A Long-Short-Term Fusion Approach For Video Cache

Rui-Xiao Zhang, Tianchi Huang, Lifeng Sun
Computer Science and Technology, Tsinghua University
Email: zhangrx17@mails.tsinghua.edu.cn, sunlf@tsinghua.edu.cn

***Abstract*—Owing to the unprecedented growth of video demands, video caching has been a basic network functionality in today's network architectures to offload backbone traffics, as well as provide users with lower access delay. Although abundant cache replacement algorithms have been proposed recently, they all suffer from a critical limitation: due to their immature rules, inaccurate feature engineering or unresponsive model update, they cannot strike a balance between the long-term history and short-term sudden events. To tackle this problem, we propose LA-E2, a long-short-term fusion cache replacement approach, which is based on a learning-aided exploration-exploitation process. Specifically, by effectively combining the deep neural network (DNN) based prediction with the online exploitation-exploration through a *top-k* method, LA-E2 can both make use of the historical information and adapt to the constantly changing popularity responsively. Through the extensive experiments in two real-world datasets, LA-E2 is demonstrated to achieve state-of-the-art performance and generalize well. Especially when the cache size is small, LA-E2 outperforms the baselines by 17.5%-68.7% higher in total hit rate.**

***Index Terms*—cache, content replacement, deep learning**

I. Introduction

Among all content transferred in the network, multimedia, especially video content, has played a dominant part. As reported by Cisco [1], video content will account for as much as 75% of the total network traffic in the future. However, the limited capacity of the backbone network has presented enormous challenges to content providers (CPs). On the one hand, a growing number of video audiences are eager to have a more timely and higher quality of viewing experience; on the other hand, CPs need to deliver videos in a more cost-effective way to make more profits. To better serve users with limited bandwidth resources and trim costs, content caching has been accepted as a promising way due to its capacity for traffic offloading.

However, although the caching technique has been used for decades, the constantly changing network conditions have brought new challenges to researchers. For example, in 5G networks, the caching storage in cellular base stations is becoming increasingly small, and more caching devices will be placed closer to end users [2]. These trends make the content caching problem more heterogeneous.

Today, most replacement algorithms are rule-based: e.g., the first in first out (FIFO), the least recently used (LRU), the least frequently used (LFU), and their variants [3]. However, the simplified rules and the ignorance of popularity make these methods hard to adapt to the request dynamics [4]. To deal with this problem, prediction-based algorithms have received more attention in recent years [5]. By making use of historical data, they extract some handcraft features to estimate the popularity of each content and replace the least popular item, i.e., the item with the lowest popularity. Nevertheless, these algorithms require significant tuning, and the insights found in one scenario may not generalize well in others. To solve these problems, some deep learning methods have been suggested in most recent work [4], [6]. By using the deep neural network (DNN), these algorithms can predict popularity without any presumptions. Unfortunately, these methods still share a key limitation: since the converge of a DNN model needs a lot of data and also a long period, the DNN model cannot detect and adapt to sudden events, such as the appearance of new hot contents.

In this paper, we propose a learning-aided exploration-exploitation (LA-E2) approach to solve the cache replacement problem. Specifically, we will highlight the following two fundamental problems. 1) How to design an approach that can strike a balance between the long-term historical information and the short-term sudden events (e.g., the appearance of hot contents); 2) how to design an approach that can generalize well to different request patterns (e.g., popularity distribution).

