Content Recommendation Algorithm Based on Double Lists in Heterogeneous Network

Jianing Chen(✉), Xi Li, Hong Ji, and Heli Zhang

Key Laboratory of Universal Wireless Communications,
Ministry of Education, Beijing University of Posts and Telecommunications,
Beijing, People's Republic of China
{chenjianing,lixi,jihong,zhangheli}@bupt.edu.cn

Abstract. Applying recommendation algorithms in mobile edge caching can further improve the utilization of the caching and relieve the pressure of the backhaul links. The key is to capture accurate user preferences which are usually influenced by the user's request record and current request. In this paper, we propose a content recommendation algorithm based on both history request record and current interest. The content, user preferences and user's requests are modeled as vectors from multiple content dimensions. Based on user's request record, we capture the user preferences vector (Pre-Vector) by using the maximum likelihood estimation. The Pre-Vector accurately reflects user preference but has hysteresis. The user current request vector (Req-Vector) can reflect the user's current interest but its accuracy is not stable. We propose the preference-based recommendation list and the request-based recommendation list based on the Pre-Vector and the Req-Vector respectively. In order to ensure the accuracy of the recommendation list, the final recommendation list is generated based on the Pre-Vector and the Req-Vector's cosine similarity. The simulation results show that, the proposed algorithm has improved caching hit rate compared with existing recommendation algorithms.

Keywords: Recommendation algorithm · User preferences · Maximum likelihood estimate · Multiple content dimensions

1 Introduction

Soaring mobile data traffic presents a big challenge for the 5th-generation (5G) mobile communication system. According to the Ericsson Mobile Report 2019 [1], the global average monthly mobile data traffic has reached 29.0 EB in 2019 Q1, and will reach nearly 164.5 EB at the end of 2024. Mobile edge caching [2,3] is one of the promising solutions to solve mobile data traffic congestion. It stores content at the edge of the networks, which shortens the physical distance between users and content, and then alleviates the pressure of the backhaul.

H. Gao et al. (Eds.): ChinaCom 2019, LNICST 313, pp. 140–153, 2020.
https://doi.org/10.1007/978-3-030-41117-6_12

Applying recommendation algorithms in the mobile edge caching is a viable solution to further improve the caching utilization and user experience, and it will also relieve the pressure of the backhaul links. In a heterogeneous network of 5G, in addition to the macro base station (MBS), many small base stations (SBS) [4,5] are densely deployed, which makes the caching space in the network edges much larger compared to the traditional cellular network. This provides more options for content recommendations, which requires an effective algorithm for higher recommendation hit rate.

The research of the recommended algorithm originated from the Internet field in the 1990s. Later, with the prosperity of e-commerce, the research of recommended algorithms has ushered in a boom. Jiang et al. [6] analyzed the key technology related to personalized recommendation in distance education, and proposed multiple recommendation algorithms. Felfernig et al. [7] analyzed the applicability of group recommendation to requirements prioritization, and made the application of group decision heuristics in the context of requirements prioritization. Shaikh et al. [8] listed various limitations of the current recommendation methods and believed that integration of semantic in recommendation techniques could provide better recommendation with proposed system.

In general, the design of the recommendation algorithm inevitably takes into account user preferences [9,10]. Also, many researchers have studied the content recommendation algorithm with additional factors, such as the distance between the user and the content, the user's social relationship, the base station's limited storage space, the association between multiple contents and so on. Wang et al. [11] considered the social relationship of the users, proposed a recommendation system for cached-enabled mobile social network, which could maximize the traffic offloading ratio. Some researchers considered the content association in mobile edge caching recommendation. Kastanakis et al. [12] proposed a content-related approach that enabled joint caching and recommendation in mobile edge caching. Some researchers apply reinforcement learning to the caching recommendation algorithm. Guo et al. [13] developed a Q-learning based algorithm to dynamically replace and recommend the files in the caching of the BS. Yang et al. [14] jointly optimized content caching and recommendation at BSs to maximize the caching gain, and proposed a hierarchical iterative algorithm to solve the optimization problem.

The recommendation algorithm can be classified into two types: the memory recommendation algorithm and the memoryless recommendation algorithm. The memory recommendation algorithm mainly summarizes the user preference based on the user's long-term request record as the recommendation basis. The memoryless recommendation algorithm extracts keywords or key information for the user's current request, and then recommend contents with the similar keywords. Time-varying user preferences make the estimation of user preferences have hysteresis in the condition of only using the memory recommendation algorithm. While only using the memoryless recommendation algorithm to estimate the user preferences has inaccuracy, especially when the user record is very limited. There is very attractive requirement to combine these two types of recommended algorithms to overcome hysteresis and inaccuracy, and then improve the utilization of the caching contents and user experiences.

