Bayesian Poisson Factorization with Side Information for User Interest Prediction in Hierarchical Edge-Caching Systems

Sajad Mehrizi, Symeon Chatzinotas

***Abstract*—Edge-caching is an effective solution to cope with the unprecedented data traffic growth by storing contents in the vicinity of end-users. In this paper, we formulate a hierarchical caching policy where the end-users and cellular base station (BS) are equipped with limited cache capacity with the objective of minimizing the total data traffic load in the network. The caching policy is a nonlinear combinatorial programming problem and difficult to solve. To tackle the issue, we design a heuristic algorithm as an approximate solution which can be solved efficiently. Moreover, to proactively serve the users, it is of high importance to extract useful information from data requests and predict user interest about contents. In practice, the data often contain *implicit feedback* from users which is quite noisy and complicates the reliable prediction of user interest. In this regard, we introduce a Bayesian Poisson matrix factorization model which utilizes the available side information about contents to effectively filter out the noise in the data and provide accurate prediction. Subsequently, we design an efficient Markov chain Monte Carlo (MCMC) method to perform the posterior approximation. Finally, a real-world dataset is applied to the proposed proactive caching-prediction scheme and our results show significant improvement over several commonly-used methods. For example, when the BS and the users have caches with storage of 25% and 10% of the total contents size respectively, our approach yields around 8% improvement with respect to the state-of-the-art approach in terms of caching performance.**

***Index Terms*—Content caching, Combinatorial optimization, Bayesian modeling, Poisson matrix factorization**

I. Introduction

Edge-caching is a promising approach to alleviate the unprecedented data traffic growth on back-haul links in cellular networks [1]. It leverages the observation that only a few number of popular contents are responsible for the majority of data traffic. Therefore, proactively caching these finite popular contents near the users can substantially relieve the overloaded network traffic and improve users quality of experiences (QoEs).

Next-generation of networks may deploy a hierarchy of cache units which are connected in a specific topology, and cooperate to address users' requests. This cooperative structure can increase the performance and save bandwidth significantly [2]. A special case is where the end-users and the BS are equipped with limited cache memories and form a hierarchical topology. By proactively caching contents at the user' cache, she can fetch the contents instantly with no delay. Therefore, user QoE boosts dramatically.

A full-fledged hierarchical cache management system requires accurate user preference prediction about contents. This task is not a trivial task and there are two main challenges that need to addressed to obtain accurate prediction. Firstly, the data requests at user-level is extremely sparse and scarce [3]. Secondly, in practice, users often do not *explicitly* provide feedback about which contents they like/dislike and only the number of requests for each content is observed. This type of data is referred to as *implicit feedback* and is quite noisy and very challenging to obtain reliable and useful patterns [4]. For example, a user who did not request a movie to watch might have done so because she dislikes the movie or just because she did not know about the movie or was unavailable to watch it. Moreover, a request for a movie from the user does not necessarily indicate she likes the movie, i.e., she might be disappointed after watching the movie.

Matrix factorization (MF), which has shown great success in recommendation systems, can be employed to address the issues [4], [5]. MF is a powerful approach which can efficiently deal with the data sparsity and filter out the noise in the data through learning the user-content interactions and provide accurate prediction. Therefore, our main focus in this work is to develop an efficient MF for user interest learning for hierarchical caching.

A. Prior Work

Real-world caching systems usually deploy simple caching policy strategies that refresh the cache continuously during the delivery phase. Prevalent examples are Least Recently Used (LRU) or Least Frequently Used (LFU) [6]. Albeit simple, they often are designed for single cache and are not suitable for complicated network topologies. Moreover, they disregard patterns in content requests therefore do not have satisfactory performance.

Recently, to boost the caching gain, wide variety of polices have been designed with different goals for various network architectures. For instance, caching polices at BS-level are designed to minimize backhual and transmit power costs [7], [8], maximize content delivery rate [9] and maximize the amount of traffic load supported by the network [10]. At user-level, different polices are investigated to maximize the offloading gain of device to device (D2D) networks [11], minimize the average download latency of D2D networks [12] and minimize the energy consumption [13]. More recently, caching polices at unmanned aerial vehicle (UAV) are designed to maximize users' throughput [14], and minimize caching and content retrieval costs [15]. The majority of works assumed

user interest about contents is perfectly known in advance. However, user interest is unknown in practice and needs to be predicted from data requests.

To predict future content requests at BS-level, linear models in [16], [17] and non-linear models using deep neural networks in [18]–[21] have been proposed. The developed models however may not predict the user-level requests accurately. In other words, the models require huge number of historical data which is not the case at user-level. Learning user-level interest is important for various polices e.g., user-level and D2D caching [12].

More recently, inspired by the impressive success of MF in recommendation systems, MF has been used in caching systems for user preference learning [1], [22]–[25]. Traditional MF models assume that the data is generated based on Gaussian likelihood and learning is preformed using maximum a posteriori (MAP) or Bayesian approaches. There are two main issues with Gaussianity assumption which undermine its performance. Firstly, it assumes the data is continuous and therefore puts zero mass on integer values. However, user requests are naturally count data and Gaussian model ignores this. Secondly, it treats the requested and non-requested values (non-zero and zero values in the user-content matrix) equally. Since, in practice, the user-content matrix is sparse with large number of zeros, the Gaussian model puts more emphasis on zero values which leads to overestimation of user activity distribution [26].

To mitigate the aforementioned issues, the authors in [26] proposed a hierarchical Poisson factorization (HPF) model suitable for count, sparse and massive data; and it provides significant improvement in terms of accuracy and computational complexity with respect to Gaussian factorization models. The model can yield inference algorithms which scale linearly in terms of non-zero observations. This indicates that latent structures can be effectively learned for massive sparse data. The HPF model has been extended to Poisson tensor factorization to capture time-varying nature of content popularity in caching systems in [27], [28].

An important limitation of the HPF model is that it cannot exploit the available side information which content are associated with. For example, for movie contents, we may know their genres, release dates and etc. Since users request contents based on their features, it is expected that similar contents in their feature space have similar request patterns in the observation space [29]. Incorporating this prior into the model can tackle sparse data requests to improve the prediction accuracy.