To address the above concerns, we argue that the cache replacement problem should be treated as not only a popularity prediction problem, but also an online exploration and exploitation process, that is, using prediction method to extract long-term popularity features from historical information, while using online E2 to deal with short-term popularity changes at the same time. The critical insight behind our approach is: from an empirical observation, we find that even though the prediction-based model may not adapt to the sudden event appearance and lead to an inaccurate prediction of the best replacement (i.e., the item with the lowest popularity), it can still use historical information to form a small subset of candidates which contains the best choice. As a result, LA-E2 then utilizes an online E2 technique to identify the most suitable choice in a real-time way. Through the above methods, LA-E2 can strike a balance between the prediction, which focuses more on long-term popularity information, and exploration, which focuses more on short-term sudden events. In summary, the main contributions are as follows:

1. We analyze the potential of a high-performance replacement approach. We deeply study how to use LA-E2 to alleviate the fundamental problems which previous algorithms

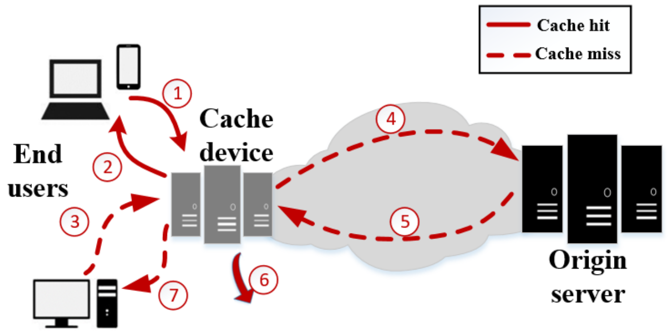


Fig. 1. A typical video caching system. The solid line shows a cache hit, while the dash line shows a cache miss.

still face.

2. We propose LA-E2, a cache replacement approach, which can both extract long-term popularity features without any predefined processing, and adapt itself to short-term sudden popularity changes. To the best of our knowledge, we are the first to combine the deep learning method with the E2 process to solve the cache replacement problem.

3. Through the extensive experiments in two real-world datasets, we find that LA-E2 can achieve state-of-the-art performance. Especially, when the cache size is small, LA-E2 outperforms the baselines by 8.4%-72.3% and 17.5%-68.7% in two datasets, respectively.

II. Related work

Content caching has been a basic network functionality in today's network architectures, such as the content delivery network (CDN) [7]–[9] and 5G networking [10]. Due to the simplicity of implementation, most currently used caching algorithms are still based on FIFO, LRU, and LFU [3]. However, the effectiveness of these algorithms requires the presumption of the request patterns (e.g., Poisson arrival), which is always problematic in the real world.

At the same time, some data-driven algorithms have been proposed in recent years. For example, in [5], authors propose a popularity-based learning algorithm for cache replacement. [11] predicts time-varying popularity through collaborative filtering. However, since these algorithms need handcraft feature engineering and pre-processing, they cannot be generalized well to deal with different scenarios.

Inspired by the success of the deep learning method, the latest work has paid attention to using DNN to solve the caching problem. In [4], authors propose an approach which directly uses long-short-term-memory (LSTM) network to predict popularity (denoted as "DeepCache"). In [6], researchers use a sequence-to-sequence model, also based on LSTM, to predict future characteristics of each content. Nevertheless, these algorithms may suffer from prediction bias due to unresponsive model update, which separates them from the optimal solution.

III. Cache replacement profiles

A. System overview

Fig 1 shows a typical caching system. As depicted, the system consists of three parts: the origin server, the cache device, and the users. The origin server is possessed by a CP and contains all video contents. The cache device is located close to users and able to be an edge server or a smart router. At each time, when a user requests a video content (①/③), it will first go to the cache device and see whether the content can be accessed. If accessible (②), the request will be directly served, which is called a cache hit; otherwise, a cache miss occurs, and the cache device will go to the origin server (④) to fetch the content (⑤), and then a replacement is needed to make room for it (⑥). Finally, the user will get served by the new content(⑦).

B. Problem formulation

Without loss of generality, we only consider one single cache device, which can be easily extended to multiple ones. Suppose that a CP has an N-contents set, which is denoted as $\boldsymbol{C} = \{1, 2, ...i, ..., N\}$, and $\boldsymbol{R} = \{R_t, t \in \mathbf{Z}\}$ represents the successive contents requested by end users. Just like [4], [5], in this paper we also regard all contents as the same size, and the cache device can only cache up to K different contents. In practice, K is much less than N. The cached contents at timeslot t are denoted as $\boldsymbol{C}\hat{(\boldsymbol{t})} = \{C_1\hat{(t)}, C_2\hat{(t)}, ..., C_K\hat{(t)}\}$.

