Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.036440
Article

Tech Science Press

A Time Pattern-Based Intelligent Cache Optimization Policy on Korea Advanced Research Network

Waleed Akbar, Afaq Muhammad and Wang-Cheol Song*

Department of Computer Engineering, Jeju National University, Jeju-si, 63243, Korea
*Corresponding Author: Wang-Cheol Song. Email: philo@jejunu.ac.kr
Received: 30 September 2022; Accepted: 14 November 2022

Abstract: Data is growing quickly due to a significant increase in social media applications. Today, billions of people use an enormous amount of data to access the Internet. The backbone network experiences a substantial load as a result of an increase in users. Users in the same region or company frequently ask for similar material, especially on social media platforms. The subsequent request for the same content can be satisfied from the edge if stored in proximity to the user. Applications that require relatively low latency can use Content Delivery Network (CDN) technology to meet their requirements. An edge and the data center constitute the CDN architecture. To fulfill requests from the edge and minimize the impact on the network, the requested content can be buffered closer to the user device. Which content should be kept on the edge is the primary concern. The cache policy has been optimized using various conventional and unconventional methods, but they have yet to include the timestamp beside a video request. The 24-h content request pattern was obtained from publicly available datasets. The popularity of a video is influenced by the time of day, as shown by a time-based video profile. We present a cache optimization method based on a time-based pattern of requests. The problem is described as a cache hit ratio maximization problem emphasizing a relevance score and machine learning model accuracy. A model predicts the video to be cached in the next time stamp, and the relevance score identifies the video to be removed from the cache. Afterwards, we gather the logs and generate the content requests using an extracted video request pattern. These logs are pre-processed to create a dataset divided into three-time slots per day. A Long short-term memory (LSTM) model is trained on this dataset to forecast the video at the next time interval. The proposed optimized caching policy is evaluated on our CDN architecture deployed on the Korean Advanced Research Network (KOREN) infrastructure. Our findings demonstrate how adding time-based request patterns impacts the system by increasing the cache hit rate. To show the effectiveness of the proposed model, we compare the results with state-of-the-art techniques.

Keywords: Multimedia content delivery; request pattern recognition; real-time; machine learning; deep learning; optimization; caching; edge computing

1 Introduction

According to Cisco white papers [1], there is a 35% increase in Internet users and a 60% increase in IP devices, and six hundred million more mobile devices from 2018 to 2023. The broadband speed will double from 45 to 110 Mbps, and the cellular speed will triple from 13.2 to 43.9 Mbps. The higher Internet users, the more social media applications, resulting in an increased number of IP networks and rapidly affecting the velocity of big data generation. Many social media streaming applications need very low latency and high bandwidth consumption. Similarly, real-time applications such as virtual reality, augmented reality, and voice recognition requires ultra-low network latency and high bandwidth. The network must carefully manage this immense increase in big data to reduce congestion. These requirements will further expand the difficulty faced by network administrators. Therefore, there needs to be a mechanism that can cope with such low latency and high bandwidth requirements.

The Content Delivery Network (CDN) is a distributed infrastructure that provides users with real-time content [2]. It can efficiently handle applications with low latency, high speed, and maximum bandwidth utilization requirements. The industry has deployed CDN to store data in user proximity to fulfill users requests. The CDN reduces the backbone network traffic load and decreases the request count toward the data center. Providing the content from CDN-edge means having comparatively less latency than the data center. According to Gartner [3], in 2023, more than 75% of the data will be processed at the edge. Therefore, we need strategies and algorithms to process and manage the resources on edge servers optimally. The edge can exist anywhere between the data center and the user. An edge device one hop away from the user, can provide very low latency with higher processing speed. But it can reduce the coverage area of the CDN-edge, which means more edge devices are required to cover different areas. Therefore, a suitable caching policy is needed to fulfill the network demands. Over the past few years, there have been many proposed caching policies, such as LRU (Least Recently Used) [4], LFU (Least Frequently Used) [5], and ILP (Integer Linear Programming) [6]. Since we live in the era of (Artificial Intelligent) AI, caching policies based on machine learning (ML), deep learning (DL), and reinforcement learning (RL) can extract the inherent features of data and then act accordingly [7]. The above-mentioned network requirements can be satisfied to take advantage of AI capabilities on a CDN-edge.

Many AI-proposed techniques optimize content caching on edge servers [8–10]. The authors in [6] employ machine learning for caching the multi-media content and location prediction to enrich the customer experience [11]. Utilizes the Q-Learning methodology for assisted caching in multiple (Multi-Access Edge Computing) MEC servers [12]. Proposes a dynamic programming knapsack optimization algorithm for video caching and real-time transcoding problems. However, none of these methods specifically discuss the user request arrival pattern. Although in [13], the author claims to extract the user request pattern from the customer transaction dataset, which is unsuitable in caching scenarios. Moreover, a recent survey on edge cache optimization emphasizes the significance of a time-based request pattern [14]. In this regard, we propose an optimized CDN-edge caching policy that extracts a time-based request pattern from the publicly available dataset captured over 25 years. Furthermore, while assessing the CDN optimization technique's performance, another vital aspect is the evaluating environment. Generally, researchers evaluate their proposed methods using either a simulation or a lab-based environment. However, in the case of a CDN scenario, such environments are inappropriate because they do not reflect real-world constraints such as network delay, link bandwidth, and traffic congestion. Therefore, our proposed testbed scenario consists of a CDN server, CDN-edge, and user devices representing the real-world use case on KOREN Software-Defined Infrastructure (SDI).

Our approach aims to optimize the utilization of cache on edge. The main contributions of our work are as follows:

- A publicly available dataset is analyzed to extract a time-based request pattern for its execution in the CDN scenario.
- A data center is built consisting of 1300 videos with their meta information.
- A CDN-edge architecture is created on KOREN.
- A dataset is created by collecting the logs from the CDN scenario.
- The problem is formulated mathematically as a cache hit maximization problem with environmental constraints.
- A video removal mechanism is created based on time, genre, and video size attributes.