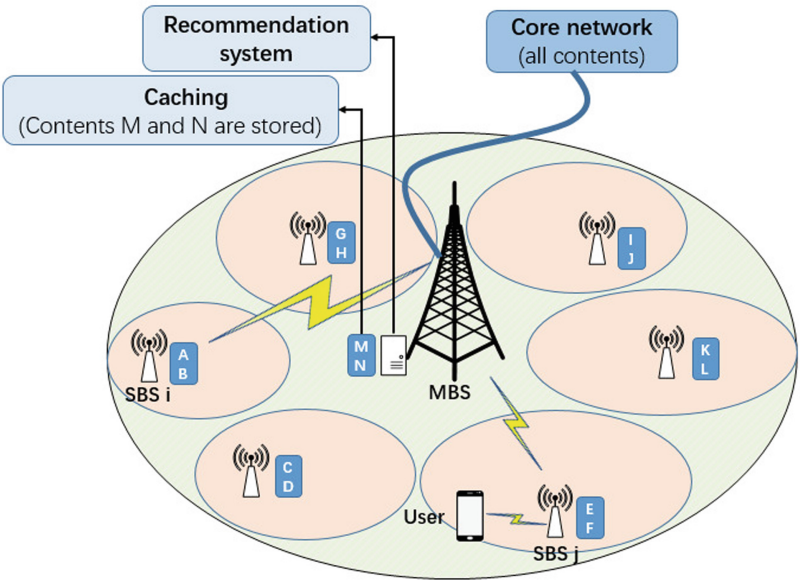

Fig. 1. System model.

In this paper, we investigate the time-varying features of user preference in heterogeneous network, and propose a content recommendation algorithm based on double lists (CRADL) to optimize the recommended hit rate. We assume several SBSs and a MBS collaboratively store and recommend contents to the user in a typical scenario. The optimization problem to maximize the caching recommended hit ratio is analyzed and formulated. The user preference vector (Pre-Vector) is captured by maximum likelihood estimation of user's request record. Then we design the preference-based recommendation list and the request-based recommendation list based on Pre-Vector and the user current request vector (Req-Vector). In order to ensure the accuracy of the recommendation, we need to find a balance between Pre-Vector and Req-Vector. Based on their cosine similarity, we selectively integrate the two lists into one list and recommend to the user. Simulation results show that the proposed algorithm has the better performance in the hit rate compared to the existing caching algorithms in heterogeneous network.

2 System Model

As shown in Fig. 1, we deploy a number of BSs in a region. The MBS located in the middle of the region. The MBS manages these SBSs and knows what content is cached in these SBSs. And we install the recommendation system application in the MBS. We make these BSs collaboratively store the popular content. The MBS manages $m-1$ SBSs in a the region, all BSs are represented

by sets $S = \{s_1, s_2, s_3, s_4, ..., s_m\}$. The user will select the closest BS to access. To simplify, we set the caching space size of each BS is c. The core network has all the contents, assuming that the total number of the contents is k, these contents are represented by $F = \{F_1, F_2, F_3, F_4, ..., F_k\}$. In the case of multiple users, the probability that these contents are requested is subject to the Zipf distribution. The probability P_i of the file ranked i is requested by users is:

$$P_i = \frac{1/i^r}{\sum_{p=1}^{k}(1/p^r)} \tag{1}$$

where r is the zipf index, $r > 0$.

To simplify, we set the size of each content is ξ, the BS j can store l files, the cache space of BS j is always bigger than or equal to the total size of the contents it stores. Then we get a constraint:

$$C1 : \sum^{l} \xi \leqslant c \tag{2}$$

And in reality, the number of cached contents of these m BSs is much smaller than the number of contents of the entire network. We get the another constraint:

$$C2 : mc \leqslant C_{network} \tag{3}$$

where $C_{network}$ is the total number of contents in the core network.

For a single user, we believe that there is a strong connection with the user requests is the user preferences. In order to fully describe user preferences, we intend to measure user preferences from D dimensions, the user preferences vector is represented as: $U = \{u_1, u_2, u_3, u_4, ..., u_D\}$. Similarly, the content F_j vector is modeled as: $F_j = \{f_1, f_2, f_3, f_4, ..., f_D\}$.