At the same time, for each request R_t, we use an indicator $\mathbb{1}$ to identify whether it is a cache hit or a cache miss, which satisfies $\mathbb{1} \in \{0, 1\}$. When it is a cache hit denoted as $R_t \in \boldsymbol{C}\hat{(\boldsymbol{t})}$, we set $\mathbb{1} = 1$. Otherwise $\mathbb{1} = 0$, and a replacement is needed. Here we denote the evicted content as e_t, which is determined by the replacement algorithm. The objective of the problem is to get as many cache hits as possible. Formally, the caching problem is defined as follows:

$$\max_{e_t} \frac{1}{T} \sum_{t=1}^{T} \mathbb{1}_{\{R_t \in \boldsymbol{C}\hat{(\boldsymbol{t})}\}} \tag{1}$$

$$s.t. \quad \begin{cases} K \ll N, \\ e_t \in \{C_1\hat{(t)}, C_2\hat{(t)}, ..., C_K\hat{(t)}\}. \end{cases} \tag{2}$$

C. The potential of a better replacement approach

To design a better replacement approach, we first compare existing algorithms in two real-world datasets (for the completeness of paper structure, we present the profiles of the datasets in §V-A). Notably, by using the optimal solution [12], which needs to know all the future request sequence and cannot be obtained in reality, we quantify the potential gain of a better approach.

The algorithms involved in the comparison are as follows:

FIFO: The first-in-first-out algorithm which is rule-based. The algorithm will record the request time of each content. When the replacement is needed, the earliest cached item will be evicted first.

K-LRU: A variation of the least recently used (LRU) algorithm. The algorithm is that when the cache device has

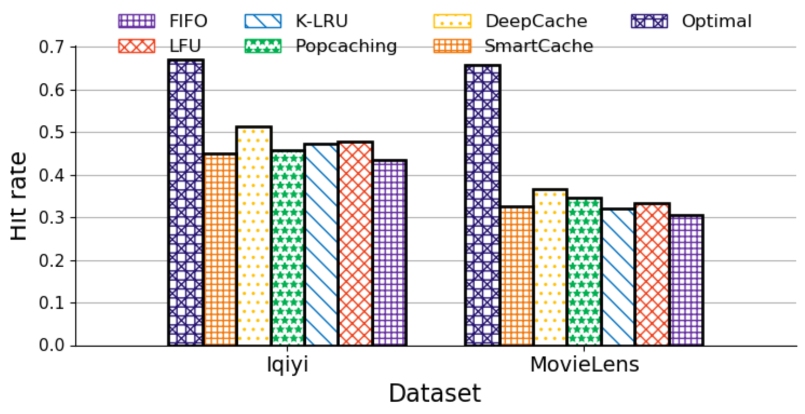


Fig. 2. Evaluation existing cache performance on two real-world datasets

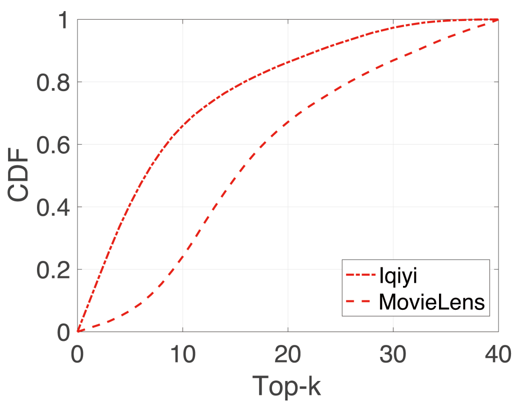


Fig. 3. The distribution of *the optimal solution is in top-k prediction results*

been full of data, the data that have not been requested for the longest time will be evicted first. Its modification K-LRU is to solve the "cache pollution problem". The core idea of K-LRU is to modify the criterion of "recently used once" to "recently used K times".

