Knowledge-Based Systems 181 (2019) 104791

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Adaptive resource prefetching with spatial–temporal and topic information for educational cloud storage systems☆

Qionghao Huang [a], Changqin Huang [a,b,*], Jin Huang [a], Hamido Fujita [c,d]

[a] South China Normal University, Guangzhou, China
[b] Department of Educational Technology, Zhejiang Normal University, China
[c] Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Viet Nam
[d] Software and Information Science Iwate Prefectural University, Takizawa, 020-0693, Japan

HIGHLIGHTS

- A novel topic model is proposed to estimate the resource popularity in terms of educational topics.
- An adaptive prefetching inference approach is designed to predict popular educational resources.
- An efficient prefetching management mechanism is introduced to support geo-distributed cloud storage systems.

ARTICLE INFO

Article history:
Received 17 April 2019
Received in revised form 28 May 2019
Accepted 29 May 2019
Available online 31 May 2019

Keywords:
Spatial–temporal locality
Topic information
Semantic inference
Prefetching mechanism
Educational resources

ABSTRACT

Prefetching proactively resources at datanodes within distribution networks plays a key role in improving the efficiency of data access for e-learning, which requires assistance from semantic knowledge on educational applications and resource popularity. To capture such information, we should exploit the characteristics of education end-users' requests with high spatial–temporal locality. This paper aims to develop resource popularity modeling techniques for enhancing the performance of educational resource prefetching. Specifically, a novel topic model, built on an accelerated spectral clustering and an ontology concept similarity, is proposed to support resource access based on semantic features, including topic relevance and spatial–temporal locality. Using the proposed model, an adaptive prefetching inference approach is presented to associate possible popular resources in the future data requests. Also, an efficient prefetching management mechanism incorporating with replica techniques is suggested to design resource cloud storage systems for geo-distributed educational applications. Experiments over a simulation setting and a real-world case study with seven million users across China are carried out. Results demonstrate that the proposed method performs favorably compared to the state-of-the-art approaches.

1. Introduction

Online educational resources have been experiencing an explosive growth in recent years, while cloud storage systems based on geo-distributed datacenters, as a promising solution to manage such big volume of data, gain a widely use in educational applications in the era of cloud computing [1–3]. Meanwhile, network traffic of these applications also grows dramatically [4], and imposes a heavy burden on core networks, resulting in a long latency of resource access and poor experience for end-users [5]. It is an effective method to alleviate the traffic burden, reduce access latency, and provide a favorable QoS (Quality of Services) for users through proactive resource prefetching or caching at the datanodes bypass core networks [6,7]. Due to the vast amount of educational resources delivered through core networks, we cannot expect to perform prefetching all the resources for local end-users by using few datanodes. Thus, it is interesting and necessary to develop some effective prefetching strategies to place educational resources with high-frequency requests at datanodes that reduce resource access latency for end-users. To achieve this goal, we need a method to characterize the resource popularity in this working domain. Most of the previous studies focus on the use of statistical features to model the resource popularity. For instance, some works assume that resource popularity follows

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.05.034.

* Corresponding author at: Department of Educational Technology, Zhejiang Normal University, China.
E-mail addresses: cqhuang@zju.edu.cn (C. Huang), h.fujita@hutech.edu.vn (H. Fujita).

https://doi.org/10.1016/j.knosys.2019.05.034
0950-7051/

some probability distribution, and adopt certain statistical or machine learning techniques, such as a cost-effective scheme [8], collaborative similarity [9], PDE (Partial Differential Equation) & ODE (Ordinary Differential Equation) analysis [10], a transfer learning-based approach [11], a linear learning model [12], to estimate popularity of resources from historical data. It is observed that these methods have not taken domain knowledge into consideration for resource popularity modeling. In the educational domain, we can take advantages of characteristics from learning behaviors, such as spatial–temporal patterns resulted from topic oriented online resource demands [13,14]. Here, we refer to some statistical or semantic patterns of resource requests within a specific time slot and location as the spatial–temporal locality [15,16]. Such spatial–temporal locality usually indicates regularity of the curriculum activities that users engaged in, commonly focusing on certain educational topics at some period and place [17,18]. Thus, it is of great significance to develop an efficient prefetching mechanism incorporating with such valuable information for maximizing the efficiency of datanode storage within high-speed distribution or access networks. To practically leverage such information to estimate the popularity of resources, the key is to discovery these educational topics related to the corresponding locality. Thus, it is of great significance to develop an efficient prefetching mechanism incorporating with such valuable topics for maximizing the efficiency of datanode storage within high-speed distribution networks.

In general, the locality occurs while topic-oriented resources are accessed. Accordingly, it is appropriate that a topic model is built to uniformly describe the access locality and topics. Concretely, taking large-scale user queries or data request logs as the initial input, the model firstly leverages an accelerated spectral clustering algorithm to detect possible educational topics. Then an ontology similarity based algorithm is employed to improve the model in distinguishing real topics from candidate educational topics. With the proposed model, a relevant prefetching inference approach is presented to match possible popular resources. Furthermore, an adaptive prefetching management mechanism, including the selection of datanodes for prefetching, the calculation of window size for prefetching, and the operations for prefetching with different lifecycles of topics, is suggested to design cloud storage systems for geo-distributed educational applications. The proposed algorithms are both evaluated by simulation experiments and a real-world educational application named *WorldUC*[1](*World-wide City of Universities*[2]) with more than seven million online users across China. The results show that our algorithms promise a state-of-the-art performance and promote the efficiency of cloud storage systems for education applications greatly.

Our contributions are summarized as follows:

1. By employing an accelerated spectral clustering and an ontology concept similarity, a novel topic model is proposed to estimate the resource popularity for exploring educational topics behind the spatial–temporal locality.
2. With respect to the spatial–temporal locality, an adaptive light-weighted semantic prefetching inference approach is designed to predict popular educational resources.
3. Through incorporating with replica techniques, a systematic and efficient prefetching management mechanism is introduced to support cloud storage systems for geo-distributed educational applications.

[1] http://www.worlduc.com/.

[2] https://itunes.apple.com/cn/app/da-xue-cheng-yun-ke-tang/id1340840752.

The remainder is organized as follows: Section 2 reviews related work, Section 3 formulates the problems in this paper, Section 4 describes the system architecture and solution overview, Section 5 details algorithms for prefetching management. Section 6 reports the results of experiments, followed by the conclusion in Section 7.

2. Related work

The optimization of data placement in geo-distributed data-intensive applications is an important solution to lower the implication (limited bandwidth or monetary cost) brought by the transmission network besides job scheduling [19]. Prefetching techniques incorporating with replica management can mitigate the implication in such application environments [20]. Prefetching techniques are commonly used in different levels of cyber-based systems [21], such as instruction prefetching [22], in-memory prefetching [23], disk prefetching [24], and online resource prefetching [25–27]. As for resource prefetching or caching in network environments, there are lots of research on the prefetching in the edge of mobile networks [28–30], in distribution networks [31], and in geo-distributed datacenters [32]. It is convinced that a few popular resource items contribute to most of the views [6]. Therefore, it is of great importance to predict the future resource popularity to prefetch or cache popular resources within limited data storage capacity, while resource popularity is often unknown in advance.

Some works on resource popularity estimation assume that the resource popularity profile is unknown but time invariant [9,11,33]. In [9] study, the researchers employed a retention value and number of common clients to estimate the popularity of video segments from history logs, and provided a means for efficient dissemination of personalized video streams in resource-constrained environments. In [33,34] study, the traffic characteristics of YouTube was investigated and discovered that the popularity could be modeled by a Weibull distribution. In [11], the popularity profile was estimated by a transfer learning approach while leveraging prior information from a surrogate domain. However, most of real-world applications are running in dynamic environments, and hence the aforementioned works may not be applicable within dynamic systems.

Another works that take dynamic factors (such as the temporal variation of user preference) into consideration have achieved satisfactory performance in estimating the resource popularity profile [12,35]. In [35], considering the dependence of user's preferences on their context, the resource popularity was estimated by using contextual multi-armed bandit optimization. In [12], the study incorporated resource feature and location characteristic to predict future resource hit rate through a linear model. For the lack of a deep analysis of *reasons* behind the popularity, the above works are still not adaptive enough to be applicable in educational applications. Semantic techniques can fully exploit valuable information of resources, webpages, data request logs, and *etc.* to perform resource prefetching or caching, and achieve a promising performance [27,31,36]. However, these works are designed for general domains, which may be unwieldy and inefficient in a specific domain, especially the one that has distinct characteristics.

A more light-weighted and efficient solution to predict popular resources in the future data requests is possible with the assistance of the spatial–temporal locality of data access in educational applications. With such knowledge, a practical prefetching management is proposed to promote the efficiency of cloud storage systems for educational applications in this paper.

Table 1
Notations.

Notations	Meanings
c	A concept
$\boldsymbol{C}$	A cluster of concepts
$\boldsymbol{C}_i^{ca}$	A collection of candidate central concepts for a concept cluster $\boldsymbol{C}_i$
c_i^c	A central concept of $\boldsymbol{C}_i$
$\boldsymbol{\mathcal{A}}$	A partition of service districts for educational cloud storage systems, and $a_{i'}$ denotes its element
$\boldsymbol{S}$	A sequence of concepts
$\boldsymbol{S}_{i,j}$	*STRC* for Concept $c_{i,j}$ within $\boldsymbol{C}_i$, formulated as $\boldsymbol{S}_{i,j} = TSeq(c_{i,j})$, and $c_{i',j}$ denotes its element
$\mathcal{O}$	An ontology of educational resources
$\boldsymbol{TD}$	A collection of **T**arget **D**atanodes to prefetch or cache educational resources for users in District $a_{i'}$
$\hat{\boldsymbol{\mathcal{S}}}$	The ordered resources to be prefetched
z	The window size or data volume of prefetching

3. Problem formulation

As discussed before, to employ the spatial–temporal locality to estimate the popularity of resources, it is vital to model educational topics for the resultant locality. Most of topic models gain a low explainability in their outputs and cannot be applied to trace educational topics that cause a spatial–temporal locality [37,38]. Thus, we leverage ontology techniques to build a more explainable topic model to trace topics for the resultant locality. Before detailing the model, some notations (Table 1) and definitions are introduced firstly.

Concept is a node or entity in an ontology of educational resources.

Sequence of Topic Relevant Concepts (*STRC*) for a concept is a sequence of concepts from a certain operation between a concept and the other concepts within a concept cluster. The generation of *STRC* is detailed in Section 5.2.2.

Candidate central concept is a concept which similarity between its *STRC* reaches a defined threshold value. The similarity between concepts is calculated by Eq. (4), and there are usually more than one candidate central concept within a concept cluster.

Central concept is a concept that *represents* a topic of a concept cluster as much as possible, and it is chosen from candidate central concepts.

With the above notations, a topic in terms of an ontology $\mathcal{O}$ of educational resources is defined as

$$\mathcal{T} = (\boldsymbol{C}_i, c_i^c, \boldsymbol{S}_i^c, \varepsilon_1, \mathcal{O}), \tag{1}$$

where ε_1 is a threshold to determine whether $\boldsymbol{C}_i$ contains a topic, and its relevant metric is defined by Eq. (18).

Thus, a topic collection $\boldsymbol{\mathcal{T}}$ that causes spatial–temporal locality at District $a_{i'}$ within a time slot Δt is expressed as

$$\boldsymbol{\mathcal{T}}_{\Delta t,i'} = \{\mathcal{T}_1, \ldots, \mathcal{T}_k\}, \tag{2}$$

where k is the cardinality of $\boldsymbol{\mathcal{T}}_{\Delta t,i'}$.