3 Problem Formulation

For individual users, the requesting content is more susceptible to their personal preferences. Some users like to listen to Hip-pop, while others like light music and so on. In [15], the author measures a content from multiple dimensions, quantifies the content in each dimension. And the user's request is normally distributed in each dimension. Inspired by [15], we decide to model the user's request as: in the case of unchanging user preferences, the content requested by the user follows a normal distribution in each dimension and the mean of the vectors of these request records is equal to the user preferences vector. Suppose the user sends x requests ($q = \{q_1, q_2, q_3, ..., q_x\}$) with the same user preferences, we get third constraint:

$$\begin{aligned} C3 : \{u_1, u_2, u_3, u_4, ..., u_D\} &= \frac{\sum_{i=1}^{x} q_i}{x} \\ &= \frac{\sum_{i=1}^{x} \{f_1, f_2, f_3, f_4, ..., f_D\}_i}{x} \end{aligned} \tag{4}$$

The user preferences are time-varying. We quantify the user preference duration as the number x of requests, x is a random positive integer value. After the user sends x requests, the user preferences may change to a new random user preferences. The probability that the user preference changes is β, $0 \leqslant \beta \leqslant 1$.

There are two situations that trigger the recommendation system. One is that when the user finishes watching a video or listens to a song, the recommended system in MBS will recommend relevant content to the user. The other is that when the content requested by the user is not stored in the BS and the collaborative BSs, the recommended system will recommend the cached content to the user.

To recommend the user what the user wants, we need to define the similarity between the content. The similarity between content i and content j is defined as:

$$P_{i,j} = \frac{F_i \cdot F_j}{\|F_i\| \cdot \|F_j\|} \tag{5}$$

The user acceptance of recommended content is related to content similarity and content popularity. When the cached content j is recommended to the user, the probability that a user accepts content j is defined as:

$$P_{u,j} = \omega * (\frac{U \cdot F_j}{\|U\| \cdot \|F_j\|})^{\varphi} \tag{6}$$

where ω is the popularity impact coefficient, $\omega = (\frac{P_j}{P_1})^{\phi}$. ϕ is the Popularity impact index, $\phi = 0$ means the user is not affected by popularity. φ is the cosine matching impact index, $\varphi = 0$ means the user is only affected by popularity. $\frac{U \cdot F_j}{\|U\| \cdot \|F_j\|}$ is the user preferences vector and the recommended content vector's cosine matching value.

The recommendation system will recommend z cached contents to the user in the form of a list. In order to conform to human aesthetics, generally 3 to 8 contents are appropriate. Then the probability that the user accepts the recommendation can be expressed as:

$$\begin{aligned} P_{U_t,list} &= 1 - \prod_{i=1}^{z} (1 - p_{ut,i}) \\ &= 1 - \prod_{i=1}^{z} [1 - (\frac{P_i}{P_1})^{\phi} * (\frac{U_t \cdot F_i}{\|U_t\| \cdot \|F_i\|})^{\varphi}] \end{aligned} \tag{7}$$

where U_t is the current user preferences. We hope to maximize the effectiveness of the cached content, that is, to find the recommended hit rate maximum, $max\{P_{U_t,list}\}$.

4 Content Recommendation Algorithm Based on Double Lists

4.1 User Preferences Estimating

To maximize $P_{U_t,list}$, we need to estimate the user preference as accurately as possible from user's records. The estimated user preferences is represented by U_o and it is generated by the user's app, and it is sent to the MBS when a request is sent. Suppose the mobile app counts the user's recent N requests. In any of these D dimensions, the quantized values of these requests are normally distributed. We take the i-th dimension of the N records, denoted by $Q_{N,i} = \{f_{1,i}, f_{2,i}, f_{3,i}, ..., f_{N,i}\}$.