LFU: The algorithm will keep track of the requested content. By calculating the access frequency in a certain time window, the content which is requested for the least times will be evicted first.

SmartCache [13]: An algorithm which converts the caching problem as a multi-armed bandit problem, and uses the UCB method to solve it. LA-E2 is similar to this algorithm, as both of them use UCB. However, instead of only relying on the UCB algorithm to make popularity prediction, LA-E2 can both capture the long-term dynamics and short-term popularity shifts.

Popcaching [5]: This algorithm first makes feature extraction by introducing a "context-vector", and then it will maintain and update the popularity of the content with the same "context vector". In the original paper, the context vector is the normalized view count per day.

DeepCache [4]: The state-of-the-art algorithm. This algorithm predicts the popularity of each content by using the deep LSTM network. After inputting the recent requests sequence, DeepCache will evict the content with the lowest popularity.

According to their fundamental ideas, we classify them into three types: the rule-based (FIFO, K-LRU, LFU, SmartCache), the traditional prediction-based (Popcaching) [5], and the DNN-based (DeepCache) [4]. Their performance is depicted in Fig 2, and we can find the following observations:

- Comparing with traditional rule-based and prediction-based algorithms, DeepCache performs the best in both the Iqiyi dataset and MovieLens dataset.
- The state-of-the-art algorithm (i.e., DeepCache) can only reach 40%-70% performance of the optimal in hit ratio, and there is an ample space for improvement.

The outstanding performance of DeepCache in first observation can be explained: the rule-based algorithms ignore or only consider the current popularity of the content that actually cannot be directly treated as future popularity. For instance, FIFO doesn't consider future popularity, while LFU only considers the popularity in a specific time window. At the same time, we can find that comparing with LFU, Popcaching performs better in MovieLens dataset but worse in the Iqiyi dataset. It means that although Popcaching uses feature engineering to predict popularity, the features are still handcraft and cannot generalize well to deal with different request patterns. Instead, DNN-based DeepCache can make decisions almost only relying on raw input data, so that it can generalize to different datasets.

At the same time, to explain the second observation, we dive deeper into the DNN-based DeepCache. We present the distribution of *the optimal solution is in top-k prediction results* in Fig 3. We find that the DNN-based algorithm cannot narrow down to the single optimal replacement option in neither Iqiyi dataset nor MovieLens dataset. However, the optimal one is often among the top-k predicted options. As shown in Fig 3, in the Iqiyi dataset, the probability of the optimal replacement included in top-8 is more than 60%, while 5% if we only pick the option with the minimal popularity (top-1). Similar observations can also be obtained in the MovieLens dataset.

The reason behind the prediction bias of *top-1* is: the converge of a DNN model needs both a large amount of data and time as well, so the model should be updated periodically on coarse timescales. However, the request popularity is constantly changing, which means the model updated in coarse time cannot identify sudden events such as the appearance of new hot contents. Meanwhile, as the long-term popularity can be captured regardless of the update time, the DNN-based algorithm can still predict the *top-k* well based on historical information.

Inspired by the above observations, we conclude that instead of only using a DNN-based predictor, which may suffer from imprecise prediction due to its coarse time update, we should also use an online exploration-exploitation (E2) process to trim the prediction bias and adapt to the real-time popularity changes.

IV. The design of LA-E2

A. LA-E2 overview

The intuition behind combining DNN-based prediction and online E2 technique is that a prediction-only approach cannot deal with the sudden events responsively as it needs a long time to update, while an online E2-only approach cannot well capture the long-term popularity patterns as it estimates the popularity in an oversimplified way (e.g., average). Following this idea, we pick the *top-k* unpopular alternatives from the DNN-based predictor, which contain the long-term popularity information. Then, we employ the online E2 process and combines the short-term pattern to identify the item with the lowest popularity. As a result, LA-E2 is enabled to both consider long-term historical request patterns and adapt itself to short-term popularity changes. In details, LA-E2 consists of the following phases:

- **Phase A:** Collecting historical request information, which will be used to characterize the request pattern.
- **Phase B:** Inputting historical information to the DNN to predict the future popularity of each content.
- **Phase C:** Selecting the top-k candidates through the prediction results.
- **Phase D:** Online E2 on the top-k replacement options using multi-armed bandit (MAB) techniques.

