Retrospective Higher-Order Markov Processes for User Trails

Tao Wu
Purdue University
West Lafayette, IN
wu577@purdue.edu

David F. Gleich
Purdue University
West Lafayette, IN
dgleich@purdue.edu

ABSTRACT

Users form information trails as they checkin with a geolocation, rate items, or consume media. A common problem is to predict what a user might do next for the purposes of guidance, recommendation, or prefetching. Markov chains models have been widely used methods to study such sequences of data. First-order Markov chains are easy to estimate, but lack accuracy when history matters. Higher-order Markov chains, in contrast, have too many parameters and suffer from overfitting the training data. Fitting these parameters with regularization and smoothing only offers mild improvements. In this paper we propose the retrospective higher-order Markov process (RHOMP) as a low-parameter model for such sequences. This model is a special case of a higher-order Markov chain where the transitions depend retrospectively on a single history state instead of an arbitrary combination of history states. There are two immediate computational advantages: the number of parameters is linear in the order of the Markov chain and the model can be fit to large state spaces. Furthermore, by providing a specific structure to the higher-order chain, RHOMPs improve the model accuracy by efficiently utilizing history states without risks of overfitting the data. We demonstrate how to estimate a RHOMP from data and we demonstrate the effectiveness of our method on various real application datasets spanning geolocation data, review sequences, and business locations. The RHOMP model uniformly outperforms higher-order Markov chains, Kneser-Ney regularization, and tensor factorizations in terms of prediction accuracy.

KEYWORDS

Higher-order Markov chains; Tensor factorization; User models

1 INTRODUCTION

User trails record sequences of activities when individuals interact with the Internet and the world. Such data come from various applications when users write a product review [25], checkin at a physical location [13, 42], visit a webpage, or listen to a song [8]. Understanding the properties and predictability of these data helps improve many downstream applications including overall user experiences, recommendations, and advertising [1, 19]. We study the

prediction problem and our goal is to estimate a model to describe and predict a set of user trails.

Markov chains are one of the most commonly studied models for this type of data. For these models, each checkin place, website, or song is a state. Users transition among these states following Markov rules. In a first-order Markov model, the transition behavior to the next state of the sequence only depends on the current state. Higher-order Markov models include a more-realistic dependence on a larger number of previous states, and multiple recent studies found that first-order Markov chains do not fully capture the user behaviors in web browsing, transportation and communication networks [12, 32]. Furthermore, ignoring the effects of second-order Markov dynamics has significant negative consequences for downstream applications including community detection, ranking, and information spreading [2, 32].

The downside to higher-order Markov models is that the number of parameters grows exponentially with the order. (If there are N states and we model m steps of history, there are N^{m+1} parameters.) So, even if we could accurately learn the parameters, it is already challenging to even store them. (Some practical techniques include low-rank and sparse approximations, but these pose their own problems.) Second, since the number of model parameters grows rapidly, the amount of training data required also grows exponentially with the order m [12]. Acquiring such a large volume of training data is usually impossible. Lastly, determining the amount of history to use itself is hard [27], and selecting a large value of m could severely overfit the data, thus making the learned model less reliable.

One strategy to address these limitations of higher-order Markov chains is a variable order Markov chain [6] where the relevant history length changes among states. There is a fitting algorithm that can automatically determine an appropriate order for each state, however it requires substantial computation time [31] which restricts it to applications with only a small number of states [5, 12, 14].

Smoothing and regularization methods [11] like Kneser-Ney smoothing and Witten-Bell smoothing are additional approaches to make the higher-order Markov chain more robust. These methods are widely applied in language models for predicting unseen transitions. We will compare against the behavior of the Kneser-Ney smoothing in our experiments and show that our method has a number of advantages.

In this paper we propose the retrospective higher-order Markov process (RHOMP) as a simplified, special case of a higher-order Markov chain (Section 3). In this type of Markov model, the process retrospectively choses a state from the past m steps of history, and then transitions as a first-order chain conditional on that state from history. This assumption helps to restrict the total number of parameters and protect the model from overfitting the correlations between history states. Specifically, this model corresponds to

choosing m different first order Markov chain transition matrices, one for each step of history, as well as an associated probability distribution. Consequently, the number of parameters grows linearly with the size of history while preserving the higher-order nature. We also show there are important connections between our model and the class of pairwise-interaction tensor factorization models proposed by Rendle et al. [29, 30] (Section 3.2).

We design an algorithm to select an optimal model from training data via maximum likelihood estimation (MLE). For the second-order case with two steps of history, this yields a constrained convex optimization problem with a single hyperparameter α. We derive a projected gradient descent [16] algorithm to solve it. It requires only a few iterations to converge and each iteration is linear in the training data. We select the hyperparameter by fitting a polynomial to the likelihood function as a function of the parameter and select the global minimum. Thus, our RHOMP process does not require any parameter tuning and is scalable to applications with tens of thousands of states. In addition, both the process of updating the gradients and model parameters parallelize over the training data.