With a topic model, we can develop effective algorithms to solve out several parameters ($\hat{\mathcal{S}}$, z, $\boldsymbol{TD}$) in the prefetching process. Thus, in this paper, we employ an accelerated spectral clustering and an ontology concept similarity metric to obtain $\boldsymbol{\mathcal{T}}_{\Delta t,i'}$ from large-scale logs of users' queries or annotations from data requests. Using the constructed collections of topics, we propose an adaptive semantic inference approach to get the resource collection $\hat{\mathcal{S}}$. Then z and $\boldsymbol{TD}$ is calculated based on the intensity of topics and the workload of cloud storage systems.

4. System architecture

As depicted in Fig. 1, the architecture of an educational cloud storage system in this paper mainly consists of two components: essential modules and prefetching management. Essential modules consist of *User Interface*, *Data Request Controller*, *Data Storage Severs*, and *Replica Management*. The proposed **P**refetching **M**anagement (PM) in this paper is a system optimization module for cloud storage systems to promote the efficiency of data access, and the main parts of it are briefly described below.

1. **Spectral Clustering Based Topic Detection and Selection**: to apply the topic model in Eq. (2) to estimate the popularity of resources, an algorithm based on an accelerated spectral clustering and ontology concept similarity is proposed to detect candidate topics (Section 5.1). With obtained collections of topics, central concepts and *STRC*s are also solved out to support the semantic inference of prefetching resources (Section 5.2).
2. **Light-weight Semantic Prefetching Inference**: with central concepts and *STRCs* from collections of topics, a light-weight semantic prefetching inference approach is proposed to generate a sequence prefetching educational resources (Section 5.3.1).
3. **Prefetching Management with Topic Lifecycles**: after the inference of prefetching resources, the prefetching window size and target datanodes are calculated for performing prefetching operations (Section 5.3.2). A systematic prefetching management mechanism incorporating with replica management and topic lifecycles is introduced to enhance the performance of educational cloud storage systems (Section 5.3.3).

Relevant algorithms to realize prefetching management for educational cloud storage systems are detailed in the following (Section 5).

5. Algorithms for prefetching management

In this section, we detail the algorithms to solve the problems mentioned in Section 3, including the construction of Eq. (2), the prefetching inference of $\hat{\mathcal{S}}$, the calculation of z and $\boldsymbol{TD}$, and a prefetching management mechanism incorporating with lifecycles of topics.

5.1. Candidate topic detection

In this part, an accelerated spectral clustering algorithm is employed to obtain clustered concepts (denoted as $\{\boldsymbol{\mathcal{C}}_1, \boldsymbol{\mathcal{C}}_2, \ldots, \boldsymbol{\mathcal{C}}_k\}$) from massive users' queries or data request logs. Each concept cluster $\boldsymbol{\mathcal{C}}_i$ form a candidate topic that may result in a spatial–temporal locality in data access of users. We utilize a HowNet-based ontology of education resources [39,40] as $\mathcal{O}$ in Eq. (1). In the following, we detail the detection of candidate topics.

5.1.1. Concept extraction using educational ontology

The workflow of concept extraction is shown in Fig. 2. The proposed accelerated spectral clustering algorithm takes concepts extracted from massive users' queries or data request logs from some service district as an initial input.

Data Request Controller extracts a certain number of keywords from annotations associated with user requested educational resources. Extracted keywords are mapped into the corresponding concepts in educational ontology through a matching algorithm [41], then output a collection of concepts.

Assuming that there are m sessions of users' requests within a time slot Δt in some service district, a collection of concepts after a matching algorithm is denoted as

$$\boldsymbol{C}_{m,\Delta t} = \{c_{1,1}, c_{1,2}, c_{1,3}, \ldots, c_{i,1}, c_{i,j}, c_{i,3}, \ldots, c_{m,1}, c_{m,2}, c_{m,n}\}, \tag{3}$$

where $c_{i,j} (i, j \in \mathbb{N})$ denotes the jth concept mapping to the ith session.

The system performs a topic relatedness analysis on $\boldsymbol{C}_{m,\Delta t}$, and the topic detection process is executed when the relatedness reaches a defined threshold value.

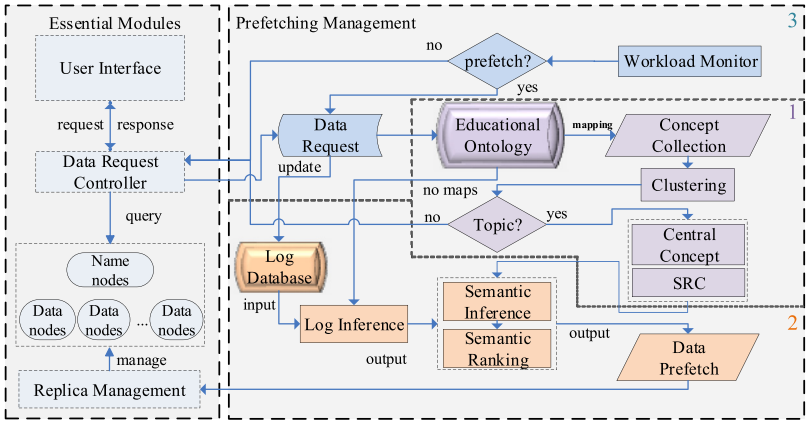

Fig. 1. The architecture of data prefetching system.

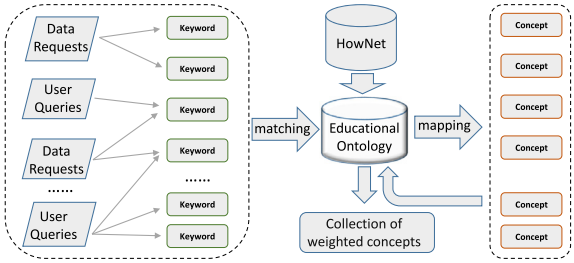

Fig. 2. The concept extraction of users' queries or requests by using educational ontology.

5.1.2. Topic relatedness analysis of concept collection

To minus a computation cost, the system randomly samples m' concepts ($m' \ll m$) from $\boldsymbol{C}_{m,\Delta t}$ to perform topic relatedness analysis. A sampled collection of concepts is denoted as $\boldsymbol{C}^{sa} = \{c_k^s \mid c_k^s \in C_{m,\Delta t}, 1 \le k \le m'\}$. $Sim(a, b)$ denotes a function that calculates the similarity between Concept a and Concept b in the educational ontology [39].

$$Sim(a,b) = \frac{\alpha(depth(a) + depth(c))}{\alpha(depth(a) + depth(b)) + dist(a,b) + \mid depth(a) - depth(b) \mid}. \tag{4}$$

By applying $Sim(a, b)$ among the concepts in $\boldsymbol{C}^{sa}$, a similarity matrix of $\boldsymbol{C}^{sa}$ is expressed as

$$M^{sa} = \left(Sim(c_i^s, c_j^s)\right)_{m' \times m'}, \tag{5}$$

where $Sim(c_i, c_i) = 1.000$, $Sim(c_i, c_j) = Sim(c_j, c_i)$, $c_i, c_j \in \boldsymbol{C}^{sa}$.

Then the mean concept similarity of $\boldsymbol{C}^{sa}$ is given by

$$\overline{Sim(C^{sa})} = \frac{2\sum_{j=2}^{m'}\sum_{i=1}^{j-1} Sim(c_i, c_j)}{m(m-1)}. \tag{6}$$

A threshold ε_1 (obtained by simply analyzing historical data) is preset to determine whether there exists topics in $\boldsymbol{C}_{m,\Delta t}$. That is if $\overline{Sim(\boldsymbol{C}^{sa})} \ge \varepsilon_1$ maintains in most of the sampling processes, then it performs the candidate topic detection, or *Data Request Controller* continues to monitor users' queries or requests from the next time slot $\Delta(t+1)$. In the following, we present the process of candidate topic detection.

5.1.3. Accelerated spectral clustering based detection

There are many resultant problems when vectorizing concepts for designing algorithms, such increasing the dimension of data, causing loss of information. Therefore, we propose a spectral clustering algorithm that uses a similarity function to construct the affinity matrix rather than other techniques based on vector-form concepts. To deal with a large-scale concept collection, we combine weighted k-means with optimized time-stochastic subspace sampling to speed up the clustering process. The algorithm is detailed in the following.

Let set $\boldsymbol{V} = \{v_1, \ldots, v_i, \ldots, v_n\}$ denote a $\boldsymbol{C}_{m,\Delta t}$ after removing repeated concepts within it. Instead of using other distance metrics (such as Gaussian Kernel) relying on vector-form concepts, we form the elements $\omega_{i,j}$ of the affinity matrix A by Eq. (4), and the matrix A is written as

$$A = (\omega_{i,j})_{n \times n}, \tag{7}$$

where $w_{i,j} = Sim(v_i, v_j)$ is defined as the similarity between v_i and v_j, and $\omega_{i,j} = \omega_{j,i}$. And its degree matrix D is a diagonal matrix as follows

$$D := diag(A\mathbf{1}_n), \tag{8}$$

where $diag(.)$ is to generate a diagonal matrix by a vector input, and $\mathbf{1}_n \in \mathbb{R}$ is a vector with all its elements equal to 1.

The goal of clustering is to separate $\boldsymbol{V}$ into k subgraphs, $\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_k$, where $\boldsymbol{V}_i \cap \boldsymbol{V}_j = \emptyset$ and $\boldsymbol{V}_1 \cup \boldsymbol{V}_2 \cup \cdots \cup \boldsymbol{V}_k = \boldsymbol{V}$. For convenience, we define a membership matrix $U = \{u_1, \ldots, u_k\}^T \in \mathbb{R}^{k \times n}$ to represent the separation results $\{\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_k\}$, where $u_i^T \in \mathbb{R}^{1 \times n}$ is the ith row of U to tell the members of cluster V_i. Besides, the matrix $Z \in \mathbb{R}$ subjects to $Z = (z_1, \ldots, z_k)^T = (UDU^T)^{-1/2}U$, where $z_i = u_i(u_i^T Du_i)^{-1/2}$, and $tr(\cdot)$ represents the trace of a matrix. Thus, the objective function of a traditional algorithm for spectral clustering named *NormalizedCut* [42] is rewritten as

$$\begin{aligned} NCut(\boldsymbol{V}_1, \ldots, \boldsymbol{V}_k) &= \sum_{i=1}^{k} \frac{u_i^T A u_i}{u_i^T D u_i} = \sum_{i=1}^{k} \frac{u_i^T}{(u_i^T D u_i)^{1/2}} A \frac{u_i}{(u_i^T D u_i)^{1/2}} \\ &= \sum_{i=1}^{k} z_i^T A z_i = tr(ZAZ^T). \end{aligned} \tag{9}$$

Let $\tilde{Z} = ZD^{1/2}$, then Eq. (9) is rewritten as

$$tr(ZAZ^T) = tr(ZD^{1/2} \cdot D^{-1/2}AD^{-1/2} \cdot D^{1/2}Z^T) = tr(\tilde{Z}AD^{-1/2}\tilde{Z}^T). \quad (10)$$

Therefore, to use the objective function to get the *Best Cut* is expressed as

$$\max_{\tilde{Z}} tr(\tilde{Z}D^{-1/2}AD^{-1/2}\tilde{Z}^T). \quad (11)$$

The solution of Eq. (11) can be transformed into getting k_1 eigenvalues and corresponding eigenvectors $(f_1, f_2, \ldots, f_{k_1})$ when Eq. (11) is the maximum by using Lagrange multipliers. However, it becomes inefficient when it comes to large-scale datasets, which often happens in educational applications. Thus, a fast algorithm for large-scale spectral clustering is urgent to address the problem.