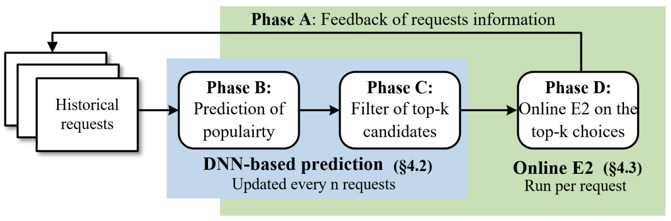
To better illustrate LA-E2, we also present the above phases in Fig 4. We can see that **Phase B, C** are enforced through DNN-based prediction (shown in blue), which are updated on coarse timescales (every n requests), and **Phase D** is executed by online E2 process (shown in grey), which is run and updated per request. We leave out **Phase A** due to its clarity. The details of **Phase B, C** are described in §IV-B, and we discuss **Phase D** in §IV-C.

B. DNN-based prediction

Phase B: Popularity prediction In this phase, LA-E2 will input the historical information collected in **Phase A** and output the popularity prediction of each content. Inspired by the recent success of the deep learning method in sequence prediction area [4], [6], we use LSTM, a widely used RNN structure, as the popularity predictor. By using the built-in forget and memory gate, LSTM can extract popularity features directly from raw input data, and generalize to different request patterns by itself as well.

We apply *softmax* function to represent the popularity distribution of each video content [4], and we use the cross-entropy to estimate the loss and train our neural network. For space limitations, we recommend readers to [4], [14] for more LSTM's training details. As a default setting, we use a 3-layer LSTM, each with 128 hidden units, and the model is updated every 1000 requests.

Phase C: Top-k candidates filtering Whenever a replacement is needed, we will evict the least popular video content. Instead of using the single replacement option in **Phase B**, LA-E2 will sort the cached items by predicted popularity in ascending order and pick the *top-k* unpopular candidates.



Phase A: Feedback of requests information
Historical requests
Phase B: Prediction of popuilarity
Phase C: Filter of top-k candidates
Phase D: Online E2 on the top-k choices
DNN-based prediction (§4.2)
Updated every n requests
Online E2 (§4.3)
Run per request

Fig. 4. Overview of LA-E2 cache replacement approach

Through this phase, LA-E2 can select out the candidates based-on long-term historical information.

C. Phase D: Online E2

Multi-armed bandit problem: We first make a brief introduction to the Multi-armed-bandit (MAB) problem. Consider a bandit with a certain number of arms, and for each arm, the reward distribution is unknown. At each timestep, one gambler needs to decide which arm should be pulled, and the object is to optimize the long-term average reward. To handle this problem, UCB (Upper Confidence Bound) algorithms are proposed. The key idea behind UCB algorithms is to always opportunistically choose the arm with the highest upper confidence bound of reward, which will naturally tend to use arms with high expected rewards (exploitation) or with high uncertainty (exploration).

UCB for cache replacement: We first illustrate that the cache replacement problem is a generalization of the classical MAB problem. Each content in the cache corresponds to an arm. In other words, there are K arms in total. At each time, LA-E2 observes the popularity of each content, which is equivalent to the reward, and makes a decision of replacement, which is equivalent to pulling an arm. Since the popularity of different contents varies with time, we use the sliding-window UCB (SW-UCB) algorithm [15] to adapt to this non-stationary process. At timeslot t, for each content i, the policy constructs a UCB, which contains two parts as follows:

