u(t)$, which means the behavior vector of student u' at time t' is similar with that of student u at time t. Students from different departments may behave differently because of the discipline characteristic, which will lead to a slight difference in normal online time. For example, students from the software engineering department may tend to spend more time online than the other students. So, we also take the profile information into consideration, the symbol $x_u(t)$ here is equal to the vector $x_u(t)$ in Section 4.1. And the formula sim (a, b) is the similarity value of vector a and vector b.

Considering the calculation amount, we do not compare every behavior of all students at all the time. Instead, we first aggregate students' behavior into k categories. When we need to find the similar behavior set S of the behavior vector $x_u(t)$, we first find the category the behavior vector $x_u(t)$ belongs to; let us assume the category is c, and then we start to calculate the similarity between $x_u(t)$ and all the other behavior vectors $x_{u'}(t')$ in the category c. Finally, we keep those behavior vectors that have similarity greater than a threshold in the set S; based on which, we will get the normal online time $n_u(t)$.

4.4. Model Comparison. The idea of LIA and NIA is direct, and the target of these two models is to find how students' behavior and internet addiction level influence their daily online time. The LIA model is easier to train because it has fewer parameters than the NIA model. Though the NIA model is much more powerful, it is hard to train the network as there are so many parameters.

The idea of the CIA conforms to our intuitive thinking that those who spend more time online than the normal time are more likely to be addicted to the internet. However, it is hard to find the normal online time. In this paper, we calculate the normal online time of a student u on a specific day t by averaging the online time of those students who behave similarly with u on t. The correctness of internet addiction calculation may be influenced by the precision of the clustering results.

5. Experiments

5.1. Data Description. Our data come from a Chinese college, including students' consuming records in the school restaurant and internet access records. Besides, they also include the personal attributes information of students such as department, gender, and age.

The consumption records consist of students' profiles, time, place, and amount of one consumption. Students have various consumption behaviors like normal dining, snack, shower, and deposit. Here we consider deposit is a special behavior, which is saving money to the school card. The behavior category can be identified through the place where consumption behavior takes place. For example, consumption in the school restaurant must be normal dining behavior, and consumption in the bathhouse must be shower behavior. Therefore, we first divide the places into different categories and then extract the consuming amount on dining, snack, shower, deposit, and total consuming amount per hour from the consuming records. We also count students' daily consumption frequency.

Besides, students can access the internet using campus Wi-Fi only when they get authenticated. Based on the authentication record, we extract the time student accesses the campus Wi-Fi per hour. And such time is approximate to the time they spend on the campus. Similarly, at each time when a student visits a website, a connection record is generated. When the visit is completed, there will be a disconnected record. Based on these records, we can extract the student's actual online time and the average gap between two internet access per day. After feature extraction, combining the daily consuming behavior and online behavior (actual online time is excluded), the behavior of a student in a day can be represented as a vector. We also represent every student with the one-hot method using their profile information.

Due to some reasons, we do not have students' internet access records in the dormitory and library. It is considered that students' activities are mainly centralized around classrooms and canteens as well as some college student activity centers. In class, students need to listen to the teachers most of the time, and at the restaurant, they always play with a phone to kill time. Therefore, the actual online time we extract is mainly about the entertainment. Intuitively, the entertainment time is suitable to be used to calculate the internet addiction level.

We choose the records of undergraduate students enrolled in 2016 and 2015 from September 1, 2018 to November 11, 2018. After dropping students with record number less than 35 days, there are 3767 students. The first 50 records are used for training, and the left records are used for testing. Students' profile representation and daily behavior vector are shown in Table 1.

Table 1: Features used in experiments.

Type	Feature	Dimension	Representation
Profile	Gender	2	One hot
	Department	61	
	Age	8	
Consumption	Dining amount	24	Statistical value
	Snack amount	24	
	Shower amount	24	
	Deposit amount	24	
	Total amount	24	
	Frequency	1	
Internet	Wi-Fi access time	24	Statistical value
	Internet access gap	1	

Table 2: Regression results.

Model	Feature		
	ia−	ia	ia+
LIA	0.000056	**0.000048** (14.3%)	0.000050 (10.7%)
NIA	0.000092	**0.000083** (9.8%)	0.000086 (6.5%)
CIA	0.000138	**0.000127** (8.0%)	No such condition

"ia−" refers to the baseline experiment; "ia" represents the second experiment; "ia+" stands for the third experiment.

5.2. Internet Addiction Calculation. LIA, NIA, and CIA models can be used to study the internet addiction level by predicting students' online time every day. To show the correctness of our models, we conduct several experiments.

For LIA and NIA models, we conduct three experiments. The first experiment removes the internet addiction and regular behavior part of LIA and NIA models and predicts students' daily online time using students' behavior data and profile information, which is considered as a baseline. The second experiment only takes internet addiction into consideration. For LIA, it means using the naive LIA model, and as for NIA, it means there is only one item of the input of the private part. The last experiment takes internet addiction and regular behavior into consideration. For LIA, it means using LIA with regular behavior model, and for NIA, it means there is 8 items of the input of the private part. For the CIA model, we conduct two experiments: the first experiment uses the average online time in the similar behavior set S as the prediction, which is considered as a baseline. And the other experiment first calculates the internet addiction value of each student using equation (14) and then predicts students' online time using their neighbors' actual online time and the internet addiction value a_u by equation (13).

For the linear model, the value of λ is set to 0.6, and μ is set to 0.4. For the neural network model, the activation function of the hidden layer is $f(x) = x$, and the activation function of the output layer is $f(x) = \tanh(x)$. In addition, the number of public hidden layer units is 10, and the number of private hidden layer units is 2. The learning rate is set to 0.01, and the number of the epoch is 40. Note that for the third experiment of the NIA model, we set the learning rate to 0.05 which will get the best prediction accuracy. For the clustering-based model, the threshold is set to 0.7 and the cluster number is set to 50. The MSE performance of each method is shown in Table 2.

From the results in Table 2, we know that no matter which model, the prediction accuracy increases with our internet addiction assumption. Such results guarantee the correctness of our internet addiction assumption. However, for the LIA and NIA models, adding the assumption of regular behavior, the accuracy does not improve compared to the results without such an assumption. One possible reason is that there is some volatility in students' behavior; however, LIA and NIA are not able to model it. Generally, the results of the neural network model and clustering-based model are worse than that of the linear model. Maybe it is because the linear model is strong enough to represent the relationship between students' behavior, internet addiction, and online time. And there are too many parameters in the neural network model, which is not easy to train. Though clustering students into several categories before calculating the similarity will reduce the computing complexity, the prediction results depend on the clustering results, and that may cause some error. The bias of the clustering results may be a reason leading the worst prediction accuracy of the CIA model.

5.3. Internet Addiction Verification. In this section, we conduct some experiments to verify the correctness of the methods we propose. First, we show the consistency of the internet addiction value we calculated using the models we proposed and the value evaluated through the psychological scale. And then, we devise regression and classification tasks to verify the critical role the internet addiction value we calculated plays on daily online time prediction task.

5.3.1. Comparison with the Psychological Scale. In psychology, researchers usually use the internet addiction scale to measure if people are addicted to the internet. Therefore, we use a questionnaire to test if a student is an internet addict and compare the results calculated by the questionnaire with that by our method.

In consideration of the national condition of China, we choose the internet addiction scale devised by professor Fan [22], which is widely used in Chinese psychological researches. As the situation today is not exactly the same with that several years ago, we cut some questions on that scale and only keep five necessary questions. And we use 4 points Likert scale to measure the degree of each question. See Table S1 in the Supplementary Material section for the details of the scale we used.

After giving the questionnaire to students, we retrieve 128 questionnaires, which are enough to analyze students' internet addiction levels in the psychological field. The students who complete the questionnaire consist of 78 males and 50 females, and there are around 81 students in grade 3 and 47 students in grade 4, which shows the samples are evenly distributed.

To show the effectiveness of the new scale we use, we calculate the reliability and validity of our scale, which are two dimensions to test if a scale is credible to use in psychology. The reliability and validity of our scale are 0.789 and 0.731 separately. The higher the value of the reliability and validity is, the better the scale is, and 0.7 means that our scale is credible enough to test the internet addiction.

On the principle of voluntariness, we did not force students to write down their student id or name. Since there are only 39 students who volunteer to give us their student id, we mainly compare those students' results judged by the psychological scale and that by our methods. There are five questions on our scale. Because we use 4 points Likert scale to measure, the total grade is 20. The greater the grade a student gets, the more likely this student is addicted to the internet. We define those whose grade less than ten is not addicted to the internet, and the others are internet addicts. As for the results calculated by the LIA model, we consider those whose value greater or equal than 0.45 is addicted to the internet. The threshold of NIA and CIA models is set to 0.5 and 0.35 separately. 0.45, 0.5, and 0.35 are approximate to the average value of the corresponding method. We use F1 score to evaluate the consistency between the results of the LIA model, the NIA model, the CIA mode, and the psychological scale. The results are shown in Table 3.

From Table 3, we see that all the internet addiction values calculated through these three models are consistent with the results evaluated from the psychological scale. Particularly, though the CIA model performs poorly in the internet addiction calculation task, comparing with the NIA model, the internet addiction value of the CIA model is more consistent with the psychological scale results. Such results show the correctness of our methods and give us a clue that the relationship among behaviors is an important factor when calculating the internet addiction value.

5.3.2. Online Time Prediction. Based on our assumption, internet addiction is a hidden variable, which will influence students' daily time online. Therefore the learned internet addiction value should be a useful feature to predict students' online time. We devise two tasks to verify the correctness of our learned internet addiction value.

The aim of the regression task is to predict students' daily online time. The baseline experiment takes the daily behavior vector and the profile information as the input. The contrast experiment predicts the daily online time using students' internet addiction value, daily behavior vector, and profile information. For the classification task, it is similar to the regression task. First, the records are divided into two parts: one part with online time greater than or equal to the average online time, the other part with an online time less than the average online time. The aim of the classification task is to predict which part online time belongs to. The experiment settings are the same as the regression task. The methods used in the regression task and classification task consist of the decision tree (DT), support vector machine (SVM), k-nearest neighbors (KNN), random forest (RF), gradient boosting decision tree (GBDT), bagging and extremely randomized trees (ET).

MSE is used as the evaluation method for the regression task, and F1 score for the classification task. The results are shown in Tables 4 and 5.

Table 3: F1 score between the results of our methods and psychological scale.

Model	LIA	NIA	CIA
F1 score	0.71	0.63	0.71

From Table 4, we observe that, for the regression task, the SVM model gets a huge mean square error. One possible reason may be that it is not suitable for this task, so we will ignore the SVM results in the discussion below. After adding the internet addiction value calculated by LIA and CIA models, all the prediction accuracies lift. And after adding internet addiction value calculated by NIA, although the promotion of prediction accuracy is not as remarkable as that of adding the value calculated by LIA or CIA models, most of the methods still get some promotion.

For the classification task, no matter which internet addiction value is added to the behavior vector, except for the effect of the SVM method has not changed, the effect of all the other methods has evidently been improved.

Generally speaking, after adding the internet addiction value calculated by LIA, NIA, or CIA, both regression and classification tasks get a remarkable promotion, which shows the effectiveness of the internet addiction value learned by the models we propose.

5.4. Internet Addiction Analysis. To show the internet addiction situation in college, we analyze the distribution of internet addiction and the differences of the internet addiction level among different groups such as different gender and department. Because the naive LIA model has the best prediction accuracy when studying students' internet addiction value and the value learned through the naive LIA model is the most consistent with the psychological results, the following analysis is based on the value calculated by the naive LIA model.

5.4.1. Internet Addiction Distribution. Figure 2(a) illustrates the number of students with respect to the calculated internet addiction value. The greater the internet addiction value is, the more serious students' addiction towards the internet is. We observe that internet addiction distribution is similar to a normal distribution. To show the distribution of the internet addiction value clearly, we delete the value greater than 0.7 or less than 0.2, which is shown in Figure 2(b). If we define internet addiction less than 0.45 is normal, from Figure 2(b), we observe that most of the students are addicted to the internet with different levels.

5.4.2. Internet Addiction Differences among Groups. To reveal the differences in internet addiction between genders, we count the average internet addiction value of different genders. And we also count the average online time of different genders. Figure 3 shows that girls spend more time on the internet than boys. However, boys are more addicted to the internet than girls. Such a result is consistent with the finding in the psychological field. Wei et al. investigated the

Table 4: Regression task.

Feature model	ia–	ia (LIA)	ia (NIA)	ia (CIA)
DT	0.000076	**0.000061** (19.7%)	0.000072 (5.3%)	0.000064 (15.8%)
SVM	0.004114	0.003636 (11.6%)	0.003677 (10.6%)	**0.003024** (26.5%)
KNN	0.000065	**0.000064** (1.5%)	0.000066 (–1.5%)	**0.000064** (1.5%)
RF	0.000040	0.000039 (2.5%)	0.000040 (0%)	**0.000038** (5%)
GBDT	0.000042	**0.000039** (7.1%)	0.000042 (0%)	0.000040 (4.8%)
Bagging	0.000041	**0.000039** (7.3%)	0.000041 (0%)	**0.000039** (7.3%)
ET	0.000068	**0.000057** (16.2%)	0.000065 (4.4%)	0.000065 (4.4%)

"ia–" refers to the baseline experiment and "ia (LIA)" stands for the experiment with the internet addiction value learned by the naive LIA model, which gets the best results in the internet addiction calculation task using the LIA model. "ia (NIA)" represents the experiment with the best internet addiction value learned by the NIA model without regular behavior consideration, which gets the best results in the internet calculation task using the NIA model. Similarly, "ia (CIA)" refers to the experiment with the internet addiction value learned from the clustering-based model.

Table 5: Classification task.