A distance function between v_i and v_j is defined as

$$d_k^2(v_i, v_j) = 2(1 - Sim(v_i, v_j)). \quad (12)$$

Let $K = (K_{i,j})_{n\times n} \in \mathbb{R}$, where $K_{i,j} = d_k(v_i, v_j)$, and k is the number of the clusters. Therefore, to get *Best Cut* of $\boldsymbol{V} = \{\boldsymbol{V}_1, \ldots, \boldsymbol{V}_k\}$ equals to minimize the distance between data points and central points within clusters. To relate the problem to *Normalized Cut*, we define the objective function as

$$min\{\boldsymbol{V}_1, \ldots, \boldsymbol{V}_k\} = \sum_{i=1}^{k}\sum_{v_j\in V_i} w_j d_k^2(v_j, c_i) = \sum_{i=1}^{k}\sum_{j}^{n} w_j u_{i,j} d_k^2(v_j, c_i), \quad (13)$$

where w_j is a weight for Concept v_j, c_i is the center of Cluster $\boldsymbol{V}_i$, and $u_{i,j}$ is an element of the membership matrix U.

Let $Y \in \mathbb{R}^{k\times n}$ be a weighted membership matrix, where $Y = (y_1, \ldots, y_k)^T = UW$, and $W \in \mathbb{R}^{k\times n} : W = diag(w_1, \ldots, w_n)$. Let $s_i = y_i^T \mathbf{1}_n$ and $L \in \mathbb{R}^{k\times k} : L = [diag(s_1, \ldots, s_n)]^{-1}$. Then we normalize Y with L, we get

$$\tilde{Y} = (\tilde{y}_1, \ldots, \tilde{y}_k)^T = L^{1/2}UW^{1/2}.$$

Therefore, Eq. (13) is rewritten as

$$\max_{\tilde{Y}} tr(\tilde{Y}W^{-1/2}KW^{-1/2}\tilde{Y}^T). \quad (14)$$

From Eq. (11) and Eq. (15), they are equivalent to each other, that is we can use Eq. (15) to get a *best cut* of V without doing the time-consuming eigen-decomposition of a laplacian matrix [43]. The work [44] provides an effective way to solve Eq. (15), from which Eq. (15) is rewritten as

$$\max_{\tilde{Y}} tr(\tilde{Y}W^{-1/2}KW^{-1/2}\tilde{Y}^T) = \sum_{i=1}^{k}\sum_{j=1}^{n} w_j u_{i,j} d_k^2(v_j, c_i)^2 \in \tilde{\mathcal{H}}_k, \quad (15)$$

where $\tilde{\mathcal{H}}_k \subset \mathcal{H}_k = span(d_k(v_i, \cdot), \ldots, d_k(v_n, \cdot))$, namely $\tilde{\mathcal{H}}_k$ is generated from a subspace of the space generated by the whole concepts.

It is practical to sample m concepts ($m \ll n$) to approach the approximate solution of Eq. (15) if the quality of sampling is good enough. And we leverage the spatial–temporal locality in educational applications to optimize the sampling process to get a fast convergence. The sampling process is formulated as follows:

$$m = \varsigma \sum_{i=0}^{t'} n_i \varphi(\kappa \frac{log_{s_i}(n_i)}{s_i} - \chi\kappa \frac{log_{s_{i-1}}(n_{i-1})}{s_{i-1}}) \quad (16)$$

where $n_0 = 0$, s_i is the number of sessions in time slot i, ς is a scalar from sampling, $\varphi(\cdot)$ is a distribution function proportional to the size of input, κ is a topical relevant coefficient, and χ is a decay coefficient. *tSample(·)* denotes this sampling process.

With the aforementioned notations and equations, the whole workflow in this subsection is summarized in Alg. 1. Each cluster forms a candidate topic, and it is determined whether to be a topic in the next subsection.

Algorithm 1: Accelerated Spectral Clustering Based Candidate Topic Detection

Input: $\boldsymbol{C}_{m,\Delta t}$: a collection of concepts
m: the number of concepts in sampling
k: the number of clusters
miter: the maximum of iteration
Output: $\{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots \boldsymbol{C}_k\}$ //a collection of concept clusters that may form topics

$\tilde{\boldsymbol{V}} = \{\tilde{v}_1, \ldots, \tilde{v}_m\} \Leftarrow tSample(\boldsymbol{C}_{m,\Delta t})$ //generate a subset from sampling;
$\{\tilde{K}, \hat{K}\} \Leftarrow \{K = D^{-1}AD^{-1}\}$ // compute $\tilde{K}, \hat{K}$;
$T \Leftarrow \tilde{K}\hat{K}^{-1}$ // compute T;
$U \Leftarrow Rand(U)$ & $t \Leftarrow 0$ // initiate U and the iteration index t;
while *U doesn't change* ***or*** $t >$ *miter* **do**
  $t = t + 1$;
  $Y = UW$ & $\hat{Y} = LY$;
  $\alpha = \tilde{Y}T = \hat{Y}\tilde{K}\hat{K}^{-1}$;
  for $j = 1, \ldots, n$ **do**
    $i' = \arg\min_{i\in[k]} d_k^2(v_j, c_i)$;
    $U_{i,j} = 0, i = 1, \ldots, k$;
    $U_{i,j} = 1, i = i'$;
  end for
end while
$\{\boldsymbol{V}_1, \ldots, \boldsymbol{V}_k\} \Leftarrow U$;
$\{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_k\} \Leftarrow \{\boldsymbol{V}_1, \ldots, \boldsymbol{V}_k\}$;
return $\{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_k\}$

5.2. Topic and central concept selection

Repeated concepts are removed in the aforementioned candidate topic detection, the output $\{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_k\}$ of Alg. 1 may not cause a spatial–temporal locality in Δt. Therefore, we take repeated concepts into consideration to distinguish concept clusters (denoted as $\{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_{k_1}\}$) that form a topic from $\{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_k\}$. With $\{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_{k_1}\}$, central concepts and *STRC*s also are solved out in the following.

5.2.1. Selection of central concept

The weight of a concept means the repeated times of the concept within $\boldsymbol{C}_{m,\Delta t}$, and it is denoted as $Weight(c_{i,j}) = \Omega_{i,j}$. Thus, the weight of $\boldsymbol{C}_i$ is defined as

$$Weight(\boldsymbol{C}_i) = \sum_{i'=1}^{Size(\boldsymbol{C}_i)} c_{i'}, c_{i'} \in \boldsymbol{C}_i. \quad (17)$$

The density of a cluster is used to tell if there exists a topic within a cluster,

$$Density(\boldsymbol{C}_i) = \frac{Weight(\boldsymbol{C}_i)}{Size(\boldsymbol{C}_i)}, \quad (18)$$

where if $Density(\boldsymbol{C}_i) \geq \varepsilon_1$, then $\boldsymbol{C}_i$ exists a topic. ε_1 is a threshold value related to the optimized dynamic relation to the workload.

After such a selection, the concept cluster collection is rewritten as $(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_{k_2})$, then a central concept and its *STRC* would be obtained within each cluster.

The similarity between $c_{i,j}$ and $\boldsymbol{S}_{i,j}$ within Cluster $\boldsymbol{C}_i$ is expressed as

$$SSS(\boldsymbol{S}_{i,j}) = \sum_{i'}^{Size(\boldsymbol{S}_{i',j})} \Omega_{i,j} Sim(c_{i,j}, c_{i',j}), \quad (19)$$

and its mean and variance for $c_{i,j}$ and $\boldsymbol{S}_{i,j}$ are written as

$$SAS(\boldsymbol{S}_{i,j}) = \frac{SSS(\boldsymbol{S}_{i,j})}{Size(\boldsymbol{S}_{i,j})}, \quad (20)$$

$$\sigma(\boldsymbol{S}_{i,j}) = \sqrt{\frac{\sum_{i'=1}^{Size(\boldsymbol{S}_{i,j})} (\Omega_{i',j})^2 (Sim(c_{i,j}, c_{i',j}) - SAS(\boldsymbol{S}_{i,j}))^2}{Size(\boldsymbol{S}_{i,j})}}. \quad (21)$$

A candidate central concept is chosen by applying the following two steps:

- Step I : choose $c_{i'}$ from $\boldsymbol{C}_i$, and compute the similarity between $c_{i'}$ and the other concepts in $\boldsymbol{C}_i$, and remove the concepts which similarity with $c_{i'}$ is less than ε_3, and generate $\boldsymbol{S}_i$ for $c_{i'}$ from the rest concepts;
- Step II : calculate $SSS(\boldsymbol{S}_i)$, $SAS(\boldsymbol{S}_i)$, $\sigma(\boldsymbol{S}_i)$, keep $c_{i'}$ as a candidate concept if $SSS(\boldsymbol{S}_i) \geq \varepsilon_4$ and $\sigma(\boldsymbol{S}_i) \geq \varepsilon_5$. ε_4 is a threshold value directly proportional to the number of sessions, and ε_5 is a threshold value related to the intensify of topic relatedness in Eq. (25).

After the generation of $\boldsymbol{C}_i^{ca}$, c_i^c for $\boldsymbol{C}_i$ is obtained from $\boldsymbol{C}_i^{ca}$ by performing the calculation of the centrality of candidate concepts, and the centrality is given by

$$D(c_i) = \sum_{j}^{i-1} Sim(c_i, c_j) + \sum_{j=i+1}^{Size(\boldsymbol{C}_i^{ca})} Sim(c_i, c_j). \quad (22)$$

Therefore, the central concept c_i^c of $\boldsymbol{C}_i^{ca}$ is defined as

$$c_i^c = \{c_i | D(c_i) = max\,(D(c_1), \ldots, D(c_h))\}. \quad (23)$$

5.2.2. Generation of sequence of topic relevant concepts

With c_i^c, its *STRC* is generated from the amalgamation of candidate concepts. During the amalgamation, the similarity between c_i^c and the concepts in *STRC*s of candidate central concepts are recalculated, and its value is given by

$$ISM(c_i^c, c_{i',j}) = max(Sim(c_i^c, c_{i',j}), Sim(c_i^c, c_{i'}^{ca}) \cdot Sim(c_{i'}^{ca}, c_{i',j})). \quad (24)$$

The workflow of the amalgamation is shown in Fig. 3, and the procedure to select a central concept and its *STRC* is summarized in Alg. 2. In the following section, we detail the prefetching technique based on central concepts and *STRC*s.