The rest of the paper is structured as follows: Section 2 provides the literature review. Section 3 describes the problem statement and mathematical model of the proposed system. Section 4 presents the algorithm design, and Section 5 discusses building the CDN scenario with the generation of a dataset with the details of the supervised learning model. Section 6 evaluates the performance of the proposed methodology and discusses the results. The last section concludes our research with future directions.

2 Related Work

The authors in [15] propose an LRU-based cache replacement strategy. The LRU strategy is merged with the priority queue and re-access weight mechanism. Also, a Bayesian network theory-based prefetching strategy is implemented to reduce access latency and improve the quality of service. The proposed methodology is evaluated on the campus network, but the data center is usually placed in a geographically remote location in real-world scenarios. The authors do not consider user preferences, video profiles, and user request patterns over time. In [16], the author has proposed a machine learning-based smart caching and location prediction method for improving user experience. It caches multi-media content's temporal and spatial information on the Internet of Things (IoT) equipment. A real-time database is used to locate information. However, no real-time-based user request pattern is considered, and simulation-based evaluation is done. Reference [11] presents utilized multi-agent reinforcement learning techniques for cooperative content caching in MEC servers and download weight latency is used as a cache reward. MEC servers use Q-learning to make caching decisions in a multi-agent environment. The authors use an inappropriate representation of a Poisson distribution-based user request and evaluate their proposed work using a simulated environment. Authors in [13] concatenate the convolutional neural network (CNN), bidirectional deep neural network (BNN), and fully connected neural network (FCNN) models. A combined model is used for feature extraction, dimension reduction, timestamp-based user request association, and prediction performance improvements. They claim to use the models in the online function. However, these models are computationally intensive and complex; therefore, they are inappropriate in online scenarios. Also, the user request pattern is extracted from the customer transaction dataset that does not reflect the legit video request pattern. Moreover, the proposed scheme is not evaluated in a CDN environment. In [12], the dynamic programming and knapsack algorithm is employed to optimize the solution for video caching and real-time transcoding problems. The authors consider the attributes of video representations, video popularity level, and user profiling for user request patterns. Their objective is to maximize the cache hit ratio, and a simulation is done to evaluate the performance of the proposed system. Unrealistic assumptions about the edge resources are made. Further, Zipf distribution-based content popularity is considered that is impractical. In [17], the authors designed a cooperative caching model based on multiple cooperative caching edge servers. Machine learning and greedy algorithm-based content delivery strategies are designed for cooperative caching servers. The optimization objective is to minimize the average content transmission delay. The simulation environment is set up to evaluate the performance of the proposed scheme. The authors use a fixed user request pattern and Zipf distribution-based content popularity.

In [18], the authors integrate a social content-centric network (SocialCCN) with mobile edge caching. A multi-head attention-based model uses the social relationship, geographical information, and historical request features for popularity prediction. It incorporates the encoder-decoder structure with BiLSTM, and caching strategy is based on the average value of multi-step popularity prediction. However, it needs the details of the environmental setup and information on how their proposed scheme has been evaluated in the CDN environment. In [19], a cyclic genetic ant colony optimization algorithm for the CDN environment is presented. The access count, access frequency, and data size are the multi-objectives for the optimization algorithm. The algorithm efficiently offloads the data with minimum access and latency and processes the high access and latency data. However, there is no added information about the video request patterns and profiles. The authors in [20] utilize an artificial neural network (ANN) to find the relationship of content demand over the time stamp. Then, an integer linear programming-based solution is proposed to formulate the latency problem. Although they numerically evaluate the results, they do not consider content and user profiling. Moreover, they do not evaluate their proposed method in the CDN scenario.

Table 1: Comparison with state-of-the-art techniques

Reference	Technique Used	What is cached?	Environment	Dataset	Request Pattern
[10]	Reinforcement Learning	Video	Simulation	MovieLens	Poisson
[12]	Dynamic Programming	Video	Simulation	No	No
[13]	Deep-RNN	User Request	Not Tested	Customer Transaction	Bi-RNN
[15]	Priority LRU	File	Campus	No	Random
[16]	ML with (LRU/LFU)	Multimedia Content	Simulation	No	No
[17]	Neural CF	Content	Not Tested	No	Fixed
[18]	Encoder-Decoder with Bi-LSTM	Content	Simulation	No	SONETOR
[19]	Ant Colony Optimization	Data	Numerical Evaluation	No	No
[20]	RNN with Linear Programming	Data	Numerical Evaluation	No	Random
[21]	Deep-RL	Content	Simulation	Resource Utilization	No
Our	Proposed Scheme	Video	Real-time CDN	We generate dataset	Time-based

Table 1 shows a comparison of all the proposed techniques. The above algorithms optimize the CDN-edge caching policy. The previous studies utilize either traditional or machine learning techniques for CDN cache optimization. A few methods [11,12,16,17] are evaluated in the simulation environment and others [18,19] numerically. The following proposed approaches [13,18–20] are not assessed in the CDN environment, only the model proposed in [15] is evaluated in the campus-level CDN environment. However, it still does not reflect real-world cases, whereas, in typical CDN scenarios, the data center is usually placed in a geographically remote location. In addition, a user request pattern is essential when considering the edge caching scenario. The previous studies have used random, Poisson distribution, fixed, and Recurrent Neural Network (RNN) based patterns. However, they have yet to consider a timestamp in their solution. The importance of a time-based request pattern is also highlighted in [22,23]. For example, users tend to watch more entertainment videos at night than educational ones. Similarly, most users work in offices, universities, and companies during the daytime. To this end, it is important to consolidate the timestamp with the request pattern.

Additionally, [13,15,16] also mention the algorithms to optimize the cache replacement strategies. But multiple AI models will put a huge computational workload on the CDN-edge. Therefore, our proposed model combines AI techniques with the computational workload.