We evaluate the effectiveness of RHOMP models in experiments with real datasets including product reviews, online music streaming, photo locations, and checkin business types (Section 5.1). We primarily compare algorithms in terms of their ability to predict information from testing data and use accuracy and mean reciprocal rank as the two main evaluation metrics. These experiments and results show that the RHOMP model achieves superior prediction results in all datasets (Section 5.2) compared with first and second order chains. For even higher-order chains, RHOMP shows stable performance with one exception (Section 5.4) where the data only has short sequences.

Code and data for this paper are available at: https://github.com/wutao27/RHOMP.

Remark. Recently Kumar et al. [22] proposed the Linear Additive Markov Process (LAMP) that is closely related to our framework. Specifically, our RHOMP model is the same as the generalized LAMP (GLAMP) model from that reference. We learned about this paper after our submission to KDD. The papers share a number of related technical results about the models, and we discovered the related work [24, 26, 39, 43] based on their manuscript. The main difference is that in this paper we focus on the general form that allows to learn different Markov chains for each step of history. In addition we connect the RHOMP model with a particular tensor factorization to a higher-order Markov chain.

2 PRELIMINARIES

We begin by formally reviewing the problem of user trail prediction. Then we will review relevant background on Markov chain models.

2.1 Problem Formulation

We denote a user trail as a sequence over a discrete state space $s = (s_1, s_2, \cdots)$ with each element $s_i \in \{1, 2, \cdots, N\}$. Here N is the total number of states. The sequence can represent, for instance, a user's music listening history with each state denoting a song/artist, or a user's checkin history from social network with each state denoting a location. Given a specific user trail up to time $t - 1$: $s = (s_1, s_2, \cdots, s_{t-1})$ with $t \geq 2$, the task is to predict the next

state at time t based on a large set of user trails for training: $\mathcal{S} = \{s^{(1)}, s^{(2)}, \cdots\}$, where each $s^{(i)}$ is an individual trail.

2.2 Markov Chain Methods

An m−th order Markov chain is defined as a stochastic process $\{X_t, t = 1, 2, \cdots\}$ on the state space: $\{1, 2, \cdots, N\}$ with the property that the next transition only depends on the last m steps. Formally,

$$\Pr\left(X_t = i \mid X_{t-1} = i_{t-1}, \cdots, X_1 = i_1\right)$$
$$= \Pr\left(X_t = i \mid X_{t-1} = i_{t-1}, \cdots, X_{t-m} = i_{t-m}\right).$$

An $(m + 1)$-order transition tensor $\underline{P}$ with size N characterizes the above Markov chain, with $\underline{P}_{i,j,\cdots,k}$ denoting the probability of transitioning to state i given the m current history states $(j, \cdots, k)$. The model with $m = 1$ is called the first-order Markov chain and similarly it can be described by an $N \times N$ transition matrix P.

In order to use a Markov chain for the prediction problem, we need to estimate the transition matrix P. Given a set of users trails $\mathcal{S} = \{s^{(1)}, s^{(2)}, \cdots\}$, the maximum likelihood estimator (MLE) of the probability $P_{i,j}$ for a first order chain is given by [12]:

$$P_{i,j} = \frac{c(i,j)}{\sum_\ell c(\ell, j)}$$

where $c(i, j)$ denotes the number of instances where the states j and i were consecutive in all trails. For the case of higher-order Markov chain, it is well-known that any higher-order ($m > 1$) Markov chain X_t is equivalent to a first-order Markov chain Z_t by taking a Cartesian product of its state space. This simplifies the parameter estimations and we may replace the original states with the Cartesian product states:

$$\underline{P}_{i,j,\cdots,k} = \frac{c(i, j, \cdots, k)}{\sum_\ell c(\ell, j, \cdots, k)},$$

where now $c(i, j, \cdots, k)$ counts the number of instances of the sequence $k, \cdots, j, i$ in the training data.

Returning to the prediction task itself, Markov chain methods take as input the history states of a trail and lookup the probabilities for all future states in the matrix P or tensor $\underline{P}$. This becomes a ranked list of states with the highest probability on top.

3 RETROSPECTIVE HIGHER-ORDER MARKOV PROCESSES

The goal of the retrospective higher-order Markov process (RHOMP) is to strike a balance between the simplicity of the first order Markov model and the high-parameter complexity of the higher-order Markov model. Nevertheless, it is important for the model to account for higher-order behaviors because these are necessary to capture many types of user behaviors [12, 32]. Towards that end, the RHOMP model describes a *structured* higher-order Markov chain that results in a compact low-parameter description of possible user behaviors. We describe this formally for the case of a second-order history (and discuss largely notational extensions to higher-order chains in Section 3.4).