Feature model	ia–	ia (LIA)	ia (NIA)	ia (CIA)
DT	0.960643	0.997667 (3.9%)	0.997989 (3.9%)	**0.998899** (4.0%)
SVM	0.960773	0.960773 (0%)	0.960773 (0%)	0.960773 (0%)
KNN	0.959270	0.970958 (1.2%)	**0.981605** (2.3%)	0.973924 (1.5%)
RF	0.967783	0.978654 (1.1%)	0.979268 (1.2%)	**0.981382** (1.4%)
GBDT	0.959812	0.961584 (0.2%)	0.960936 (0.1%)	**0.962591** (0.3%)
Bagging	0.965652	0.998827 (3.4%)	0.998481 (3.4%)	**0.999378** (3.5%)
ET	0.958128	0.966017 (0.8%)	**0.970712** (1.3%)	0.969175 (1.2%)

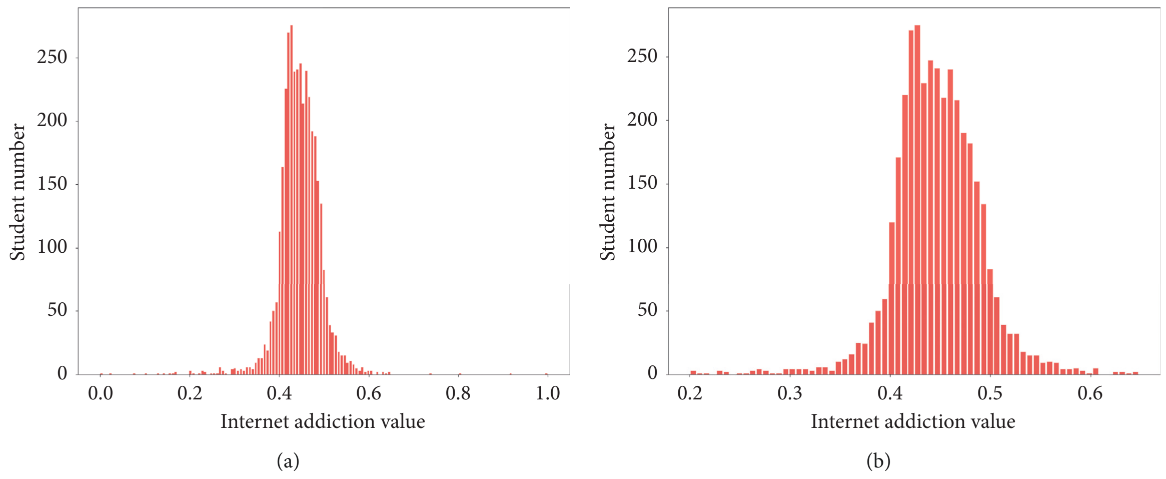


Figure 2: Internet addiction distribution. (a) Number of students with different levels of addiction. (b) Some students with the value greater than 0.7 or less than 0.2 are deleted.

internet addiction situation of the college student in Hubei Polytechnic University using questionnaires.

They point out that boys are usually not good at communication, and therefore, the communication in real life is not enough to meet their actual communication needs. The way of communication with the network as the medium is easier to control; that is, they can improve the quality and quantity of communication in this way, which meets their needs of communication. Besides, Girls are better than boys in time management ability and deal with network use time more reasonably. So boys are more addicted to the internet than girls [23]. The consistency with the findings of psychology further proves the correctness of the internet addiction value we learned.

Figure 4(a) illustrates the average internet addiction level of different departments. In general, except the internet addiction level of a few departments is extremely high, it fluctuates around 0.43. Furthermore, we statistically analyze the differences in internet addiction levels among students in different disciplines. In Figure 4(b), we can observe that there is no significant difference in internet addiction levels among students in different disciplines. The result is also consistent with the psychological finding in [23]. Experiments conducted by Wei et al. that demonstrate though there is some difference in the interpersonal health and time management ability among students in different disciplines, the difference is not significant. And the difference in internet addiction is not significant. The consistent result with

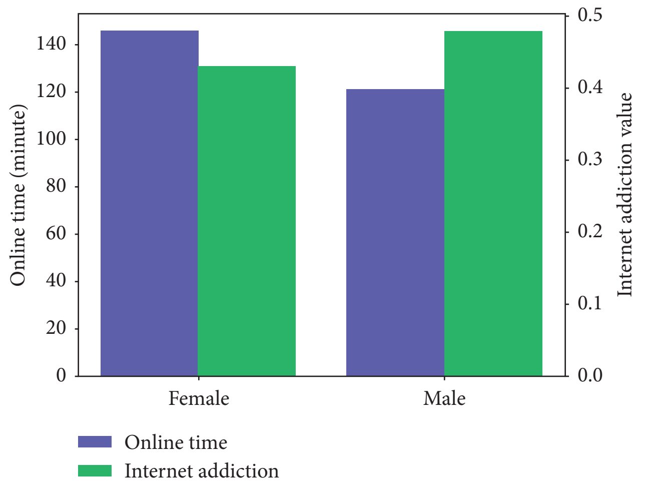


FIGURE 3: Differences of online time and internet addiction between different genders.

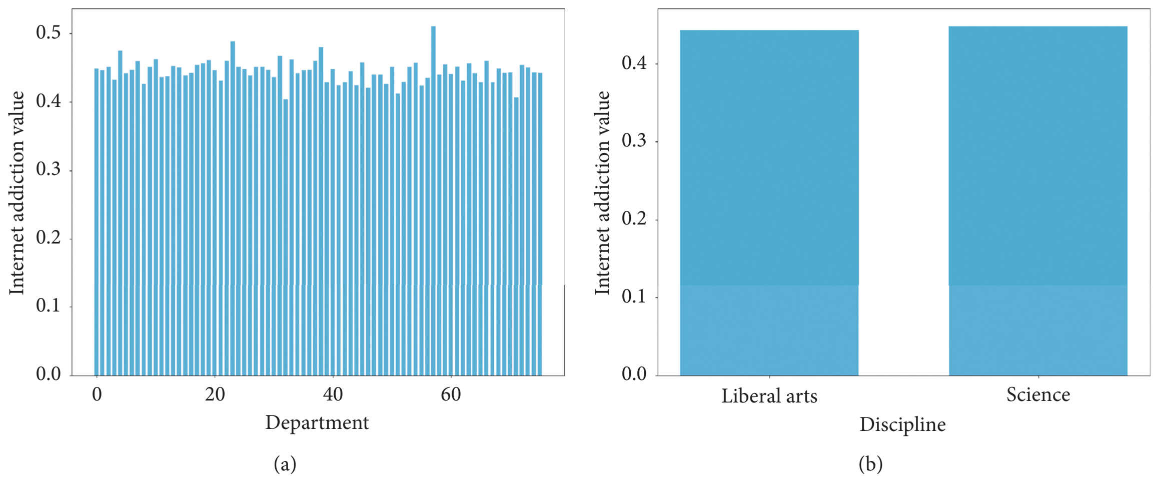


FIGURE 4: Differences of internet addiction among different departments and disciplines. (a) Department. (b) Discipline.

psychological findings is also evidence of the effectiveness of the internet addiction value we learned.

5.4.3. Effect of Internet Addiction on Online Time. The decision tree is a classical machine learning model. It is good at classification and regression tasks, and it is interpretable. Therefore, the decision tree model has plenty of applications in various fields [24–26]. To show the role internet addiction plays when predicting students' online time, we extract students' daily Wi-Fi access time, consuming amount, consuming frequency, average internet access gap, and actual online time. Then we conduct two binary classification experiments using classification and regression decision tree method: one predicts online time interval with daily Wi-Fi access time, consuming amount, consuming frequency, and average internet access gap, and the other predicts online time interval with daily Wi-Fi access time, consuming amount, consuming frequency, average internet access gap, and internet addiction value. Because the whole tree is too big to be put here, we select two representative branches. Note that all the values are normalized. The average value of the internet addiction value, consuming amount, consuming frequency, Wi-Fi access time, internet access gap, and online time is 0.45, 0.009, 0.044, 0.062, 0.004, and 0.015 separately.

From Figure 5(a), we know that Wi-Fi access time and average internet access gap are important features when predicting the online time. It is consistent with our intuitive thinking that less Wi-Fi access time and a long internet access gap will cause less online time. Figure 5(b) illustrates that after adding the internet addiction value, the value is critical for predicting daily online time. Particularly, in this branch, the relatively high internet addiction value is a reason leading to long online time.

5.4.4. Effect of Internet Addiction on Grade. The psychological research shows that internet addiction will damage students' study [1]. To show the bad influence of internet addiction and to verify the correctness of the internet addiction value we calculated, we do some statistics about the grades of those who are addicted to the internet and those who are not.

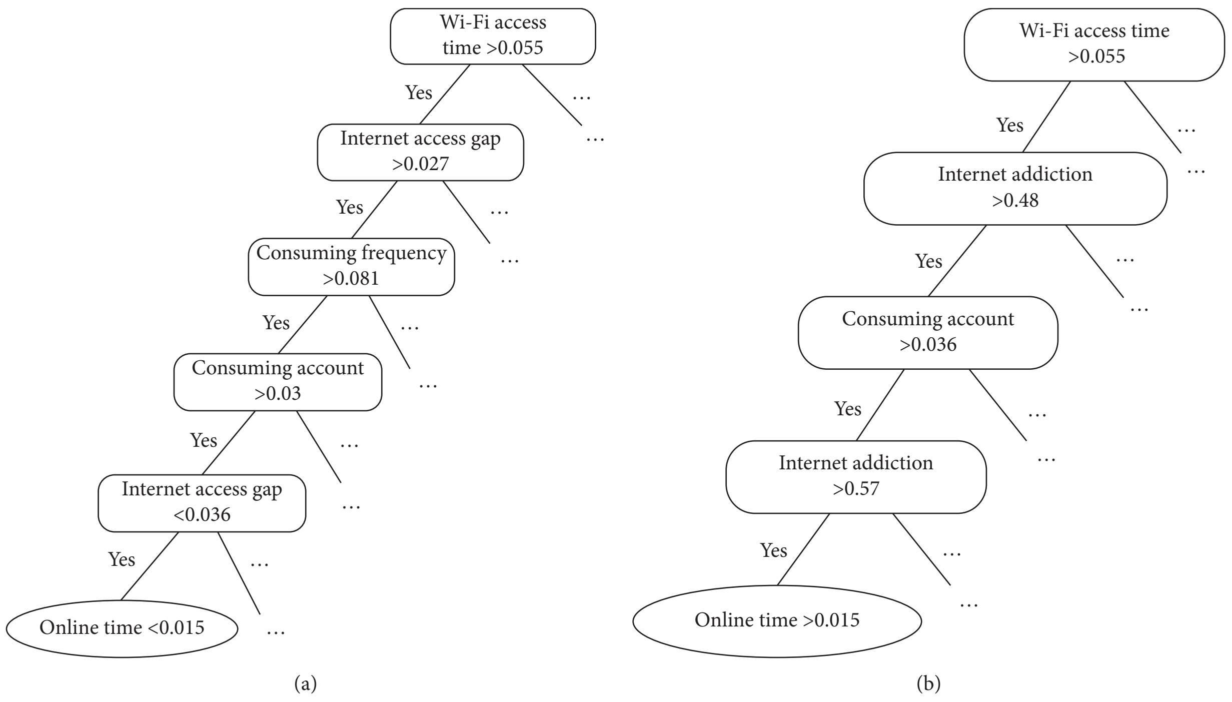


Figure 5: Decision tree with behavior and internet addiction value. (a) Prediction with behavior data. (b) Prediction with behavior data and internet addiction value learned through naive LIA.

As we mentioned before, there are only 39 students who volunteer to give us their student id, and one of them does not have any grade records, so the analysis of this part is mainly based on the grades of the remaining 38 students.

First, we define that students whose internet addiction values equal to or more than 0.45 are internet addicts, and the others are not. We divide students into two groups based on their internet addiction values. Then we calculate their average grade point of the second semester in 2018. At last, we count the average grade point and student number who failed at least one course of each group. The average grade of each student is calculated with the formula below:

$$G(u) = \sum_{c \in C(u)} \operatorname{cred}(c) * \operatorname{gp}(c), \tag{16}$$

where $G(u)$ refers to the average grade point of student u of the second semester in 2018, $C(u)$ stands for all the courses student u takes in this semester, $\operatorname{cred}(c)$ is the credit of course c, and $\operatorname{gp}(c)$ is the grade point of course c student u gets.

The analysis results are shown in Table 6.

From this table, we see that almost half of the students are addicted to the internet. And the average grade point of students who are addicted to the internet is significantly lower than the normal students. There are more students who failed the exam in the internet addicts group than that in the other group. The statistics conform to the psychological findings that internet addiction has a bad influence on students' study. Such results further verify the correctness of the internet addiction value we calculated.

Table 6: Classification task.

	Stu number	Average G	Failed stu number
Stu with ia	18	2.75	3
Stu without ia	20	3.21	1

"stu with ia" refers to the students who are addicted to the internet, and "stu without ia" refers to those who are not addicted to the internet. "Average G" stands for the average grade point of all the students in each group. "Failed stu number" is the number of students who fail at least one course in each group.

6. Conclusions

In this paper, we estimate college students' internet addiction levels quantitatively using their behavior data on the campus. Specifically, we define the internet addiction value as a hidden variable which will affect students' online time and formulate the problem as a regression problem.

Along this line, we first propose a linear internet addiction (LIA) model, which depicts the linear relationship among students' internet addiction level, behavior data, and time they spent online. To model the nonlinear relationship, we also provide a neural network internet addiction (NIA) model. Besides, we also develop a clustering-based internet addiction (CIA) model, which calculates the internet addiction based on the differences between students' actual online time and normal online time. These three models also take students' regular behavior and the similarity among students' behavior into consideration.

Finally, we conduct excessive experiments on a real-world dataset from a Chinese college, and the experimental results demonstrate the effectiveness of our model. The analysis results are consistent with some psychological findings, which also verify the correctness of the models we propose.

Data Availability

The behavior data used to support the findings of this study have not been made available due to privacy concerns.

Disclosure

It is an extension of the paper *Using Behavior Data to Predict the Internet Addiction of College Students* [27] which is published in the International Conference on Web Information Systems and Applications (WISA) in 2019.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key R&D Program of China (no. 2017YFC0803700) and NSFC grants (nos. 61532021 and 61972155).

Supplementary Materials

Table S1. Internet usage survey of college student. (*Supplementary Materials*)

References

[1] Internet addiction disorder, 2019, https://en.wikipedia.org/wiki/Internet_addiction_disorder.

[2] N. Upadhayay and S. Guragain, “Internet use and its addiction level in medical students,” *Advances in Medical Education and Practice*, vol. 8, pp. 641–647, 2017.

[3] Y. Xue, Y. Dong, M. Luo et al., “Investigating the impact of mobile SNS addiction on individual’s self-rated health,” *Internet Research*, vol. 28, no. 2, pp. 278–292, 2018.

[4] A. Fumero, R. J. Marrero, D. Voltes, and W. Peñate, “Personal and social factors involved in internet addiction among adolescents: a meta-analysis,” *Computers in Human Behavior*, vol. 86, pp. 387–400, 2018.

[5] M. Z. Malak, A. H. Khalifeh, and A. H. Shuhaiber, “Prevalence of Internet Addiction and associated risk factors in Jordanian school students,” *Computers in Human Behavior*, vol. 70, pp. 556–563, 2017.

[6] W. He, A. Qi, Q. Wang et al., “Abnormal reward and punishment sensitivity associated with Internet addicts,” *Computers in Human Behavior*, vol. 75, pp. 678–683, 2017.

[7] Y. Zhang, X. Qin, and P. Ren, “Adolescents’ academic engagement mediates the association between Internet addiction and academic achievement: the moderating effect of classroom achievement norm,” *Computers in Human Behavior*, vol. 89, pp. 299–307, 2018.

[8] F. Zhao, Z.-H. Zhang, L. Bi et al., “The association between life events and internet addiction among Chinese vocational school students: the mediating role of depression,” *Computers in Human Behavior*, vol. 70, pp. 30–38, 2017.

[9] K. Wang, Y. Jin, H. Wang, H. Peng, and X. Wang, “Personalized time-aware tag recommendation,” in *Proceedings of the 32th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 459–466, AAAI-18, New Orleans, LA, USA, February 2018.