We design a CDN architecture on top of KOREN-SDI to overcome the challenges mentioned above to represent a real-world scenario. We establish a data center consisting of 1300 videos from eight different categories. Additionally, we extract a time-based content request pattern from the publicly available dataset. Furthermore, we create video profiles to integrate the effect of size, length, and type of video.

3 System Model Definition and Problem Formulation

This section explains the caching model with the formulation of the relevance score. The integration of the proposed system model with machine learning is also presented. The discussion of the hit ratio maximization problem using relevance score and machine learning concludes the section. The basic steps followed by the proposed model are mentioned in Algorithm 1. Initially, the proposed system waits for the video request. Once the video request arrives, it checks for the requested video in a cache. If found in the cache, a requested video will be provided to the user. Suppose that a requested video is not available in the cache. In that case, a reactive caching function is activated to download the video from the data center and provide the video to the user. In case of a cache miss scenario, a proactive caching function is activated to predict the request video for the next time stamp. A REACTIVE CACHING () and PROACTIVE CACHING () functions detail is provided in section IV.

Algorithm 1: Cache Optimization

1: Wait until the video request arrived
2: When a request for video v has arrived
3: **if** Requested video v is available in the cache, then
4: Provide the requested video v to the user
5: Record the log, such as the time of the request, video id, and video genre
6: **else**
7: Video v ← call REACTIVE CACHING (v)
8: Provide the requested video v to the user
9: call PROACTIVE CACHING ()
10: **end if**

3.1 System Model

A CDN environment is a design that consists of a data center, edge server, and end device on KOREN-SDI to evaluate the proposed scheme in a real-world environment. Furthermore, a 24-h video request pattern is extracted from the public dataset and executed on edge to generate the user request. Fig. 1 shows the overview of our proposed architecture. The main components are the end-device to generate user requests, CDN-edge for video caching, and the data center for video repository. Initially, a 24-h video request pattern is extracted from publicly available datasets [24,25] and placed on the end-device to generate a request in a time-based pattern. A request generator module is activated on the user's device to create a request. Initially, a request is sent to CDN-edge; the request handler on the CDN-edge receives the request and checks for content in the local video repository. If video content is found, it is provided to the user. Otherwise, the CDN-edge generates a new request to the data center. On the data center, the request handler receives the content request from the CDN-edge, finds the requested content in the central video repository, and forwards the requested content to the response generator. Response generators send

the content to CDN-edge. Upon receiving the content from the data center, the CDN-edge provides the content to a user. On the end device, the content handler receives video from CDN-edge and stores it.

$$R_i = \begin{cases} 1, & if\ requested\ video\ is\ (i) \\ 0, & otherwise \end{cases} \tag{1}$$

$$R_i^g = \begin{cases} 1, & if\ requested\ video\ is\ (i) \\ 0, & otherwise \end{cases} \tag{2}$$

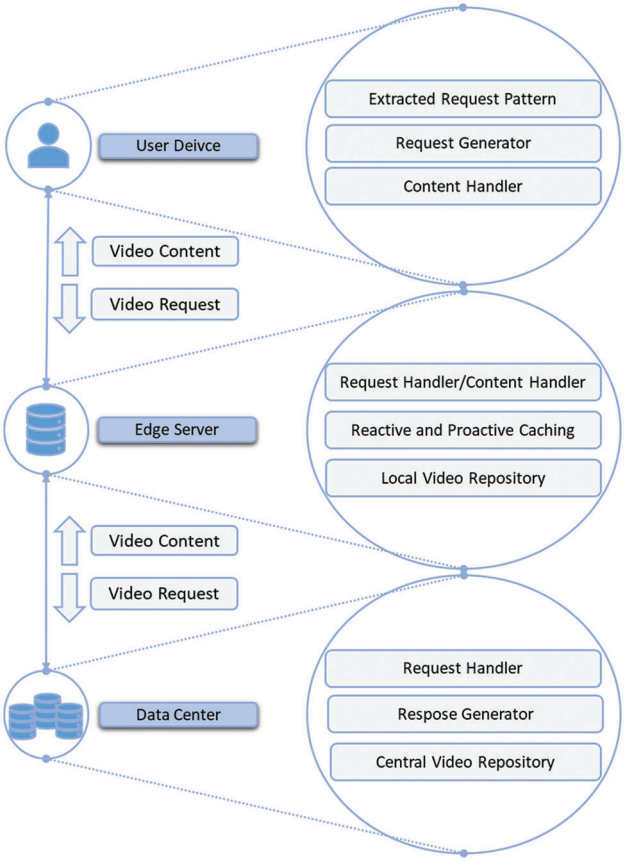

Figure 1: An overview of a proposed system with all basic components

A $V = \{v_1,\ v_2,\ v_3,\ \ldots,\ v_N\}$ is the set of videos that make up a data center, and N is the total number of videos. Similarly, the set of the genre is denoted as $G = \{g_1,\ g_2,\ g_3,\ \ldots,\ g_3\}$, where K is the total number of genres. The symbol for the total number of videos that are currently cached is M. Cache and video storage sizes are denoted by the symbols S_C and S_i. The morning period, which lasts from 4:00 AM to 9:00 AM; the day period, which lasts from 9:00 AM to 5:00 PM; and the evening period, which lasts from 5:00 PM to 4:00 AM, are the three time periods that make up a 24-h period. The mathematical representation of time slots is defined as $t\ \epsilon\ T$ where $T = (Morning,\ Day,\ Night)$. There are two binary variables R_i and R_i^g whose definition is provided in Eqs. (1) and (2), respectively. R_i is used to indicate whether or not a video is the requested video. Similarly, the R_i^g function will show whether or not a requested video belongs to the g genre.