The Maximum Likelihood Estimation (MLE) can be used to estimate the parameters of the model. The goal of MLE is to find a set of parameters that maximize the probability that the model will produce observations:

$$argmax\left\{p(Q_{N,i}; \mu_i, \sigma^2)\right\} \tag{8}$$

where $p(Q_{N,i}; \mu_i, \sigma^2)$ is a likelihood function, indicating the probability of observation data appearing under the parameter μ_i. We assume that each observation is independent and then:

$$p(f_{1,i}, f_{2,i}, f_{3,i}, ..., f_{N,i}; \mu_i) = \prod_{k=1}^{N} p(f_{k,i}; \mu_i, \sigma^2) \tag{9}$$

For convenience, Eq. 9 becomes a logarithmic function form (it obeys normal distribution):

$$\begin{aligned} p(Q_{N,i}; \mu_i, \sigma^2) &= ln(p(Q_{N,i}; \mu_i, \sigma^2)) \\ &= -\frac{Nln(2\pi)}{2} - \frac{Nln(\sigma^2)}{2} - \frac{1}{2\sigma^2}\sum_{k=1}^{N}(f_{k,i} - \mu_i)^2 \end{aligned} \tag{10}$$

Find the derivative of Eq. 10 and let it equal 0:

$$\frac{\partial ln(p(Q_{N,i}; \mu_i, \sigma^2))}{\partial \mu} = \frac{1}{2\sigma^2}\sum_{k=1}^{N}(f_{k,i} - \mu_i) = 0 \tag{11}$$

Finally, we get:

$$\mu_i^\star = \overline{Q_{N,i}} = \frac{1}{N}\sum_{k=1}^{N} f_{k,i} \tag{12}$$

Similarly, we can get the values of D dimensions and get the estimated vector μ:

$$\mu = \{\mu_1^\star, \mu_2^\star, \mu_3^\star, ..., \mu_D^\star\} \tag{13}$$

There may be highly deviating requests in this set of data, which would be removed by using the Grubbs criteria. After removing highly deviating requests, calculate μ again, and that is the estimated user preferences U_o (Pre-Vector):

$$U_o = \{\mu_1, \mu_2, \mu_3, ..., \mu_D\} \tag{14}$$

4.2 Content Recommendation Algorithm Based on Double Lists

Now we have U_o (Pre-Vector), and the quantized current request is represented as F_c (Req-Vector). After the MBS obtains the user's request F_c (Req-Vector) and U_o (Pre-Vector), the MBS calculates the cosine similarity degree between U_o and all cached content. According to the descending order of the cosine similarity degree, we can get the preference-based recommendation list (P-list). Similarly, we use the same method to calculate the cosine similarity degree between F_c and all cached content, finally we have the request-based recommendation list (R-list). The two lists is just the initial formation of the recommended lists. When the requested content belongs to hot content, we are not sure whether the user likes the type of content or just the hot content. We introduce popularity to tailor the two lists after matching. The popularity value exists only on the network side and the MBS, and the MBS periodically updates the popularity value of the content cached in the region. The matching degree is calculated as follows:

$$\begin{aligned} P-list : p_{uo,j} &= \omega * (\frac{U_o \cdot F_j}{\|U_o\| \cdot \|F_j\|})^{\varphi} \\ R-list : p_{fi,j} &= \omega * (\frac{F_i \cdot F_j}{\|F_i\| \cdot \|F_j\|})^{\varphi} \end{aligned} \tag{15}$$

After that, we get two lists. We descend the P-list: $\{p_{uo,1}, p_{uo,2}, ...\}$, $P_{uo,1}$ indicates the matching degree of the content with the highest matching degree in P-list. Similarly, we have the R-list: $\{p_{fi,1}, p_{fi,2}, ...\}$.

The cosine similarity between the user preferences vector U_0 and the user's requested content F_i is:

$$q_{uo,fi} = \frac{U_o \cdot F_i}{\|U_o\| \cdot \|F_i\|} \tag{16}$$

If the user preferences vector U_o is similar to the request F_i, $q_{uo,fi}$ is high, we would consider the P-list as a recommended reference. If the user preferences vector U_o is far from the request F_i, $q_{uo,fi}$ would be low, we would consider the R-list as a recommended reference. In order to flexibly recommend caching to users, we need to merge the two lists.

Knowing that a recommended list contains z content, we divide the cosine similarity of U_o and F_i evenly into $z+1$ levels. In this paper, we assume that a recommendation list contains 4 content, $z = 4$. According to different cosine matching degrees, we take N_{uo} top contents from U-list and N_{fi} top contents from R-list, to form a new recommendation list, where $N_{uo} + N_{fi} = z$.

For example, When the cosine similarity is in level V, $0.8 \leqslant q_{uo,fi} \leqslant 1$, we believe that the user preferences have not changed at this time, and still use the top four contents of P-list as recommendations. As shown in Table 1.