[10] C. Guan, X. Lu, X. Li et al., “Discovery of college students in financial hardship,” in *Proceedings of the 2015 IEEE International Conference on Data Mining*, pp. 141–150, IEEE, Atlantic City, NJ, USA, November 2015.

[11] H. J. Ye, D. C. Zhan, X. Li et al., “College student scholarships and subsidies granting: a multi-modal multi-label approach,” in *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 559–568, IEEE, Barcelona, Spain, December 2016.

[12] K. Wang, X. Zhao, H. Peng, and X. Wang, “Bayesian probabilistic multi-topic matrix factorization for rating prediction,” in *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, pp. 3910–3916, New York, NY, USA, July 2016.

[13] Y. Zhu, H. Zhu, Q. Liu, E. Chen, H. Li, and H. Zhao, “Exploring the procrastination of college students: a data-driven behavioral perspective,” in *Database Systems for Advanced Applications*, pp. 258–273, Springer, Cham, Switzerland, 2016.

[14] J. Peng, Y. Zhou, X. Sun et al., “Social media based topic modeling for smart campus: a deep topical correlation analysis method,” *IEEE Access*, vol. 7, pp. 7555–7564, 2018.

[15] G. N. Burlak, J. Hernandez, A. Ochoa et al., “The use of data mining to determine cheating in online student assessment,” in *Proceedings of the Electronics, Robotics and Automotive Mechanics Conference (CERMA’06)*, vol. 1, pp. 161–166, IEEE, Cuernavaca, Mexico, September 2006.

[16] S. Abdi, H. Khosravi, and S. Sadiq, “Predicting student performance: the case of combining knowledge tracing and collaborative filtering,” in *Proceedings of the International Conference on Educational Data Mining*, Buffalo, NY, USA, July 2018.

[17] W. Liu, C. Y. Bao, and B. Wen, “Investigation on the Internet dependence of undergraduates and analysis of correlative causes,” *Chinese General Practice*, vol. 13, no. 8, pp. 2485–2487, 2010.

[18] M. A. Albahar, “Skin lesion classification using convolutional neural network with novel regularizer,” *IEEE Access*, vol. 7, pp. 38306–38313, 2019.

[19] A. Almuhareb, W. Alsanie, and A. Al-Thubaity, “Arabic word segmentation with long short-term memory neural networks and word embedding,” *IEEE Access*, vol. 7, pp. 12879–12887, 2019.

[20] S. Al-Dahidi, O. Ayadi, M. Alrbai, and J. Adeeb, “Ensemble approach of optimized artificial neural networks for solar photovoltaic power prediction,” *IEEE Access*, vol. 7, pp. 81741–81758, 2019.

[21] J. Yuan, Y. Jin, W. Liu, and X. Wang, “Attention-based neural tag recommendation,” in *Database Systems for Advanced Applications (DASFAA)*, pp. 350–365, Springer, Cham, Switzerland, 2019.

[22] F. Fan and Y. Bai, “A study on the internet dependence of college students: the revising and applying of a measurement,” *Psychological Development and Education*, vol. 24, no. 2, pp. 187–203, 2005.

[23] Y. Y. Wei, G. S. Huang, Z. B. Xie et al., “Research on the relationship between students’ internet dependence and loneliness by taking Hubei polytechnic university as an example,” *Journal of Liuzhou Vocational & Technical College*, vol. 3, 2018.

[24] J. Cheng, G. Li, and X. Chen, “Research on travel time prediction model of freeway based on gradient boosting decision tree,” *IEEE Access*, vol. 7, pp. 7466–7480, 2018.

[25] M. B. B. Heyat, D. Lai, and F. I. K. Y. Zhang, “Sleep bruxism detection using decision tree method by the combination of C4-P4 and C4-A1 channels of scalp EEG,” *IEEE Access*, vol. 7, pp. 102542–102553, 2019.

[26] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, “Fast HEVC to SCC transcoder by early CU partitioning termination and decision tree-based flexible mode decision for intra-frame coding,” *IEEE Access*, vol. 7, pp. 8773–8788, 2019.

[27] W. Peng, X. Zhang, and X. Li, “Using behavior data to predict the internet addiction of college students,” in *Web Information Systems and Applications*, pp. 151–162, Springer, Cham, Switzerland, 2019.

Hindawi
Scientific Programming
Volume 2019, Article ID 5941096, 20 pages
https://doi.org/10.1155/2019/5941096

Research Article

An Intelligent Data Analysis for Recommendation Systems Using Machine Learning

Bushra Ramzan,[1] Imran Sarwar Bajwa,[1] Noreen Jamil,[2] Riaz Ul Amin,[3] Shabana Ramzan,[4] Farhan Mirza,[5] and Nadeem Sarwar[6]

[1]*Department of Computer Science & IT, The Islamia University, Bahawalpur 63100, Pakistan*
[2]*Department of Computer Science, FAST National University, Islamabad, Pakistan*
[3]*Faculty of Computing, BUITEMS, 83100 Quetta, Pakistan*
[4]*Department of Computer Science, Govt. Sadiq College Women University, Bahawalpur, Pakistan*
[5]*School of Engineering, Computer & Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand*
[6]*Department of Computer Science, Bahria University, Lahore, Pakistan*

Correspondence should be addressed to Imran Sarwar Bajwa; imran.sarwar@iub.edu.pk

Received 27 May 2019; Revised 13 August 2019; Accepted 30 August 2019; Published 31 October 2019

Guest Editor: Aibo Song

In recent times, selection of a suitable hotel location and reservation of accommodation have become a critical issue for the travelers. The online hotel search has been increased at a very fast pace and became very time-consuming due to the presence of huge amount of online information. Recommender systems (RSs) are getting importance due to their significance in making decisions and providing detailed information about the required product or a service. To acquire the hotel recommendations while dealing with textual hotel reviews, numerical ranks, votes, ratings, and number of video views have become difficult. To generate true recommendations, we have proposed an intelligent approach which also deals with large-sized heterogeneous data to fulfill the needs of the potential customers. The collaborative filtering (CF) approach is one of the most popular techniques of the RS to generate recommendations. We have proposed a novel CF recommendation approach in which opinion-based sentiment analysis is used to achieve hotel feature matrix by polarity identification. Our approach combines lexical analysis, syntax analysis, and semantic analysis to understand sentiment towards hotel features and the profiling of guest type (solo, family, couple etc). The proposed system recommends hotels based on the hotel features and guest type for personalized recommendation. The developed system not only has the ability to handle heterogeneous data using big data Hadoop platform but it also recommends hotel class based on guest type using fuzzy rules. Different experiments are performed over the real-world datasets obtained from two hotel websites. Moreover, the values of precision and recall and F-measure have been calculated, and the results are discussed in terms of improved accuracy and response time, significantly better than the traditional approaches.

1. Introduction

In the modern era of advancing web technologies, the recommender systems (RSs) have turned the notice of the business society and the common man towards itself due to its significance and importance in the e-commerce and achievement of superior customer's approval. Nowadays e-commerce is believed to be strongly connected to the customer's satisfaction, and an ultimate success is always dependent on customer loyalty. The same is with the online booking and reservation systems being a main component of the tourism industry. Mariani et al. [1] discussed that the most powerful and popular industry which has a major impact on total GDP of world economy is tourism. Tourists around the world are always looking for best hotels for their residence during the tours which keeps the recommender systems as their primary choice to obtain best available hotel choices for online reservations well before reaching their destinations in order to avoid any future residential trouble in hotels.

Liu et al. [2] have discussed that, in recent past, some recommender systems are made in order to facilitate tourists to get a list of hotel recommendations before making any booking. The nature of most of the data on Internet and web is heterogeneous becoming a hurdle for recommender systems because conventional recommender systems are dealing with homogenous data only which compromises the performance of hotel recommendation systems. Complex data in the multiple forms such as numeric, text, and visuals require developers' attention to develop recommender systems dealing with the heterogeneity of data. Li et al. [3] observed that recently few recommenders are available in the market having some capabilities to deal with heterogeneous data in which they have used ratings obtained from a customer feedback but do not include reviews, votes, ranks, and video views from the user feedback available on social media. In the proposed approach, we have also used multitypes of user feedbacks such as votes and YouTube video views. Our proposed recommender system gives two-fold novelty and advantage; first, it uses a hotel feature matrix to recommend a suitable hotel to a user on the basis of both quantitative (numerical) and qualitative (textual) features by using machine learning classification to achieve true recommendations; it mines user contextual information and extracts sentiments from reviews by analyzing the other travelers' reviews together with the ranks, votes, and YouTube video views to improve the recommendation accuracy. Second, a fuzzy module provides the recommendation of hotels in a particular type of user such as solo, family, business, friends, and couple because recommendations will be different based on type of user trip and user preference. Like for a family, "room," "food," and "cleanliness" facilities are the main preferences but for a single guest, facilities like "pool," "spa," and "gym" may have a greater preference. Similarly "WiFi" and "computer" can be an important feature for the user who is on a business trip.

Zhang and Mao [4] suggested that recommender systems are developed to achieve true and relevant recommendations. Relevant recommendation means the recommendation which is according to the customer's preference and choice. Usually, a recommender system uses customer ratings and reviews from the previous data considering the hotel's attributes or features. So the main challenge in this paper is to develop an intelligent approach which processes and analyzes large heterogeneous web data to achieve true hotel recommendations which are relevant to the customer's choice.

While dealing with diverse nature of data in our multifeature hotel recommendation system, the main challenge was the opinion mining/sentiment analysis of users' reviews to calculate a polarity score which represents the degree of likeness or dislikeness about a hotel by a user. A typical recommender usually banks on previous users' ratings about the hotel's attributes or features, but our proposed recommender also uses reviews, numerical rank votes, and video views to take true results of users' multitype feedback. The polarity score provides a textual side of user's opinions about a particular hotel. To handle the diversity of heterogeneous data as the presented approach uses both numeric data as well as textual data, a big data solution involving Hadoop was used in our approach because it efficiently handles data heterogeneity and data diversity in a better way. We have defined the guest type (solo, family, business, friends, and couple) as the main part of this research. We will not only consider different rating parameters but also apply feature based sentimental analysis on user's reviews. For example, Trip Advisor allows travelers to rate hotels on several options such as such as location, room, cleanliness, service, and staff. The process of extracting opinions from textual reviews is called as sentiment analysis/opinion mining.

There are number of studies present in the literature to perform sentiment analysis with the state-of-the-art methods to handle reviews to provide recommendations. Machine learning classification is the one of the most useful techniques for the sentiment classification of categorized text into positive, negative, or neutral categories. In machine learning technique, training and testing datasets are essential. A training dataset is used to learn the documents, and the test dataset is used to validate the performance. There are four main types of machine learning to classify reviews as shown in Table 1.

Within the field of data analytics, machine learning is part of a piece known as predictive analytics. Machine Learning algorithms are not series of processes serially executed to produce a predefined output. They are instead series of processes aiming to "learn" patterns from past events and build functions that can produce good predictions, within a degree of confidence.

Machine learning based-sentiment analysis or classification is used to classify and provide recommendations for the users. In supervised machine learning techniques, two types of data sets are required: training dataset and test data set. An automatic classifier learns the classification factors of the document from the training set, and the accuracy in classification can be evaluated using the test set. The key step in the supervised machine learning technique is feature selection. The classifier selection and feature selection determine the classification performance.

The main purpose of our recommender system is to provide suggestions and recommendations which are truly based on customer's preference and choice. Koren et al. [5] has focused on the quality of recommendations. It is suggested by the author that when a large number of user ratings and reviews are used to be processed to provide efficient and true recommendations, then quality of recommendations is significantly important [6]. Booking through online systems and recommenders has increased in recent years, and multinational organizations are working over this domain to take maximum advantage. A recommendation system (RS) helps customers not only finding appropriate hotels but also it is benefitting in all domains such as movies, books, and all sorts of other different products and items. Different types of data, i.e., hotels, movies, and music, can be processed by RS. The generic recommender architecture is presented in Figure 1. Hsieh et al. [6] have explained in detail that different recommender systems are built using different methods and

Table 1: Comparison of Machine learning approaches.

No.	Machine learning approaches	Description
1	Supervised learning	It uses previous data as input variable to predict the most probable output value for new data, depending upon those associations learned from the previous data sets. (i) *Regression* is a form of predictive modeling technique which examines the relationship between a dependent variable and an independent variable. (ii) *Classification* is the technique in which the algorithm learns from the data input given to it and then uses this learning to classify and produce new observation.
2	Unsupervised learning	It uses unlabeled data that have no historical labels to train the algorithm. The purpose is to find some structure within it by exploring the data. (i) Clustering groups a set of objects in such a way that objects in the same group are more similar to each other in some respect than to those in other groups. (ii) *Dimensionality reduction* removes useless data before analysis. This is used to remove redundant data and outliers.
3	Reinforcement learning	(i) With this approach, the algorithm discovers through trial and error which trials produce the best rewards. (ii) It is often used for gaming, navigation, and robotics.
4	Deep learning	(i) Deep Learning helps in training computers to deal with the problems that are not well defined. (ii) Deep learning and neural networks are often used in speech and image recognition applications.

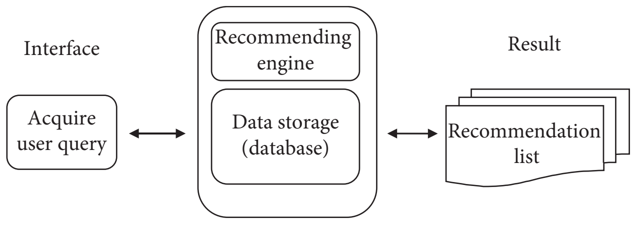


Figure 1: Generic architecture of a recommender system.

algorithms which uses customers' previous data consisting reviews and ratings about the different products to obtain true recommendations. Burke [7] has discussed about different types of recommendation algorithms. It explained that there are two recommendation techniques. First is collaborative filtering (CF), and the other is content-based filtering (CBF). Mixer of these two algorithms is called as hybrid filtering.

Lops et al. [8] have explained that the most widespread recommender filtering technique is collaborative filtering; however, users' preferences and choices are presented by their linked points in the content-based recommender system. CF works by collecting user ratings for items in a given domain and calculating similarities between users or items in order to provide relevant recommendations. Ekstrand et al. [9] elaborate that collaborative filtering uses a class of methods and utilizes preferences of other users which they have expressed for the same items to recommend items to the active user. This technique can also be beneficial in all other domains where the customer preferences can randomly change.

Collaborative filtering algorithms can be of two kinds, i.e., item-based recommender system and user-based recommender system. Item-based recommenders compare item similarities, and the user-based recommenders instead compare user similarities in the recommendation process.

According to Zhang et al. [10], collaborative filtering- (CF-) based approach is a very successful technology in all RSs. The papers [11, 12] haev reported that there are three fundamental challenges faced by CF approaches such as

(i) *Cold start* challenge rise where an item appears which has not been rated before, recommendations cannot be made for it or when a new user without any prerecorded profile appears [13, 14]

(ii) *Sparsity* challenge appears when there are numerous items but too less rating values available in the initial stage of recommendation [11, 14]

(iii) *Scalability* challenge appears when users' and items' data are very big to process [11].