3.2 Relevance Score

All of the requested data cannot be stored in the cache due to the cache's storage capacity being significantly less than that of the data center. As a result, there should be a way to replace the old videos in the cache with the ones that have just been requested. All cached videos are given a relevance score, and the video with the lowest relevance score is removed from the cache. A relevance score is based on three attributes: time, genre, and size. All these attributes are separately calculated for each time slot (t).

$$f_1 = \frac{\sum_{i=1}^{M} R_i}{M}, \quad \sum_{i=0}^{M} S_i < S_c \tag{3}$$

$$f_2 = \frac{\sum_{i=1}^{M} R_i^g}{M}, \quad g \in G \tag{4}$$

$$f_3 = \frac{S_v - min(S_i)}{max(S_i) - min(S_i)}, \quad i \in n \tag{5}$$

The time and genre attribute of the video is derived using empirical probability [26,27]. The empirical probability states *the ratio of the number of outcomes of event A to the total number of trials*. That is used for the time attribute as *the ratio of the total number of requests for video i to the total number of requests*. For genre attribute, it is used as *the ratio of total number of requests for video of genre* (g) *to the total number of requests*. The mathematical formulation of the time and genre attributes is shown in Eqs. (3) and (4), respectively. The video (v) size attribute is calculated as the normalized value of all requested videos, as shown in equation Eq. (5).

$$E(f_j) = -ln(m)^{-1} \sum_{i=1}^{m} \left[P(f_j) * lnP(if_j)\right] \tag{6}$$

$$w_j = \frac{1 - E(f_j)}{3 \; - \; [E(f_1) + E(f_3) + E(f_3)\,]} \tag{7}$$

$$rel_{score} = [w_1 f_1 + w_2 f_2 + w_3 (1 \; - \; f_3)] \tag{8}$$

The information entropy [28] is stated as "*how much surprise there is in an event. Those events that are rare (low probability) are more surprising and therefore have more information than those events that are common (high probability)*". The weights are assigned to each attribute based on the definition of information entropy to determine how each attribute affects the relevance score as shown in Eq. (6). The weight of each attribute is obtained through information entropy, as shown in Eq. (7). In denominator of Eq. (7), the sum of all the attributes entropies is subtracted from three because there are three attributes in total. $E(f_j)$ and W_j represents the information entropy and weight of j^{th} attribute, respectively, and $f_j \in (f_1, f_2, f_3)$. The constant value in the denominator of Eq. (6) denotes the total number of attributes in our system. Eq. (8) determines the relevance score based on the video's attribute value and weight. The relevance score measures how good the video is in the current context. The context comprises the current time slot (t) and cached videos. The aim is to locate and remove a video with a minimum relevance score.

3.3 LSTM Architecture

LSTM is the most widely used algorithm for time series prediction because it can memorize prior predictions. In our case, the model needs to take the preceding predict previous to account when forecasting the next time stamp. Fig. 2 shows a simplistic representation of the LSTM. It comprises basic gates that determine which information is allowed to get through and which is denied. The ignoring gate

is used to remove irrelevant results from the predictions. The forgetting gate separates the current predictions from the old ones that were kept in memory. The selection gate determines which part of the prediction will be the output. There are two additional activation functions: the hyperbolic $(tanh)$ function and the logistic (σ) function. A hyperbolic function squashes data between −1 and 1, while a logistic function squashes values between 0 and 1.

$$f_t = \sigma\left(W_f \,.\, [h_{t-1},\ x_t] + b_f\right) \tag{9}$$

$$i_t = tanh[\sigma\ (W_i \,.\, [h_{t-1},\ x_t] + b_i)\ ] \tag{10}$$

$$\tilde{C}_t = \sigma\ (W_c \,.\, [h_{t-1},\ x_t] + b_c) \tag{11}$$

$$C_t = f_t\ *\ tanh(i_t) + C_t\ *\ C_{t-1} \tag{12}$$

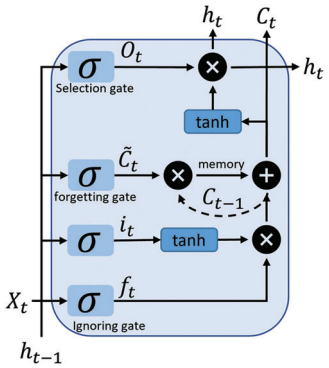

Figure 2: The internal structure of LSTM cell

The output of the ignoring gate, represented by the value of f_t, is calculated using Eq. (9). Eqs. (10) and (11) show how to calculate the values of i_t and $\tilde{C}_t$ in a similar way. Based on f_t, i_t, and $\tilde{C}_t$, Eq. (12) is utilized to calculate the updated cell state C_t at time stamp t.

$$o_t = \sigma\ (W_o \,.\, [h_{t-1},\ x_t] + b_o) \tag{13}$$

$$h_t = o_t\ *\ tanh(C_t) \tag{14}$$

As per Eq. (13), the value of the selection gate is calculated using the previous hidden state and current cell input with sigmoid used as an activation function. As shown in Eq. (14), the current hidden state is the addition of the selection gate output with the current cell state.

$$V_t = W_v \,.\, h_t + (b_t) \tag{15}$$

$$\hat{h}_t = softmax(V_t) \tag{16}$$

To find the final output $\left(\hat{h}_t\right)$ of the LSTM model, the SoftMax activation function is added as the last layer activation. Before the activation is applied, the weighted matrix (W_v) is multiplied and bias (b_t) is added to the output (h_t). Eqs. (15) and (16) represent the steps of final prediction of LSTM model.

3.4 Objective Function

To maximize cache hits, we build a multi-objective problem based on the relevance score and accuracy of the machine learning model under the constraints of total cache capacity and caching decision variables. Therefore, Eq. (17) defines the cache maximization problem:

$$\max_{HitRatio} \frac{max(acc_{score}) + (1 - min(rel_{score}))}{2} \tag{17}$$

$$s.t.\ C1 : \sum_{i=0}^{m} [S_i * X_i] < S_c$$

$$C2 : X_i = \begin{cases} 1, & if\ video\ (i)\ in\ cache \\ 0, & otherwise \end{cases}$$

In this case, C1 confirms that the total amount of storage for all cached videos should not exceed the total amount of storage for the cache, and C2 ensures that a function only evaluates cached videos.