It should be pointed out that in order to avoid duplication, if a content is in the top ranking in both lists, then after the content is taken in U-list, the content is not considered in R-list.

Table 1. The division of $q_{uo,fi}$ $(z = 4)$

Level	Level V	Level VI	Level III	Level II	Level I
$\boldsymbol{q_{uo,fi}}$	$[0.8, 1]$	$[0.6, 0.8)$	$[0.6, 0.4)$	$[0.4, 0.6)$	$[0, 0.2)$
$\boldsymbol{N_{uo}}$**:**$\boldsymbol{N_{fi}}$	4:0	3:1	2:2	1:3	0:4

After synthesizing the two lists into one list, we can get the theoretical recommendation acceptance probability:

$$\begin{aligned} P_{list} &= 1 - \prod_{j=1}^{z} (1 - p_{list,j}) \\ &= 1 - [\prod_{j=1}^{N_{uo}} (1 - p_{uo,j})] * [\prod_{k=1}^{N_{fi}} (1 - p_{fi,k})] \end{aligned} \tag{17}$$

5 Simulation Results and Discussions

In this section, we will fully evaluate the performance of the CRADL from different angles. And we will analyze the impact of several important parameters on CRADL performance. For comparison, we also simulate the other three recommended algorithms: the hottest caching recommendation algorithm (Hot-Algorithm), caching recommendation algorithm based on user preference (Pre-Based) and caching recommendation algorithm based on user's current request (Req-Based). The Hot-Algorithm means that the MBS would always recommend the hottest contents to the user. The Pre-Based means that the MBS would recommend contents to the user only based on the user's preference. The Req-Based means that the MBS would recommend contents to the user only based on the user's current request.

We consider setting parameters in the range of $200 * 200$ m: only 1 user, number of BSs $M = 7$, and each BS's default caching capacity $c = 30$ contents. This creates a relatively dense network in which the SBSs are randomly distributed. The core network has all the contents, $C_{network} = 500$ *contents*. The Zipf index is set to $r = 0.9$ by default. The duration of user preference is quantified as: the number of requests to continue in the case of constant preference, the number of requests is random values 1–20. The probability that the user preference changes is $\beta = 0.1$. The Pre-Based algorithm and CRADL consider $N = 10$ user

historical requests. We set the Popularity impact index $\phi = 1$ and the cosine matching impact index $\varphi = 1$. And we assume that the user does not move and selects the nearest base station access.

In order to get close to the reality, we followed the qqmusic's classification to the music: electronic music, pop music, Chinese Style Music, ballad, Hip-Hop, country music and so on, a total of 15 categories, which means vector dimension $D = 15$. We assume that the core network has a total of 500 contents, $k = 500$. And we quantified the top 500 songs of qqmusic and got 500 15-dimensional vectors of the top 500 songs. The quantitative criteria are as Table 2 (in terms of a single dimension such as Hip-Hop).

Table 2. The quantitative criteria (Hip-Pop as an example)

Degree	Not Hip-Pop	A little Hip-Pop	Normal Hip-Pop	Very Hip-Pop	Super Hip-Pop
Values	0–0.2	0.2–0.4	0.4–0.6	0.6–0.8	0.8–1

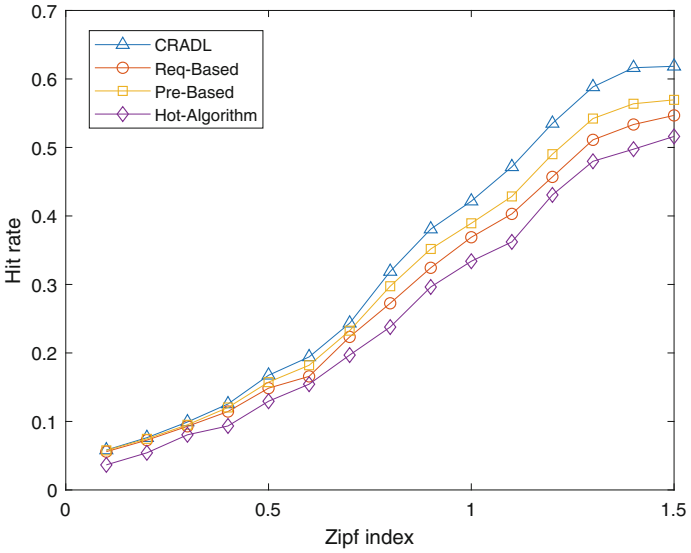

Fig. 2. The recommended hit rate changes as the Zipf index changes.