B. Contributions

The present paper introduces an efficient Poisson MF which can exploit the content features to enhance the prediction accuracy and caching performance. In particular, our contributions can be summarized as follows:

- We formulate a hierarchical caching policy to minimize the overall data traffic in the network for transmitting contents to the end-users. The policy is a combinatorial programming problem which is NP-hard and computationally expensive to solve in practice, especially when the number users or contents is large. To overcome the issue, we propose a low complexity heuristic method to approximate the solution of the original problem.
- To proactively serve users, we introduce a Bayesian Poisson factorization model that enhances the prediction accuracy of user interest about contents by simultaneously exploiting the content features and historical data requests.
- We then perform posterior approximation using MCMC sampling method which can be efficiently implemented. Thanks to the property of Poisson factorization, the inference problem scales linearly with the number non-zero requests.
- Throughout our simulations, we show that the presented proactive caching-prediction algorithm outperforms several commonly-used methods on a real-world dateset.

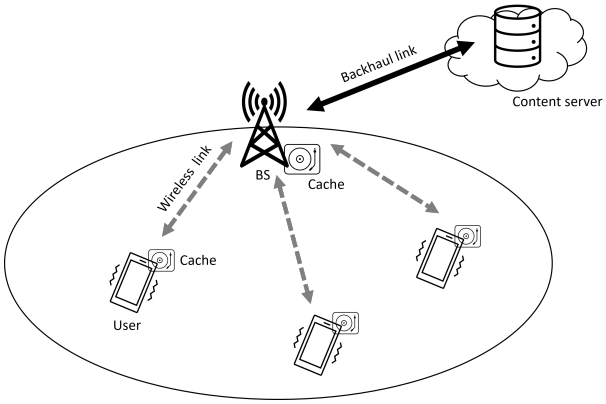

Fig. 1: A system model for hierarchical caching

This paper is organized as follows. The system model and problem statement are described in Section II. In Section III, our probabilistic Poisson factorization model is introduced. In Section IV, we apply MCMC for the inference task. Finally, Section V shows the simulation results and Section VI concludes the paper.

II. System model and Problem Formulation

Consider a cache-enabled communication network that consists of a base station (BS) and M users, as shown in Fig. 1. We denote $\mathcal{M} \triangleq \{1, ..., M\}$ the set of users where $m = 1, .., M$, where m refers to user m. The content server contains a library of N contents. Let denote $\mathcal{N} \triangleq \{1, ..., N\}$ the set of contents where $n = 1, .., N$, where n refers to content n. The BS and the device of user n are equipped with cache memories with limited capacities which can store a finite number of contents. We denote S_b and S_m the cache sizes of BS and user m respectively. In practice, the size of user's cache is much smaller than the size of BS cache i.e. $S_b > S_m$.

A hierarchical content caching strategy is considered where a user can retrieve contents from his local cache, the BS cache or the content server. In particular, users submit random requests towards contents. If the requested content by user m is available at his device, the user retrieves the content immediately. Otherwise, the request is sent to the BS via the

wireless link and the user is served if the content is available at the BS cache. In this case, l_{mn} data unit needs to be transferred from the BS to user m due to requesting content n. Here, the data unit defined as $l_{mn} = y_{mn}s_n$ where s_n is the size of content n and y_{mn} is the number of requests for content n by user m. If the requested content is available neither at the user's cache nor the BS cache, the BS fetches the content from the content server via the backhaul link and the user is served via the wireless link. In this case, additional l_{mn} data unit needs to be transmitted from the backhaul link.

Transferring data in the network is costly therefore for an efficient content delivery strategy the contents need to be stored as close as possible to the end-users in order to minimize the data traffic load. To achieve this goal, we formulate the following caching policy:

$$\min_{x_{mn}w_m} \sum_{n=1}^{N}\sum_{m=1}^{M}(1-x_{mn})[l_{mn}+(1-w_n)l_{mn}] \tag{1a}$$

$$\text{s.t}: \sum_{n=1}^{N} x_{mn}s_n \leq S_m, \quad \forall m \in \mathcal{M}, \tag{1b}$$

$$\sum_{n=1}^{N} w_n s_n \leq S_b, \tag{1c}$$

$$w_n \in \{0,1\}, \quad \forall n \in \mathcal{N}, \tag{1d}$$

$$x_{mn} \in \{0,1\} \quad \forall n \in \mathcal{N}, \forall m \in \mathcal{M}. \tag{1e}$$

The constraints in (1b) and (1c) guarantee the cache capacity at the users and the BS respectively. Moreover, w_n is the cache decision variable indicating that content n is stored at the BS cache if $w_n = 1$ otherwise $w_n = 0$. Likewise, x_{mn} is the cache decision variable indicating that content n is stored at the cache of user m if $x_{mn} = 1$ otherwise $x_{mn} = 0$. Moreover, constraints (1d) and (1e) denote the space restriction for w_n and x_{mn} respectively.

The optimization problem in (1) is binary non-linear programming and intractable to solve in polynomial time. The problem, however, can be equivalently reformulated as the following programming:

$$\min_{x_{mn},w_m,\gamma_{mn}} \sum_{n=1}^{N}\sum_{m=1}^{M} -x_{mn}l_{mn} + \gamma_{mn}l_{mn} \tag{2a}$$

$$\text{s.t}: (1b), (1c), (1d), (1e), \tag{2b}$$

$$\gamma_{mn} + x_{mn} \leq 1, \tag{2c}$$

$$\gamma_{mn} + w_n \leq 1, \tag{2d}$$

$$x_{mn} + w_n + \gamma_{mn} \geqslant 1, \tag{2e}$$

$$0 \leq \gamma_{mn} \leq 1, \tag{2f}$$

which we used the change of variable $\gamma_{mn} = (1-x_{mn})(1-w_n)$ and introduced additional linear constraints (2c)-(2f) into the problem. The optimization problem in (2) is binary linear programming and can be solved by the available solvers [30]. However, due to combinatorial nature of the problem and considering the huge dimentionality in practice, i.e., when the number of users or contents is large, it may not be computationally efficient to use solvers. To efficiently solve the formulated caching policy problem, we instead propose a heuristic method as an approximate solution in Algorithm 1.