4 Algorithm Design

According to the hit ratio maximization model, the proposed algorithm considers the hybrid strategy, which combines proactive and reactive caching. The reactive algorithm is invoked to fulfill the video request in case of a cache miss. Proactive caching is activated to pre-fetch the video that the machine learning model has predicted in time $(t - k)$. For both strategies, a minimum relevance score is utilized to remove a video from the cache in order to make room for new videos. The reactive caching steps used during the cache miss scenario are listed in Algorithm 2. In case of a cache miss scenario, CDN-edge forwards the video request to the data center. Upon receiving the video, CDN-edge verifies the available space. If no free space is available, video (d) will be removed from the CDN-edge the make the space. Video (d) is selected based on the minimum relevance score recorded in the video log. Based on machine learning prediction, Algorithm 3 describes the proactive caching procedures. Before the next request is due, it retrieves the predicted video and puts it in the cache. The machine learning model predicts the next video to arrive. No further action is done if the predicted video is already in CDN-edge. If the video is unavailable in the cache, it is downloaded from the data center. In Algorithm 4, the procedures for calculating the relevance score are listed. It measures how relevant a video is in the given context. The relevance score is calculated for all available videos on CDN-edge. An empirical frequency is measured for genre and time attributes; a normalized value is calculated for size attributes. After that, entropy and weight are calculated for each attribute using Eqs. (6) and (7), respectively. In the end, the relevance score is calculated, and a video with a minimum relevance score is selected for removal.

Algorithm 2: Reactive Caching

1: Forward video v request to data center

2: On video v arrival from the data center

3: **if** Available cache space ≥ video v size, then

4: Store the video v in the cache

5: Available cache space = currently available space-video v size

6: RETURN video v to user

7: **else**

(Continued)

Algorithm 2 (continued)

8: video d ← call RELEVANCE SCORE CALCULATION ()
9: delete video d from the cache
10: Record the log, such as time of deletion, video id, and video genre
11: Go to line 3
12: **end if**

Algorithm 3: Proactive Caching

1: video i ← ML PREDICTION () (Eq. (9))
2: **if** the video i already in the cache, then
3: EXIT ()
4: **else**
5: while Available cache space ≤ video i size do
6: video d ← call RELEVANCE SCORE CALCULATION ()
7: delete video d from the cache
8: Record the log, such as time of deletion, video id, and video genre
9: end while
10: Store the video i in the cache
11: Available cache space = currently available space-video i size
12: **end if**

Algorithm 4: Relevance Score Calculation

1: Create an empty list as lsrel
2: **for** Every video v **in** the cache **do**
3: f1 ← Calculate the empirical frequency of time attribute (Eq. (3))
4: f2 ← Calculate the empirical frequency of genre attribute (Eq. (4))
5: f3 ← Calculate the normalized value of the size attribute (Eq. (5))
6: E(fj) ← Calculate info. The entropy of every attribute (fj) (Eq. (6))
7: Wj ← Calculate the weight of every attribute (fj) (Eq. (7))
8: relscore ← Calculate the relevance score of video v (Eq. (8))
9: ls_rel ← Append (relscore)
10: **end for**
11: video d ← minimum (ls_rel)
12: RETURN video d

5 Dataset Generation and Pre-Processing

This section discusses request pattern generation, log collection, and real-time dataset building.

5.1 Video Request Pattern

For a better prediction, the context of the environment needs to be integrated. The time of day strongly relates to the video request count. Typically, more entertainment videos are requested at night and education in the daytime. To understand the relationship between time and request count, we analyze two publicly available datasets from Kaggle [29]. The dataset is "MovieLens20M" [24] with 20 million records, and "The Movie Dataset" [25] with 26 million records. A 24-h request pattern extracted from these datasets depicts the mapping of request counts on timestamps, as shown in Fig. 3. The Y-axis and X-axis represent the user request count per hour and hour value, respectively. The request count tends to increase during the evening and peak at night around 9:00 PM. Similarly, it tends to decrease late at night and reaches a minimum around 9:00 AM. This 24-h pattern is divided into three time slots, a morning time slot from 4:00 AM to 9:00 AM, a day time slot from 9:00 AM to 5:00 PM, and a night time slot from 5:00 PM to 4:00 AM. A similar time slot-based prediction model is proposed in [29,5]. These three intervals are our different scenarios where the caching policy will adapt accordingly.

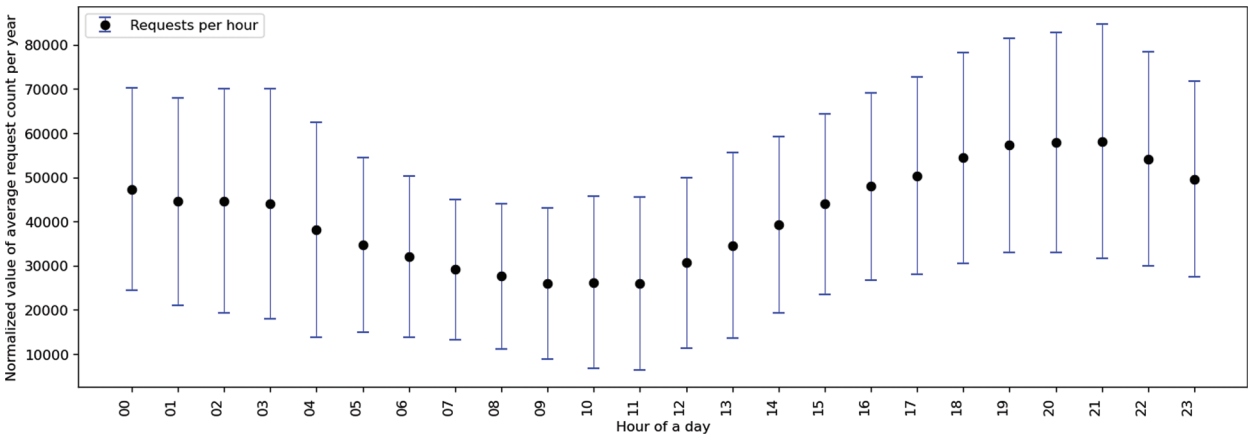

Figure 3: An extracted 24-hour time-based request pattern extracted from a publicly available dataset

5.2 Video Repository

We built a repository of 1300 videos from eight genres downloaded from popular YouTube channels such as BTS, T-Series, and Cocomelon. The genres are animation, movies, interviews, education, sports, films, news, travel, and music. The video meta-information is as follows: a rating, number of views, video length, and video size. Videos with a size of 50 MB or less are downloaded. Then, all videos are assigned a unique video id, and later on, these video ids are mapped to a request pattern generated in the previous step. Complete detail about the dataset features is listed in Table 2.