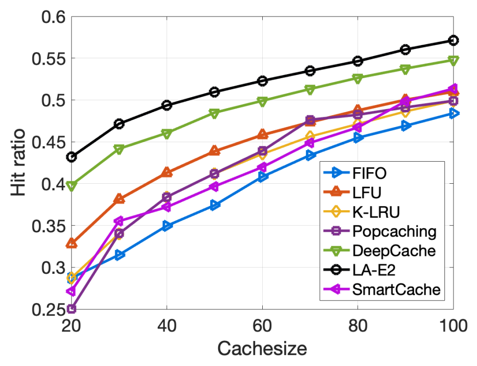
$$UCB_{t,i} = \bar{X}_t(\gamma, i) + P_t(\gamma, i) \tag{3}$$

in which the first part is the average empirical popularity of content i in a fixed-size horizon τ, and the second part is the padding function [15], which is used to represent the uncertainty of the popularity estimation. The average empirical popularity is defined as:

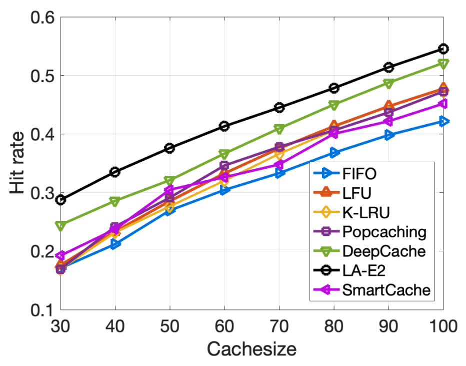
$$\bar{X}_t(\tau, i) = \frac{1}{\tau} \sum_{s=t-\tau+1}^{t} \gamma^{t-s} \mathbb{1}_{\{R_s = i\}} \tag{4}$$

here, $\gamma \in (0, 1)$ is a discounted factor. The use of γ is consistent with the fact that the content that arrives more recently is more likely to be popular in the future. Notably, the empirical popularity is not the same as the popularity calculated by the **Phase B**, as it characterizes the short-term pattern. The padding function is defined as

$$P_t(\tau, i) = B\sqrt{\frac{\log(t \wedge \tau)}{N_t(\tau, i)}} \tag{5}$$



(a) Evaluation in the Iqiyi dataset



(b) Evaluation in the MovieLens dataset

Fig. 5. LA-E2 can achieve state-of-the-art performance in both datasets. As illustrated, it outperforms the baselines by 8.3%-72.3% and 17.5%-68.7% in the Iqiyi dataset and MovieLens dataset, respectively.

k	5	7	10	15	20
Iqiyi	0.413	0.424	**0.432**	0.381	0.357
MovieLens	0.387	0.395	0.402	**0.413**	0.381

TABLE I
CACHE HIT RATE UNDER DIFFERENT *top-k*

where $t \wedge \tau$ denotes the minimum of t and τ, and B is a hyperparameter. $N_t(\tau, i)$ is the number of times that content i has been replaced in recent horizon τ, and the smaller it is, the longer content i has stayed in the cache, and it can be calculated as:

$$N_t(\tau, i) = \sum_{s=t-\tau+1}^{t} \mathbb{1}_{\{i=e_s\}} \tag{6}$$

It is notable that different from classical MAB problem which aims to choose the arm with the maximal expected reward, we aim to select (i.e., evict) the arm (i.e., the content) which has the minimum expected reward (i.e., popularity), so the parameter B should be negative, and the policy in **Phase D** can be formulated as:

$$e_t = \arg\min_i \bar{X}_t(\gamma, i) + B\sqrt{\frac{\log(t \wedge \tau)}{N_t(\tau, i)}}, B < 0 \tag{7}$$

From the formulation, we can see that the content with low popularity (i.e., small $\bar{X}_t(\gamma, i)$) and the content that has stayed in the cache for a long time (i.e., small $N_t(\tau, i)$), are likely to be replaced. The former is the exploitation process, while the latter is the exploration process. This process can be clearly interpreted: if a content suddenly becomes popular, it will be less likely to be replaced, as it has been accessed many times and is attached with a higher $\bar{X}_t$. Meanwhile, if a content has been replaced many times (i.e., a large $N_t(\tau, i)$), it should have been treated as the popular content. Since only the cached content can be replaced, it has been at least requested for $N_t(\tau, i) - 1$ times. Therefore, a large $N_t(\tau, i)$ should also be less likely to be replaced, and LA-E2 will tend to replace other contents that have not been replaced, which is an exploration. Due to the E2 process is run per request, LA-E2 can deal with the sudden popularity changes responsively.