Most of the recommender systems suffer from the cold start problem because users usually do not provide adequate ratings to hotels to enable collaborating filtering based recommendation, which can lead to an issue called as cold start problem. We have proposed the hotel recommender system that mines contextual information and sentiments from reviews and recommend the travelers the name of the hotels based on their preferences, by analyzing the other travelers' reviews together with the rating value, ranks, votes, and YouTube video views to improve the recommendation accuracy. Opinion-based sentiment analysis resolves this issue by considering four types of contexts: (i) guest type, which can be "business," "couple," "solo," "group," and "family"; (ii) hotel name; (iii) location; and (iv) rating about different hotels. We have used the approach where collaborative filtering technique aggregates with the sentimental analysis, to provide personalized hotel recommendation. Opinion-based sentiment analysis calculates the polarity of each sentence of review to find its score effectively solving the problem of cold start and improving the accuracy.

Similarly, one cannot imagine manually sorting through thousands of comments, customer support conversations, or customer reviews, as there are too much data of hotels available to process. Sentiment analysis allows to process data at scale in an efficient and cost-effective way improving the scalability issue. Machine learning-based sentiment analysis or classification is used to classify and provide recommendations to the users. In the classification technique, the system learns from the data input given to it and then uses this learning to classify new recommendations. In supervised machine learning techniques, two types of data sets are required: training dataset and test data set. An automatic classifier learns the classification factors of the document from the training set, and the accuracy in classification can be evaluated using the test set. The key step in the supervised machine learning technique is feature selection. The classifier and feature selection determines the classification performance. Some researchers have introduced another approach known as cluster-based approach [6, 11, 15]. Collaborative filtering based on clustering reduces the computation time and focuses only on time efficiency improvement as the clustering phase is performed offline.

The main idea here is to develop a recommender system which helps users to find hotels according to their preference and choice using previous users' reviews and ratings. Due to the availability of extensive web data, the main problem while processing thousands of items and its related information is the storage and the time efficiency. In order to deal with this problem, an intelligent and efficient item-based collaborative filtering recommendation system is proposed which uses Hadoop platform along with NoSQL database to improve the performance and efficiency while dealing with a huge number of hotel data. We have performed various experiments to achieve performance gains of Hadoop with improved response time of the recommender resolving the scalability problem [11].

Previous researches lack accuracy of true recommendations. The reason for less accuracy is the use of only quantitative data such as likes and ratings which ignored the qualitative aspect of user feedback and challenge the reliability of the previous recommender's accuracy. Here, it is important to mention that both aspects of likeness of users can provide us with true and accurate recommendations. So there is a need to consider both quantitative as well as the qualitative aspects of multitype user feedback in the recommender. The both qualitative and quantitative aspects of likeness can be achieved by using not only ratings but also text reviews, votes, and video views that were not covered by the previous approaches. Some other gaps and deviations are also tabulated in Table 2. It signifies differences in the previous approaches and our approach.

The proposed system is capable of storing a large number of hotels data efficiently and provides improved time-efficient recommendations. This paper intends to provide four main contributions as stated in the following:

(i) The proposed recommender system helps in achieving true hotel recommendations through processing and analyzing of large heterogeneous web data, i.e., both ratings (numerical) and reviews (textual) using opinion mining approach and fuzzy approach to produce relevant recommendations according to customers' type and choice

(ii) Development of a web-based recommender for hotel recommendation which integrates linked data of external resources (hotel websites) containing hotel information available online

(iii) The proposed approach optimizes performance in the proposed hotel recommendation system using NoSQL Cassandra database in Hadoop environment

(iv) The dataset is obtained from two different sources (websites) such as TripAdvisor.com and Expedia.com

We are using opinion-based approach which is used to classify the text into three sentiment expressions such as "Positive," "Negative," and "Neutral" with the help of SentiWordnet Wordnet dictionaries. There were number of challenges that arise during the processing of reviews and extraction of the features from textual reviews. Some of the challenges are as follows:

(i) Dealing with the big data which consist of the textual reviews describing opinions given by the people for the hotels

(ii) Casual informal languages, abbreviation/emoji/slang, or use of emoticons

(iii) Spelling mistakes/typing errors

(iv) Ambiguous reviews given by a customer, e.g., I have never lived in a hotel quite like this one before! Ambiguity: we cannot understand whether the hotel is the best or the worst

(v) Reviews containing hashtags

(vi) Detecting polarities of hidden sentiments of a customer in a given review

The remaining paper is further organized and structured as follows. Related work and allied concepts are described in Section 2. Collaborative filtering and recommender system are discussed to cover the previous

Table 2: Deviation of different approaches.

Source	Multitype feedback	Ratings	Reviews	Votes	Video views	Polarity scores	Tf-Idf	Fuzzy logic	Multi data sources
[4]	✓	✓	✓	×	×	×	×	×	✓
[6]	×	✓	×	×	×	×	×	×	×
[16]	×	×	✓	×	×	×	×	×	×
[17]	✓	✓	✓	×	×	✓	×	×	×
[18]	×	×	×	×	×	×	×	✓	×
[19]	×	✓	×	×	×	×	×	×	×
Proposed approach	✓	✓	✓	✓	✓	✓	✓	✓	✓

work. A General recommender model is also explained. The methodology to design a hotel recommender is explained in Section 3. It explains the proposed hotel recommender components. The detailed description of preliminary experiments with testing and training datasets along with result is presented in Section 4. System overview is also provided in the same section. A comparative analysis of the proposed approach with the previous studies is presented in Section 5. Conclusion and future work are provided in Section 6.

2. Related Work

The previous work related to the recommender system is discussed in this section. Prior research describes the related concepts of recommenders such as information filtering and recommendation algorithms previously used to develop recommender systems [20] which help to understand and realize the need of recommenders in the modern era of web technologies.

2.1. Recommender Systems. Recommender systems help end users to help discover products and services that they are looking for. Tan and He [13] have proposed a physical resonance procedure, named resonance similarity (RES), as a novel approach. This novel similarity provides superior predictive accuracy in comparison to the traditional similarity measures used for users' evaluations. Fasahte et al. [21] have discussed that unrated items can be recommended and predicted by using different filtering techniques, and they have conducted experiments using Trip Advisor dataset. Additionally, they presented hybrid approach uses rating data and textual content to predict the user behavior. Crespo et al. [22] discuss that Sem-Fit uses the customers' experience point of view in order to apply fuzzy logic methods to relating customer and hotel characteristics, represented by means of domain ontologies and affect grids. Hu et al. [23] measured the fineness of rating predictions and evaluated the performance of the Context Aware Personalized Hotel (CAPH) recommender using ratings and reviews of Trip Advisor data.

The Hwang et al. [24] have used the Trip Advisor reviews in the semantic-based Latent Dirichlet Allocation (LDA) method to perform a hotel review for the hotel management systems and to obtain a distinguishable performance to the Term Frequency-Inverse Document Frequency (TF-IDF) method. The author has used all types of features of the hotels and concluded that the word-based LDA method has high precision compared to the LDA method. Sandeep et al. [25] performed twitter community sentiment analysis to obtain real-time sentiments of the common people to represent both existing and potential customers. The proposed method using monthwise sentiment score of twitter hash tags of Indian telecom operators successfully predicted their growth rate in terms of subscriber addition. Meng et al. [26] have proposed the method called KASR (Keyword Aware Service Recommendation) which is implemented in Hadoop and cloud for the big data analysis of reviews to improve the time efficiency and the scalability in big data projects [27, 28].

The text mining techniques combined with tracking and browsing are used to develop a personalized hotel recommendation by Lin et al. [29]. A useful stochastic programming model using multiple regression analysis was designed to lower the search cost of the customers by Rianthong et al. [30] It is concluded that the review rating, prices, and utility of the hotels are needed to be taken at the upper side of the sequence. Lal and Baghel [31] explore the most relevant and crucial features for sentiment classification and group them into seven categories, named as basic features, seed word features, TF-IDF, punctuation-based features, sentence-based features, N-grams, and POS lexicons. Sharma et al. [17] have used customer's reviews and preferences from booking.com to determine the hotel rating using previous users' data in a multicriteria review-based recommendation system approach and NLP technique. Chang et al. [16] hypothesized surrounding environments hotel recommendations. CATPAC (content analysis program) was used to analyze users' reviews and ratings, and SPSS (Statistical Package for Social Sciences) was used to analyze the users' ratings with regression analysis and analysis of variance (ANOVA) to check customer loyalty.

Jannach et al. [32] worked on the recommender system which utilized the regression-based methods and item-based models for accurate recommendations. Ibrahim et al. [33] presented a personalized intelligent information model to examine hotel services. Bouras and Tsogkas [34] have used the user clustering Word Net-enabled k-means algorithm to recommend improved news articles. Chen and Chuang [18] optimized the performance of a ubiquitous hotel recommender system by using a nonlinear and fuzzy programming approach over the hotels dataset. Fasahte et al. [21] have used a Lexicon-based approach to identify sentiments towards the hotel's aspects within the defined context, and they also explained how unrated items can be used for recommendation using the item-based CF technique. Both rating data from the user's review and rating data of user's in a hybrid recommendation approach are used to analyze the user's behavior. Valcarce et al. [35] have used Cassandra as the platform of the distributed big data recommendation application and the MySQL Cluster for comparison.

User's previous data for the different products and items like hotels, books, and articles are gathered in the systems using collaborative filtering algorithms. Rankboost algorithm and cluster-based collaborative filtering are used to develop a hotel recommendation system proposed by Huming and Weili [11] to get recommendation according to users own choice. For the quantitative and qualitative analyses, the data were obtained from hoteltravel.com.

There are a number of recommender systems which are developed by researchers and developers who have used the collaborative filtering algorithms and techniques [36–42] to provide recommendation service. Collaborative and content-based filtering uses the knowledge of the users to calculate the correlation with other users and to perform certain deductions in the feature space [19, 43]. In this paper, the proposed hotel recommender system is highly efficient and somewhat bridges the gap using hotel feature extraction using natural language opinion mining analysis.

Nilashi et al. [44] have used PCA-ANFIS (Principal Component Analysis-Adaptive Neuro-Fuzzy Inference System) and EM (Expectation Maximization) to develop a recommender system and implemented them in the tourism domain based on multicriteria collaborative filtering technique. Data were obtained from TripAdvisor website to perform experiments and achieve high accuracy and high time efficiency for the recommendation of the hotel. Phorasim and Yu [45] have used k-means and collaborative filtering approach which results in more precise and less time-consuming recommendations as compared to the existing traditional one. Kögel [46] discussed that, to obtain relevant suggestions in real time, a model-driven software engineering approach is used which collects data from different sources and combines it. Do et al. [47] survey common techniques for implementing model-based approach so as to achieve high accuracy. Yibo et al. [48] have built a hybrid recommendation model for movie recommendation using sentiment analysis on spark platform which outperforms the traditional models in terms of various evaluation criteria.

3. Proposed Hotel Recommender System

The proposed system uses the heterogeneous nature of data (textual and numerical) crawled in from World Wide Web (www). Data are obtained from the selected hotel websites (data sources) containing the keywords present in the active user search query. A web crawler was used to download the requested data and store the obtained data in a NoSQL database Cassandra for further processing. The data are usually found in the form of numbers (such as votes, ranks, and number of video views) and text (such as reviews and comments). To get true recommendations, our system has used ranks, votes, and reviews data to extract hotel features from it.

The system works in two parallel ways. The numeric ranks and votes of hotels from each selected data source are normalized. On the other hand, review data are processed using natural language processing package for review mining, and features are extracted in the form of a hotel feature matrix. Further, numerical polarity scores are computed for these extracted features using SentiWordNet and average polarity score is calculated. Now weighted average polarity scores are calculated by aggregating normalized rank score, voting score, and polarity score. Finally, recommendations are computed by applying the fuzzy logic approach. We have defined a fuzzy set containing certain fuzzy rules to calculate the final score to find out the guest type (solo, family, business, friends, couple, etc.) for the hotel. The proposed hotel recommendation approach is shown in Figure 2. The final recommendations of the hotels based on a particular guest type in one of the five different classes are displayed.

3.1. Feature Extraction Process. First of all, before deriving reviewer's feature preferences, we first analyze raw textual reviews and convert them into structured form to extract opinion-based feature. Reviews from any website are usually extracted into a Json file which is then loaded into the system database. We have studied different types of methods for mining the feature-based opinions from textual reviews and found that NLTK package is most appropriate tool to extract features from hotel reviews. Natural language toolkit NLTK is a Python library to make programs that work with natural language. The library can perform different operations such as tokenizing, stemming, classification, parsing, tagging, and semantic reasoning. We have used NLTK 3.3 Version in this paper. Following steps have been performed for identifying numerous features from the hotel reviews:

(i) Extracting features from a review and grouping synonymous features
(ii) Finding and assigning value to the opinions that are associated with various features in the review
(iii) Assigning these features a value in the normalized range

The review data are converted in comma separated values to be available in easily readable form, i.e., natural language data format. The proposed system needs to perform a natural language processing to extract hotel features based on the previous guests' opinion. We have performed following four steps to process a natural language text review as follows:

(i) Lexical analysis
(ii) Syntax analysis
(iii) Semantic analysis
(iv) Feature extraction

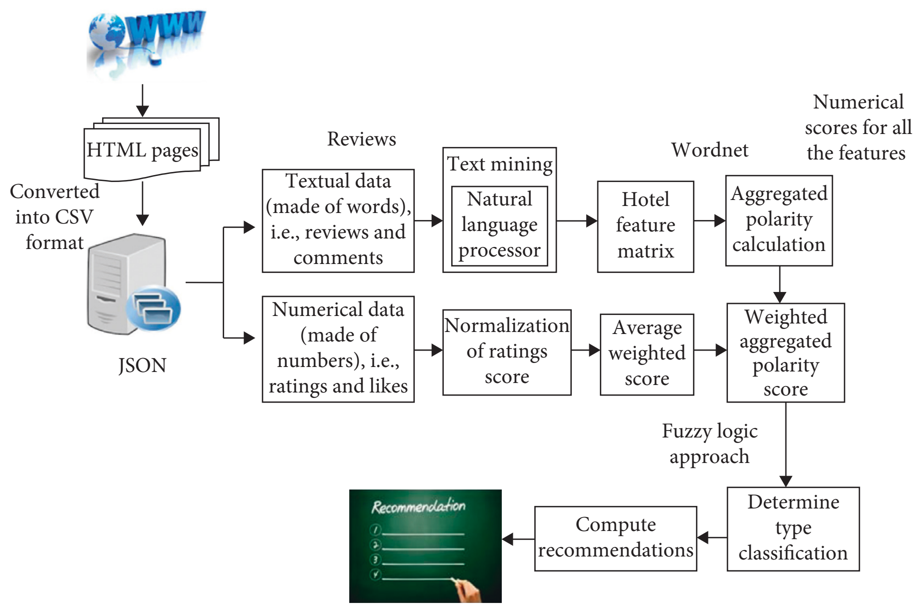

HTML pages
Converted into CSV format
JSON
Reviews
Textual data (made of words), i.e., reviews and comments
Text mining
Natural language processor
Wordnet
Hotel feature matrix
Numerical scores for all the features
Aggregated polarity calculation
Numerical data (made of numbers), i.e., ratings and likes
Normalization of ratings score
Average weighted score
Weighted aggregated polarity score
Fuzzy logic approach
Determine type classification
Compute recommendations
Recommendation

Figure 2: The proposed approach for hotel recommendation.