Algorithm 2: Selection of Central Concept and Generation of *STRC*

Input: $(\boldsymbol{C}_1, \boldsymbol{C}_2, ..., \boldsymbol{C}_k)$ // a collection of concepts
Output: $\boldsymbol{C}^c, \boldsymbol{S}^c$ // a collection of central concepts and their *STRC*s
1 $(\boldsymbol{C}_1, \boldsymbol{C}_2, ..., \boldsymbol{C}_{k_2}) \Leftarrow (\boldsymbol{C}_1, \boldsymbol{C}_2, ..., \boldsymbol{C}_k)$// use Eq. (18) and ε_1;
2 $\boldsymbol{C}^c \Leftarrow \{\emptyset\}, \boldsymbol{S}^c \Leftarrow \{\emptyset\}$ // initialization;
3 **for** $i \leq k_2$ **do**
4 $\quad \boldsymbol{C}_i^{ca} \Leftarrow \boldsymbol{C}_i$ // choose candidate central concepts from $\boldsymbol{C}_i$;
5 $\quad c_i^c \Leftarrow \boldsymbol{C}_i^{ca}$ // choose the central concept from $\boldsymbol{C}_i^{ca}$ using Eq. (23);
6 $\quad \boldsymbol{C}^c \cup c_i^c \Leftarrow c_i^c$ // collect the central concept;
7 $\quad \boldsymbol{S}_i^c \Leftarrow \{\boldsymbol{C}_i^{ca}, c_i^c\}$ // generate the sequences of topic relevant concepts;
8 $\quad \boldsymbol{S}^c \cup \boldsymbol{S}_i^c \Leftarrow \boldsymbol{S}_i^c$ // collect the sequences of topic relevant concepts;
9 **end for**
10 **return** $\boldsymbol{C}^c = \{c_1^c, ..., c_{k_2}^c\}$ *and* $\boldsymbol{S}^c = \{\boldsymbol{S}_1^c, ..., \boldsymbol{S}_{k_2}^c\}$

5.3. Semantic-based resource prefetching

In this section, a semantic-based prefetching technique is presented, which consists of the inference of educational resources to be prefetched, the calculation of prefetching size z and a collection of target datanodes ***TD***, and prefetching management with topic lifecycles as well.

5.3.1. Inference of popular educational resources

Specifically, caching and prefetching are uniformly described in the view of popularity of resources [6], namely our prefetching algorithm also keeps popular cached resources in the target datanodes. User requests within ΔT will be further analyzed from a spatial perspective. Firstly, user requests are partitioned into different groups in accordance with their service districts. Assuming that a central concept of District $a_{i'}$ in some time slot is c_i^c, **T**opic **R**elatedness(TR) between $a_{i'}$ and c_i^c in ΔT is defined as

$$TR(c_i^c, a_{i'}, \Delta T) = \sum_{t=n}^{T} max(\{Sim(c_i^c, c_j) \cdot \frac{SSS(TSeq(c_j))}{m_t}\}) \cdot \beta_t, \quad (25)$$

where m_t is the total amount of user requests within $a_{i'}$, c_j is a central concept in Δt within $a_{i'}$, which similarity with the current central concept c_i^c meets $Sim(c_i^c, c_j) \geq \varepsilon_3$ ($c_i^c = c_T$ when $t = T$), and β_t ($0 \leq \beta \leq 1$) is a coefficient of time decay, the bigger the closer to the current ΔT, and vice versa.

It is important for the inference of popular resources to leverage relation information of concepts in the educational ontology. To reduce computation cost and complexity of the inference, we propose a light-weighted inference approach based on four kinds of semantic relations we defined in the view of an educational ontology, and they are *supclass*, *subclass*, *sibclass*, and *hybclass*.

Supclass is that a concept is a parent node of a central concept, *subclass* is that a concept is a child node of the central concept, *sibclass* is that a concept is a brother node of a central concept, and *hybclass* is that a concept contains two or more the aforementioned relations with a central concept.

Therefore, we develop three rules for the inference of relations of concepts:
Rule I: *SubClassOf* (c_x, c_y), *SubClassOf*(c_y, c_z) → *hasGrandFather*(c_x, c_z).
Rule II: *SubClassOf*(c_x, c_z),*SubClassOf*(c_y, c_z) → *hasSibling*(c_x, c_y).
Rule III: *SubClassOf*(c_x, c_k), *SubClassOf*(c_y, c_z), *hasSibling*(c_k, c_z) → *hasCousin*(c_x, c_y).

Therefore, the process of inference is stated in the following:

- Part I: (1) find out the concepts in $\boldsymbol{S}_i = TSeq(c_i^c)$ that have a direct connection with c_i^c. (2) based on *Rule I* and the educational ontology, find out the concepts that have *supclass* or *subclass* with c_i^c. (3) based on *Rule I*, *Rule II*, and the educational ontology, find out the concepts that have *sibclass* with c_i^c.
- Part II: (1) find out the concepts in $\boldsymbol{S}_i = TSeq(c_i^c)$ that have a indirect connection with c_i^c. (2) choose a certain amount of the indirect concepts in terms of the similarity with (c_i^c). (3) based on *Rule I*, *Rule II*, *Rule III* and the educational ontology, find out the concepts that have *sibclass* with the chosen concepts.

With the output concepts (denoted as $\boldsymbol{S}_I$) from the above inference, the sequential order of them is another critical problem to be solved. We propose a sorting method for such a concept collection, which takes both the relatedness and intensity of a topic within the current time slot into consideration, and the process is shown as follows:

- Part I: (1) take a central concept as the origin point, and then label parent nodes as the *1*st layer, the parent nodes of parent nodes as the *2*nd layer, and so on. (2) label child nodes as the *-1*st layer, the child nodes of child nodes as the *-2*nd layer, and so on.
- Part II: (1) assume that the top layer of c_i^c with $\boldsymbol{S}_i^c$ is the lth layer, and the lowest layer is the l'th layer. (2) calculate the total similarity of each layer with c_i^c, denoted as $Sim(c_i^c, f), l' \leq f \leq l$.
- Part III: (1) calculate the relatedness of $c_{v,j}$ (the jth concept of the vth layer, included the concept in $\boldsymbol{S}_i^c$ and the output

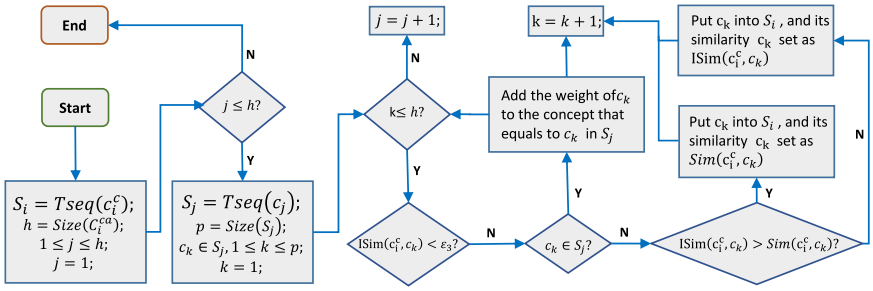

Fig. 3. The amalgamation of *STRC*s for the central concept c_i^c.

of the above inference), which is calculated by

$$P(c_i^c, c_{v,j}) = \frac{Sim(c_i^c, v)}{\sum_{l}^{f=l'} Sim(c_i^c, f)} \cdot Sim(c_x, c_{v,j}) \cdot H(c_i^c, a_{i'}, \Delta T). \tag{26}$$

- Part IV: (1) merge the same concepts in $\boldsymbol{S}_I$ and add up their relatedness. (2) sort the concepts in the merged $\boldsymbol{S}_I$ with their relatedness and denote the output as $\boldsymbol{S}^c$.

There are several search engines for different knowledge representation schemes, such as taxonomies, ontologies, thesauri, graphs, mind maps, among others [45]. Ontology techniques provide us a better choice for the design of search engine and good compatibility with the inference approach, and besides some meta data mentioned in [45], concepts associated with learning objects are also included in meta data. $\boldsymbol{S}^c$ is mapped the annotated resources of education in the cloud storage system through semantic search algorithms [46]. After removing the existed resources in target datanodes, the educational resources to be prefetched $\hat{\boldsymbol{\mathcal{S}}}$ therefore are obtained.

5.3.2. Calculation of window size and target datanodes

The popularity of resources is not the only factor for prefetching, but also the workload of datanodes, such their bandwidth, capacity of *CPU*, and workload of *I/O*. The relative capacity is used to indicate the ability to provide services (such as data transmission, storage) between Datanode i and j, and it is given by Eq. (27).

$$Load(i, j) = P_{i,j}^{BW} \cdot \lambda_{BW} + (1 - P_j^{CPU}) \cdot \lambda_{CPU} + (1 - P_j^{I/O}) \cdot \lambda_{I/O}. \tag{27}$$

where P^{BW} is a theoretical bandwidth between Datanode i and Datanode j, P_j^{CPU} is the percentage of usage of *CPU* of Datanode j, P_j^{CPU} is the percentage of usage of *I/O* of Datanode j, and $\lambda_{BW}, \lambda_{CPU}, \lambda_{I/O}$ are coefficients to lower the implication caused by different metrics.

As shown in Fig. 4, if the user requests from District $a_{i'}$ cause a certain temporal locality, the system chooses the datanodes which bandwidth with clients from the district are bigger than a certain threshold value to form the **C**andidate **S**et ($\boldsymbol{CS}$= $\{b_i \mid 1 \le i \le p, i, p \in \mathbb{N}\}$) of target datanodes to store the prefetched resources, and the datanodes that store the replica of the education resources to be prefetched is the **R**eplica **S**et ($\boldsymbol{RS}$= $\{d_i \mid 1 \le j \le q, j, q \in \mathbb{N}\}$), $\boldsymbol{CS} \cap \boldsymbol{RS} = \emptyset$. The workload between Datanode b_i and Datanode d_i is expressed as $Load(b_i, d_i)$, while $Load(b_i, a_{i'})$ denotes the mean workload between b_i and clients from $a_{i'}$. The default replica factor of the system is three (namely $p = 3$), and the workload between $\boldsymbol{CS}$ and $\boldsymbol{RS}$ is expressed as

$$Load(\boldsymbol{CS}, \boldsymbol{RS}) = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \otimes \begin{pmatrix} b_1 & b_2 & \dots & b_p \end{pmatrix} \tag{28}$$

$$= \left(Load(d_i, b_j)\right)_{3\times p}, \tag{29}$$

where the operator $\otimes$ means the workload between datanodes, and the workload between $\boldsymbol{CS}$ and $a_{i'}$ is denoted as

$$Load(\boldsymbol{CS}, a_{i'}) = \begin{pmatrix} b_1 \\ \dots \\ b_p \end{pmatrix} \otimes a_{i'} \tag{30}$$

$$= (Load(b_i, a_{i'}))_{p\times 1}. \tag{31}$$

The total workload between $a_{i'}$ and d_j is expressed as

$$Load(a_{i'}, d_j) \mid b_i = Load(a_{i'}, b_i) + Load(b_i, d_j), \tag{32}$$

where b_i is a possible target datanode to store prefetching resources, then $\boldsymbol{TD}$ is solved out by Eq. (33).

$$\begin{aligned} \boldsymbol{TD} = \{b_i \mid \{&Load(a_{i'}, d_j) \mid b_i \\ &= max\{Load(a_{i'}, d_j) \mid b_1, \dots, Load(a_{i'}, d_j) \mid b_p\}\}\}. \end{aligned} \tag{33}$$

Not all the educational resources in $\boldsymbol{\mathcal{S}}$ are necessarily to be prefetched to $\boldsymbol{TD}$, and the window size of prefetching is determined by the topic relatedness of data requests and the workload of the running system, which is given by

$$z = \left(\frac{\sum_{i=1}^{n} TR(c_i^c, a_{i'}, \Delta T)}{n \cdot \sum_{t=1}^{T} \beta_t} \alpha + \frac{Load(a_{i'}, d_j) | b_{i'}^s}{2(\lambda_{BW} + \lambda_{CPU} + \lambda_{I/O})} \right) \cdot Size(\hat{\boldsymbol{\mathcal{S}}}), \tag{34}$$

where $\hat{\mathcal{S}}$ is the data size of $\hat{\mathcal{S}}$, c_i^c is the central concept of the time slot Δt (n central concepts), β_t is the same decay factor as in Eq. (25), α and γ are coefficients to control the influence brought by the popularity of concepts relevant to central concepts and the workload of the system, and $\alpha + \gamma \le 1$.