Figure 2 shows that as the Zipf index changes, the probability that the recommendation list is accepted by the user is on the rise. Because as the Zipf index increases, the probability of users accepting recommended content is also increasing, which results in an increase in the hit rate of the four methods. When calculating user preferences, Pre-Based refers to the user's last 10 requests, while Req-Based simply treats the request content vector (F_c) as a user preference

vector. So the user preferences calculated by Pre-Based are closer to real user preferences than Req-Based, thus the Pre-Based's hit rate is higher than the Req-Based's. And CRADL is combines the advantages of the Pre-Based and Req-Based, so CRADL has the highest hit rate.

The Req-Based algorithm takes into account the user's current request, so the acceptance probability of the Req-Based algorithm is larger than the Hot-Algorithm. The Pre-Based algorithm considers the user $N = 10$ historical requests, the Pre-Based algorithm can accurately capture the user's preference, so it has higher hit rate. And CRADL has always the highest hit rate. Because CRADL considers the user's current request and the user preference, the appropriate caching recommendation list can be flexibly generated when the current request differs from the user's preferences.

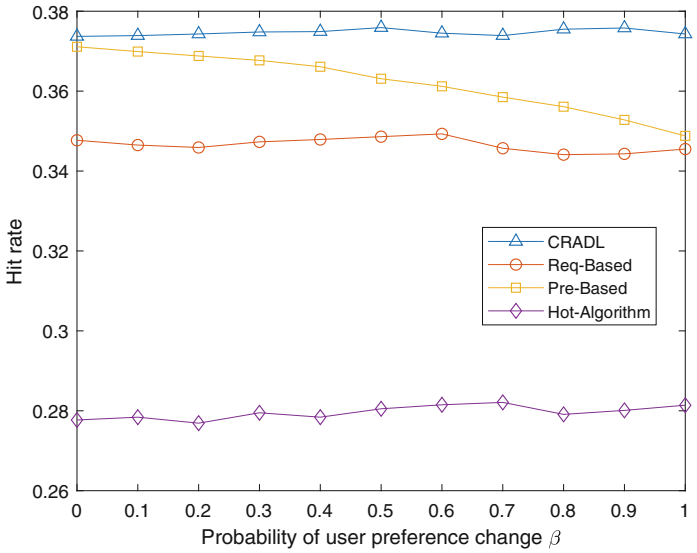

Fig. 3. The recommended hit rate changes as the probability of user preference change β changes.

Figure 3 shows that as the user preference change probability β changes, the probability that the hit rate of the recommended list also changes. Both Req-Based algorithm and Hot-Algorithm are memoryless, so changes in the probability of change in user preference can't affect the performance of Req-Based algorithm and Hot-Algorithm. For the Pre-Based algorithm, as the increase in β means that the user preference changes more frequently, the user preference estimated by the historical request matches the current request less. So the hit rate of Pre-Based algorithm would drop, from 37.1% to 34.8%. CRADL can consider both user preference and the current request, it can adapt to changes in user preference rate and maintain a 37–38% caching recommendation hit rate.

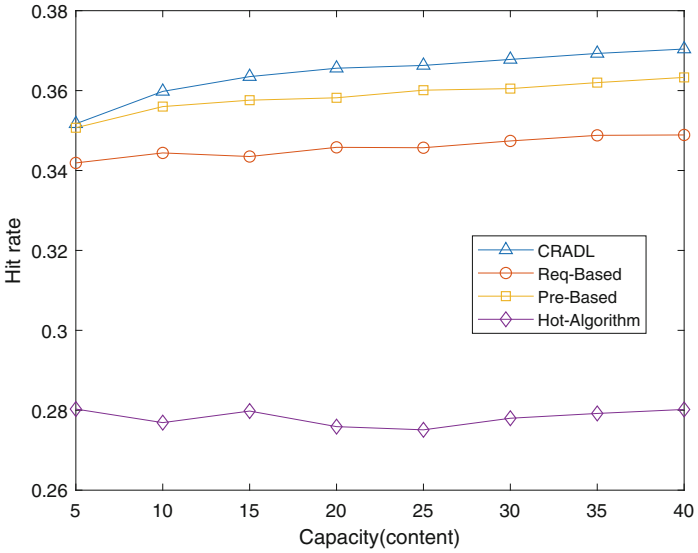

Fig. 4. The recommended hit rate changes as BS's caching capacity c changes.