3.1.1. Lexical Analysis. In lexical analysis, the streams of characters are taken as input and streams of tokens are generated as an output.

(1) Tokenizing. The hotel reviews are available in the form of paragraph which contains a number of sentences or strings. These strings are tokenized into tokens or lexicons. These tokens are usually words but can also be numbers or symbols. Usually all whitespace characters are removed from the sentences, and alphabets or numbers are considered as a single token. These tokens further go through POS tagger to get different parts of speech called as morphemes. For example, a verb "feels" is stored as "feel + s" and a noun "vegetables" is stored as "vegetable + s." Afterwards, morphemes are lexically analyzed by a parse tree (Tables 3 and 4).

Table 3: Lexical analysis of the review.

Review strings	Rooms of the hotel are big. Food is delicious. Hotel location is best. There is no Internet.
POS tagging	The/DT Room/NN of/IN the/DT hotel/NN is/VBZ big/NN./. Food/NN is/VBZ delicious/JJ./. Hotel/NN location/NN is/VBZ best/JJ./. There/EX is/VBZ no/DT internet/NN

Table 4: Parse tree of a review sentence.

```
(ROOT
    (S
      (NP
          (NP (DT the) (NN Room))
          (PP (IN of)
            (NP (DT the) (NN hotel))))
      (VP (VBZ is)
          (NP (NN big)))
      (. .)))
```

3.1.2. Syntax Analysis. In this step of analysis, all the sentences and the phrases of the paragraph of reviews are authenticated in consultation with the defined grammatical rules in the English language. This paper uses Google Spell Check to correct the grammatical errors and typos in the crawled reviews. The misspelled words in review text are corrected by using a statistical spell checker (http://norvig.com/spell-correct.html) and removing the duplicates and unnecessary punctuation marks in sentences, e.g., !, ?, etc. We have used the publicly recognized POS tagger to remove some noisy information contained in the review text such as syntactical errors and mistakes. The principal parts of all sentences are also identified in this phase, i.e., object part, subject part, and verb part. Parse tree and typed dependencies are also generated in this phase. Syntactic dependency parser (http://nlp.stanford.edu/software/lex-parser.shtml.) can also return the syntactic dependency relations between the words in a sentence.

(a) Word stemming: there are many words which are derived from actual words called as derived words. Stemming is used to reduce derived words into their root forms. Lancaster Stemmer has been used in our work. Python NLTK provides WordNet Lemmatizer that uses the WordNet Database to lookup lemmas of words. The part of speech is first detected before getting the actual root word. In this process, the part of speech of a word is first determined and different normalization rules are applied for each part of speech.

(b) Extraneous word removal: reviews usually contain words which do not have significant meaning in extracting features for any product or item such as "the," "a,", "also," "about," "an," "at," "to," etc. These words are removed. There is not any globally approved library for list of stop words present in the

English language. To overcome this issue, we have developed library for such words in java of our own.

(c) Shorten exaggerated word.

(d) Words which have same letter repeating more than two times in a single word and not present in the lexicon are shortened to the meaningful word with the repeating letter occurring only once like exaggerated word “NOOOOOO” is reduced to “NO.”

(e) Part of speech tagging: the words with similar grammatical properties are classified through part of speech tagging system. Each word in the review is separated and tagged according to part of speech it belongs to. The words are tagged as Singular Nouns, Plural Nouns, Verbs, Adjectives, Adverbs, etc. The NLP package returns tags like NN stands for singular common nouns, NP stands for singular proper nouns, etc.

3.1.3. Semantic Analysis. In the semantic analysis, all the tagged words among the sentences of the review are extracted in some sort of tabular form. It is decided in this particular stage that what actions are performed by the particular subject and a number of attributes related to every object are also identified. Output of the semantic analyzer as in Table 5 contains a semantic table which is generated by the input review text on the basis of the parse tree generated in the previous phase.

If there is a noun in the sentence, then we take it as a feature and store current sentence under this feature. The extracted information in the semantic table is used to express the features of the hotel. To find the polarity value for all the features from the user reviews using opinion mining approaches, a hotel feature matrix is obtained as in Table 6.

3.2. Polarity Detection. We identify the polarity of each review in the collection reviews using the NLTK library and calculate aggregated polarity score for each feature based on each review for every hotel from selected websites. We have started conducting feature-based opinion mining of every review, where opinion indicates positive, neutral, or negative sentiment that a reviewer expressed on a feature based on opinion words, as there are multiple opinion words (great, nice, and awesome) that are related with each feature (location, room, and food) in a review. We assess every opinion word’s sentiment strength which is also called polarity value. In this analysis, the values of review features and their associated opinions in terms of polarity are derived and shown in Table 7.

3.2.1. TF-IDF Generation. After the preprocessing of textual reviews and removing all repetitive entries and unnecessary stop words, Tf-Idf (term frequency -inverse document frequency) needs to be generated for each review. We compute weights of each item in the review using term frequency-inverse document frequency (TF-IDF) technique to determine which terms might be the most representative and

Table 5: Semantic analysis.

No.	Tagged words	Value
1	Room	Big
2	Food	Delicious
3	Location	Best
4	Internet	Not available

Table 6: Hotel x feature matrix after NL processing.

No.	Hotel-ID	Review-ID	Location	Price	Room	Food	Staff
1	Hotel-1	R-1	2	2	2	1	2
2	Hotel-1	R-2	1	1	1	2	1
3	Hotel-1	R-3	0	3	2	1	3
4	Hotel-2	R-1	2	0	2	1	2
5	Hotel-2	R-2	1	1	3	2	1
6	Hotel-2	R-3	2	2	2	1	2
7	Hotel-3	R-1	1	1	2	2	2
8	Hotel-3	R-2	0	1	1	2	1

Table 7: Polarity matrix.

No.	Review-ID	Worst	Great	Shame	Awesome	Nice	Label
1	R-1	2	1	2	1	1	Negative
2	R-2	1	1	1	2	1	Positive
3	R-3	0	3	2	1	3	Positive
4	R-1	2	0	2	1	2	Negative
5	R-2	1	1	3	2	1	Neutral

occur frequently in the collection of documents as well as which words are less representative and rarely occurring. TF-IDF is computed for each term word occurring in the collection of reviews. TF(t) is defined as follows:

$$\mathrm{TF}(t) = \frac{\text{number of times term } t \text{ appears in a document}}{\text{total number of terms in the document}}. \quad (1)$$

While IDF for a term (t) is given as follows:

$$\mathrm{IDF}(t) = \log \frac{\text{total number of documents}}{\text{number of documents consisting term } t}. \quad (2)$$

Now, we have to weigh down the frequent terms and find out the rare ones, by computing TF-IDF weight. The TF-IDF weight is the product of TF(t) and IDF(t):

$$\mathrm{TF} - \mathrm{IDF\ weight} = \mathrm{TF}(t) * \mathrm{IDF}(t). \quad (3)$$

SentiWordNet is a dictionary that tells, rather than the meaning, the sentiment polarity of a review. For detecting the polarity and subjectivity of different hotel reviews and to get the polarity and subjectivity, we have used SentiWordNet, a publicly available analyzer of the English language that contains opinions extracted from a WordNet database. We separate our collection of reviews to extract words (hotel features) and assigned all representative occurring under the appropriate hotel features as explained in previous steps, find positive (pos), negative (neg), and neutral (neu) terms to calculate the sentiment score. SentiWordNet is included with Python's NLTK package and provides WordNet synsets with sentiment polarity. WordNet gives different types of semantic associations between words, which are used to calculate sentiment polarities. In simple words, sentiment analysis is the process of quantifying something which is qualitative in nature such as textual reviews. The sentiment score of a term (pos or neg) is multiplied by TF-IDF weight to calculate overall sentiment score (polarity) of terms in the document and is given as follows:

$$\text{overall sentiment (polarity)} = \text{sentimentscore} * \text{TF} - \text{IDF weight}. \tag{4}$$

The overall sentiment polarity score (negative or positive) explains how many features are positively or negatively important in the hotel review. As there are multiple opinion words that are related with each feature in a review, a weighted average value is calculated which acts as the weight to represent the overall positive or negative polarization of the review. If the polarity score of a feature in the reviews of a hotel is greater than zero, then the feature is the positively polarized; if it is less than zero, then it is negatively polarized; and if it is equal to zero, then it represents the neutrality. We have calculated the polarities of all the reviews of the hotels taken from different data sources.

The polarity of the reviews Pr of a hotel from the selected website can be calculated by taking the difference of the aggregated polarity score of positive reviews posr and the aggregated polarity score of negative reviews negr of that particular hotel h_n from the particular selected website w_o:

$$\text{Polarity}_{rm}(P) = \text{sgn}\left[\left|\sum_{m=1}^{n}(\text{posr}_m)\right| - \left|\sum_{m=1}^{n}(\text{negr}_m)\right|\right], \tag{5}$$

where $\text{posr}_m \wedge \text{negr}_m \in h_n$ and $h_n \in w_o$.

Then, we will take aggregated polarity score of textual reviews of each hotel from each selected website. Aggregation is the process of combining things. That is, putting those things together so that we can refer to them collectively:

$$\text{aggregated polarity}_{hn}(A) = \sum_{m=1}^{n} P_{rm}. \tag{6}$$

The weighted average of the aggregated polarity by total reviews of the respective hotel from the selected website and the weight score of ranks and votes will be calculated as follows:

$$\text{weigted average polarity}_{hn}(B) = \frac{A_{hn}}{T} + (\text{aggregated ranks}_{wo}) + (\text{aggregated votes}_{wo}). \tag{7}$$

Here, T is the total number of reviews of hotel h_n from hotel website w_o:

$$\text{aggregated weighted average polarity}_{hn}(C) = \sum_{O=1}^{N}(B_{wo}) + (\text{Likes}_{hn}),$$
$$\text{average aggregated weighted average polarity}_{hn}(D) = \frac{C_{hn}}{N}. \tag{8}$$

where N is the total number of selected hotel websites containing large number of hotel reviews.

$$\text{Final score}_{hn}(F) = \frac{r_{hn} - \min(r_{\text{hn}})}{\max(r_{hn}) - \min(r_{hn})} * 10. \tag{9}$$

where F is the final value of the normalized average aggregated score r_{hn} of the hotel (h_n) from number N of selected websites.

3.3. *Type Classification and Recommendation.*

The classification is done by calculating the final score. The reviews words are matched with the dictionary words, and if it is a positive word, then score will be +1; if negative word, then score will be −1, otherwise 0. The final recommendation is achieved by using the fuzzy logic approach. The fuzzy sets theory provides a framework for the representation of the uncertainty of many aspects of human knowledge. For a given element, fuzzy set theory presents the degrees of membership to a set. For example, if we have the set of solo guests, then we can consider that a person who likes to do gym or take massage in hotel belongs to such a set with a degree of 1 or a person who likes to have a ghazal night or cinema in the hotel must be a couple guest and belongs to a set in some other degree. The purpose of our recommender is to provide hotel recommendations based on some expert criteria using the fuzzy set. The first step of the recommendation process consists of representation of knowledge about how the hotels are selected. This knowledge is expressed using fuzzy sets.

In the first step, the fuzzy sets are defined based on the expert knowledge. The expert explains the characteristics of the hotels and the characteristics of the customers in terms of fuzzy sets. The second step consists of providing recommendations using a previously built hotel-feature-rating matrix which is usable by a recommender system based on a collaborative filtering technique.

3.3.1. *Fuzzy Set.*

To represent the degree of membership of a certain hotel to a certain class, fuzzy set theory is used. The final recommendation is achieved by using the fuzzy logic approach based on the fuzzy rules by calculating the final score to provide the class of the hotel based on guest type (solo, couple, etc.) as shown in Figure 3. Fuzzy rules are defined as follows:

Rule 1: if $F > 8$ then Hotel Type is "R (Recommended)"

Rule 2: else if $F > 6$ and $F \leq 8$ then Hotel Type is "BR (Best Recommended)"

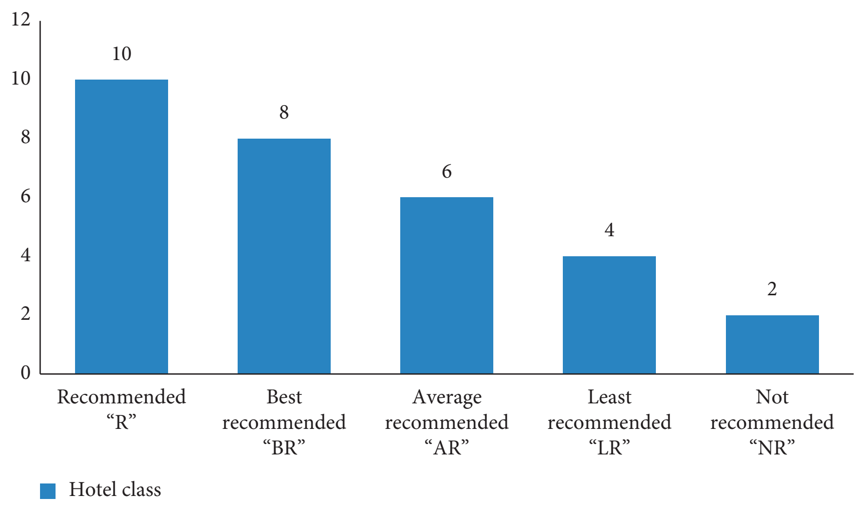


FIGURE 3: Hotel recommendation class.

Rule 3: else if $F > 4$ and $F \leq 6$ then Hotel Type is "AR (Average Recommended)"

Rule 4: else if $F > 2$ and $F \leq 4$ then Hotel Type is "LR (Least Recommended)"

Rule 5: else Hotel Type is "NR (Not recommended)"

4. Implementation Setup

4.1. NoSQL Storage. As our recommender is designed to deal with heterogeneous types of data, a database is required which can store this diverse nature of data. We have Cassandra database to store data used in the proposed recommender. Review pages which match the Query keywords are downloaded and are stored in the NoSQL database in Hadoop. The dataset used in this study is crawled in from the external resources such as hotel website of Trip Advisor and Expedia. The data of hotels are saved in comma separated value format (CSV). So, it is converted into the JSON format to increase its readability. The users' textual reviews and ratings assigned by existing users recorded as rating score, likes, or star ranks are stored in Cassandra. The ranks score can vary between the different scales of 1 to 5 or 1 to 10. Normalized ranks are calculated in this paper. The Cassandra in the designed approach will decrease the execution time represented in milliseconds (ms). The dataset contains hotel reviews and users feedbacks in the form of votes, ranks, and YouTube video views. The data are collected in the steps shown in Figure 4.