5.3 Logs Collection

We send 80,000 video requests from the end device based on the extracted video request pattern. If the requested video is available, the edge server satisfies the request rather than a data center. The records for each of these operations are kept, and the time it takes to download a video is also measured. A dataset is created afterward by mapping these logs to user request patterns; a complete description of the dataset is shown in Table 3.

Table 2: Data center attributes with their description

Attributes	Description
Number of videos	1300
Video genre	Interview, education, sports, films, animation, NEWS, music, and travel
Video attributes	Size, length, rating, and views
Video size	Between 5 to 50 MBs
Video rating	1, 2, 3, 4, 5

Table 3: Dataset features with description

Attributes	Description
Video ID	To uniquely identify the video.
Timestamp	Time of requested video
Genre	Type of video (listed in Table 2)
Rating	How popular is the video (most popular is the highest number)
Views	How many requests for video are received
Length	Time duration of the video
Size	Video storage size of the disk
Latency	Time required to download a video
Latency (avg)	Average time required to download a video in the time slot (t)
Ratio	Total number of requests for single video/total number of requests for all videos

5.4 Pre-Processing and Model Training

Fig. 4 shows the overall system flow from data pre-processing to model training and evaluation. The dataset is further refined and restructured in the data cleaning and feature selection phase. In this phase, the irrelevant features are dropped, and the essential features are selected using a correlation matrix score. The data is sorted with a timestamp to make the sequential prediction feasible. As mentioned above, we calculate the relevance score using equations [30]. After completion of pre-processing step, the dataset has video-id, timeslot, latency, and relevance score as essential features. We aim to predict the videos for the next time slot; therefore, we further divide the datasets into train and test. 20% of the data is left for testing, and 80% is used for training. The capacity of the LSTM model to keep the prior output in memory and apply it for the subsequent prediction is the main factor driving its selection. If dataset features have a strong correlation, LSTM can be trained on a small amount of data [30]. To benefit from these characteristics, several researchers applied LSTM in a variety of areas, including language identification [31], picture detection [32], and forecasting [33]. We selected LSTM as our predicted model while keeping these capabilities in mind. A dataset built through log collection was used to train the model. The model hyper-parameter values are count layer (2), output classes (325), features (4), and step-size (8).

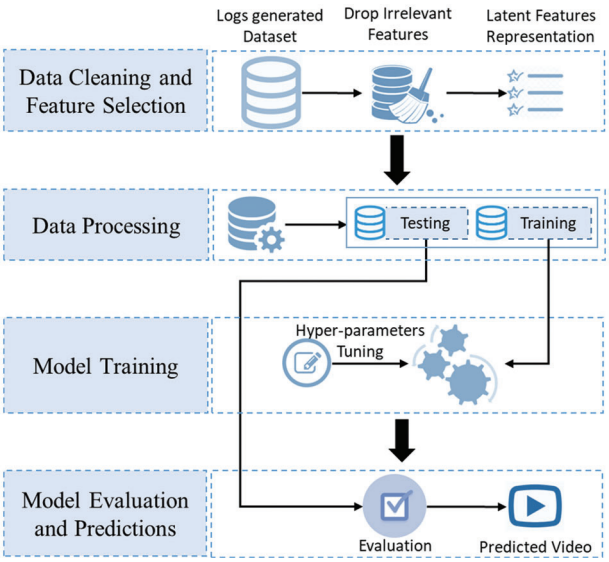

Figure 4: The data pre-processing with model training and prediction

6 Performance Evaluation and Results

The testbed environment, mathematical model verification, system model performance evaluation, and results are all covered in this section.

6.1 Testbed Environment

The testbed environment is built on top of the KOREN infrastructure. The topology has ten virtual switches in the KOREN, one for each city, interconnected using 10G links. We have ten physical nodes (edge nodes) deployed in ten cities connected through virtual switches. All outgoing city traffic passes through the directly connected switch. The Open Daylight SDN controller is used to deploy virtual flow switches in the KOREN testbed. The flow rules are deployed using reactive forwarding based on the requested end device in the initial step. After that, a communication link between the data center and CDN-edge is created. The content request is generated from the end device and sent to the CDN-edge. The content request will be provided to a user if found on the edge. Otherwise, the request is fulfilled by the data center.

6.2 Mathematical Model Verification

To validate our proposed model, we apply the data validation approach. To examine how an objective function is defined in Eq. (17) behaves, we choose seven different values between 0 and 1. These numbers have been chosen: 0, 0.1, 0.2, 0.4, 0.6, 0.8, and 1. To determine which combination produces the best outcome, we examine the outcome of the objective function for every combination. When accuracy is one and relevance score is zero, an objective function yields its optimal value. That indicates that the most irrelevant video is removed from the cache, and machine learning is predicted with the highest accuracy. When an objective function produces a value of zero, it indicates that the user requests the deleted video from the cache in the next timestamp and that the machine learning-predicted video did not result in a hit scenario. Similarly, the definition of the relevance score function is evaluated as stated in Eq. (8). The following possible values are considered for each attribute: 0, 0.25, 0.50, 0.75, and 1. The weights $(w_1, w_2, and\, w_3)$ are computed in accordance with Eq. (7). We examine the results of every possible combination in an effort to determine the optimum attribute values and weights. According to

the relevance equation, if a single attribute returns its maximum value, the relevance score will be more than 50%. For instance, if video v is the most-watched video, the time attribute value will be 1, and the resulting relevance score will be more than 50%. Similarly, to this, the relevance score will be higher than 90% if two attributes return their maximum value. For instance, if video v in the currently cached video is the most popular and has a smaller size, the resulting relevance values will be greater than 90%.