Algorithm 1 Cache replacement using LA-E2

```
1: /* Phase A */
2: Collecting historical requests $R_{his}$.
3: if a cache replacement is needed then
4:     /* Phase B */
5:     Popularity prediction by using $LSTM$
6:     /* Phase C */
7:     Select out top-k candidates $Top$-$K$
8:     /* Phase D */
9:     for each $item_i \in TopK$-$K$ do
10:        Calculate the UCB value $ucb_i$ for $item_i$ using Eq(4)
11:    end for
12:    $item_i = \min_{item_i} ucb_i$
13: end if
```

V. EXPERIMENT

A. Dataset

Iqiyi dataset: The dataset is collected from Iqiyi[1], which is one of the most popular CPs in China. The time span of the dataset is two weeks, and it contains more than 1 billion requests from over 2 million viewers, and the unique video contents are more than 0.2 million.

MovieLens dataset: MovieLens[2] is a website in which users can rate and make comments on different movies. Inspired by the previous work [5], we also use the comment data over time to simulate request sequence.

B. Evaluation and discussion

We first discuss the influence of parameter k of *top-k*. Intuitively, a larger k indicates that LA-E2 will make the decision depending more on the online E2 process and pay more attention to the short-term popularity information; while a smaller k will enforce LA-E2 to pay more attention to long-term popularity. As a result, there will a trade-off between

[1]www.iqiyi.com
[2]movielens.org

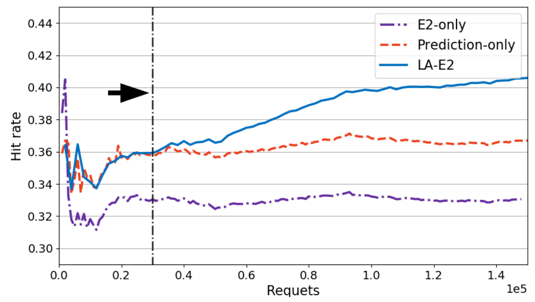


Fig. 6. The performance of LA-E2 over time in the MovieLens dataset. To better demonstrate the effectiveness of LA-E2, we also conduct the ablation study. The results illustrate that both prediction and online E2 are essential.

long-term prediction and short-term adaptation. The results are shown in Table I, and we can find that $k = 10$ and $k = 15$ are the best choices for Iiqiyi dataset and MovieLens dataset, respectively. The reason for the results can be explained: since MovieLens is from users' comments, it is more likely to be influenced by recent topics, and therefor it needs a larger k to focus more on short-term popularity changes.

We then compare the hit rate of LA-E2 and other baselines under various cache sizes, and the results are represented in Fig 5(a) and Fig 5(b). We can see that our proposed LA-E2 outperforms all the benchmarks in both the Iqiyi dataset and MovieLens dataset, thus generalizing well. Especially when the cache size is small, the performance improvement of LA-E2 can achieve 8.3%-17.5% against the second-best algorithm and 50.32%-72.3% against others,

To better illustrate LA-E2, we also conduct an ablation study in Iqiyi dataset. In Fig 6, we show the cache hit rate of three ablation groups over time, and the three groups are: E2-only group (without DNN-based popularity prediction), the prediction-only group (without online exploration and exploitation), and LA-E2 group. It is notable that before the 30000-th request comes, LA-E2 is the same as E2-group (i.e., without online E2), and as expected, their performance is almost the same. Then, both prediction and online E2 are applied to LA-E2 group, and as depicted in Fig 6, the LA-E2 group starts to increase significantly and continuously outperforms other groups. Another observation is that E2-only group significantly outperforms the other two groups at the beginning of the experiment, and then starts to suffer from performance degradation. This can be explained: at the beginning, there are no enough data for the DNN predictor to converge, and thus the prediction-only group and LA-E2 group get unsatisfactory result. After collecting enough historical requests, the strength of popularity extraction of DNN becomes obvious, and the DNN starts to work, while in this time, E2-only group keeps in a low performance as it cannot well extract popularity features from long-term historical information.