In our proposed application, the processing of the designed web recommender application is explained in the following steps:

(i) Start the process

(ii) The active user queries the system by inputting as per the search criteria and guest types such as solo, couple, and business

(iii) Then, the system checks for the previous users' data (ratings, ranks, and reviews) matching the query from the web in the system database

(iv) The system filters the query data by matching query in the external web sources available

(v) If match with the query, then collect metadata

(vi) Save the metadata in the NoSQL database

(vii) If it does not match, then discard it

(viii) Repeat until all matched metadata are found

(ix) End

4.2. System Overview. The proposed recommender system application is made up of three main components. The first is the external resources, then the front end, and the other is backend as shown in Figure 5. The dataset contains hotel reviews and ranks which are taken from the external hotel websites of Trip Advisor and Expedia. Reviews are divided into training and testing data sets to verify the improved performance of the proposed methodology using Hadoop plat form and Cassandra database. The complete data are stored into the proposed system database using web crawler written in java in the developed methodology.

In order to get best recommendations related to the users' choice and desire, our hotel recommendation system is developed with certain methods and techniques and also uses some open source tools such as Hadoop platform and Cassandra. The application is accessible online from any platform, and it uses development environment based on reliable open source tools. The computing environment is also discussed in Table 8. In the proposed application, the user can query providing search criteria to get their desire recommendations. The system provide recommendations by using the review polarity scores and ratings calculated using the reviews users data stored in Cassandra which matched the active user query.

4.3. Computing Resources. The resources required to test our proposed system also include some reliable open source tools, for example, Cassandra, Hadoop, amd PHP. The system specifications are given in Table 8.

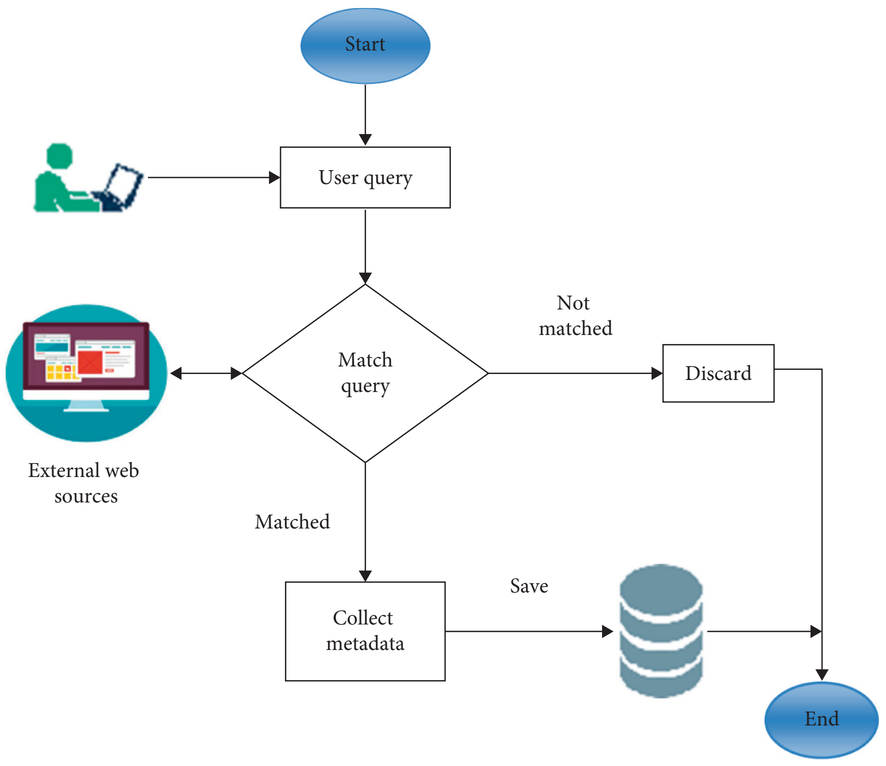


FIGURE 4: Data collection.

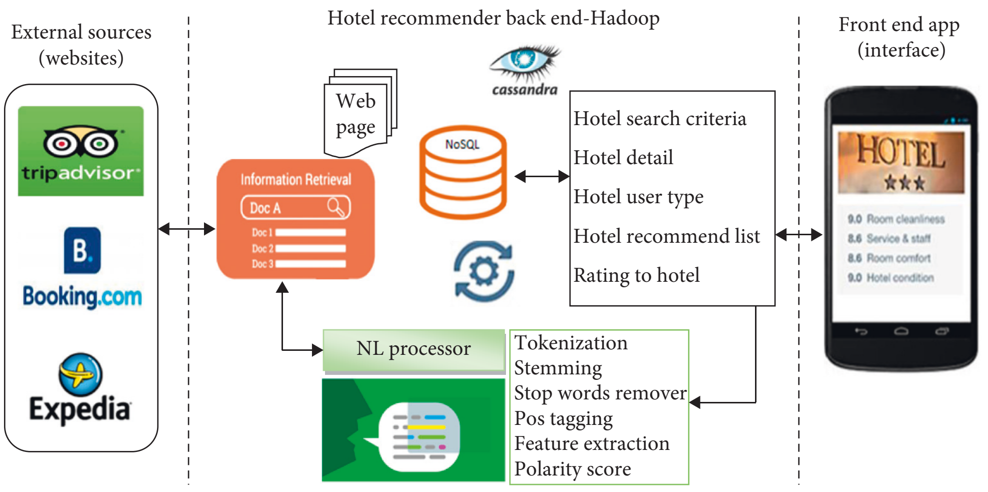


FIGURE 5: Proposed hotel recommender architecture.

TABLE 8: System specifications.

The system specifications are 4- Core. i7-3770, 3.40 GHz, 16 GB RAM, 500 GB disk
It is implemented with CentOS 7, Hadoop 2, Apache Cassandra 3.11, java 1.8, and PHP 7.1, PHP-Cassandra extension

4.4. Web Service and Methods. The designed hotel recommendation application will be accessible through a web page. HTTP connections are used to perform a number of web services using appropriate determinant URIs. The linked data are gathered by the application when a GET request is submitted by a URI, with the specified method providing user authentication and the required query parameters (QPs). The corresponding response message HTTP is formatted in JSON keeping the data uniformity and returned to App. When a user requires any data from the web page, certain specified URIs and HTTP connections are used to connect a user. The user made a request, and it is submitted through the appropriate HTTP method by using the necessary parameters required by that particular method, and then after processing, the requested data are displayed on the front end page of the web application. All the methods used and their parameters in URIs are shown in Table 9.

The hotel data are converted into JSON format to provide information about the hotel to produce recommendation out of large number of hotels available. During the data transmission, the web services and methods are protected by SSL over the recommender web application.

4.5. Experiment and Results. We have used two reliable data repositories (Trip Advisor and Expedia) containing significant number of ranks, ratings, and reviews to represent heterogeneity of data, i.e., textual reviews and numerical ratings and ranks. These data scores contain data for 8000 of the most popular hotels and collection of hotel reviews and ratings which is useful in our experiments. After data preprocessing, TF-IDF generation, and polarity detection using SentiWordNet, we have computed the polarity scores for textual reviews obtained from each selected website. We have illustrated the data obtained from the selected hotels and the corresponding data source and its processing in the proposed methodology in Table 10.

In Table 8, the selected external data source used in our work such as Trip Advisor is represented by "D1" and Expedia is represented as "D2." Similarly, the corresponding selected hotels are represented as Mandarin Oriental, New York "H1," Amsterdam Court Hotel "H2," Hotel Metro "H3," Millennium Hilton "H4," and Belnord Hotel "H5." We have calculated polarity scores of the textual reviews for each hotel from the selected websites. Normalized rank score (scale 1–5) and the voting score are also calculated. Voting score represents the number of votes given by the customer to each hotel. Hotel names and data sources with their corresponding IDs are shown in the table.

The normalized rank scores of a selected hotel from two selected data sources such as Trip Advisor "D1" and Expedia "D2" are plotted in Figure 6, and it has been noted that the Expedia has high ranks comparatively in comparison with Trip Advisor.

Polarity scores of textual reviews of hotels taken from different selected data sources are calculated and aggregated in Table 11. Weighted average polarity is achieved by adding normalized ranks and votes in the average polarity (calculated by taking average of aggregated polarities).

The power of social media websites such as twitter, YouTube, and Facebook is also creating a shift in the way travelers seek out suggestions and tips before making any booking decision for certain hotel. Videos contain hotels pictures as well as present hotel services which may also affect the behavior of customers before selection of hotel and also have impact on the hotel rating. That is why we have used YouTube video views in our work to present the heterogeneous approach. We added the number of views in weighted average polarity to calculate the aggregated weighted average polarity for quality recommendations. The final scores along with hotel classes are shown in Table 12.

These heterogeneous data sources such as ranks, votes, textual reviews, and views are computed using the proposed approach, and final rank scores are obtained as shown in Figure 7. The hotels are classified based on the final ranking score. The same is explained in Table 12, and hotel class is identified based on the fuzzy set used in our work.

The final rank score "F" of H1 hotel "SpringHill Suites Denver Downtown" is greater than 8 that is why it is placed in class "R." The H3 hotel "Mandarin Oriental New York" lies in "BR"class because "F" score is greater than 6 and is less than 8. Whereas, the H2 hotel "Amsterdam Court Hotel," H4 hotel "Millennium Hilton," and H5 hotel "Belnord Hotel" scores are greater than 2 and less than 4 so it lies in class "LR."

Figure 8 represents the class of each hotel taken from selected data sources against the final rank score computed based on the guest types such as solo, family, and couple.

Figure 7 represents the each recommended criteria score against the selected data source and the proposed system using heterogeneous data.

This designed web application provides the customers the opportunity to obtain their desire hotel out of a large number of hotels available. The recommender system searches on the basis of system defined criteria depending upon the guest type. The list of recommended hotels generated by the proposed system is displayed in Figure 9.

5. System Performance Evaluation

5.1. Evaluation Metrics. Some evaluation metrics are used to evaluate the accuracy of the proposed system. These evaluation metrics convey that the results obtained by the proposed system are accepted by the targeted users. Mostly performance evaluation uncovers what needs to be improved before the product goes to market. Without performance analysis, software application is likely to suffer from issues such as running slow while several users use it simultaneously or if the system does not respond quickly. So we have performed performance analysis to demonstrate that the proposed system produces effective recommendations; there are number of metrics but we have used three metrics which are used in this study.

Precision is the first measure used to evaluate our system represented by the following equation:

$$\text{precision} = \frac{A}{C + A} \times 100\%. \tag{10}$$

According to the user feedbacks in the system, we assume that A represents the total number of recommended hotels liked by the user and C represents the number of hotels which are not liked by the user. "$C + A$" will be the total number of hotels recommended by the system to the same user. The precision can be calculated by taking the ratio of the number of the recommended hotels which are liked by the user over the total number of hotel recommended by the system. The Recall rate is defined as follows as per the same assumptions made above:

Table 9: Methods used in the web service.

Method	Description	QPs	HTTP method
Search	Searches by the given criteria to get a list of available hotels data	*Searchname*	GET
Topratings	Displays the specified hotels along with their ratings	*Ratings. Hotel id*	GET
Hoteldetail	Provides hotel details with name ID and region	*hotelId*	GET
Recommendhotel	Provides a list of recommended hotels	*Id/name*	GET
Getratings	Requires a list of hotels previously visited by users and replies by the ratings data	*numRating*	GET

Table 10: Polarity scores, rank scores, and voting scores.

Selected hotel	Data source					
	Trip advisor (D1)			Expedia (D2)		
	Polarity score	Normalized rank score	Voting score	Polarity score	Normalized rank score	Voting score
SpringHill Suites, Denver Downtown (H1)	31	3.9	209301	23	5	268231
Amsterdam Court Hotel (H2)	−5	3.4	38821	7	4	63420
Mandarin Oriental, New York (H3)	17	3.2	111620	24	5	127023
Millennium Hilton (H4)	−4	2.8	17023	−3	3.5	35622
Belnord Hotel (H5)	16	3.5	29441	22	5	41323

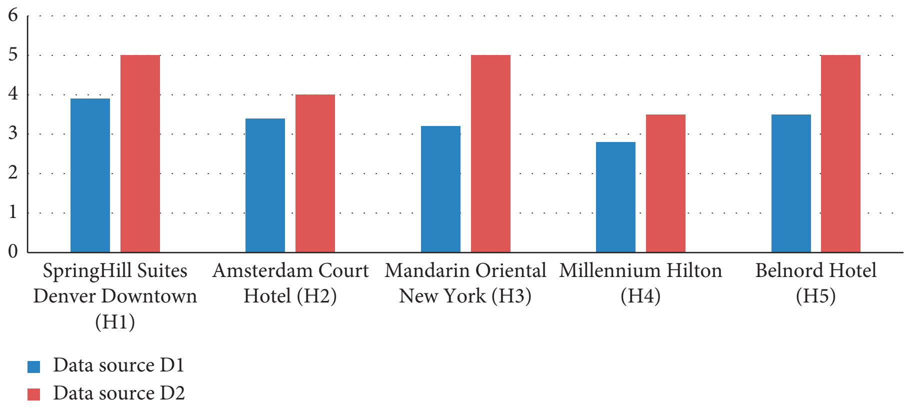


Figure 6: Difference in ranks scores.

Table 11: Weighted average polarity computation.

Hotel	Reviews		Aggregated polarity		Average polarity		Weighted average polarity	
	D1	D2	D1	D2	D1	D2	D1	D2
H1	1309	1519	31	23	0.023	0.015	209304.92	268236.01
H2	396	456	−5	7	0.012	0.015	38824.41	63424.01
H3	1189	998	17	24	0.014	0.024	111623.21	127028.02
H4	537	337	−4	−3	0.007	0.008	17025.80	35625.50
H5	971	1117	16	22	0.016	0.019	29460.50	41328.01

Table 12: Computation of final score.

Hotel ID	YouTube views	Aggregated weighted average polarity	Final score	Hotel class
H1	331025	569795.465	8.93234	R
H2	89023	140147.21	3.92215	LR
H3	284230	403555.615	7.63217	BR
H4	56056	82381.65	2.08245	LR
H5	78124	113518.255	3.12871	LR

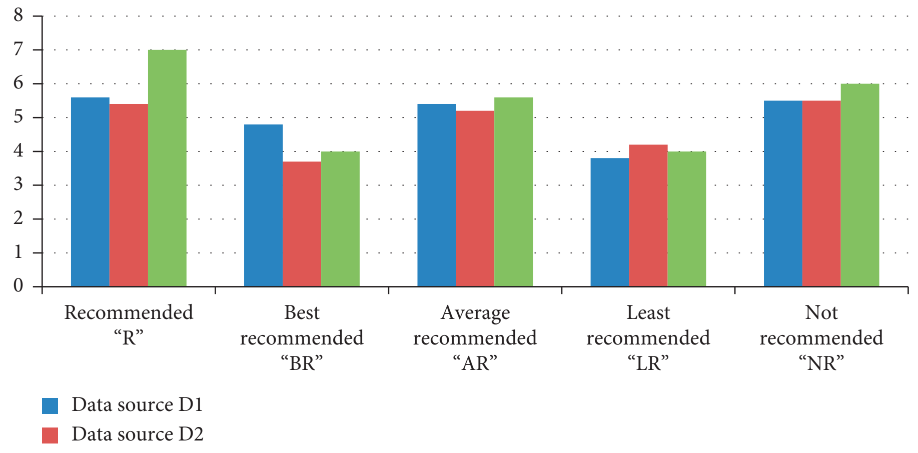


Figure 7: Comparison of hotel scores.