Algorithm 1: A heuristic algorithm for solving (1)

1 Sort $L_m = [l_{m1}, ..., l_{mN}]$ in decreasing order and store their sorted indices into $\Pi_m = [\pi_{m1}, ..., \pi_{mN}]$, $\forall m \in \mathcal{M}$;
2 Set $x_{m,n} = 0$, $\forall m \in \mathcal{M}, n \in \mathcal{N}$;
3 **for** $m \leftarrow 1$ **to** M **do**
4 Set $\bar{s} = 0$;
5 **while** *True* **do**
6 Compute $\bar{s} = \bar{s} + s_{\pi_{mn}}$;
7 **if** $\bar{s} > S_m$ **then**
8 break;
9 **end**
10 Set $x_{m,\pi_{mn}} = 1$;
11 **end**
12 **end**
13 Compute residual global contents' importance $q_n = \sum_{m=1}^{M}(1-x_{mn})l_{mn}$, $\forall n \in \mathcal{N}$;
14 Sort $Q = [q_1, ..., q_n]$ in decreasing order and store their sorted indices into $\Pi = [\pi_1, ..., \pi_N]$;
15 Set $w_n = 0$, $\forall m \in \mathcal{M}$;
16 Set $\bar{s} = 0$;
17 **while** *True* **do**
18 Compute $\bar{s} = \bar{s} + s_{\pi_n}$;
19 **if** $\bar{s} > S_b$ **then**
20 break;
21 **end**
22 Set $w_{\pi_n} = 1$;
23 **end**

Specifically, in line 1, for each user m, we sort vector $L_m \triangleq [l_{m1}, ..., l_{mN}]$ in decreasing order. From line 2 to 12, the cache of user m is filled with contents with the highest values in order, independent of the other user, such that to satisfy the cache capacity constraint in (1b). Note that l_{mn} measures the amount of data unit requested by user m for content n and indicates the importance of the content for the user. Therefore, it is reasonable to store contents with the highest values first. In line 13, we compute the aggregated data unit requests for content n, q_n, by the users that have not stored it at their cache. In line 14, we sort vector $Q \triangleq [q_1, ..., q_n]$ in decreasing order. From lines 15 to 23, the BS cache is filled with contents with the highest values in order, such that to satisfy the cache capacity constraint in (1c). Similarly, q_n measures the global importance of content n for the users therefore it is reasonable to store contents with the highest values first. The worst case computational complexity of the heuristic algorithm is dominated by the sorting operations in line 1 and is $\mathcal{O}(MN \log N)$, which is linear in terms of the number of users and super-linear in terms of the number of contents. Therefore, the algorithm is very scalable to use in practice.

In order to solve the heuristic algorithm or problem (2), it is essential to know user interest about contents, i.e., y_{mn}, which is unknown in practice. To tackle the issue, we design a proactive caching strategy with the ability to predict user interest. In particular, the users historical requests are collected

at the BS and their interest about contents are predicted. Subsequently, using the predictions, the caching policy is solved. All the computations for prediction and solving the caching policy are performed at off-peak hours, for example at midnight, at the BS. Finally, the users are served at peak hours during the day based on the caching decision. In the next section, we focus on developing an efficient and accurate prediction algorithm.

III. Poisson Factorization

In this section, we describe a Poisson factorization model for user interest prediction.

We denote $\mathbf{Y} \in \mathbb{Z}^{M\times N}$ to be observation matrix with element y_{mn}. We also assume that each content m is described by D-dimensional feature vector $\mathbf{z}_n = [z_{z,1}, ..., z_{n,D}] \in \mathbb{R}_+^{D\times 1}$ which contains content category, release date and so on. In addition, it is assumed that user preference is fixed over time (we can assume it does not considerably change over short time intervals, e.g. a few hours or days). In practice, matrix $\mathbf{Y}$ is sparse i.e., the majority of elements are zero and noisy. The goal is to denoise the observation matrix to uncover the true user preference about contents.

In MF approach, each user and content is represented by a set of latent features (or factors) which can capture user preference and content popularity respectively. In particular, in HPF model, user and content are represented in a same K-dimensional non-negative latent space. Let $\mathbf{u}_m = [u_{m,1}, ..., u_{m,K}] \in \mathbb{R}_+^{K\times 1}$ and $\mathbf{v}_n = [v_{n,1}, ..., v_{n,K}] \in \mathbb{R}_+^{K\times 1}$ be the user and content latent representations. The HPF model assumes that the element y_{mn} is generated based on the Poisson likelihood as [26]:

$$y_{mn} \sim \text{Pois}(\mathbf{u}_m^T \mathbf{v}_n). \tag{3}$$

Additionally, the latent factors are assumed to have the following hierarchic priors for users and contents:

$$u_{mk} \sim \text{Gam}(\alpha_0, a_m), \quad a_m \sim \text{Gam}(\delta_0, \delta_0/\delta_0'), \tag{4}$$

$$v_{nk} \sim \text{Gam}(\gamma_0, b_m), \quad b_n \sim \text{Gam}(\eta_0, \eta_0/\eta_0'), \tag{5}$$

where k is the kth column of latent factor vectors $\mathbf{v}_n$ and $\mathbf{u}_m$. Putting additional priors over the rate of gammas for each user and content, i.e., a_m and b_n, can capture the data heterogeneity. Moreover, due to using the gamma density, the latent factors have sparse representations which leads to improved generalization performance.

An issue with the HPF model is that it cannot exploit content features and therefore it does not perform satisfactorily. To incorporate content features, we use a similar strategy as proposed in [29]. The key idea is to construct a prior distribution for content latent factors such that to encourage contents with similar features to have similar latent factors. However, we cannot directly use the prior proposed in [29] as it is specifically designed for continuous latent factors but the Poisson factorization model requires a prior with non-negative support. To design such a prior for content latent factors, we first introduce the following lemma from [31].

Lemma 1: If x is a non-central chi-squared random variable with centrality parameter δ and degree of freedom ν then its density function can be expressed as a mixture of Poisson and central chi-squared densities as:

$$x \sim \chi^2_{\nu+2h}, \quad h \sim \text{Pois}(\frac{1}{2}\delta). \tag{6}$$

Using lemma 1 and considering that central chi-squared distribution is a special case of gamma distribution, we assume that content latent factors have the following prior:

$$v_{nk} \sim \text{Gam}(\nu + h_{nk}, \beta), \quad h_{nk} \sim \text{Pois}(\beta \mathbf{b}_k^T \mathbf{z}_n), \tag{7}$$

where gamma density is replaced with chi-squared density to enhance the flexibility of the prior. To ensure that the prior in (7) is a non-degenerate distribution, ν needs to be a non-zero value which we set $\nu = 0.01$. An important property of prior (7) is that, as we will see, it helps to derive closed-form expressions for all conditional densities during MCMC sampling algorithm which we can leverage for an efficient inference.