The system sequentially prefetches popular resources with the order of $\hat{\mathcal{S}}$, and the prefetching process will be halted if the size of the prefetched exceeds the window size given by Eq. (34).

5.3.3. Prefetching management with topic lifecycles

A topic in user requests usually experiences three states (retention, transition, fading) after its generation in a lifecycle of topic:

1. Retention: a topic also exists in the previous time segment.
2. Transition: Topics in two connected time segments share a certain similarity but with different central concepts.
3. Fading: a topic has been taken place in the previous time slots but fades in the following time slots.

The prefetching strategies for each state of topics are shown in the following.

- Retention: if the prefetching has taken for the topic, the inference of prefetching concepts would cover more indirect concepts based on the workload of the system.
- Transition: if the similarity between the central concept in Δt and the central concept in $\Delta(t-1)$ is bigger than ε_3 and

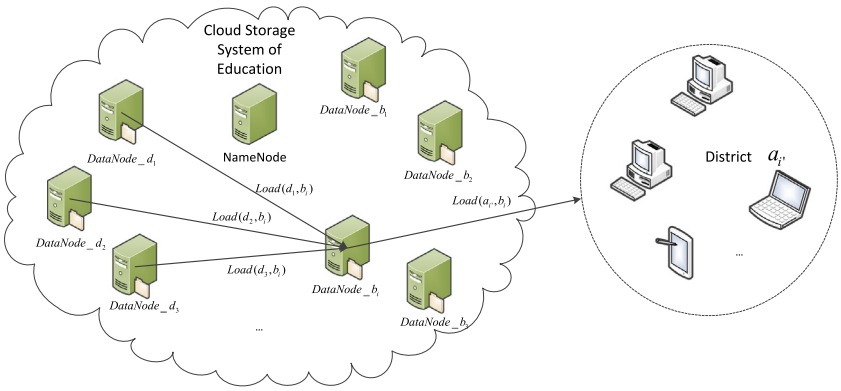

Fig. 4. The selection of target datanodes.

less than 1, the transition has taken place between central concepts, and the system would prefetch popular resources based on the $\mathcal{S}$ in Δt.

- Fading: if the central concept in Δ_t have a lower similarity (less than ε_3) with the central concepts in $\Delta t-1$ and $\Delta(t-2)$, the topic in $\Delta(t-1)$ and $\Delta(t-2)$ may be fading. The system would prefetch popular resources of the $\mathcal{S}$ in Δ_t firstly, and prefetch the popular resources of the previous time slot when the system stays idle. If the fading continues in $\Delta(t+1)$, the system would stop the prefetching of educational resources for the faded central concept, and remove the part of prefetched resources when there is not enough storage space in the target datanodes. Otherwise, the system would prefetch the $\mathcal{S}$ in $\Delta(t+1)$ firstly, and then resume the prefetching of the $\mathcal{S}$ in $\Delta(t-1)$.

Prefetching management incorporated with lifecycles of topics is summarized in Alg. 3.

Algorithm 3: Prefetching Management with Lifecycles of Topics

Input: Topics in different time slots
Output: Operations of prefetching
if *Can the system resources allow to perform prefetching* **then**
  Find out the topics in Δt;
  if *Does there exist the same topics in the previous time slots* **then**
    Cover more indirect concepts in the inference; ;
  end if
  if *Does there exist the similarity topics between* Δt *and* $\Delta(t-1)$ **then**
    if *Does there exist the similarity topics between* $\Delta(t-1)$ *and* $\Delta(t-2)$ **then**
      else
        if *Does there exist the similarity topics between* Δt *and* $\Delta(t-2)$ **then**
          Set the state of the prefetching for the topics before $\Delta(t-2)$ enable;
        else
          Set the state of the prefetched resources for the faded topics in $\Delta(t-2)$ removable;
        end if
      end if
  else
    if *Does there exist the similarity topics between* Δt *and* $\Delta(t-2)$ **then**
      Suspend the prefetching for the topics before $\Delta(t-1)$;
    else
      Suspend the prefetching for the topics in $\Delta(t-1)$;
    end if
  end if
  Prefetch the educational resources for the central concepts in Δt;
end if

The related algorithms for the prefetching of educational resources, based on a spatial–temporal locality in user requests and an adaptive semantic inference approach, have fully presented in the above several sections, and the workflow of semantic-based prefetching for educational cloud storage systems is summarized in Fig. 5. We report the experiment results of the proposed techniques in the next section.

6. Performance evaluation

This section reports the results from both simulation experiments and a real-world case study on an educational application named *WorldUC* [47].

6.0.4. Evaluation metrics

In our experiments, we use several metrics to evaluate the performance of the proposed techniques. Some complicated metrics are described below in detail.

Normalized Mutual Info(NMI) and **Adjusted Rand Score (ARS)** [48]: to evaluate the performance of clustering algorithms, two scoring metrics are employed, the normalized mutual info score (NMI) and the adjusted rand score (ARS). The scores of them ranges between 0 and 1, with 1 representing perfect cluster prediction and 0 representing largely independent cluster prediction from the true clusters.

AVERAGE HIT RATIO(AHR): Hit ratio indicates how many data requests are responded by targeted datanodes. *AHR* at time t is calculated by taking the average of all users' hit ratio, by:

$$AHR = \frac{1}{U}\sum_{i=1}^{N}\frac{q_i(t)}{q_i(t)+p_i(t)}, \tag{35}$$

where U is the number of users, $q_i(t)$ and $p_i(t)$ denotes the number of requests responded by targeted datanodes and remote servers at time t, respectively.

BYTE HIT RATIO(*BHR*): besides *AHR*, *BHR* is also to estimate the performance of prefetching or caching algorithms, which is calculated by:

$$BHR = \frac{\sum_{i\in\mathbb{N}} B_{pre}(i)}{\sum_{i\in\mathbb{M}} B_{total}(i)} \tag{36}$$

where $B_{pre}(i)$ is the data size of user i responded from prefetching servers, and $B_{total}(i)$.

AVERAGE DELIVERY LATENCY(ADL): Delivery latency means the how long that users can obtain their demand resource. Accordingly, the *ADL* is calculated by the average of all users' transmission delivery latency during the experiment, by

$$ADL = \frac{1}{U}\sum_{i=1}^{U}\sum_{v_j\in H_i(t)}\tau_{v_j}^{i}, \tag{37}$$

where $\tau_{v_j}^{i}$ is accessing latency of v_j by user i.

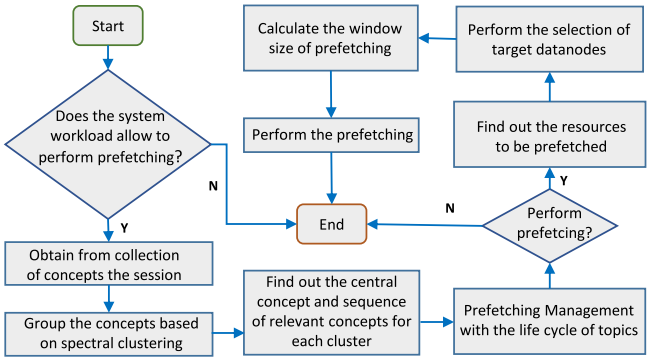

Fig. 5. Prefetching management with lifecycles to topics.

6.1. Baseline methods

In the educational topic detection part, the topic model is built on graph structural clustering techniques, and interconnectivity-based clustering algorithms (e.g. spectral clustering) achieve a better performance compared with other distance-based clustering algorithms (e.g. kNN) in the graph structural clustering [49]. To show that the proposed algorithm (denoted as *SC-SM*) can achieve a satisfactory performance within a reasonable usage of time and memory, we compare *SC-SM* with other three state-of-the-art graph structural clustering algorithms in terms of time consumption, memory usage, sampling performance, and accuracy. The compared algorithms are briefly described in the following:

- **SC-VEC** [50] : a spectral clustering algorithm with vector-form concepts as data points [51], the dimension of vector-form concept is 200 in the experiments;
- **SC-NC** [50]: a spectral clustering algorithm with Eq. (4) as a similarity function to generate affinity matrix;
- **SC-WK** [44]: a similar accelerated spectral clustering algorithm with random sampling subspace, and Eq. (4) also is used to form the affinity matrix in the experiments.

Integrating caching techniques with prefetching algorithms can significantly reduce latency of remote access [6]. Statistical techniques [25] and machine learning methods [52,53] are usually employed to realize a weighting function for file replacement. Therefore, we compare the proposed algorithm (denoted as **STLP**) with other typical statistical and machine learning based algorithms, and they briefly described in the following:

- **AGG (PBM)** [25]: divide the space into prefetching buffer and cache buffer, and perform document replacement and prefetching based on a weight function.
- **mART1 (LRU)** [52]: adapt a least recently used algorithm for caching replacement, and use modified ART1 for clustering based prefetching.
- **pART2(PBR)** [53]: introduce probability-based policy for caching replacement, and use adaptive resonance theory (ART) to design a prefetching model.

6.2. Simulation evaluation

6.2.1. Simulation settings

In this part, we use keyword collections collected from simulate queries and requests to evaluate the performance of our algorithms. The topics of learning activities are chosen from records in an online learning platform named *Moodle*,[3] and the learning process of learners is modeled by the learning paths of users on an e-learning cloud space described in the work [54]. The user queries or data requests are generated with a probability $\frac{b}{a}\left(\frac{i}{a}\right)^{b-1} e^{-i/a^b}$ during the learning process [55], where $a = 286, 152$ and $b = 0.53$ in this simulation. With these settings, we generate 5 simulate datasets as shown in Table 3, the number of clusters is to evaluate the performance of the algorithm proposed in Section 5.1.3, and the number of topics is to evaluate the performance of the algorithm proposed in Section 5.1.2.

The clustering algorithms run on the *DEll* workstation with the specification as Table 2 shown. In particular, most of the compared algorithms are used or modified from the widely-used python machine learning package scikit-learn [56].

Table 2
Specification of workstation.

Brand	CPU	Memory size	Operating system	Software
Dell	Intel Xeon E5 3.60 GHz	18 GB	Ubuntu 16.04	Python 3.6

Table 3
Data characteristics simulate datasets.

Datasets	Num. of keywords	Num. of clusters	Num. of topics
SDataset1	5 683	7	4
SDataset3	9 298	13	7
SDataset4	56 535	18	10
SDataset5	150 072	23	12

6.2.2. NMI of clustering algorithms

We ran the algorithms for 15 times and the average *NMI* of the statics results are in shown Table 4. Both *SC-VEC* and *SC-NC* could run on the small-scale dataset *Sdataset1* and *Sdataset2* by using an eigen-decomposition method to perform clustering, but failed on the much larger dataset *Sdataset3* and *Sdataset4*. Because the space complexity of *SC-VEC* and *SC-NC* is $O(n^2)$, there are simply about 23.8 GB to run the algorithms when an element (double float type) of the feature matrix for Sdataset3 needs 8 bytes to store, which would lead to the memory out of usage. As for *SC-WK* and *SC-SM*, they can still run on much larger datasets with limited memory for their sampling-based design, and the advantage of this kind of algorithms in memory usage is well depicted in Fig. 6. From the results, *SC-NC* outperforms *SC-VEC* in both two datasets, a reason for this is that it may be the information loss causing by a vectorizing of concepts. Thus, using a concept similarity to generate affinity matrix is better choice in this context. With the number of sampling points increasing, the clustering results of *SC-WK* and *SC-SM* also get better, while *SC-SM* achieves the same performance with less sampling points.