VI. Conclusion

In this paper, we propose a novel approach named LA-E2 to solve the cache replacement problem. By using a DNN predictor, LA-E2 can well extract long-term historical features, and predict popularity without presumptions. At the same time, by using online E2 process, LA-E2 can adapt itself to the constantly changing request patterns responsively. The extensive simulation on two real-world datasets proves that LA-E2 can significantly outperform state-of-the art algorithms, which illustrates the effectiveness and generalization as well.

Acknowledgement: This work is supported by NSFC under Grant 61936011, 61521002 the National Key R&D Program of China (No. 2018YFB1003703), and Beijing Key Lab of Networked Multimedia.

References

[1] Cisco Visual Networking Index, "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper," 2016.

[2] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[3] Muhammad Zubair Shafiq, Alex X Liu, and Amir R Khakpour, "Revisiting caching in content delivery networks," in *ACM SIGMETRICS Performance Evaluation Review*. ACM, 2014, vol. 42, pp. 567–568.

[4] Cong Zhang, Haitian Pang, Jiangchuan Liu, Shizhi Tang, Ruixiao Zhang, Dan Wang, and Lifeng Sun, "Toward edge-assisted video content intelligent caching with long short-term memory learning," *IEEE Access*, vol. 7, pp. 152832–152846, 2019.

[5] Suoheng Li, Jie Xu, Mihaela Van Der Schaar, and Weiping Li, "Popularity-driven content caching," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[6] Arvind Narayanan, Saurabh Verma, Eman Ramadan, Pariya Babaie, and Zhi-Li Zhang, "Deepcache: A deep learning based framework for content caching," in *Proceedings of the 2018 Workshop on Network Meets AI & ML*. ACM, 2018, pp. 48–53.

[7] George Pallis and Athena Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM*, vol. 49, no. 1, pp. 101–106, 2006.

[8] Rui-Xiao Zhang, Tianchi Huang, Ming Ma, Haitian Pang, Xin Yao, Chenglei Wu, and Lifeng Sun, "Enhancing the crowdsourced live streaming: a deep reinforcement learning approach," in *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2019, pp. 55–60.

[9] Rui-Xiao Zhang, Ming Ma, Tianchi Huang, Haitian Pang, Xin Yao, Chenglei Wu, Jiangchuan Liu, and Lifeng Sun, "Livesmart: A qos-guaranteed cost-minimum framework of viewer scheduling for crowdsourced live streaming," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 420–428.

[10] Konstantinos Poularakis, George Iosifidis, Leandros Tassiulas, et al., "Approximation algorithms for mobile data caching in small cell networks.," *IEEE Trans. Communications*, vol. 62, no. 10, pp. 3665–3677, 2014.

[11] Ejder Baştuğ, Mehdi Bennis, and Mérouane Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *arXiv preprint arXiv:1405.5974*, 2014.

[12] Richard L Mattson, Jan Gecsei, Donald R Slutz, and Irving L Traiger, "Evaluation techniques for storage hierarchies," *IBM Systems journal*, vol. 9, no. 2, pp. 78–117, 1970.

[13] Sabrina Muller, Onur Atan, Mihaela Van Der Schaar, and Anja Klein, "Smart caching in wireless small cell networks via contextual multi-armed bandits," pp. 1–7, 2016.

[14] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[15] Aurélien Garivier and Eric Moulines, "On upper-confidence bound policies for non-stationary bandit problems," *arXiv preprint arXiv:0805.3415*, 2008.