6.3 Model Performance Evaluation

The two sub-systems will determine how the proposed system is evaluated. The two subsystems are the relevance score computational model, which has a mathematical definition, and the machine learning model, which forecasts the video for the next time stamp. The previous section has explained how the relevance score computational model is evaluated. Here, we will discuss the machine learning model and demonstrate the proposed system evaluation later. For the proposed model evaluation, we generate the request in sequential order. Based on the sequential list, we have the information about the video v to be requested at $(t + n)$, therefore, w. Therefore, video v on time t at the edge. The request can be executed from the edge when the user requests the video v on the next timestamp. We measure the prediction of the proposed model using an accuracy metric. The trained model is individually applied on the CDN-edge to make the prediction. Later the model performance is measured using the hit ratio. Fig. 5 shows the hit ratio comparison results of our proposed models. The morning and day datasets have a lower hit ratio, which might be due to the smaller dataset size and fewer training examples per class. With an increase in dataset size, the hit ratio can be improved. Fig. 6 shows the comparison of the proposed caching policy with state-of-the-art schemes. The morning (M) and day (D) dataset performance scores are relatively lower due to the smaller dataset size. The increase in dataset size can improve the performance. The best performing lowering model has trained on the night (N) dataset and achieved 94% accuracy, which is better than any other existing model.

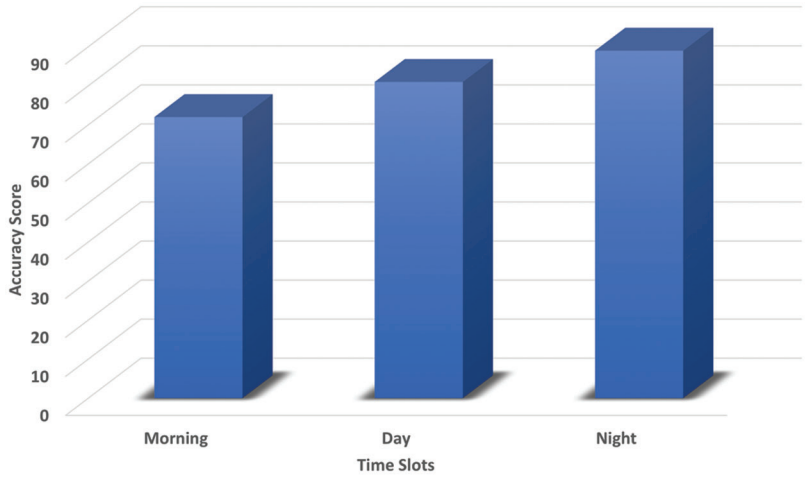

Figure 5: Hit ratio of time-slot datasets

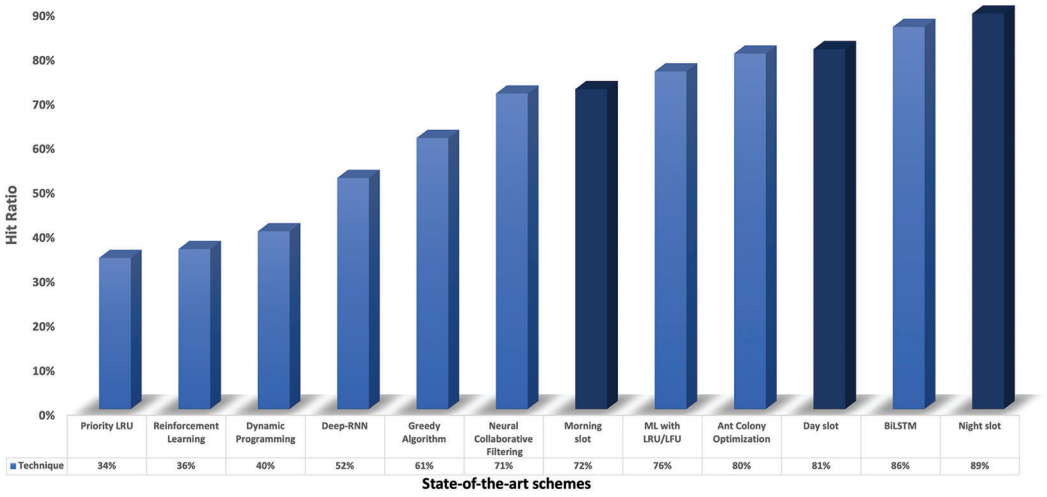

Figure 6: Comparison with state-of-the-art schemes

7 Conclusion and Future Works

This paper proposes an algorithm that optimally utilizes the edge storage resource to increase the chances of request fulfillment from the CDN-edge. The optimal use of the storage is the major problem in CDN as the edge resources are limited. Therefore, our CDN-edge caching scheme optimally selects the video that is least affected by future requests and deletes it from the edge. A selected video is based on video weight and a time-based video and genre profile. Then, an extracted request pattern is appended with the proposed scheme to inherit the time-based features. Furthermore, a request pattern is configured on the user device to generate request in sequential order. A dataset is built after the pre-processing of operation logs. Finally, an LSTM model is trained and optimized with different hyper-parameter values using GridSearchCV. Moreover, the proposed scheme is tested on a CDN architecture built on top of KOREN SDI. The comparison results show the proposed method outperforming the state-of-the-art techniques on hit ratio. A more complex time-based pattern can be extracted in future work, and different dataset features can be utilized to enhance model training. A hierarchical or multi-level edge can be deployed to test the model in a more realistic environment. An extended version of this algorithm can be applied to optimize the network resources.

Acknowledgement: This research was supported by the 2022 scientific promotion program funded by Jeju National University.

Funding Statement: This research was supported by the 2022 scientific promotion program funded by Jeju National University.

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

References

[1] Cisco. "Cisco annual internet report (2018–2023) white paper," 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf.