The prior distribution in (7) encourages contents with similar features to have similar latent factors. In particular, the mean and the variance of v_{nk} are:

$$E[v_{nk}] = \frac{\nu}{\beta} + \mathbf{b}_k^T \mathbf{z}_n, \tag{8}$$

$$Var[v_{nk}] = \frac{\nu + 2\beta \mathbf{b}_k^T \mathbf{z}_n}{\beta^2}. \tag{9}$$

As it can be seen, in expectation, the contents' latent factors will have similar values if they have similar features. For simplicity, we assume that the latent factors are a linear function of content features. Additionally, parameter β controls their variances. In other words, when β is large, the latent factors are strongly encouraged to have similar values. On the other hand, when β is small, the latent factors are allowed to be more varied. The parameters β and $\mathbf{b}_k$ are unknown and need to learned from data. Therefore, we use gamma for their priors as:

$$b_{dk} \sim \text{Gam}(\kappa_0, \kappa_0/\kappa_0'), \tag{10}$$

$$\beta \sim \text{Gam}(\vartheta_0, \vartheta_0/\vartheta_0'). \tag{11}$$

Finally, the complete probabilistic feature-based HPF is depicted in Fig. 2.

The Bayesian inference aims to compute the posterior distribution of all the model's unknown variables in the light of data requests which is proportional to:

$$\prod_{m,n} p(y_{mn}|\mathbf{u}_m, \mathbf{v}_n) \prod_{n,k} p(v_{nk}|h_{nk}, \beta) p(h_{nk}|\mathbf{b}_k, \beta)$$
$$\prod_{m,k} p(u_{mk}|a_m) \prod_{m} p(a_m) \prod_{k,d} p(b_{dk}) p(\beta). \tag{12}$$

Computing the posterior is intractable since several high-dimensional integral are involved. In the next section, we develop an efficient inference algorithm for posterior approximation.

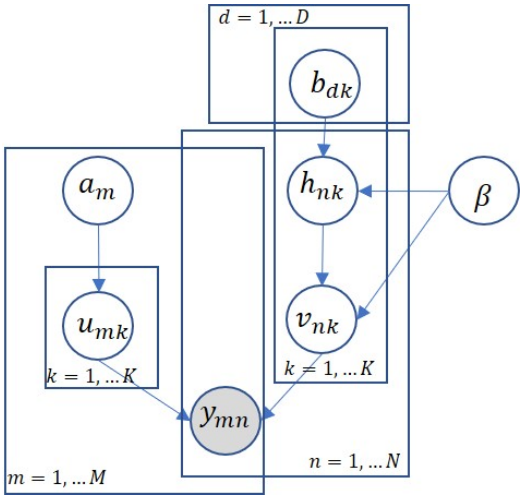

Fig. 2: Probabilistic feature-based HPF model.

IV. Posterior Inference

In this section, we deploy a MCMC method for posterior approximation. One of the simplest MCMC algorithms is the Gibbs sampling, which cycles through the variables, sampling each one from its distribution conditioned on the current values of all other variables [32]. In the following, we derive closed-form conditional posterior densities (CPDs) through which sampling can be done easily.

- CPD of u_{mk}. The following lemma is quite useful to compute the CPD.
  Lemma 2: [33], Let x_1,..., x_K be samples from Poisson distributions with rates of respectively λ_1,..., λ_K. Then, $x = x_1 + ... + x_K$ is a Poisson distribution with rate of $\lambda = \lambda_1 + ... + \lambda_K$. Moreover, conditioned on x, the joint $(x_1, ..., x_K)$ follows a multinomail distribution with $(\frac{\lambda_1}{\sum_k \lambda_k}, ..., \frac{\lambda_K}{\sum_k \lambda_k})$ event probabilities and x number of trails.
  Using lemma 2, we can reformulate (3) as:

$$y_{mnk} \sim \text{Pois}(u_{mk} v_{nk}), \tag{13}$$

  where y_{mnk} is a latent such that $y_{mn} = \sum_k y_{mnk}$. Now, using (13), the CPD of u_{mk} can be computed as:

$$\begin{aligned} p(u_{mk}|.) &\propto u_{mk}^{\alpha_0 - 1} e^{-a_m a u_{mk}} \prod_{n=1}^{N} u_{mk}^{y_{mnk}} e^{u_{mk} v_{nk}} \\ &\propto u_{mk}^{\sum_n y_{mnk} + \alpha_0 - 1} e^{u_{mk}(\sum_n v_{nk} + a_m)} \end{aligned} \tag{14}$$

  which is in the from of a gamma density given by:

$$p(u_{mk}|.) = \text{Gam}(\sum_n y_{mnk} + \alpha_0, \sum_n v_{nk} + a_m). \tag{15}$$

- CPD of v_{nk}: Using (13), we obtain

$$p(v_{nk}|.) \propto v_{nk}^{\sum_m y_{mnk} + h_{nk} + \nu - 1} e^{v_{nk}(\sum_m u_{mk} + \beta)}, \tag{16}$$

  which is in the from of a gamma density given by:

$$p(v_{nk}|.) = \text{Gam}(\sum_m y_{mnk} + h_{nk} + \nu, \sum_m u_{mk} + \beta). \tag{17}$$

- CPD of h_{nk}: It can be computed as:

$$\begin{aligned} p(h_{nk}|.) &\propto \frac{\beta^{\nu + h_{nk}}}{h_{nk}! \Gamma(\nu + h_{nk})} v_{nk}^{\nu + h_{nk} - 1} (\beta \mathbf{b}^T \mathbf{z}_n)^{h_{nk}} \\ &\propto \frac{1}{h_{nk}! \Gamma(\nu + h_{nk})} (4\beta^2 v_{nk} \mathbf{b}^T \mathbf{z}_n)^{2h_{nk}}, \end{aligned} \tag{18}$$

  where $\Gamma(.)$ is gamma function. The expression is in the form of a Bessel density [34] given by:

$$p(h_{nk}|.) = \text{Bessel}(\nu - 1, 2\beta\sqrt{v_{nk} \mathbf{a}^T \mathbf{z}_n}), \tag{19}$$

  with $\nu > 0$.
- CPD of b_{kd}: The CPD does not have a closed-form expression. However, with the help of lemma (2), similar to (13), we reformulate (7) as:

$$h_{nkd} \sim \text{Pois}(b_{kd} z_{nd}), \tag{20}$$

  where h_{nkd} is a latent variable such that $h_{nk} = \sum_d h_{nkd}$. Now, using (20), the CPD of u_{mk} can be computed as:

$$p(b_{kd}|.) = \text{Gam}(\kappa_0 + \sum_n h_{nkd}, \kappa_0/\kappa_0' + \beta \sum_n z_{nd}). \tag{21}$$