3.1 The Retrospective Process

The specific structure that a RHOMP describes is a retrospectively first-order Markov property. For some intuition, suppose that a web
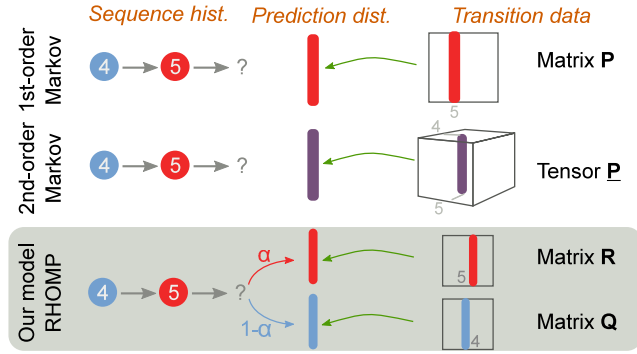
Figure 1: An illustration of Markov chain methods and our proposed RHOMP model.

surfer had visited a search-query result page and then clicked the first link. In the RHOMP model, the user will first determine if they are going to continue browsing from the search-result page or the first link—hence users have the power to retrospect over history. Once that decision has been made, the user will behave in a first-order Markovian fashion that depends on *if* the user returned to the previous state or remained on the current state. Formally, suppose that the chain has recently visited states j and k. The RHOMP is a two-stage process that *first* selects a single history state. Since there are only two states, we model this selection as a weighted coin-toss where the probability of picking j is α and so picking k happens with probability $1 - \alpha$. Once we have the history state, then the RHOMP transitions according to a transition matrix that is specific to that step of the history. Thus

$$\Pr\left(X_t = i \mid X_{t-1} = j, X_{t-2} = k\right) = \alpha R_{i,j} + (1 - \alpha)Q_{i,k},$$

where R models the transitions from the current state (when those are selected) and Q models the transitions from the previous state (when those are selected). See Figure 1 for illustration. We summarize this in the following definition:

Definition 3.1. Given $0 \leq \alpha \leq 1$ and two stochastic matrices R, Q, a second-order retrospective higher-order Markov process will transition from state j with history state k as follows: (i) with probability α it transitions according to R with the current state j, and (ii) with probability $1 - \alpha$ it transitions according to Q with the previous state k.

This model has a number of useful features. For instance, it is easy to compute the stationary distribution as the following theorem shows.

THEOREM 3.2. *Let α, R, Q be a second-order RHOMP model. Consider the stationary distribution $\mathbf{x}$ in terms of the long-term fraction of time the process spends in a state:*

$$x_i = \lim_{t \to \infty} \frac{number\ of\ times\ X_t = i}{t} \quad for\ each\ i = 1 \dots N.$$

Such a distribution $\mathbf{x}$ always exists. Moreover, it is unique if $\alpha R + (1 - \alpha)Q$ is an irreducible matrix.

PROOF. Because the RHOMP is a special case of a second-order chain, we can use the relationship with the first-order chain on the

Cartesian product space to establish that a distribution $\mathbf{x}$ always exists. This follows because the long-term distribution of a first-order, finite-state space Markov chain always exists (though there could be multiple such distributions) [37]. Let $X_{i,j}$ for all $1 \leq i, j \leq N$ be any limiting distribution of the product state space, and $\mathbf{x}$ be either of the corresponding marginal distribution such that $\sum_j X_{j,k} = x_k$ or $\sum_k X_{j,k} = x_j$. Note that both of these marginals result in the same distribution because we use the long time average to define $X_{i,j}$. Then we have:

$$x_i = \sum_j X_{i,j} = \sum_j \sum_k (\alpha R_{i,j} + (1 - \alpha)Q_{i,k})X_{j,k}$$
$$= \sum_j \alpha R_{i,j}x_j + \sum_k (1 - \alpha)Q_{i,k}x_k = (\mathbf{Px})_i$$

where P is defined as $\alpha R + (1 - \alpha)Q$. So the limiting distribution $\mathbf{x}$ follows $\mathbf{x} = P\mathbf{x}$, and it is unique if the corresponding Markov chain P is irreducible. ☐

In Section 3.3, we show how to compute a maximum likelihood estimate of R and Q from data.

3.2 A Tensor Factorization Perspective

We originally derived this type of RHOMP via a tensor factorization approach, but then realized that the retrospective interpretation is more direct and helpful. Nevertheless, we believe there are fruitful connections established by the tensor factorization approach. Consider the transition tensor of a second-order Markov chain: $\underline{P}$ is a 3-mode, $N \times N \times N$, non-negative tensor such that

$$\sum_i \underline{P}_{i,j,k} = 1 \text{ for all } 1 \leq j, k \leq N. \tag{1}$$

This imposes a set of N^2 equality constraints. If we wanted to use traditional low-rank tensor approximations such as PARAFAC or Tucker [21] to study large datasets, then we would need to add a large number of constraints to the fitting algorithms in order to ensure that the factorization results in a stochastic tensor that we could use for a second order Markov chain. This approach was extremely challenging.