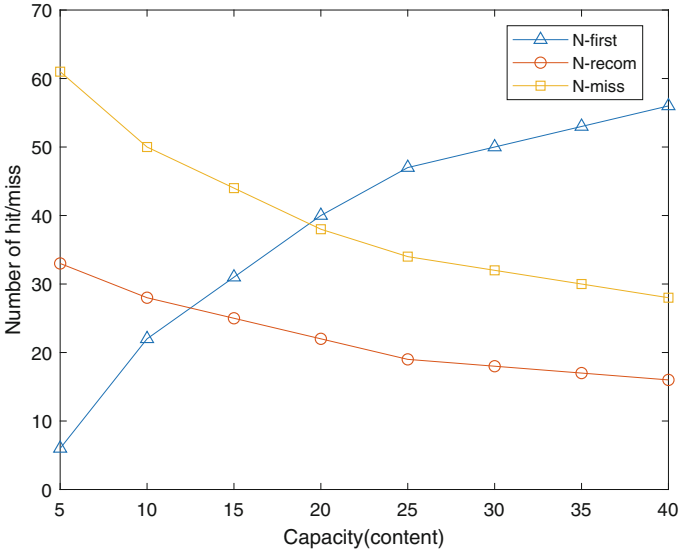

Fig. 5. The caching first hit rate changes as BS's caching capacity c changes.

Figure 4 shows that as BS's caching capacity c changes, the probability that the hit rate of the recommended list changes slowly. In the previous chapters, we know that the probability of acceptance of users is affected by the popularity of the content. For the content at the end of the popularity ranking, even if the cosine matching degree is high, the probability of being accepted is not high because the popularity is too low, so the content with highly popular and highly cosine-matched is easier to accept. That means, even if the caching space of the BSs is increased, the influence of the long tail effect is too small, which makes the recommended hit rate increase little.

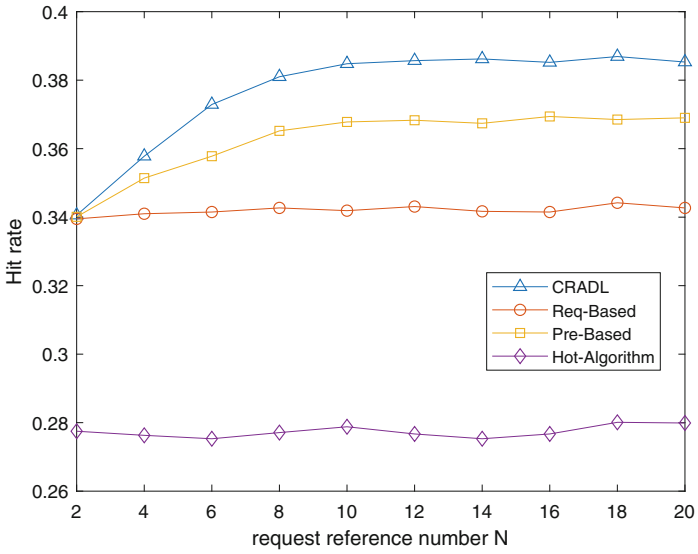

Fig. 6. The recommended hit rate changes as the number of historical records reference N changes.

But as for the number of the caching first hits (when caching recommendation system is not triggered), with the expansion of the BSs' caching space, the number of the caching first hits has a big improvement, as shown in Fig. 5. In each set of tests, the user generated 100 requests. The number of successful recommendations N_{recom} plus the number of recommended failures N_{miss} plus the number of caching first hits N_{first} equals 100, $N_{first} + N_{recom} + N_{miss} = 100$.

Figure 6 shows that as the number of user record N changes, the Pre-Based algorithm and CRADL's recommended hit rate has improved, and then stabilized. Both Req-Based algorithm and Hot-Algorithm are memoryless, so the number of historical records reference N can't affect the performance of Req-Based method and Hot-Algorithm. As can be seen from Fig. 6, when $N < 10$, the Pre-Based algorithm's hit rate and CRADL's hit rate begin to increase. When $N = 10$, the hit rate of CRADL is approximately 38.5% of the saturation value,

and the saturation value of the Pre-Based algorithm is 36.9%. It can be determined that in our scenario, the accuracy of the user preference can be estimated when $N > 10$. When $\beta = 0.1$, on average, every time a user sends 10 requests, he will change his preferences once. So when $N > 1/\beta = 10$, the curves for CRADL and Pre-Based are flat.