- CPDs of β, a_m, y_{mnk} and h_{nkd}: It is not difficult to see that their CPDs have the following forms:

$$\begin{aligned} p(\beta|.) = \text{Gam}(&\vartheta_0 + NK\nu + 2\sum_{nkd} h_{nkd}, \\ &\vartheta_0/\vartheta_0' + \sum_{nd} z_{nd} + \sum_{n,k} v_{nk}) \end{aligned} \tag{22}$$

$$p(a_m|.) = \text{Gam}(\delta_0 + K\alpha_0, \delta_0/\delta_0' + \sum_m u_{mk}) \tag{23}$$

$$p(y_{mn1}, ..y_{mnK}|.) = \text{Multi}(y_{mn}, \mathbf{p}_{y_{mn}}) \tag{24}$$

$$p(h_{nk1}, ..h_{nkD}|.) = \text{Multi}(h_{nk}, \mathbf{p}_{h_{nk}}) \tag{25}$$

  where $\mathbf{p}_{y_{mn}} = [\frac{u_{m1} v_{n1}}{\sum_k u_{mk} v_{nk}}, ..., \frac{u_{mK} v_{nK}}{\sum_k u_{mk} v_{nk}}]$ and $\mathbf{p}_{h_{nk}} = [\frac{b_{k1} z_{k1}}{\sum_d b_{kd} z_{kd}}, ..., \frac{b_{kD} z_{kD}}{\sum_d b_{kd} z_{kD}}]$.

Overall, the Gibbs sampling algorithm takes the form as in Alg. 2.

Samples from the beginning of the Markov chain, i.e., burn-in period, are usually highly correlated and should be discarded to make sure that they are independent and the chain is in the stationary distribution [32]. The samples after burn-in period are used to predict the interest of user m for content n, as:

$$E\{y_{mn}\} = \frac{1}{I - I_{burn}} \sum_{i=I_{burn}+1}^{I} \mathbf{u}_m^{(i)^T} \mathbf{v}_n^{(i)} \tag{26}$$

where I and I_{burn} are the total number of samples and burn-in samples respectively. The more samples there are, the more closely the distribution of the samples matches the desired posterior distribution and the more accurate prediction is achieved. In practice, however, the algorithm is run only for fixed number of iterations or until the available computational resources are exhausted. Moreover, I_{burn} is usually set to half of I in order to obtain safe independent samples.

Algorithm 2: Gibbs sampling algorithm

Initialize $\mathbf{b}_k^0, \beta^0, v_{nk}^0, u_{mk}^0$, $\forall m = 1,..,M, n = 1,..,N, k = 1,..,K$,;
for $i \leftarrow 1$ **to** I **do**
  for $m \leftarrow 1$ **to** M *and* $n \leftarrow 1$ **to** N *such that* $y_{mn} > 0$ **do**
    Sample $\hat{y}_{mnk}^i \sim p(\hat{y}_{mnk} | v_{nk}^{i-1}, u_{mk}^{i-1})$ $\forall k = 1,..,K$;
  end
  for $m \leftarrow 1$ **to** M **do**
    Sample $u_{mk}^i \sim p(u_{mk} | \sum_n v_{nk}^{i-1}, \sum_n \hat{y}_{mnk}^i)$, $\forall k = 1,..,K$;
    Sample $a_m^i \sim p(a_m | \sum_k u_{mk}^i)$;
  end
  for $n \leftarrow 1$ **to** N **do**
    for $k \leftarrow 1$ **to** K **do**
      Sample $v_{nk}^i \sim p(v_{nk} | \sum_m u_{mk}^i, \sum_m \hat{y}_{mnk}^i)$;
      Sample $h_{nk}^i \sim p(h_{nk} | v_{nk}^i, \mathbf{b}^{i-1})$;
      Sample $h_{nkd}^i \sim p(\hat{h}_{nkd} | h_{nk}^i, v_{nk}^i, b_{kd}^{i-1})$, $\forall d = 1,..,D$;
    end
  end
  Sample $b_{kd}^i \sim p(b_{kd} | \sum_n v_{nk}^i, \beta^{i-1}, \sum_n h_{nkd}^i)$, $\forall k = 1,..,K, d = 1,..,D$;
  Sample $\beta^i \sim p(\beta | \sum_{nk} v_{nk}^i, \sum_{nkd} h_{nkd}^i)$;
end

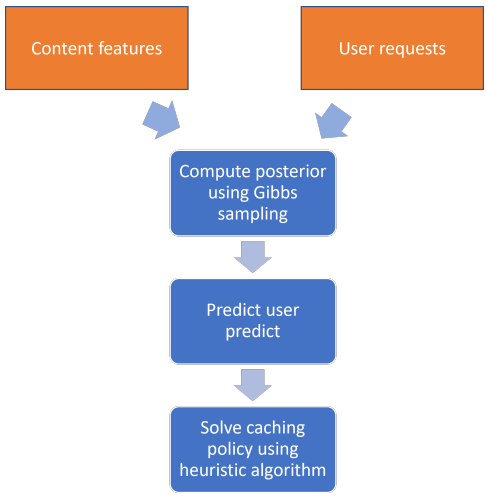

Fig. 3: The overall workflow of prediction-caching

The value in (26) can then be used as prediction for y_{mn} to solve the caching polices in Section II. Fig.3 presents an overview of our work-flow scheme for user interest prediction and caching. It consists of three main steps. In the first step, the Gibbs sampling in Alg. 2 is run to compute the posterior distribution using content features and users requests. In the second step, using (26), user interest about contents is predicted. In the third step, decisions about which contents should be cached is taken using the heuristic solutions provided by Alg.1.

C	10	20	50	100
FHPF	**0.0446**	**0.0834**	**0.1640**	**0.2516**
HPF	0.0445	0.0817	0.15248	0.2429
NMF	0.00648	0.0096	0.0257	0.0317
WMF	0.0063	0.00965	0.0256	0.0316
Autorec	0.0361	0.0642	0.0869	0.1029

TABLE I: Recall@C on Movielens 1m

A. Computational complexity:

We compute the per-iteration worst case computational complexity of the Gibbs sampling algorithm for approximating the posterior distribution. The per-iteration time complexity is dominated by three terms as follows. i) the time complexity of sampling $\hat{y}_{mn}$, $\forall m = 1,..,M, n = 1...,N$, is $\mathcal{O}(Ky_{nnz})$ where y_{nnz} is the number of non-zero requests. ii)The time complexity of sampling the parameters of $\mathbf{u}, \mathbf{v}$ is $\mathcal{O}(NK+MK)$. iii). The time complexity of sampling the rest of parameters can be ignored . Overall, the time complexity of the Gibbs sampling is $\mathcal{O}(Ky_{nnz} + MK + NDK)$.