Instead, consider a pairwise interaction tensor factorization (PITF) as proposed by Rendle et al. [30] with the following form:

$$\underline{P}_{i,j,k} = \sum_\ell A_{i,\ell}^{(J)}B_{j,\ell}^{(I)} + \sum_\ell A_{i,\ell}^{(K)}C_{k,\ell}^{(I)} + \sum_\ell B_{j,\ell}^{(K)}C_{k,\ell}^{(J)} \tag{2}$$

where matrices $A^{(J)}, A^{(K)}, B^{(I)}, B^{(K)}, C^{(I)}, C^{(J)} \in \mathbb{R}^{N \times k}$. We notice that last term in (2) is the interaction between the current state j and the previous state k, and it contributes only a constant determined by the pair (j, k). In the applications of prediction, we can drop this term because it does not affect the relative ranking for the future state i. So the factorization model becomes:

$$\underline{P}_{i,j,k} = \sum_\ell A_{i,\ell}^{(J)}B_{j,\ell} + \sum_\ell A_{i,\ell}^{(K)}C_{k,\ell} \tag{3}$$

with $A^{(J)}, A^{(K)}, B, C \in \mathbb{R}^{N \times k}$.

To see the relationship with our RHOMPs, denote $\tilde{\alpha}\tilde{R} = A^{(J)}B^\top$ and $(1 - \tilde{\alpha})\tilde{Q} = A^{(K)}C^\top$ with $0 \leq \tilde{\alpha} \leq 1$. Then the result of a PITF factorization with stochastic constraints is:

$$\underline{P}_{i,j,k} = \tilde{\alpha}\tilde{R}_{i,j} + (1 - \tilde{\alpha})\tilde{Q}_{i,k} \tag{4}$$

It is easy to verify that if both $\tilde{R}$ and $\tilde{Q}$ are stochastic matrices, then the corresponding tensor $\underline{P}$ is a transition tensor following (1). The following theorem shows that from any nonnegative $\tilde{R}$ and $\tilde{Q}$, we can construct such stochastic matrices.

Theorem 3.3. *Assuming there exist nonnegative matrices $\tilde{R}$ and $\tilde{Q}$ such that the transition tensor $\underline{P}$ can be decomposed in the form of (4), then there exist $0 \leq \alpha \leq 1$ and stochastic matrices R, Q such that $\underline{P}_{i,j,k} = \alpha R_{i,j} + (1 - \alpha)Q_{i,k}$.*

Proof. Denote $\sum_i \tilde{R}_{i,j} = \tilde{r}_j$ and $\sum_i \tilde{Q}_{i,k} = \tilde{q}_k$ for all $1 \leq j, k \leq N$. Because $1 = \sum_i \underline{P}_{i,j,k} = \tilde{\alpha}\tilde{r}_j + (1 - \tilde{\alpha})\tilde{q}_k$ for all $1 \leq j, k \leq N$, we have $\tilde{r}_1 = \tilde{r}_2 = \cdots = \tilde{r}_N = \tilde{r} \geq 0$, $\tilde{q}_1 = \tilde{q}_2 = \cdots = \tilde{q}_N = \tilde{q} \geq 0$ and $\tilde{\alpha}\tilde{r} + (1 - \tilde{\alpha})\tilde{q} = 1$. If $\tilde{r} = 1, \tilde{q} = 1$ then the original matrices $\tilde{R}$ and $\tilde{Q}$ are stochastic. Otherwise we can set

$$\alpha = \tilde{\alpha}\tilde{r}; \quad R = \tilde{R}/\tilde{r}; \quad Q = \tilde{Q}/\tilde{q}$$

where R and Q are stochastic. Then we have

$$\alpha R_{i,j} + (1 - \alpha)Q_{i,k} = \tilde{\alpha}\tilde{R}_{i,j} + \frac{(1 - \tilde{\alpha}\tilde{r})\tilde{Q}_{i,k}}{\tilde{q}}$$
$$= \tilde{\alpha}\tilde{R}_{i,j} + (1 - \tilde{\alpha})\tilde{Q}_{i,k} = \underline{P}_{i,j,k}$$

So (α, R, Q) forms a valid factorization for $\underline{P}$, the bound on α follows from $\tilde{\alpha}\tilde{r} + (1 - \tilde{\alpha})\tilde{q} = 1$ from (4). □

Consequently, the RHOMP form also arises from the PITF approach when constrained to model stochastic tensors.