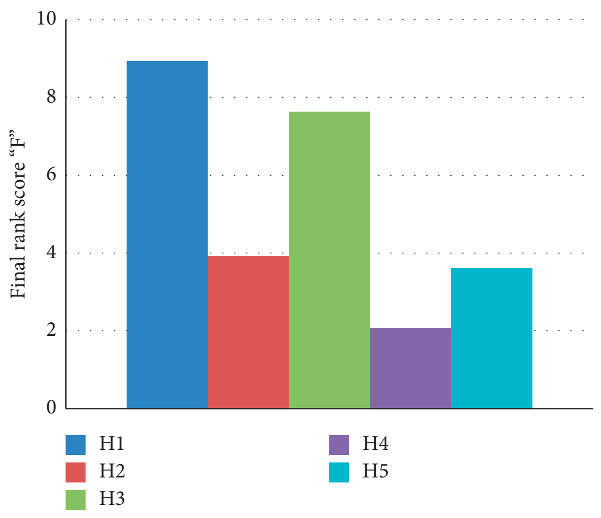
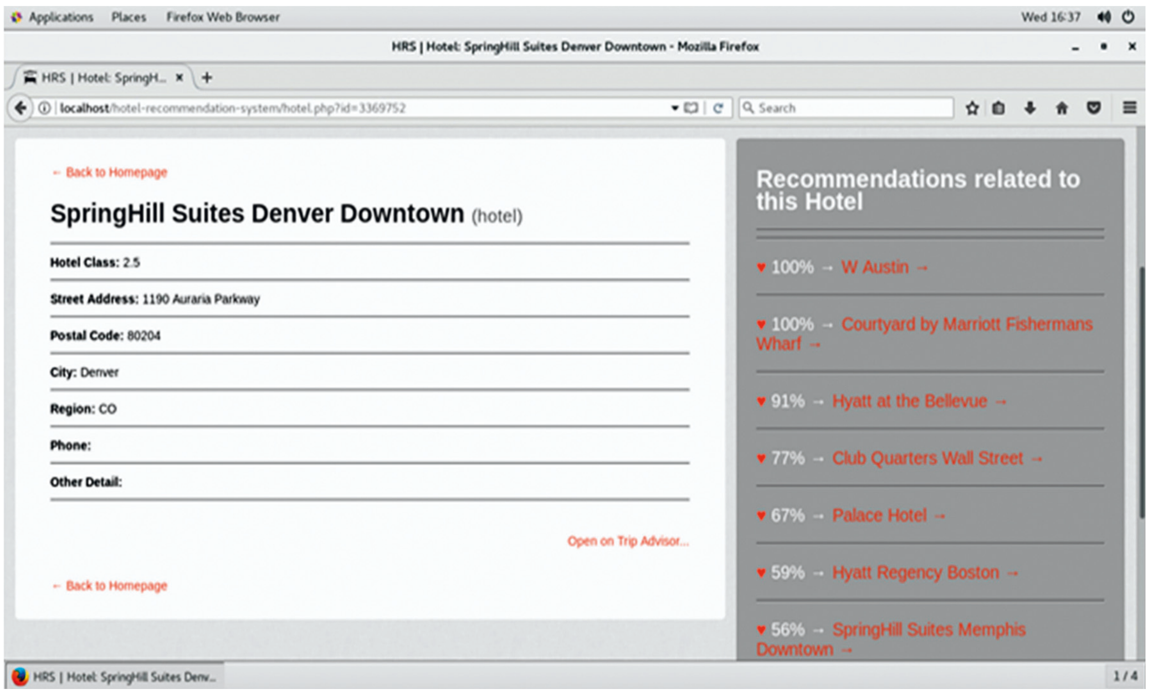


Figure 8: Proposed hotel recommender final ranking.


Applications Places Firefox Web Browser Wed 16:37

HRS | Hotel: SpringHill Suites Denver Downtown - Mozilla Firefox

HRS | Hotel: SpringH...

localhost/hotel-recommendation-system/hotel.php?id=3369752

← Back to Homepage

SpringHill Suites Denver Downtown (hotel)

Hotel Class: 2.5

Street Address: 1190 Auraria Parkway

Postal Code: 80204

City: Denver

Region: CO

Phone:

Other Detail:

Open on Trip Advisor...

← Back to Homepage

Recommendations related to this Hotel

♥ 100% → W Austin →

♥ 100% → Courtyard by Marriott Fishermans Wharf →

♥ 91% → Hyatt at the Bellevue →

♥ 77% → Club Quarters Wall Street →

♥ 67% → Palace Hotel →

♥ 59% → Hyatt Regency Boston →

♥ 56% → SpringHill Suites Memphis Downtown →

HRS | Hotel: SpringHill Suites Denv... 1 / 4

Figure 9: List of recommendations.

$$\text{Recall} = \frac{A}{B + A} \times 100\%. \quad (11)$$

where "B" represents the number of hotels targeted but are not recommended to the user and $B + A$ includes all the hotels which the user may possibly like. Recall and Precision ratios are opposite to each other. Perfect Recall rate of "1" means comparatively low Precision rate. High Precision but lower Recall gives extremely accurate recommendation. The third accuracy measure used is F-measure represented by the following equation:

$$\text{F} - \text{measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (12)$$

The F-measure is the Harmonic Mean of Precision and Recall and is used to evaluate the accuracy and efficiency of the results generated by the proposed recommender system.

We have used a subset of actual dataset which contains 100 hotels with 500 users from each of the selected websites as an exemplary dataset to test and train our proposed system. The metadata are divided into testing data (40%) and training data (60%). Depending on these data parts, we initially performed the calculations on the underlying 40% of the data of the exemplary metadata in the form of chunks of 20% of data for testing of the developed system in Table 13. We incrementally included the metadata of the remaining of the 60% dataset into the system for training the system. The accuracy metrics Precision, Recall, and F-measure with each increment are measured for exemplary data and recorded.

Figure 10 shows the F-measure values of recommendations under the proposed mechanisms with exemplary small data. The proposed recommender system supports computations to evaluate how it would influence the recommendation accuracy. The system with NoSQL dataset and proposed machine learning approach using sentiment analysis provides accurate recommendations, and its F-measure ratio value is 0.950 as the initial exemplary dataset used is very small containing 100 hotels with 500 users so such a huge improvement in terms of Precision, Recall, and F-measure is obtained. As the percentage of data increases, the values of performance measures decrease.

After testing the system with initial exemplary metadata, now the actual complete number of hotels in the used dataset obtained from Trip Advisor and Expedia contains 8000 hotels and 10 million users' data. As the actual complete dataset is large, the accuracy metrics values are comparatively decreased but still show promising results as recorded in Table 14. It shows that the proposed recommender using sentiment analysis approach provides accurate and quality recommendations to users. Graphical representation is also shown in Figure 11.

5.2. System Processing Time. Performance analysis is performed to ensure that software application will perform well under their expected workload. Most performance problems revolve around speed, response time, load time and poor scalability. Response time is often one of the most important attributes of an application. A slow running application will lose potential users. Performance testing is done to make sure an app runs fast enough to keep a user's attention and interest.

We have performed performance testing by varying patterns of workload (number of users). We have performed different experiments with different number of users. For the better understanding, we have taken data of 12 users from different professions and different age level to use and perform experiments over the proposed system. They have also used other recommenders, trivago.com, hotels.com and yatra.com to get a recommended list of hotels of their choice. Their satisfaction level is evaluated by taking their opinions about searches over each of these recommenders. Processing time over the developed system is also recorded during these searches. Participants have performed provide feedback which is recorded in Table 15. We have classified the satisfactory level into the 3 classis i.e. less satisfactory (L), satisfactory (S), and highly satisfactory (H). Feedback is taken against three capabilities i.e. time efficient (TE), relevance of recommended hotels with the user choice (RoR), cost effectiveness (CE).

Results presented in Table 15 shows satisfactory level of the participants in which most of users has selected best category to the proposed recommender system.

When the user entered the query in the proposed recommender to obtain best hotel recommendation providing a guest type according to their choice, the system performs computation using proposed machine learning sentiment analysis to collect the required recommendations. The proposed system has shown promising results in terms of improved response time comparing traditional recommenders. The system also calculates the time stamps as load time, search time, and execution time represented as milliseconds (ms). The outcomes generated in terms of response time are recorded in Table 16 and shown in Figure 12. The sum of load time and search time together is called as execution time. The proposed system outperforms and takes very less time giving the list of recommendations.

The total average query processing time taken by the proposed system with maximum workload is 2.6592 ms. Our approach along with Cassandra NoSQL database is efficient and helps to reduce the total processing time. Results indicate that the system is performing efficiently which also changes people opinions about using recommenders.

The system contributes efficiently to help users while searching hotels they like. The system is designed to achieve true recommendations and to impact customer behavior positively in terms of accuracy of recommendations with the need of the user. If true recommendations are provided, it will definitely increase the customer's satisfaction level helping them making their business decisions.

6. Comparative Analysis

The comparison in terms of time and evaluation metrics is performed in this section. It compares the proposed approach with the previous traditional approaches. A statistical analysis of the performance evaluation metrics such as the F-measure, Precision, and Recall is performed. To assess the

Table 13: Evaluation metrics with exemplary dataset.

Incremental data update using Euclidean similarity (%)	F-measure	Recall	Precision
20	0.966	0.954	0.978
40	0.956	0.941	0.972
60	0.951	0.936	0.968
80	0.938	0.921	0.956
100	0.924	0.898	0.951
Avg. ratio	0.950	0.930	0.965

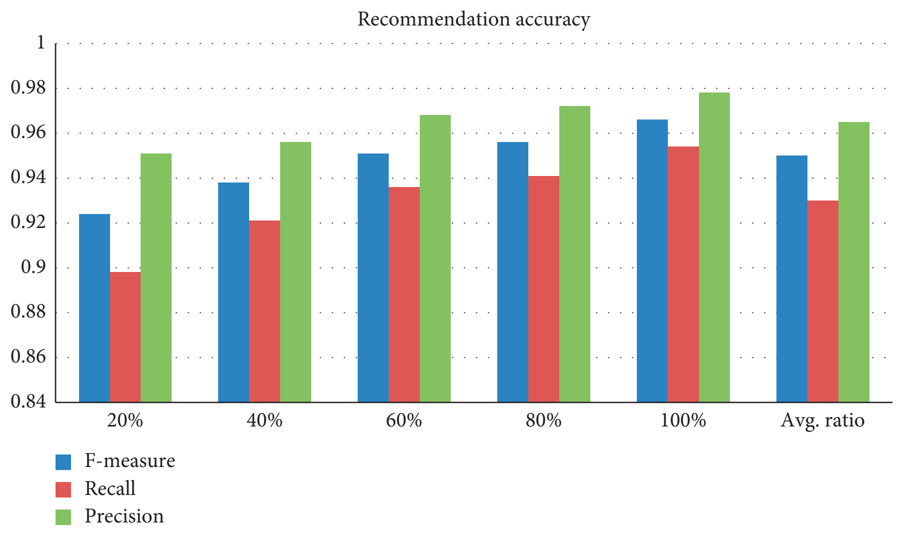


Figure 10: Performance metrics with exemplary dataset.

Table 14: Evaluation metrics with complete dataset.

Incremental data update using Euclidean similarity (%)	F-measure	Recall	Precision
20	0.825	0.791	0.877
40	0.789	0.728	0.861
60	0.771	0.726	0.821
80	0.729	0.679	0.788
100	0.688	0.631	0.756
Avg. ratio	0.76	0.711	0.821

recommendation accuracy of the proposed method, we have performed the comparison of the performance analysis metrics such as Recall, Precision, and F-measure of our hotel recommended system with a subset of actual dataset, i.e., with small exemplary dataset (100 hotels and 500 users), with complete dataset (8000 hotels 10 million users), and with the Precision, Recall, and F-measure of tradition-related recommenders and are provided in Table 17.

We have also compared the existing studies with our approach and found out promising improvement in terms of execution time of the proposed approach. The comparative analysis of the performance of the proposed hotel recommender approach with the existing related approaches found in the literature is shown in Table 18 which exhibits outcomes in terms of time improvement.

The above comparison of evaluation metric and recommendation time exhibits that the proposed approach has shown promising results. The recommendation time is reduced using the proposed approach when it is compared with the conventional approaches.

7. Conclusion and Future Work

In this paper, a novel CF recommendation approach is proposed which has the ability to handle heterogeneous data such as textual reviews, ranks, votes, and video views in a big data Hadoop environment with Cassandra database to guarantee the improved response time to generate recommendations. In the proposed system, opinion-based sentiment analysis is used to extract a hotel feature matrix and stored in a database. Our approach combines lexical analysis, syntax analysis, and semantic analysis to understand the sentiment towards hotel features. The NLTK library is used to identify the polarity of the textual reviews. The system makes use of fuzzy rules to determine the hotel class depending upon the guest type. Euclidean distance is used to

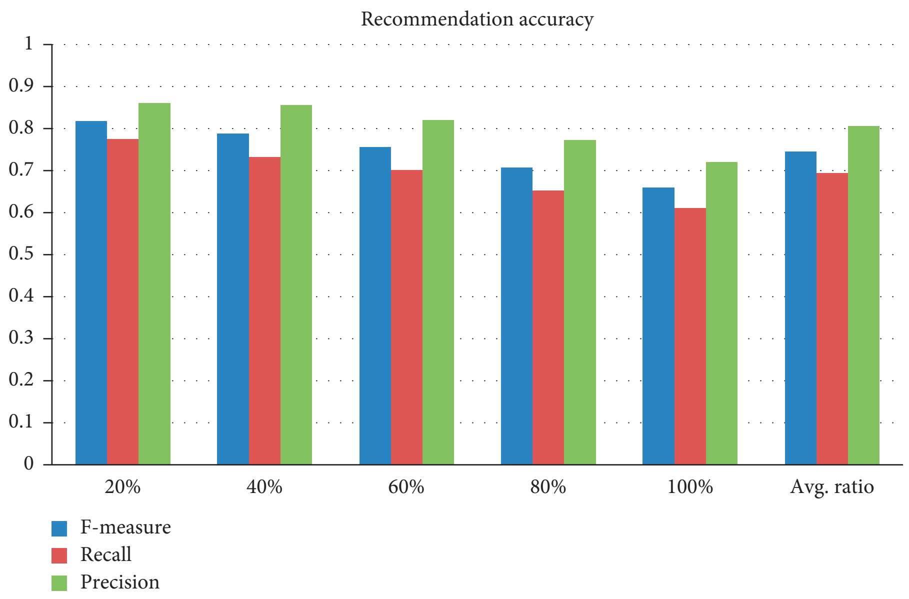


FIGURE 11: Performance metrics with complete dataset.