Speeding up via Parallel Computations: Sampling of each variable depends only on the other variables in its Markov blanket, where the Markov blanket consists of the variable's parents, co-parents, and children in the graphical representation of the Bayesian model, and is independent of other variables. For instance, sampling updates of a content factors are independent of all other contents (and similarly for a user factors). Thus, we can easily parallelize the posterior inference computations to scale up the performance of the Gibbs sampling algorithm.

V. Simulation Results

In this section, we present our simulation results on the performance of the proposed feature-based hierarchical Poisson factorization (FHPF) model in Section III and the caching policy in Section II.

The hyper-parameters of the model are set as $\alpha_0 = \delta_0 = \delta_0' = \kappa_0' = \kappa_0 = 0.3$, $\vartheta_0 = 0.01$ and $\vartheta_0' = 0.1$ unless otherwise stated. [1] We compare the performance of PHPF with four common baselines including WMF [4], HPF [26], NMF [35] and Autorec [36] for implicit data which do not exploit content features. Moreover, we consider the Movielens $1m$ dataset [37] which contains 1,000,209 ratings of approximately 3,900 movies made by 6,040 users. Movie information contains 18 genres including action, adventure, crime, musical and so on.

Similar to [26], [28], to adapt the dataset to implicit feedback scenario, a rating for movie is considered as one request for that movie. Furthermore, we select the first 80% of the dataset for training and the rest for testing the models. We set $K = 5$ in all the simulation scenarios. Moreover, we run the Gibbs sampling algorithm for $I = 500$ samples with the first half as burn-in samples.

[1] We found that with theses values the model works fine in our simulation scenarios, although they may not be optimal. Improved prediction accuracy may be found by performing extensive cross validation.

C	10	20	50	100
FHPF	**0.4589**	**0.4383**	**0.3692**	**0.3037**
HPF	0.4303	0.4272	0.3337	0.2854
NMF	0.0995	0.0791	.0822	0.0529
WMF	0.0993	0.0791	0.08221	0.0529
Autorec	0.4071	0.3650	0.2135	0.1336

TABLE II: Precision@C on Movielens 1m

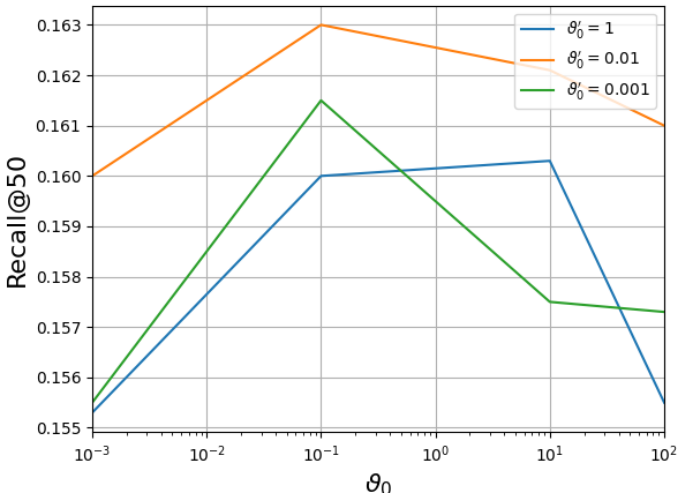

Fig. 4: Sensitivity of Recall@50 on hyper-parameters ϑ_0 and ϑ_0'

A. Prediction Performance

In this section, we show results on prediction accuracy. As performance metrics, we compute recall@C and precision@C during the test set which are defined by [38]:

$$\text{recall@}C = \frac{1}{M}\sum_m \frac{C_m^r}{C_m}, \quad \text{precision@}C = \frac{1}{M}\sum_m \frac{C_m^r}{C}$$

where C_m is the number of requested contents by user m and C_m^r is the number requested contents which are among the most C preferred contents for user m selected by the model. The recall and the precision are two metrics that are widely-used in recommendation systems literature and they measure how well an algorithm can rank contents in accordance with user preferences. Specifically, the recall method computes the portion of preferred contents that were suggested. The precision measure describes what proportion of selected contents for a user were actually preferred by the user.

Tables I and II show the recall and the precision respectively. We can see that FHPF outperforms all the other models for both recall and precision. For example, for recall@50, we obtain improvements of 7.6%, 538%, 540% and 88% with respect to HPF, NMF, WMF and Autorec respectively. This indicates that our proposed FHPF model can effectively denoise the implicit data requests and exploit content features and improve the prediction accuracy.

Next we show the sensitivity of FHPF with respect to β. For this scenario, we tune hyper-parameters ϑ_0 and ϑ_0' and investigate the recall@50 performance in Fig. 4. We can see that the prediction accuracy decreases for too small/large values of ϑ_0 and ϑ_0'. However, by comparing the results with table I, it can be observed that FHPF performs significantly better than the other benchmarks for wide ranges of ϑ_0 and ϑ_0' which shows the robustness of FHPF.