3.3 Parameter Optimization

In this section we will apply the principle of maximum likelihood to estimate the model parameters of a RHOMP (i.e., R, Q) directly from data. An alternative would be to estimate the higher-order Markov chain and use the PITF factorization as discussed in the previous section. Working directly on the RHOMP model from data has two advantages: first, the estimate corresponds exactly with the model, rather than estimate and approximate; and second, the direct approach is faster.

We first show how to compute a maximum likelihood estimate with α fixed and then discuss how to pick α. Recall that $c(i,j,k)$ is the total count of transitions moving from j to i with previous state k in the training data. With fixed α, the log likelihood of all transitions from the set $\mathcal{S}$ of user trails is:

$$\log \mathcal{L}(R, Q \mid \mathcal{S}) = \sum_{c(i,j,k)>0} c(i,j,k) \log(\underline{P}_{i,j,k})$$
$$= \sum_{c(i,j,k)>0} c(i,j,k) \log(\alpha R_{i,j} + (1 - \alpha)Q_{i,k}) \quad (5)$$

Our goal is to find a pair of stochastic matrices R, Q which minimizes the negative log likelihood, which gives us the following optimization problem:

$$\begin{aligned}
\underset{R,Q}{\text{minimize}} \quad & -\log \mathcal{L}(R, Q \mid \mathcal{S}) \\
\text{subject to} \quad & R_{i,j} \geq 0, \quad Q_{i,j} \geq 0 \quad 1 \leq i, j \leq N \\
& \sum_i R_{i,j} = 1, \quad \sum_i Q_{i,k} = 1 \quad 1 \leq i \leq N
\end{aligned} \quad (6)$$

This optimization problem is convex as the following theorem shows.

Theorem 3.4. *The negation of the log likelihood function in (5) is convex and so is the feasible region of pairs of stochastic matrices. Thus any local minima solution (R^*, Q^*) is also the solution for global mimima.*

Proof. First we verify the feasible domain of stochastic pairs (R, Q) is convex. We can check that given $0 \leq \lambda \leq 1$ and two stochastic matrices A, B, the linear combination $\lambda A + (1 - \lambda)B$ is also a stochastic matrix. This applies element-wise to the pair to verify the claim.

Now given two sets of stochastic matrices $(R^{(1)}, Q^{(1)})$ and $(R^{(2)}, Q^{(2)})$ and the corresponding linear combination $(R = \lambda R^{(1)} + (1-\lambda)R^{(2)}, Q = \lambda Q^{(1)} + (1 - \lambda)Q^{(2)})$ we have

$$-\log \mathcal{L}(R, Q \mid \mathcal{S}) = -\sum_{i,j,k} c(i,j,k) \log(\alpha R_{i,j} + (1 - \alpha)Q_{i,k})$$
$$= -\sum_{i,j,k} c(i,j,k) \log\left(\lambda(\alpha R_{i,j}^{(1)} + (1 - \alpha)Q_{i,k}^{(1)})\right.$$
$$\left. + (1 - \lambda)(\alpha R_{i,j}^{(2)} + (1 - \alpha)Q_{i,k}^{(2)})\right)$$
$$\leq -\sum_{i,j,k} c(i,j,k)\left(\lambda \log(\alpha R_{i,j}^{(1)} + (1 - \alpha)Q_{i,k}^{(1)})\right.$$
$$\left. + (1 - \lambda) \log(\alpha R_{i,j}^{(2)} + (1 - \alpha)Q_{i,k}^{(2)})\right)$$
$$= -\lambda \log \mathcal{L}(R^{(1)}, Q^{(1)} \mid \mathcal{S}) - (1 - \lambda) \log \mathcal{L}(R^{(2)}, Q^{(2)} \mid \mathcal{S}),$$

where the inequality is from the fact that $-\log$ is a convex function. So (6) is a convex problem. □

We now derive the projected gradient descent algorithm for (6), which is summarized in Algorithm 1. This involves

(1) First update R and Q based on their gradients.
(2) Since R and Q are no longer stochastic due to the above updates, the projection step is applied to project the updated R and Q back to $\ell_1 - balls$ (i.e., the stochastic property).

The gradients over R and Q are:

$$\begin{aligned}
\nabla R_{i,j} &= \frac{-\partial \log \mathcal{L}}{\partial R_{i,j}} = \sum_k \frac{-\alpha c(i,j,k)}{\alpha R_{i,j} + (1 - \alpha)Q_{i,k}} \\
\nabla Q_{i,k} &= \frac{-\partial \log \mathcal{L}}{\partial Q_{i,k}} = \sum_j \frac{-(1 - \alpha)c(i,j,k)}{\alpha R_{i,j} + (1 - \alpha)Q_{i,k}}
\end{aligned} \quad (7)$$

We accomplish the projection step using the algorithm from [16]. Note that for the sake of simplicity we present the projection step by sorting each column vector $\mathbf{w}$ from R and Q, but there is a more efficient method based on divide and conquer [16] which is linear cost to the number non-zeros in $\mathbf{w}$. However in practice sorting $\mathbf{w}$ is fast as the vector $\mathbf{w}$ is very sparse.