[3] http://moodle.scnu.edu.cn/.

Table 4
NMI of clustering algorithms on simulation datasets.

Algorithm	Num. of sampling	Simulation datasets			
		SDataset1	SDataset2	SDataset3	SDataset4
SC-VEC	100%	0.6158 (±0.0032)	0.5818 (±0.0036)	–	–
SC-NC	100%	**0.7712** (± 0.0065)	**0.7252** (± 0.0078)	–	–
SC-WK	200	0.5374 (±0.0029)	0.4739 (±0.0055)	0.3639 (±0.0025)	0.3082 (±0.0017)
	500	0.5971 (±0.0052)	0.5201 (±0.0042)	0.3966 (±0.0045)	0.3536 (±0.0081)
	1000	0.6394 (±0.0069)	0.5458 (±0.0015)	0.4223 (±0.0052)	0.3675 (±0.0021)
	2000	0.6874 (±0.0029)	0.5602 (±0.0031)	0.4652 (±0.0039)	03922 (±0.0057)
	3000	0.7169 (±0.0017)	0.6178 (±0.0030)	0.5005 (±0.0041)	0.4774 (±0.0021)
SC-SM (The proposed)	200	0.5774 (±0.0012)	0.5039 (±0.0065)	0.4139 (±0.0051)	0.3551 (±0.0029)
	500	0.6314 ± 0.0064)	0.5419 (±0.0038)	0.4682 (±0.0010)	0.3828 (±0.0035)
	1000	0.6874 (±0.0022)	0.5873 (±0.0076)	0.5164 (±0.0007)	0.4575 (±0.0022)
	2000	0.7072 (±0.0023)	0.6137 (±0.0012)	0.5749 (±0.0053)	0.5004 (±0.0045)
	3000	0.7554 (±0.0073)	0.6678 (±0.0074)	**0.5964** (±0.0063)	**0.5317** (±0.0029)

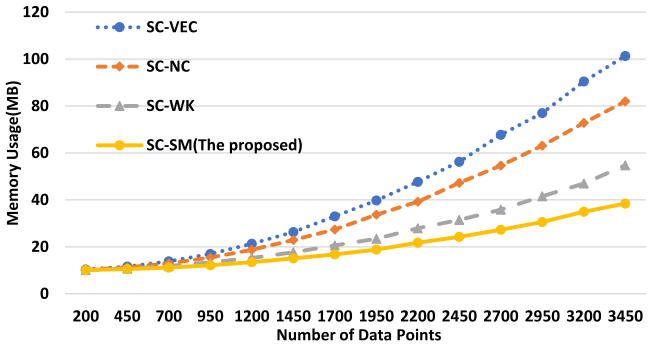

Fig. 6. The memory usage of each clustering algorithm to reach the same NMI.

6.2.3. ARS of clustering algorithms

We calculate *ARS* of the clustering algorithms while varying different datasets, and numbers of data points in the experiment, and the results is shown in Fig. 7. Fig. 7(a) and (b) show the results when the number of data points was varied. *SC-NC* achieved the highest score with complete data points involved on Sdataset1 and Sdataset2, but *SC-VEC* got a poor performance even with much more data points than *SC-WK* and *SC-SM*. Both *SC-WK* and *SC-SM* gained a reasonable performance as the number of sampling data points increased, but *SC-SM* was better. In Fig. 7(c) and (d), as the number of data points increased, all algorithms considered showed an increase in score. However, *SC-SM* consistently out-performed *SC-WK* in the varied numbers of sampling data points considered. Therefore, *SC-SM* is a better choice when dealing with large-scale datasets, which provides a reasonable performance with much less sampling data points.

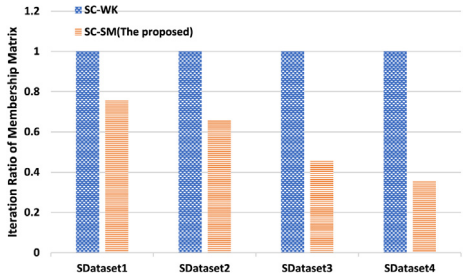

Fig. 8. The memory usage of each clustering algorithm to reach the same NMI. *SC-SM* lowers the value of iteration by applying an optimized time-stochastic sampling function compared with *SC-WK*.

6.2.4. Time cost of clustering algorithms

The time cost is an important metric for a practical algorithm. We collect the time cost information during the experiments, and the statistical results are shown in Table 5. From the results, *SC-VEC* and *SC-NC* cost much more time to perform the clustering, because they need to perform the eigen-decomposition, which is a time-consuming process. As for *SC-WK* and *SC-SM*, they both use a weighted k-means technique to optimize the objective function by leveraging a sampling process to approximate the solution, which avoids a time-consuming process of eigen-decomposition and achieves a satisfactory performance with less time cost. *SC-SM* outperforms *SC-WK* with the same sampling points while less time cost consistently. The reason is that the time cost of the clustering mainly depends on the number of sampling and the iteration of the algorithms. The sampling processing of *SC-SM* is optimized by a time-stochastic function, while *SC-WK* just uses a simply randomized process to sample the dataset. Fig. 8 shows the advantage of the optimized sampling process.

6.2.5. Accuracy of topic selection

We have compared the relative performance of our *SC-SM* with regards to other algorithms in terms of their **ACC**uracy (**ACC**) and **DIS**tance to central topics (**DIS**). The distance of central topics in the experiments is defined as the average semantic difference of between the predicted central topic and the actual central topic. Table 6 shows the results when the datasets and numbers of sampling points were varied. From the results, the *ACC* and *DIS* of the considered algorithms are positive relative to the quality of clustering results. A better accuracy of the clustering process often results in a better selection of a central topic in a cluster.

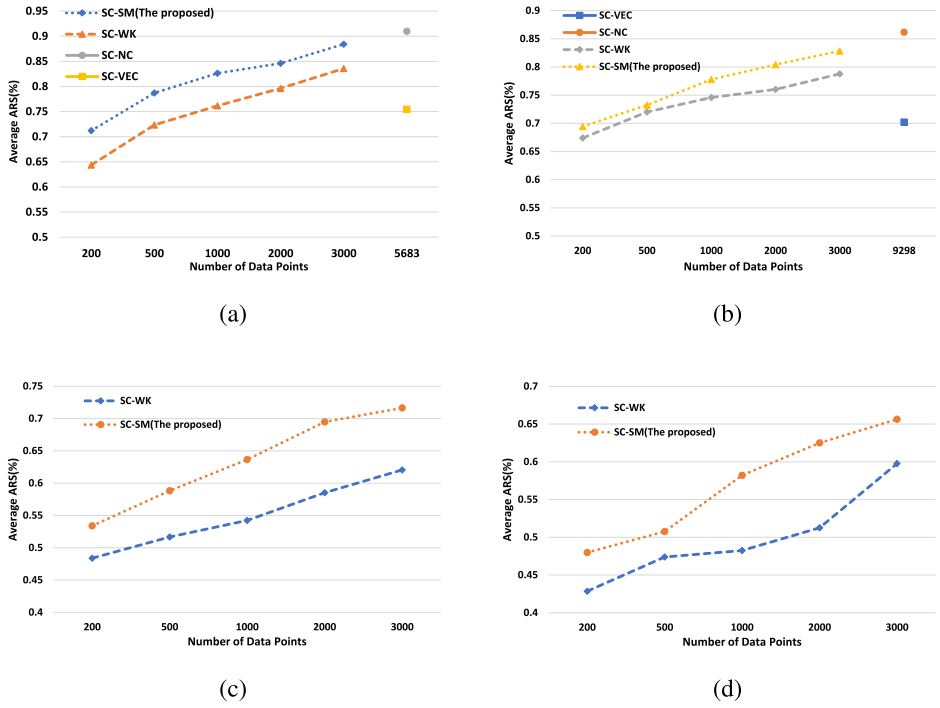

Fig. 7. The average of the adjusted rand score (ARS) for various values of the number of data points.

Table 6
Accuracy of topic selection on simulation datasets.

Algorithm	Num. of sampling	Simulation datasets							
		SDataset1		SDataset2		SDataset3		SDataset4	
		ACC	DIS	ACC	DIS	ACC	DIS	ACC	DIS
SC-VEC	100%	0.7434	0.1401	0.7244	0.1629	–	–	–	–
SC-NC	100%	**0.8643**	**0.1206**	**0.8189**	**0.1371**	–	–	–	–
SC-WK	1000	0.7325	0.1496	0.6395	0.1729	0.5160	0.1976	0.4912	0.2026
	2000	0.7805	0.1378	0.6539	0.1701	0.5589	0.1891	0.4559	0.1992
	3000	0.8100	0.1304	0.6915	0.1605	0.5942	0.1802	0.5711	0.1866
SC-SM (The proposed)	1000	0.7805	0.1453	0.6810	0.1646	0.6101	0.17886	0.5512	0.1904
	2000	0.8003	0.1384	0.7074	0.1594	0.6686	0.1671	0.6141	0. 1820
	3000	0.8485	0.1266	0.7514	0.1506	**0.7101**	**0.1628**	**0.6654**	**0.1758**

Table 5
Time cost of clustering algorithms on simulation datasets.

Num. of sampling	Algorithm	SDataset1	SDataset2	SDataset3	SDataset4
100%	SC-VEC	205.467	407.893	–	–
	SC-NC	**166.356**	**332.807**	–	–
200	SC-WK	2.168	2.745	14.742	24.367
	SC-SM[a]	**1.113**	**1.665**	**4.909**	**16.206**
500	SC-WK	2.1684	3.478	24.3673	51.583
	SC-SM[a]	**1.600**	**1.599**	**3.265**	**25.167**
1000	SC-WK	1.7940	4.929	54.288	131.254
	SC-SM[a]	**2.443**	**2.353**	**17.096**	**46.516**
2000	SC-WK	5.3040	8.595	85.379	324.358
	SC-SM[a]	**3.330**	**5.595**	**32.792**	**84.369**
3000	SC-WK	10.7233	24.819	111.211	471.211
	SC-SM[a]	**5.613**	**8.976**	**47.011**	**193.80**

[a]The proposed.

SC-SM relative to the considered algorithms achieves a better with less sampling data points and more reasonable time cost. Therefore, *SC-SM* is an effective solution to construct a topic model from these large-scale datasets.

6.3. Real-world case study

6.3.1. Study settings

In this part, we evaluate the performance of our algorithms in a real-world educational application named *WorldUC*[4] with more than seven million users across China, which provides users with rich-media educational resources ranging from primary schools to universities.

Users of *WorldUC* from 15 provinces are involved in the case study (shown in Fig. 13), the datanode setting is shown in Table 7, and the datanodes are equipped with the same specification. Both outgoing and incoming packets through the gateway router was captured. For each data accessed, the arrival time, keywords, resource size, source and destination IP addresses and port numbers were recorded for perform the prefetching analysis. Specifically, the data storage of datanodes for caching or prefetching are uniformly called as cache space in the following, we did not distinct the caching space from the prefetching space for the statistic of results in the experiments.

4 http://www.worlduc.com/.

Table 7
The setting of geo-distributed datanodes for the case study.