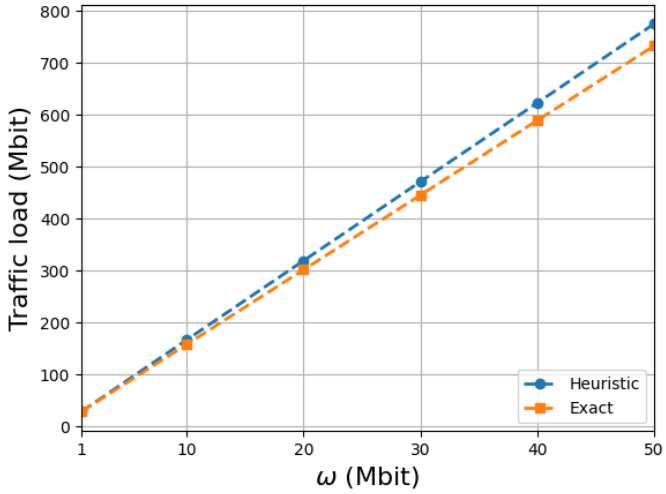

Fig. 5: Traffic load versus content sizes

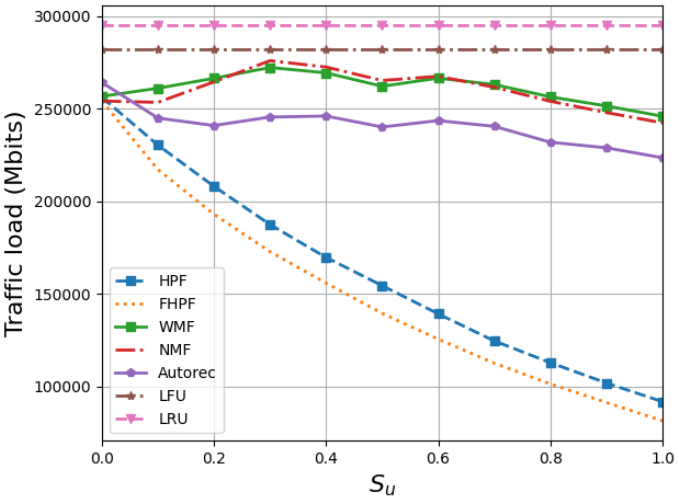

Fig. 6: Traffic load versus user cache capacity for different prediction methods

B. Caching Policy Performance

In this section, we study the performance of the caching policy formulated in Section II. We first investigate the performance of the heuristic method proposed in Algorithm 1. To solve problem (2), we use MOSEK solver embedded in CVXPY [30]. Due to huge computational complexity of the solver, we are unable to use the whole dataset with our available computation power. Therefore, we choose 2 most active users and 100 popular contents from the dataset. Since the dataset does not have movies size, we generate them from interval $[1, \omega]$ uniformly randomly, where ω is the maximum size of contents in data unit, e.g., Mbit. Moreover, we assume $S_1 = ... = S_M = S$ and $S_b = 4S = 0.25\sum_{n=1}^{N} s_n$. Fig. 5 shows the overall traffic load versus content size value ω. It can be seen that when ω is small the heuristic method performs very closely to the exact method. As ω increases the performance gap also increases. This is expected since small

error in the heuristic method can cause huge performance loss due to large contents size. Moreover, we can see that as ω increases the traffic load increases. This is because large volume of data needs to be transmitted in the network which is due to large contents size.

Next we show the performance of the prediction methods on the caching policy. Since our focus is on the prediction accuracy, we assume that all the contents have the same size of one Mbit. We choose the full dataset and use the heuristic method in Algorithm 1 due to its scalability for solving the caching policy. We also compare our method with two common caching policies, i.e., LRU and LFU. For each of these policies, we implement them for each user and the BS separately. Fig. 6 illustrates the actual network traffic load in the test set versus the cache capacity of users' device S_u, where S_u is defined as $S_u = \frac{S}{S_b}$. We also fix S_b as in previous scenario. From Fig. 6, we can see that the proposed model, FHPF, outperforms all the other methods for all range of S_u. In particular, we can see that there is a significant gap between Poisson-gamma models, i.e. HPF and FHPF, and the other approaches. This indicates that HPF and FHPF are highly expressive and accurate for modeling content requests. Additionally, we see that FHPF outperforms HPF. For example, when the BS and the users have caches with storage of 25% and 10% of the total contents size respectively, our approach yields around 8% improvement. This shows that capturing content features can improve the caching performance.

Moreover, it can been observed that traffic load generally decreases as S_u increases. This is because more contents can be stored at the users' cache which results in less traffic load on the communication links. The decreasing trend is monotonic for HPF and FHPF. On the other hand, for NMF, WMF and Autorec, the data traffic increases for some ranges of S_u as it increases. This might seem counter-intuitive as one may expect that, regardless of prediction accuracy, the caching performance should improve (or at least not degrade) by increasing the size of cache. However, we should highlight that this may not be the case for the policy formulated in Section II. Specifically, the caching decisions at the users and the BS are highly dependent and small prediction error can affect the caching decisions at both the users and the BS significantly. For an inaccurate prediction method, it is possible that the contents of BS cache will be updated with less popular contents than the ones stored before when the users' cache capacity S_u increases. In other words, storing more uninterested contents at the users' cache, by increasing S_u, can increase the error in computation of global content importance q_n in Algorithm 1, which can degrade the caching performance. We further notice that the performance of the reactive polices doesn't change as S_u increases. The reason is that, for the MovieLens dataset, a users requests a movie only once and the reactive polices, at user-level, can not predict which movie a user may request in the future. This indicates that these polices are quite efficient to use for user-level caching for the this dataset.

In Fig. 7, we show the traffic load on the backhaul link versus users' cache capacity for different prediction methods.

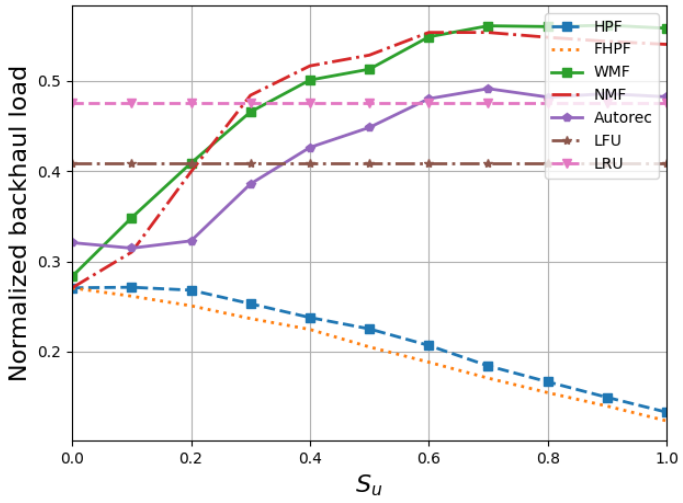

Fig. 7: Normalized data traffic on backhaul link versus cache capacity for different prediction methods

Here, the backhaul traffic load is measured as the ratio of the data requests which are served by neither the users' cache nor the BS cache compared to the total data requests, i.e. $\frac{\sum_m \sum_n (1-z_n)(1-x_{mn}) l_{mn}}{\sum_m \sum_n l_{mn}}$. It can be observed that the Poisson-gamma models, i.e. HPF and FHPF, substantially outperform the other methods. Moreover, the traffic load on the backhaul link monotonically decreases as S_u increases for HPF and FHPF while increases for a wide range of S_u for NMF, WMF and Autorec. With the same reason as we explained in previous paragraph, increasing S_u doesn't affect the performance of LFU and LRU. This again shows that Poisson-gamma models are much more accurate with respect to the other methods. Furthermore, FHPF outperforms HPF which is due to the advantage of using content features for improved prediction accuracy.