[2] C. Papagianni, A. Leivadeas and S. Papavassiliou, "A Cloud-oriented content delivery network paradigm: Modeling and assessment," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 5, pp. 287–300, 2013.

[3] Gartner. "Gartner forecasts worldwide public cloud End-user spending to reach nearly $500 billion in 2022," 2022. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2022-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-500-billion-in-2022.

[4] E. Friedlander and V. Aggarwal, "Generalization of LRU cache replacement policy with applications to video streaming," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 4, no. 3, pp. 1–22, 2019.

[5] L. Chhangte, E. Viterbo, D. Manjunath and N. Karamchandani, "Online caching and coding at the wifi edge: Gains and tradeoffs," in *Proc. IEEE WCNCW*, Seoul, Korea, pp. 1–6, 2020.

[6] Y. Wang, G. Zheng and V. Friderikos, "Proactive caching in mobile networks with delay guarantees," in *Proc. IEEE Int. Conf. on Communications (ICC)*, Shanghai, China, pp. 1–6, 2019.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[8] R. Wang, R. Li, P. Wang and E. Liu, "Analysis and optimization of caching in fog radio access networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8279–8283, 2019.

[9] L. Li, G. Zhao and R. S. Blum, "A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1710–1732, 2018.

[10] K. Bilal, S. Junaid, A. Erbad, W. Alanazi and A. Alourani, "Addressing challenges of distance learning in the pandemic with edge intelligence enabled multicast and caching solution," *Sensors*, vol. 22, no. 3, pp. 1092–1094, 2022.

[11] W. Jiang, G. Feng, S. Qin and Y. Liu, "Multi-agent reinforcement learning based cooperative content caching for mobile edge networks," *IEEE Access*, vol. 7, pp. 61856–61867, 2019.

[12] X. Chen, L. He, S. Xu, S. Hu and Q. Li *et al.,* "Hit ratio driven mobile edge caching scheme for video on demand services," in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)*, Shanghai, China, pp. 1702–1707, 2019.

[13] L. Ale, N. Zhang, H. Hu, D. Chen and T. Han, "Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5520–5530, 2019.

[14] Arxiv. "Applying machine learning techniques for caching in edge networks: A comprehensive survey," 2020. [Online]. Available: https://arxiv.org/abs/2006.16864.

[15] C. Li, M. Song, S. Du, X. Wang, M. Zhang *et al.,* "Adaptive priority-based cache replacement and prediction-based cache prefetching in edge computing environment," *Journal of Network and Computer Applications*, vol. 165, pp. 102715, 2020.

[16] Y. Tang, K. Guo, J. Ma, Y. Shen and T. Chi, "A smart caching mechanism for mobile multimedia in information centric networking with edge computing," *Future Generation Computer Systems*, vol. 91, pp. 590–600, 2019.

[17] Y. Chen, Y. Liu, J. Zhao and Q. Zhu, "Mobile edge cache strategy based on neural collaborative filtering," *IEEE Access*, vol. 8, pp. 18475–18482, 2020.

[18] J. Liang, D. Zhu, H. Liu, H. Ping and T. Li *et al.,* "Multi-head attention based popularity prediction caching in social content-centric networking with mobile edge computing," *IEEE Communications Letters*, vol. 25, no. 2, pp. 508–812, 2020.

[19] D. Wang, X. An, X. Zhou and X. Lu, "Data cache optimization model based on cyclic genetic ant colony algorithm in edge computing environment," *International Journal of Distributed Sensor Networks*, vol. 15, no. 8, pp. 1550147719867864, 2019.

[20] S. Sun, W. Jiang, G. Feng, S. Qin and Y. Yuan, "Cooperative caching with content popularity prediction for mobile edge caching," *Tehnički Vjesnik*, vol. 26, no. 2, pp. 503–509, 2019.

[21] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 31–37, 2017.

[22] R. Turrin, A. Condorelli, P. Cremonesi and R. Pagano, "Time-based tv programs prediction," in *Proc. (RecSysTV 2014)*, California, USA, vol. 14, 2014.

[23] S. Cheng and W. Wang, "Rating prediction algorithm based on user time-sensitivity," *Information*, vol. 11, no. 1, pp. 4, 2019.

[24] B. Rounak. "The MovieLens dataset," 2018. [Online]. Available: https://grouplens.org/datasets/movielens/latest/.

[25] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. System*, vol. 5, no. 4, pp. 1–19, 2015.

[26] Arxiv. "A modern introduction to probability and statistics," 2005. [Online]. Available: https://link.springer.com/book/10.1007/1-84628-168-7.

[27] T. Peng, H. Wang, C. Liang, P. Dong and Y. Wei *et al.,* "Value-aware cache replacement in edge networks for internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 9, pp. e4261, 2021.

[28] MIT Technology Review. "Claude shannon: Reluctant father of the digital Age," 2001. [Online]. Available: https://www.technologyreview.com/2001/07/01/235669/claude-shannon-reluctant-father-of-the-digital-age/.

[29] Kaggle. "Explore, analyze, and share quality data," 2022. [Online]. Available: https://www.kaggle.com/datasets.

[30] B. Zhang, G. Zou, D. Qin, Y. Lu and Y. Jin *et al.,* "A novel encoder-decoder model based on read-first lstm for air pollutant prediction," *Science of the Total Environment*, vol. 765, pp. 144507, 2021.

[31] Arxiv. "Contextual LSTM (CLSTM) models for large scale NLP tasks," 2016. [Online]. Available: https://arxiv.org/abs/1602.06291.

[32] B. Jena, S. Sexana, G. K. Nayak, L. Saba and N. Sharma *et al.,* "Artificial intelligence-based hybrid deep learning models for image classification: The first narrative review," *Computers in Biology and Medicine*, vol. 137, pp. 104803, 2021.

[33] Arxiv. "Time series forecasting using LSTM networks: A symbolic approach," 2020. [Online]. Available: https://arxiv.org/abs/2003.05672.