Province	City	Num. of datanodes	Province	City	Num. of datanodes
Guangdong	Guangzhou	5	Hunan	Chashan	7
Zhejian	Jinhua	6	Shangdong	Qingdao	3
Beijing	Beijing	4	Yunan	Kunming	2

6.3.2. NMI and ARS of clustering algorithms

Unlike the simulation datasets, it is inherently difficult to trace back to the actual central concept from real-world datasets, but we have learned that the *ACC* and *DIS* of topics are positively relevant to the performance of clustering algorithms from the above simulation results. Therefore, we carried out the experiments on real-world datasets formed from large-scale logs of users' requests or queries to evaluate the performance of *SC-WK* and *SC-SM* (*SC-VEC* and *SC-NC* fail to handle these datasets) in terms of NMI and ARS. There are 4 datasets collected from 4 service districts from different provinces, and there are about two thousand data points in each dataset. The statistical results is depicted in Fig. 9, both *SC-WK* and *SC-SM* achieved a better performance as the number of sampling data points increased, but *SC-SM* consistently outperformed *SC-WK* on 4 datasets. Thus, the advantage of the proposed algorithm still maintains in real-world datasets.

6.3.3. AHR of prefetching algorithms

Fig. 10(a) plots the overall hit ratio with for a service district with$T = 1000$ time slots, and *STLP* is outperforms the other considered algorithms. The overall hit ratio with cache size M files is defined as$\sum_{t=1}^{T} H_t / \sum_{t=1}^{T} \sum_{j=1}^{N} d_{t,j}$. Particularly, the performance of *STLP* achieved 34.27% improvement against *AGG* and 21.03% against *mART1*, 12.55% against *pART2* when the cache size is 10 files. This is due to *STLP* continuously not only learns the resource popularity but also avoiding the unstable replacement.

Fig. 10(b) plots the average hit ratio in terms of iteration times t for a service district with $M = 600$ files. The point of a curve in the figure means the *AHR* within the time window between itself and the next point. The *AHR* achieved by the considered algorithms shown in the figure is decreasing. This is due to the fact that the number of resource requests for each time slot increases when the resource library increases over time. The performance of the proposed gets better over time, which shows that *STLP* is able to identify the *most popular* resources instead of moderately *popular* resources.

6.3.4. Utilization rate of prefetching algorithms

A better utilization of limited storage space of datanodes is an important metric to estimate the performance of prefetching algorithms. Fig. 11 shows the request hit rate (*HR*) and *BHR* of three algorithms evaluated with varied percentages of capacity, where the percentage of capacity is the ratio between the size of cached resources and the size of recommended resources. From Fig. 11(a) and (b), it is seen that the *HR* and *BHR* of three algorithms increase when the size of cached resources increases, and when the size of cached resources exceeds 50%, the increase of *HR* and *BHR* becomes slow. Compared with the other considered algorithms, *STLP* achieved about 28% improvement in *HR*, and 36% in *BHR*. It is that *STLP* employs an educational topic model to explore spatial–temporal locality to estimate the popularity of educational resources, not just only statistically analyzing history logs.

6.3.5. ADL of prefetching algorithms

Fig. 12 shows the improvement of *STLP* against *AGG*, *pART2* and *mART1* with respect to the average request response time for two service districts with local datanodes under varied cache sizes. Similar to the results for the caching utilization, *STLP* achieved a better performance with the considered algorithms. *STLP*'s improvement over the average request response time compared with *pART2* and *AGG* decreases as cache size increases. As for *mART1*, *STLP*'s advantage is highest for cache size around 100 GB. This is because *mART1* also captures some degree of temporal locality but fails to capture sufficient temporal locality without a large cache size. For a 200 GB cache, the improvement over the average request response time for a service district in Guangdong is 37% for *AGG*, 20% for *pART2*, 27% for *mART1*. When the size of cached resources exceeds 700 GB, the improvement becomes slow, and this is because the cached files cover most of resources in the resource library in the case study. Fig. 13 presents the overall statistical results of the improvement of *STLP* against *pART2* in 15 provinces with respect to the average response time.

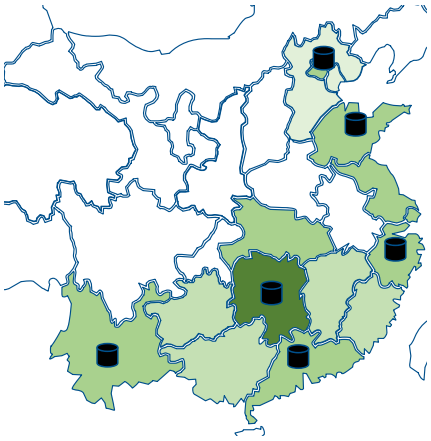

Fig. 13. Six geo-distributed datanode groups across China measured in the experiment. The color of each province indicates the amount of improvement of *STLP* as compared to *pART2* when we placed a replica at the largest city in the provinces; darker colors corresponding to higher improvements.

7. Conclusion

It is important to develop an effective way to estimate the resource popularity for enhancing the performance of prefetching, but most of the previous research have not taken domain knowledge into account and resulted in an unsatisfactory performance on the resource popularity estimation. The paper explores the idea of modeling the resource popularity estimation by taking advantages from spatial–temporal locality in education domain and incorporates resource prefetching management with such locality to improve the efficiency of data access for end-users. A novel topic model is proposed to trace educational topics for the resultant spatial–temporal locality, and its practical construction method, based on an accelerated spectral algorithm and an ontology concept similarity metric, is also presented in the paper. With the proposed model, a light-weighted semantic inference approach is proposed to predict popular educational resources in the future data requests. Moreover, a systematic and efficient prefetching management mechanism working with replica

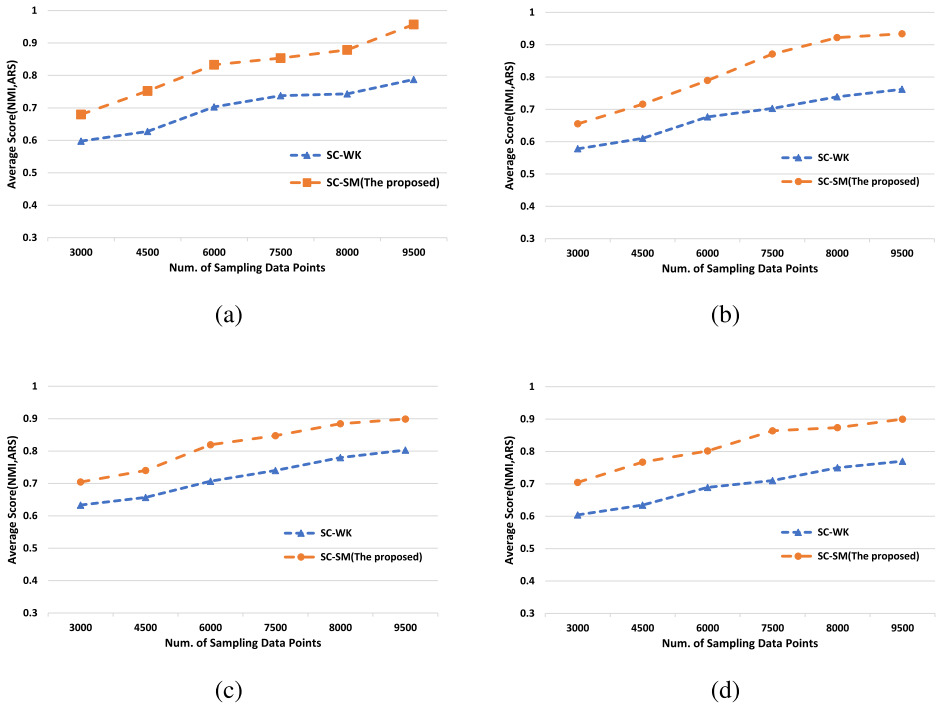

Fig. 9. The average of the normalized mutual info score (NMI) and adjusted rand score (ARS) for various values of the number of sampling data points.

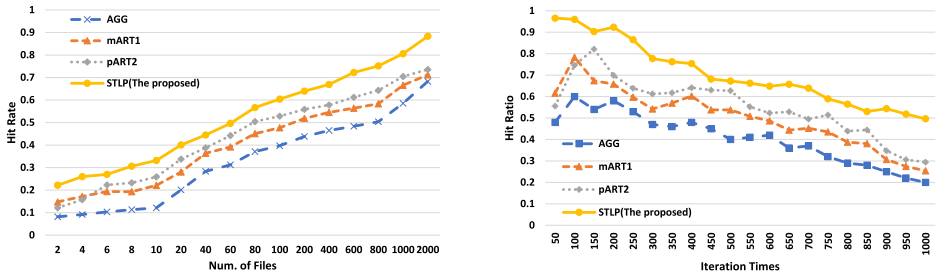

Fig. 10. Average hit ratio of caching algorithms.

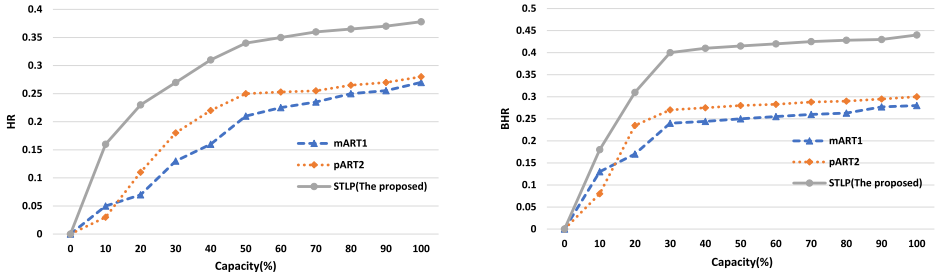

Fig. 11. The HR and BHR of three algorithms.

techniques is suggested to support geo-dispersed cloud storage systems for educational applications. Results from simulation experiments and a real-world case study show that the proposed techniques achieve a better performance compared with state-of-the-art approaches. There involve many parameter settings in the proposed techniques, which may cause a poor performance with inappropriate values of parameters. Therefore, we will focus on developing heuristic or automatic parameter setting algorithm to optimize the parameter setting problem in the future work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61877020), the Science and Technology Projects of Guangdong Province, China (No. 2015A030401087,

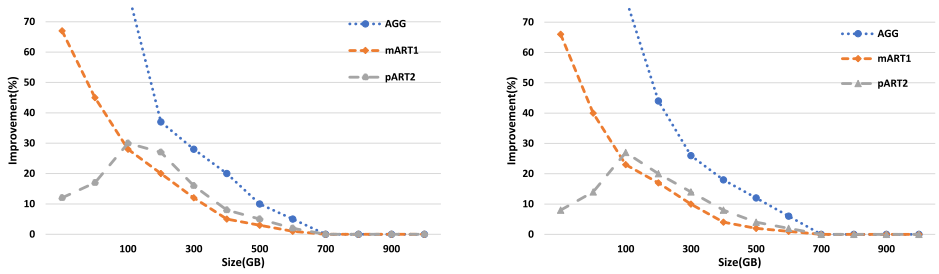

Fig. 12. Performance comparison in terms of *STLP*'s improvement over *pART2* on WorldUC traces, for caches located in Guangdong and Hunan, and for cache sizes varying between 1 GB and 1 TB.

2018B010109002), the Science and Technology Project of Guangzhou Municipality, China (No. 201904010393), and the *Golden Seed Project* of *Challenge Cup* of Extra-Curricular Academic Competition Works by South China Normal University, China (No. 19JXKC01).