6 Conclusion

In this paper, we studied the mobile edge caching recommendation system in heterogeneous network. In addition to popularity, the content requested by the users and the probability of the users accepting the recommended cached content were often highly correlated with their own preferences. Based on user preference and popularity, a question of maximizing the caching hit ratio was formulated. And a content recommendation algorithm based on multiple content dimensions was proposed. We quantified the user preferences, content, and requests from multiple dimensions. User preferences vector was derived by maximum likelihood estimation and Grubbs criterion. Then two recommended lists were generated based on user current request vector and user preferences vector. The preference-based recommendation list reflected the recent user preferences, the request-based recommendation list reflected the current user preferences. Finally, we merged two lists based on the two vectors' cosine matching. Compared with the other three recommended methods, the method we proposed could achieve a highest hit rate, whether there was a change in BS's capacity or a change in Zipf index or a change in probability of user preference change. In our scenario, we set the user preference change to be random, but in reality it was not. In the future work, we will explore the impact of $q_{uo,fi}$ division on the hit rate optimization problem, and we will investigate the connection of user preference to popularity, or other elements.

Acknowledgment. This work is jointly supported by National Natural Science Foundation of China (Grant No. 61771070), and the National Natural Science Foundation of China (Grant No. 61671088).

References

1. Ericsson Mobility Report 2019, June 2019. https://www.ericsson.com/en/mobility-report/reports/june-2019
2. Guo, F., Zhang, H., Ji, H., Li, X.: An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. IEEE/ACM Trans. Netw. **26**(6), 2651–2664 (2018)
3. Parvez, I., Rahmati, A., Guvenc, I., Sarwat, A.I., Dai, H.: A survey on low latency towards 5G: RAN, core network and caching solutions. IEEE Commun. Surv. Tutor. **20**(4), 3098–3130 (2018)
4. Li, J., Chu, S., Shu, F., Wu, J., Jayakody, D.N.K.: Contract-based small-cell caching for data disseminations in ultra-dense cellular networks. **18**(5), 1042–1053 (2019)

5. Liu, Y., Yu, F., Li, A., Ji, H., Zhang, H., Leung, V.: Joint access and resource management for delay-sensitive transcoding in ultra-dense networks with mobile edge computing. In: 2018 IEEE International Conference on Communications (ICC), pp. 1–6 (2018)
6. Jiang, P.R.Y., Zhan, H., Zhuang, Q: Application research on personalized recommendation in distance education. In: 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), vol. 13, pp. 357–360 (2010)
7. Felfernig, A., Ninaus, G.: Group recommendation algorithms for requirements prioritization. In: 2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE), pp. 59–62 (2012)
8. Shaikh, S., Rathi, S., Janrao, P.: Recommendation system in E-commerce websites: a graph based approached. In: 2017 IEEE 7th International Advance Computing Conference (IACC), pp. 931–934 (2017)
9. Chatzieleftheriou, L.E., Karaliopoulos, M., Koutsopoulos, I.: Caching-aware recommendations: Nudging user preferences towards better caching performance. In: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, pp. 1–9 (2017)
10. Sermpezis, P., Giannakas, T., Spyropoulos, T., Vigneri, L.: Soft cache hits: improving performance through recommendation and delivery of related content. IEEE J. Sel. Areas Commun. **36**(6), 1300–1313 (2018)
11. Wang, Y., Ding, M., Chen, Z., Luo, L.: Caching placement with recommendation systems for cache-enabled mobile social networks. IEEE Commun. Lett. **21**(10), 2266–2269 (2017)
12. Kastanakis, S., Sermpezis, P., Kotronis, V., Dimitropoulos, X.: CABaRet: leveraging recommendation systems for mobile edge caching. In: 2018 MECOMM, pp. 1–6 (2018)
13. Guo, K., Yang, C., Liu, T: Caching in base station with recommendation via Q-learning. In: 2017 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (2017)
14. Liu, D., Yang, C.: A learning-based approach to joint content caching and recommendation at base stations. In: 2018 IEEE Global Communications Conference (GLOBECOM), pp. 1–7 (2018)
15. Lee, M., Molisch, A.F., Sastry, N., Raman, A.: Individual preference probability modeling and parameterization for video content in wireless caching networks. IEEE/ACM Trans. Netw. **27**(2), 676–690 (2019)