Overall each iteration takes linear time in the number of unique triples (i, j, k) in the sequence data. This is upper bounded by the size of input data. We also note that the procedure of computing the gradients $\nabla R, \nabla Q$ and updating R, Q, which dominates the majority of the computation, can be paralleled.

Choosing α. To determine the value of hyperparameter α, we conduct a few trials with α chosen between $(0, 1)$. Then based on the value of the objective function, we calculate the best value of α from a polynomial interpolation of the likelihood function. Getting the global minimum of a polynomial interpolant can be done

Algorithm 1 Max. Likelihood Estimate of a 2nd-order RHOMP

Require: parameter α, step size γ_0 and transition counts $c(i,j,k)$
1: Initialize R with $R_{i,j} = \sum_k c(i,j,k)/\sum_{\ell,k} c(\ell,j,k)$, Q with $Q_{i,k} = \sum_j c(i,j,k)/\sum_{\ell,j} c(\ell,j,k)$ and $\gamma = \gamma_0$
2: **repeat**
3:     Compute the gradient matrices $\nabla R, \nabla Q$ based on (7)
4:     $R \leftarrow (R - \gamma \nabla R)$ and $Q \leftarrow (Q - \gamma \nabla Q)$
5:     **for** each column vector $\mathbf{w}$ of R and Q **do**
6:         Sort the non-zeros of $\mathbf{w}$ into $\mathbf{u}$: $u_1 \geq u_2 \geq \cdots \geq u_k > 0$
7:         Find $\rho = \max\left\{ r \leq k : u_r - \frac{1}{r}(\sum_{i=1}^{r} u_i - 1) > 0 \right\}$
8:         Define $\theta = \frac{1}{\rho}(\sum_{i=1}^{\rho} u_i - 1)$
9:         Update $\mathbf{w}$ with $w_i \leftarrow \max\{w_i - \theta, 0\}$
10:     **end for**
11:     **if** objective value decreases **then**
12:         $\gamma \leftarrow \min\{2 * \gamma, \gamma_0\}$
13:     **else**
14:         $\gamma \leftarrow 0.5 * \gamma$; re-run this iteration with the updated γ
15:     **end if**
16: **until** converge

efficiently, and polynomials can approximate arbitrary continuous functions, which renders this a pragmatic choice. Specifically we use n Chebyshev nodes as the values of α to fit the interpolant: $\alpha_k = \frac{1}{2} + \frac{1}{2}\cos(\frac{2k-1}{2n}\pi)$, $k = 1, 2, \cdots, n$. Chebyshev nodes cluster towards the endpoints of $(0,1)$, which avoids Runge phenomenon, and also admit simple and numerically stable polynomial interpolation algorithms.

Another approach for selecting the value of α is to conduct cross validation with grid search. However a different objective is needed as we could run into unseen transitions in the validation set and the likelihood would go to $-\infty$. Alternatively we can use a measurement like accuracy instead of likelihood. The main advantage of cross validation is its ability to prevent overfitting. In our experiment we find this problem does not occur, so we drop this procedure as it requires substantially more computation.

3.4 Higher-order Cases Beyond Second Order

The ideas discussed in the above sections also work for the higher-order cases with $m \geq 3$. The RHOMP model becomes:

$$\Pr(X_t = i \mid X_{t-1}=j, X_{t-2}=k,...,X_{t-m}=\ell) = \alpha_1 R_{i,j}^{(1)} + \alpha_2 R_{i,k}^{(2)} + \cdots + \alpha_m R_{i,\ell}^{(m)}$$

where $0 \leq \alpha_i \leq 1$ for $i = 1, 2, \cdots, m$, $\sum_i \alpha_i = 1$ and matrices $R^{(i)}$ for $i = 1, 2, \cdots, m$ are stochastic. Similarly, the log likelihood function can be derived as well as the gradient over each $R^{(i)}$. The projected gradient descent algorithm is then applied to update each stochastic matrix $R^{(i)}$, with a per-iteration complexity bounded by the size of the training data.