TABLE 15: An illustration of Participants Opinions.

No.	Profession	Age	kayak			Hotels.com			Booking.com			Proposed recommender		
			TE	RoR	CE	TE	RoR	CE	TE	RoR	CE	TE	RoR	CE
User1	Student	19	L	L	L	L	L	L	S	L	L	S	H	H
User2	Student	18	L	L	H	L	S	S	L	H	S	S	H	L
User3	Business	35	L	L	L	S	L	L	L	L	L	H	S	H
User4	Doctor	31	L	L	H	L	L	S	S	L	S	H	S	H
User5	Teacher	21	S	L	S	L	L	L	L	S	L	H	H	H
User6	Student	48	L	L	L	L	L	H	S	L	S	L	H	S
User7	Student	22	H	S	S	L	S	L	L	S	S	H	H	H
User8	Employee	41	L	S	L	L	H	L	L	L	H	S	H	H
User9	Student	29	L	S	L	S	S	L	S	L	S	H	S	S
User10	Doctor	39	S	L	S	L	L	L	L	S	L	H	H	L
User11	Teacher	46	L	L	S	L	L	S	S	S	S	H	H	S
User12	Student	28	S	L	S	L	L	L	L	S	L	S	H	H

TABLE 16: System response time.

No.	Loading time	Searching time	Execution time (ms)
1	3.0994	0.0461	3.5606
2	1.6212	0.0459	1.6712
3	1.0013	0.0488	1.0501
4	2.8610	0.0738	2.9348
5	3.0994	0.0455	3.1441
6	3.0994	0.0455	3.5544
7	2.8610	0.0727	2.9337
8	2.8610	0.0469	2.9079
9	3.0994	0.0024	3.1018
10	2.8610	0.0500	2.9069
11	1.9073	0.4583	1.9531
12	3.0994	0.0447	3.5544

calculate the similarities between the items and provide accurate recommendations based on the type of guest (solo, family, couple, etc.). The system takes 2.65 milliseconds to generate high-quality recommendations by reducing the system execution time. The resultant F-measure has resulted in 0.950 approximately 95% when we have run the system with exemplary data for training the system but when system is trained with the complete dataset obtained for websites, the

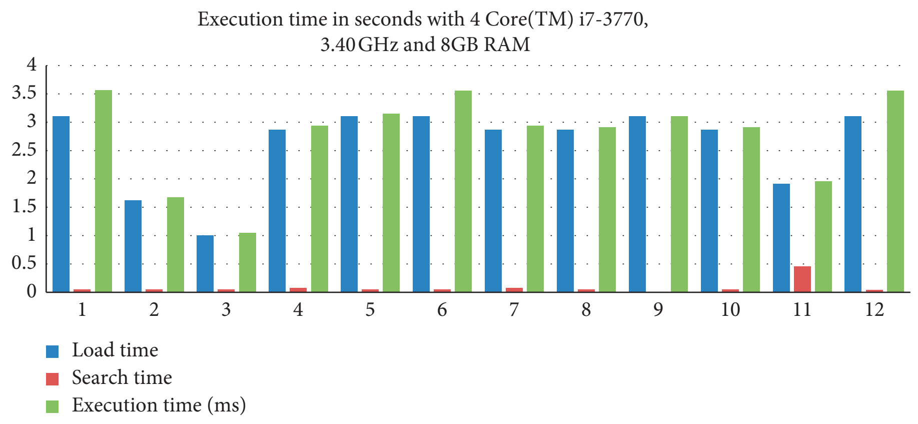


Figure 12: System response time.

Table 17: Comparison of accuracy metrics with related studies.

No.	Reference		Precision	Recall	F-measure
1	Liu et al. [2]		0.56	0.60	0.59
2	Hsieh et al. [6]		0.02	0.53	0.01
3	Zhang et al. [10]		0.25	0.34	0.35
4	Chang et al. [16]		0.43	0.29	—
5	Lin et al. [29]		0.62	0.51	—
6	Verma and Virk [19]		0.69	0.67	0.68
7	Proposed approach	With exemplary subset of dataset	0.96	0.93	0.95
		With complete dataset	0.80	0.69	0.74

Table 18: Comparison of execution time with related studies.

No.	Reference	Recommendation time
1	Bouras and Tsogkas [34]	12 sec
2	Jazayeriy et al. [37]	28 sec
3	Liu et al. [2]	27 sec
4	Proposed recommender	2.6 millisec

F- measure is somehow decreased. As the actual complete dataset is large, the accuracy metrics values are comparatively decreased to 0.745 approximately 74% but still showing promising results as recorded in Table 14. It shows that the proposed recommender using sentiment analysis approach provides accurate and quality recommendations to users.

In future, the recommender system is needed to be designed in a way that will utilize dynamic auto updated data containing the visual views, votes, and reviews online from the external websites to provide recommendations according to dynamic data found at the same time of active user query. To amplify the versatility of recommender services, this will be implemented by incorporating web cookies and using customer's navigational activities and getting feedbacks over the new recommended items.

Data Availability

The authors will provide the data used for the experiments, if requested.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. M. Mariani, D. Buhalis, C. Longhi, and O. Vitouladiti, "Managing change in tourism destinations: key issues and current trends," *Journal of Destination Marketing and Management*, vol. 2, no. 4, pp. 269–272, 2014.

[2] H. Liu, J. He, T. Wang, W. Song, and X. Du, "Combining user preferences and user opinions for accurate recommendation," *Electronic Commerce Research and Applications*, vol. 12, no. 1, pp. 14–23, 2013.

[3] M. Ibrahim and I. Bajwa, "Design and application of a multivariant expert system using Apache Hadoop framework," *Sustainability*, vol. 10, no. 11, p. 4280, 2018.

[4] J. J. Zhang and Z. Mao, "Image of all hotel scales on travel blogs: its impact on customer loyalty," *Journal of Hospitality Marketing and Management*, vol. 21, no. 2, pp. 113–131, 2012.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, 2009.

[6] M.-Y. Hsieh, W.-K. Chou, and K.-C. Li, "Building a mobile movie recommendation service by user rating and APP usage with linked data on Hadoop," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 3383–3401, 2017.

[7] R. Burke, "Hybrid recommender systems: survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[8] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: state of the art and trends," in *Recommender Systems Handbook*, pp. 73–105, Springer, Boston, MA, USA, 2011.

[9] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," *Foundations and Trends in Human-Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[10] J. Zhang, Q. Peng, S. Sun, and C. Liu, "Collaborative filtering recommendation algorithm based on user preference derived from item domain features," *Physica A: Statistical Mechanics and Its Applications*, vol. 396, pp. 66–76, 2014.

[11] G. Huming and L. Weili, "A hotel recommendation system based on collaborative filtering and rankboost algorithm," in *Proceedings of the 2010 Second International Conference on Multimedia and Information Technology (MMIT)*, vol. 1, pp. 317–320, IEEE, Kaifeng, China, April 2010.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, pp. 285–295, ACM, Hong Kong, May 2001.

[13] Z. Tan and L. He, "An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance principle," *IEEE Access*, vol. 5, pp. 27211–27228, 2017.

[14] K. Zhang, K. Wang, X. Wang, C. Jin, and A. Zhou, "Hotel recommendation based on user preference analysis," in *Proceedings of the 2015 31st IEEE International Conference on Data Engineering Workshops (ICDEW)*, pp. 134–138, IEEE, Seoul, South Korea, April 2015.

[15] L. Chen and F. Wang, "Preference-based clustering reviews for augmenting e-commerce recommendation," *Knowledge-Based Systems*, vol. 50, pp. 44–59, 2013.

[16] Z. Chang, M. S. Arefin, and Y. Morimoto, "Hotel recommendation based on surrounding environments," in *Proceedings of the 2013 IIAI International Conference on Advanced Applied Informatics (IIAIAAI)*, pp. 330–336, IEEE, Matsue, Japan, August-September 2013.

[17] Y. Sharma, J. Bhatt, and R. Magon, "A multi-criteria review-based hotel recommendation system," in *Proceedings of the 2015 IEEE International Conference on, Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)*, pp. 687–691, IEEE, Liverpool, UK, October 2015.

[18] T. Chen and Y. H. Chuang, "Fuzzy and nonlinear programming approach for optimizing the performance of ubiquitous hotel recommendation," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 2, pp. 275–284, 2018.

[19] A. Verma and H. Virk, "A hybrid genre-based recommender system for movies using genetic algorithm and kNN approach," *International Journal of Innovations in Engineering and Technology*, vol. 5, no. 4, pp. 48–55, 2015.

[20] K. Kabassi, "Personalizing recommendations for tourists," *Telematics and Informatics*, vol. 27, no. 1, pp. 51–66, 2010.

[21] U. Fasahte, D. Gambhir, M. Merulingkar, and A. M. P. A. Pokhare, "Hotel recommendation system," *Imperial Journal of Interdisciplinary Research*, vol. 3, no. 11, 2017.

[22] G. Crespo, A. L. Cuadrado, J. L. C. Palacios, R. G. Carrasco, and I. R. Mezcua, "Sem-Fit: a semantic based expert system to provide recommendations in the tourism domain," *Expert Systems with Applications*, vol. 10, no. 38, pp. 13310–13319, 2011.

[23] Y. H. Hu, P. J. Lee, K. Chen, J. M. Tarn, and D.-V. Dang, "Hotel recommendation system based on review and context information: a collaborative filtering appro," in *Proceedings of the Pacific Asia Conference on Information Systems PACIS*, p. 221, Chiayi City, Taiwan, June-July 2016.

[24] S. Y. Hwang, C. Y. Lai, S. Chang, and J. J. Jiang, "The identification of noteworthy hotel reviews for hotel management," *Pacific Asia Journal of the Association for Information Systems*, vol. 6, no. 4, 2015.

[25] R. Sandeep, S. Sood, and V. Verma, "Twitter sentiment analysis of real-time customer experience feedback for predicting growth of Indian telecom companies," in *Proceedings of the 2018 4th International Conference on Computing Sciences (ICCS)*, pp. 166–174, IEEE, Phagwara, India, August 2018.

[26] S. Meng, W. Dou, X. Zhang, and J. Chen, "KASR: a Keyword-Aware Service Recommendation method on MapReduce for big data applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3221–3231, 2014.

[27] A. Ghose, P. G. Ipeirotis, and B. Li, "Designing ranking systems for hotels on travel search engines by mining user-generated and crowdsourced content," *Marketing Science*, vol. 31, no. 3, pp. 493–520, 2012.

[28] N. M. Almeida, J. A. Silva, J. Mendes, and P. Oom do Valle, "The effects of marketing communication on the tourist's hotel reservation process," *Anatolia*, vol. 23, no. 2, pp. 234–250, 2012.

[29] K. P. Lin, C. Y. Lai, P. C. Chen, and S. Y. Hwang, "Personalized hotel recommendation using text mining and mobile browsing tracking," in *Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 191–196, IEEE, Hong Kong, China, October 2015.

[30] N. Rianthong, A. Dumrongsiri, and Y. Kohda, "Improving the multidimensional sequencing of hotel rooms on an online travel agency web site," *Electronic Commerce Research and Applications*, vol. 17, pp. 74–86, 2016.

[31] G. S. Lal and A. S. Baghel, "Efficient feature extraction in sentiment classification for contrastive sentences," *International Journal of Modern Education and Computer Science*, vol. 10, no. 5, p. 54, 2018.

[32] D. Jannach, F. Gedikli, Z. Karakaya, and O. Juwig, "Recommending hotels based on multi-dimensional customer ratings," in *Information and Communication Technologies in Tourism*, Springer, Vienna, Austria, 2012.

[33] M. Ibrahim, I. S. Bajwa, R. Ul-Amin, and B. Kasi, "A neural network-inspired approach for improved and true movie recommendations," *Computational Intelligence and Neuroscience*, vol. 2019, no. 7, Article ID 4589060, 19 pages, 2019.

[34] C. Bouras and V. Tsogkas, "Improving news articles recommendations via user clustering," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 1, pp. 223–237, 2017.

[35] D. Valcarce, J. Parapar, and A. Barreiro, "A distributed recommendation platform for big data," *Journal of Universal Computer Science*, vol. 21, no. 13, pp. 1810–1829, 2015.

[36] S. Ishtiaq, N. Majeed, M. Maqsood, and A. Javed, "Improved scalable recommender system," *The Nucleus*, vol. 53, no. 3, pp. 200–207, 2016.

[37] H. Jazayeriy, S. Mohammadi, and S. Shamshirband, "A fast recommender system for cold user using categorized items," *Mathematical and Computational Applications*, vol. 23, no. 1, p. 1, 2018.

[38] K. Sudha, M. Lavanya, and A. Kanimozhi, "Item based collaborative filtering approach for big data application," *International Journal of Scientific Research Engineering & Technology (IJSRET)*, vol. 3, no. 8, pp. 1222–1224, 2014.

[39] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[40] M. Manu and B. Ramesh, "Single-criteria collaborative filter implementation using Apache Mahout in big data,"

International Journal of Computer Sciences and Engineering, vol. 5, no. 1, pp. 7–13, 2017.

[41] S. Morozov and X. Zhong, "The evaluation of similarity metrics in collaborative filtering recommenders," in *Proceedings of the 2013 Hawaii University International Conferences Education & Technology Math & Engineering Technology*, Honolulu, HI, USA, June 2013.

[42] Q. Shambour, M. A. Hourani, and S. Fraihat, "An item-based multi-criteria collaborative filtering algorithm for personalized recommender systems," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 8, pp. 275–279, 2016.

[43] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Comparison of collaborative filtering algorithms: limitations of current techniques and proposals for scalable, high-performance recommender systems," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 1, p. 2, 2011.

[44] M. Nilashi, O. bin Ibrahim, N. Ithnin, N. H. Sarmin, and N. H. Sarmin, "A multi-criteria collaborative filtering recommender system for the tourism domain using expectation maximization (EM) and PCA-ANFIS," *Electronic Commerce Research and Applications*, vol. 14, no. 6, pp. 542–562, 2015.

[45] P. Phorasim and L. Yu, "Movies recommendation system using collaborative filtering and *k*-means," *International Journal of Advanced Computer Research*, vol. 7, no. 29, pp. 52–59, 2017.

[46] S. Kögel, "Recommender system for model driven software development," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pp. 1026–1029, Paderborn, Germany, September 2017.

[47] M.-P. T. Do, D. V. Nguyen, and L. Nguyen, "A model-based approach for collaborative filtering," in *Proceedings of the 6th International Conference on Information Technology for Education*, Ho Chi Minh City, Vietnam, August 2010.

[48] W. Yibo, M. Wang, and W. Xu, "A sentiment-enhanced hybrid recommender system for movie recommendation: a big data analytics framework," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 8263704, 9 pages, 2018.