References

[1] J.A. González-Martínez, M.L. Bote-Lorenzo, E. Gómez-Sánchez, R. Cano-Parra, Cloud computing and education: A state-of-the-art survey, Comput. Educ. 80 (2015) 132–151.
[2] A.C. Caminero, S. Ros, R. Hernández, A. Robles-Gómez, L. Tobarra, P.J.T. Granjo, VirTUal Remote laboratories management system (TUTORES): Using cloud computing to acquire university practical skills, IEEE Trans. Learn. Technol. 9 (2) (2016) 133–145.
[3] M.T. Baldassarre, D. Caivano, G. Dimauro, E. Gentile, G. Visaggio, Cloud computing for education: A systematic mapping study, IEEE Trans. Educ. 61 (3) (2018) 234–244.
[4] X. Ren, P. London, J. Ziani, A. Wierman, Datum: Managing data purchasing and data placement in a geo-distributed data market, IEEE/ACM Trans. Netw. 26 (2) (2018) 893–905.
[5] A. El Mhouti, M. Erradi, A. Nasseh, Using cloud computing services in e-learning process: Benefits and challenges, Educ. Inf. Technol. 23 (2) (2018) 893–909.
[6] W. Ali, S.M. Shamsuddin, A.S. Ismail, et al., A survey of web caching and prefetching, Int. J. Adv. Soft Comput. Appl. 3 (1) (2011) 18–44.
[7] T.X. Tran, D.V. Le, G. Yue, D. Pompili, Cooperative hierarchical caching and request scheduling in a cloud radio access network, IEEE Trans. Mob. Comput. 17 (12) (2018) 2729–2743.
[8] H.-C. Kim, D. Lee, K. Chon, B. Jang, T. Kwon, Y. Choi, Performance impact of large file transfer on web proxy caching: A case study in a high bandwidth campus network environment, J. Commun. Netw. 12 (1) (2010) 52–66.
[9] S.M. Bhandarkar, L. Ramaswamy, H.K. Devulapally, Collaborative caching for efficient dissemination of personalized video streams in resource constrained environments, Multimedia Syst. 20 (1) (2014) 1–23.
[10] Y. Xu, S.E. Elayoubi, E. Altman, R. El-Azouzi, Y. Yu, Flow-level QoE of video streaming in wireless networks, IEEE Trans. Mob. Comput. 15 (11) (2016) 2762–2780.
[11] B. Bharath, K.G. Nagananda, H. Poor, A learning-based approach to caching in heterogenous small cell networks, IEEE Trans. Commun. 64 (4) (2016) 1674–1686.
[12] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, X. Shen, Content popularity prediction towards location-aware mobile edge Caching, IEEE Trans. Multimed. 21 (4) (2019) 915–929.
[13] P.J. Muñoz-Merino, J.A. Ruipérez-Valiente, C. Alario-Hoyos, M. Pérez-Sanagustín, C.D. Kloos, Precise effectiveness strategy for analyzing the effectiveness of students with educational resources and activities in MOOCs, Comput. Hum. Behav. 47 (2015) 108–118.
[14] E. Cadme, R. Elizalde, M.C. Cabrera-Loayza, N. Piedra, Discovering interaction patterns on the use of OERs in open online courses through the application of association rules, in: Proceedings of the 5th Learning with MOOCS (LWMOOCS 2018), IEEE, 2018, pp. 140–143.
[15] F. Chinchilla, M. Lindsey, M. Papadopouli, Analysis of wireless information locality and association patterns in a campus, in: Proceedings of the International Conference on Computer Communications (INFOCOM 2004), Vol. 2, IEEE, 2004, pp. 906–917.
[16] B.J. Evans, R.B. Baker, T.S. Dee, Persistence patterns in massive open online courses (MOOCs), J. High. Educ. 87 (2) (2016) 206–242.
[17] J.-J. Lo, C.-J. Chang, H.-H. Tu, S.-W. Yeh, Applying GIS to develop a web-based spatial-person-temporal history educational system, Comput. Educ. 53 (1) (2009) 155–168.
[18] N. Li, H. Verma, A. Skevi, G. Zufferey, J. Blom, P. Dillenbourg, Watching MOOCs together: Investigating co-located MOOC study groups, Distance Educ. 35 (2) (2014) 217–233.
[19] S.A. Makhlouf, B. Yagoubi, Data-aware scheduling strategy for scientific workflow applications in iaas cloud computing, Int. J. Interact. Multimedia Artif. Intell. 5 (4) (2019) 75–85.
[20] C.Q. Huang, Y. Li, Y. Tang, R.H. Huang, A research on replica management of cloud storage system for educational resources, J. Beijing Univ. Posts Telecommun. 36 (2) (2013) 93–97.
[21] K. Grabczewski, N. Jankowski, Saving time and memory in computational intelligence system with machine unification and task spooling, Knowl.-Based Syst. 24 (5) (2011) 570–588.
[22] M. Bakhshalipour, P. Lotfi-Kamran, H. Sarbazi-Azad, An efficient temporal data prefetcher for L1 Caches, IEEE Comput. Archit. Lett. 16 (2) (2017) 99–102.
[23] X.S. Li, S.K. Yoon, J.G. Kim, S.D. Kim, A self-learning pattern adaptive prefetching method for big data applications, Sustain. Comput.: Inform. Syst. 20 (2018) 66–75.
[24] B. Cassell, T. Szepesi, J. Summers, T. Brecht, D. Eager, B. Wong, Disk prefetching mechanisms for increasing HTTP streaming video server throughput, ACM Trans. Model. Perform. Eval. Comput. Syst. 3 (2) (2018) 7.
[25] Y.F. Huang, J.M. Hsu, Mining web logs to improve hit ratios of prefetching and caching, Knowl.-Based Syst. 21 (1) (2008) 62–69.
[26] W. Ali, S.M. Shamsuddin, A.S. Ismail, Intelligent Naïve Bayes-based approaches for Web proxy caching, Knowl.-Based Syst. 31 (2012) 162–175.
[27] J. Parmar, J. Verma, State-of-art survey of various web prefetching techniques, in: Proceedings of the International Conference on Inventive Computation Technologies (ICICT 2016), Vol. 3, IEEE, 2016, pp. 1–7.
[28] N. Zhang, K. Zheng, M. Tao, Using grouped linear prediction and accelerated reinforcement learning for online content caching, in: Proceedings of the 52nd IEEE International Conference on Communications (ICC 2018), IEEE, 2018, pp. 1–6.
[29] A.I. Saleh, An adaptive cooperative caching strategy (ACCS) for mobile ad hoc networks, Knowl.-Based Syst. 120 (2017) 133–172.
[30] K. Zhang, S. Leng, Y. He, S. Maharjan, Y. Zhang, Cooperative content caching in 5G networks with mobile edge computing, IEEE Wirel. Commun. 25 (3) (2018) 80–87.
[31] W. Hu, Y. Jin, Y. Wen, Z. Wang, L. Sun, Toward Wi-Fi AP-assisted content prefetching for an on-demand TV series: A learning-based approach, IEEE Trans. Circuits Syst. Video Technol. 28 (7) (2018) 1665–1676.
[32] W. Zhang, Y. Wen, F. Liu, Y. Chen, R. Fan, Fast media caching for geo-distributed data centers, Comput. Commun. 120 (2018) 46–57.
[33] A. Abhari, M. Soraya, Workload generation for youtube, Multimedia Tools Appl. 46 (1) (2010) 91.
[34] J.J. Ramos-Muñoz, J. Prados-Garzon, P. Ameigeiras, J. Navarro-Ortiz, J.M. López-Soler, Characteristics of mobile youtube traffic, IEEE Wirel. Commun. 21 (1) (2014) 18–25.
[35] S. Müller, O. Atan, M. van der Schaar, A. Klein, Context-aware proactive content caching with service differentiation in wireless networks, IEEE Trans. Wirel. Commun. 16 (2) (2017) 1024–1036.
[36] J. Lino Monteagudo-Pereira, F. Auli-Llinas, J. Serra-Sagrista, JPIP Proxy server with prefetching strategies based on user-navigation model and semantic map, IEEE Trans. Multimedia 15 (7) (2013) 1491–1502.
[37] S. Liu, C. Ni, Z. Liu, X. Peng, H.N. Cheng, Mining individual learning topics in course reviews based on author topic model, Int. J. Distance Educ. Technol. 15 (3) (2017) 1–14.
[38] P. Wu, S. Yu, D. Wang, Using a learner-topic model for mining learner interests in open learning environments, J. Educ. Technol. Soc. 21 (2) (2018) 192–204.
[39] J. Xu, J. Liu, Y. Zhang, Word similarity computing based on hybrid hierarchical structure by HowNet, J. Inf. Sci. Eng. 31 (6) (2015) 2089–2101.

[40] J.-F. Yeh, Y.-S. Tan, C.-H. Lee, Topic detection and tracking for conversational content by using conceptual dynamic latent dirichlet allocation, Neurocomputing 216 (2016) 310–318.
[41] A.B. Rios-Alvarado, I. Lopez-Arevalo, V.J. Sosa-Sosa, Learning concept hierarchies from textual resources for ontologies construction, Expert Syst. Appl. 40 (15) (2013) 5907–5915.
[42] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[43] I.S. Dhillon, Y. Guan, B. Kulis, Weighted graph cuts without eigenvectors a multilevel approach, IEEE Trans. Pattern Anal. Mach. Intell. 29 (11) (2007) 1944–1957.
[44] H. Jia, S. Ding, M. Du, Y. Xue, Approximate normalized cuts without eigen-decomposition, Inform. Sci. 374 (2016) 135–150.
[45] P. Gaona-García, C. Montenegro-Marín, E. Gaona-García, A. Gómez-Acosta, Y. Hassan-Montero, Issues of visual search methods in digital repositories, Int. J. Interact. Multimedia Artif. Intell. 5 (3) (2018) 93–97.
[46] F. Lashkari, F. Ensan, E. Bagheri, A.A. Ghorbani, Efficient indexing for semantic search, Expert Syst. Appl. 73 (2017) 92–114.
[47] Q. Hu, Z. Han, X. Lin, Q. Huang, X. Zhang, Learning peer recommendation using attention-driven CNN with interaction tripartite graph, Inform. Sci. 479 (2019) 231–249.
[48] J. Jia, X. Xuan, B. Liu, L. Jiao, Bagging-based spectral clustering ensemble selection, Pattern Recognit. Lett. 32 (10) (2011) 1456–1467.
[49] H. Cheng, Y. Zhou, J.X. Yu, Clustering large attributed graphs: A balance between structural and attribute similarities, ACM Trans. Knowl. Discov. Data 5 (2) (2011) 12.
[50] Y. Xu, Z. Zhuang, W. Li, X. Zhou, Effective community division based on improved spectral clustering, Neurocomputing 279 (2018) 54–62.
[51] F.Z. Smaili, X. Gao, R. Hoehndorf, Onto2Vec: Joint vector-based representation of biological entities and their ontology-based annotations, Bioinformatics 34 (13) (2018) i52–i60.
[52] V. Sathiyamoorthi, V. Bhaskaran, Optimizing the web cache performance by clustering based pre-fetching technique using modified ART1, Int. J. Comput. Appl. 44 (1) (2012) 51–60.
[53] W. Feng, T.H. Kazi, G. Hu, J.X. Huang, pART2: Using adaptive resonance theory for web caching prefetching, Neural Comput. Appl. 28 (1) (2017) 1275–1288.
[54] Y. Zhou, C. Huang, Q. Hu, J. Zhu, Y. Tang, Personalized learning full-path recommendation model based on LSTM neural networks, Inform. Sci. 444 (2018) 135–152.
[55] M. Curiel, A. Pont, Workload generators for web-based systems: Characteristics, current status, and challenges, IEEE Commun. Surv. Tutor. 20 (2) (2018) 1526–1546.
[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (10) (2013) 2825–2830.