VI. Conclusion

In this paper, we formulated a hierarchical caching policy with the objective of minimizing the data traffic transmission in the network. The optimization problem is a non-linear combinatorial programming problem and difficult to solve. Therefore, we transformed it into a linear binary programming which may be solved by the available solvers. However, the formulated problem still is not scalable to solve in practice. To tackle the issue, we proposed a scalable heuristic method as an approximate solution. Since the caching policy requires user interest about contents, we subsequently proposed a Bayesian Poisson factorization model for prediction. In particular, the model exploits the available side information about contents to effectively filter out the noise in the data and provides accurate prediction. Furthermore, a Gibbs sampling algorithm was designed to efficiently compute the complicated posterior distribution. Our numerical results on a real-wold dataset demonstrate that the of proposed approach is superior to those of the traditional prediction models and caching policies.

VII. ACKNOWLEDGMENT

This work was supported by the Luxembourg National Research Fund (FNR) – PROCAST Project, ref. 11691338.

REFERENCES

[1] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[2] J. Li, T. K. Phan, W. K. Chai, D. Tuncer, G. Pavlou, D. Griffin, and M. Rio, "Dr-cache: Distributed resilient caching with latency guarantees," in *IEEE Conference on Computer Communications*, 2018, pp. 441–449.

[3] E. Baştuğ, M. Bennis, E. Zeydan, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *Journal of Communications and Networks*, vol. 17, no. 6, pp. 549–557, 2015.

[4] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 2008, pp. 263–272.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[6] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 374–398, 2003.

[7] X. Peng, J. Shen, J. Zhang, and K. B. Letaief, "Joint data assignment and beamforming for backhaul limited caching networks," in *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, Sept 2014, pp. 1370–1374.

[8] M. Tao, E. Chen, H. Zhou, and W. Yu, "Content-centric sparse multicast beamforming for cache-enabled cloud ran," *IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 6118–6131, 2016.

[9] S.-H. Park, O. Simeone, and S. S. Shitz, "Joint optimization of cloud and edge processing for fog radio access networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 11, pp. 7621–7632, 2016.

[10] X. Li, X. Wang, K. Li, Z. Han, and V. C. Leung, "Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design," *IEEE Trans. Wireless Commun*, vol. 16, no. 10, pp. 6926–6939, 2017.

[11] B. Chen, C. Yang, and A. F. Molisch, "Cache-enabled device-to-device communications: Offloading gain and energy cost," *IEEE Transactions on Wireless Communications*, vol. 16, no. 7, pp. 4519–4536, 2017.

[12] G. Kollias and A. Antonopoulos, "Joint consideration of content popularity and size in device-to-device caching scenarios," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[13] A. C. Güngör and D. Gündüz, "Proactive wireless caching at mobile user devices for energy efficiency," in *2015 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2015, pp. 186–190.

[14] D.-H. Tran, S. Chatzinotas, and B. Ottersten, "Satellite-and cache-assisted uav: A joint cache placement, resource allocation, and trajectory optimization for 6g aerial networks," *arXiv preprint arXiv:2106.05016*, 2021.

[15] X. Xu, Y. Zeng, Y. L. Guan, and R. Zhang, "Overcoming endurance issue: Uav-enabled communications with proactive caching," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1231–1244, 2018.

[16] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 915–929, 2018.

[17] N. Garg, M. Sellathurai, V. Bhatia, B. Bharath, and T. Ratnarajah, "Online content popularity prediction and learning in wireless edge caching," *IEEE Tran. Commun.*, 2019.

[18] K. N. Doan, T. Van Nguyen, T. Q. Quek, and H. Shin, "Content-aware proactive caching for backhaul offloading in cellular network," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3128–3140, 2018.

[19] M. Chen, W. Saad, C. Yin, and M. Debbah, "Echo state networks for proactive caching in cloud-based radio access networks with mobile users," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3520–3535, 2017.

[20] S. S. Tanzil, W. Hoiles, and V. Krishnamurthy, "Adaptive scheme for caching youtube content in a cellular network: Machine learning approach," *Ieee Access*, vol. 5, pp. 5870–5881, 2017.

[21] V. Fedchenko, G. Neglia, and B. Ribeiro, "Feedforward neural networks for caching: n enough or too much?" *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 139–142, 2019.

[22] Y. Wang, M. Ding, Z. Chen, and L. Luo, "Caching placement with recommendation systems for cache-enabled mobile social networks," *IEEE Communications Letters*, vol. 21, no. 10, pp. 2266–2269, 2017.

[23] Z. Zhang, C.-H. Lung, M. St-Hilaire, and I. Lambadaris, "Smart proactive caching: Empower the video delivery for autonomous vehicles in icn-based networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7955–7965, 2020.

[24] G. Li, Q. Shen, Y. Liu, H. Cao, Z. Han, F. Li, and J. Li, "Data-driven approaches to edge caching," in *Proceedings of the 2018 Workshop on Networking for Emerging Applications and Technologies*, 2018, pp. 8–14.

[25] D. T. Hoang, D. Niyato, D. N. Nguyen, E. Dutkiewicz, P. Wang, and Z. Han, "A dynamic edge caching framework for mobile 5g networks," *IEEE Wireless Communications*, vol. 25, no. 5, pp. 95–103, 2018.

[26] P. Gopalan, J. M. Hofman, and D. M. Blei, "Scalable recommendation with hierarchical poisson factorization," in *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, ser. UAI'15. Arlington, Virginia, United States: AUAI Press, 2015, pp. 326–335.

[27] S. Mehrizi, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Trend-aware proactive caching via tensor train decomposition: A bayesian viewpoint," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 975–989, 2021.

[28] S. Mehrizi, S. Chatterjee, S. Chatzinotas, and B. Ottersten, "Online spatiotemporal popularity learning via variational bayes for cooperative caching," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 7068–7082, 2020.

[29] D. Agarwal and B.-C. Chen, "flda: matrix factorization through latent dirichlet allocation," in *Proceedings of the third ACM international conference on Web search and data mining*, 2010, pp. 91–100.

[30] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[31] R. J. Muirhead, *Aspects of multivariate statistical theory*. John Wiley & Sons, 2009, vol. 197.

[32] D. Gamerman and H. F. Lopes, *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC Press, 2006.

[33] J. F. C. Kingman, "Poisson processes," *Encyclopedia of biostatistics*, vol. 6, 2005.

[34] L. Yuan and J. D. Kalbfleisch, "On the bessel distribution and related problems," *Annals of the Institute of Statistical Mathematics*, vol. 52, no. 3, pp. 438–447, 2000.

[35] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[36] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th international conference on World Wide Web*, 2015, pp. 111–112.

[37] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, p. 19, 2016.

[38] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks." *Journal of Machine Learning Research*, vol. 10, no. 12, 2009.