The biggest difference is that we are no longer able to determine the hyperparameters α_i in a simple fashion as the polynomial interpolation is only computationally efficient for one or two parameters. To address this issue, recall that in Section 3.1 we proposed the model as a retrospective walk, where the walker has probability α_k to step back $k - 1$ steps into their history and then transition according to $R^{(k)}$. Our proposal is to use a single hyperparameter $\beta < 1$ to model a decaying probability of looking back into the

history:

$$\alpha_1 = \frac{1-\beta}{1-\beta^m}, \quad \alpha_2 = \beta \frac{1-\beta}{1-\beta^m}, \quad \dots, \quad \alpha_m = \beta^{m-1}\frac{1-\beta}{1-\beta^m}.$$

(This distribution describes a truncated geometric random variable.) In our experiments for the second-order case the optimal $\alpha_1 > 1/2$ for every dataset. This offers a single step of evidence for this assumption. This β can be chosen either by the procedure of polynomial interpolation or simply using the optimal value α^* from a second-order factorization model $\beta = \alpha^*/(1-\alpha^*)$. We apply the latter approach in our experiments for RHOMP with $m > 2$.

3.5 Relationship with the Linear Additive Markov Process

Our RHOMP model was also proposed as the generalized Linear Additive Markov Process [22] as discussed in the introduction. By way of comparison, the Linear Additive Markov process, which constituted the bulk of [22], corresponds to RHOMP where $R = Q$, or a higher-order RHOMP where $R^{(1)} = R^{(2)} = \cdots = R^{(m)}$. There are additional algorithmic properties of these models shown in that reference, including results on the mixing time and an alternating fitting procedure.

4 RELATED WORK

Modeling User Trails. Early work in [28] characterized the user path patterns on the web with the tools of Markov chains. Other advanced methods include hidden Markov models (HMM) [17], variable length Markov chains [6] and association rules [1]. However the computations associated with the above methods limit them from being used in datasets with more than a few thousand states. More recent work considers the sequence prediction task with personalization, such as collaborative filtering methods [23, 33, 35] where the behavior of similar users is utilized to help the prediction, factorizing personalized Markov chains [29], TribeFlow [19] and recurrent processes [15]. In addition to the prediction problem, clustering and visualization [7], sequence classification [41], metric embedding [9, 10, 18] and hypotheses comparison [36] have also been studied. In the context of this work, we seek to improve the performance of the classic and simple Markov model by studying a structured variation.

Random Walk Models. Since our model is a special case of a higher-order Markov chain, we note that there are relationships with a variety of enhanced Markov models. First our RHOMP model defines a specific form of the Additive Markov Process (AMP) [24], where the transition probability is a summation of a series of memory functions that are restricted on the next state and one history state each. Applications of the AMP include LAMP [22] (see Section 1, 3.5), the gravity models [43], dynamical systems in physics [26, 39] where the memory function is empirically estimated for the application of binary state, and the mixed memory models [34] for language modeling where EM algorithms are applied to solve the mixture model parameters. In addition to the AMP, recent innovations include new recovery results on mixture of Markov chains [20] (a special case of HMM), which assumes a small set of Markov chains that model various classes of latent indent; and the spacey random walk [3, 4, 40] as a non-Markovian

stochastic process that utilizes higher-order information based on the empirical occupation of states.

Tensor Factorization. As already discussed, our work is directly related to the pairwise interaction tensor factorization (PITF) method proposed by Rendle in [29, 30], where the task is to generate *tag* recommendations given the {*user, item*} combination. The PITF model is learned from a binary tensor of triple {*user, item, tag*} by bootstrap sampling from pairwise ranking constrains. Our work differs in the aspect of problem formulation, model construction and parameter optimization. The RHOMP model is also a special case of both the canonical/PARAFAC and Tucker decompositions [21].

5 EXPERIMENTS

We evaluate our RHOMP method on the ability to predict subsequent states in a user trail in terms of accuracy and mean reciprocal rank (MRR) on five different types of data (Section 5.1). We then present the results of a second-order (i.e., $m = 2$) RHOMP compared with baseline methods in Section 5.2 and study over-fitting of the training data in Section 5.3. Then we study what happens for higher-order (i.e., $m > 2$) models in Section 5.4. In all cases, the RHOMP model offers a considerable improvement to existing methods.

5.1 Datasets and Evaluations Setup

The real datasets we use in our experiments cover several applications including: product reviews, online music streaming, checkin locations of social networks and photo uploads. Every dataset is publicly available. For all the datasets, self-loops are removed as we are mostly interested in predicting a non-trivial transition. Also we only consider states that show up more than 20 times. Simple statistics on each dataset are summarized in Table 1, and we now describe them individually.

LastFM [8] is a music streaming and recommendation website (last.fm). We generate user trails as listening histories regarding different artists over a one-year period (2008-05-01 to 2009-05-01).

BeerAdvocate [25] consists of beer reviews spanning more than 10 years up to November 2011 from beeradvocate.com. We study the user trail as reviews over different brewers.

BrightKite [13] was a location-based social networking website where users shared their locations by checking-in. We study the trails of location id.

Flickr [38] contains 100 million Flickr photos/videos provided by Yahoo! Webscope. We extract the user trail based on geolocation (restricted to USA) of each upload after 2008-01-01. Each longitude

Table 1: Dataset characteristics in terms of the number of states, transitions and trails

	# states	# transitions	# trails
LastFM	17,341	2,902,035	195,499
BeerAdvocate	2,324	1,348,903	35,629
BrightKite	11,465	400,340	125,437
Flickr	7,608	1,212,674	97,563
FourSQ	344	198,503	1,480

and latitude is mapped into a grid of approximate 10km by 10km, which constitutes the state.

FourSQ is a location based check-in dataset created by Yang et al [42] which contains checkins from New York City from 24 October 2011 to 20 February 2012. We extract checkin place category (e.g., bus station, hotel, bank) as state.

For experimental methods, we consider the following:

MC1, MC2 are the first-order and second-order Markov chain methods respectively, where the transition matrix is estimated based on maximum likelihood.

Kneser1, Kneser2 are the interpolated Kneser-Ney smoothing methods [11] applied on the first-order and second-order Markov chain methods respectively. This is one of the best smoothing methods for n-gram language models, where it enables higher-order Markov chain transitions to unseen n-grams. We set the discounting parameter as $n_1/(n_1+2n_2)$ by the method of leaving one out [11], where n_1 and n_2 denote the number of n-grams that appear exactly once and twice respectively

PITF is the pairwise interaction tensor factorization method [30] computed on the higher-order Markov chain estimate. Because we use ranking, we consider general positive and negative entries as valid for the factorization. We implement the fitting method ourselves to handle the sparsity in our data. We tune the regularization parameter λ during training. They are $5 \cdot 10^{-5}$ for LastFM, $1 \cdot 10^{-5}$ for BeerAdvocate, and $3 \cdot 10^{-4}$ for the other three datasets. We set the rank number k as 5% of the total number of states, which is enough to accurately capture the user behavior [30]. For SGD optimization we set the learning rate $\alpha = 0.05$, initialization $N(0, 0.01)$ and number of iterations as 10,000,000.

LME is short for Latent Markov Embedding [9]. It is a machine learning algorithm that embeds states into Euclidean space based on a regularized maximum likelihood principle. We set the dimension $d = 50$ and use default values for all other parameters (e.g., learning rate, epsilon). (We tried various values of d spanning from 2 to 100, we find as d increases the performance also gets better, for $d > 50$ the improvements are negligible. So we use $d = 50$ to make the algorithm efficient.) We use the authors' implementations.

RHOMP is our proposed method in this paper. We use initial step size as $\gamma_0 = 1$, and set $\varepsilon = 10^{-5}$ as the algorithm termination criterion when the relative improvement over log likelihood is below this point. For the hyperparameter α we use $n = 15$ Chebyshev nodes for the interpolation.

The datasets are randomly split into a training set (60%) and testing set (40%) based on keeping whole trails together. And for each dataset we conduct experiments over 5 random repetitions and present the average results. For evaluations we use accuracy over top k outputs to measure the accuracy of each method. It is calculated over all individual transitions in the testing set as

$$\text{accuracy}_k = \frac{\text{\# true transitions within top } k \text{ algorithmic results}}{\text{\# total transitions}}.$$

Besides accuracy, which measures the accuracy of the top outputs from algorithms, we also provide results on Mean Reciprocal Rank (MRR). The reciprocal rank of an output is the inverse of the rank of the ground truth answer and MRR measures the overall ranking compared to the groundtruth. For both measures, we want large scores close to 1.

Table 2: Mean Reciprocal Rank (MRR) results of various methods on all datasets. Bold indicates the best mean performance, and ± entries are the standard deviations over 5 trials. Our proposed RHOMP ($m = 2$) has the best performance in all datasets.

|	MC1	MC2	Kneser1	Kneser2	PITF	LME	RHOMP
LastFM	0.071 ± 0.001	0.068 ± 0.001	0.066 ± 0.001	0.090 ± 0.002	0.060 ± 0.001	0.062 ± 0.001	**0.100** ± 0.001
BeerAdvocate	0.080 ± 0.000	0.034 ± 0.001	0.079 ± 0.000	0.076 ± 0.001	0.068 ± 0.002	0.067 ± 0.001	**0.090** ± 0.000
BrightKite	0.551 ± 0.002	0.540 ± 0.002	0.554 ± 0.002	0.599 ± 0.002	0.444 ± 0.007	0.529 ± 0.002	**0.603** ± 0.002
Flickr	0.358 ± 0.003	0.306 ± 0.004	0.350 ± 0.001	0.379 ± 0.001	0.315 ± 0.004	0.333 ± 0.003	**0.410** ± 0.001
FourSQ	0.138 ± 0.004	0.092 ± 0.003	0.146 ± 0.005	0.155 ± 0.004	0.126 ± 0.003	0.113 ± 0.002	**0.181** ± 0.003
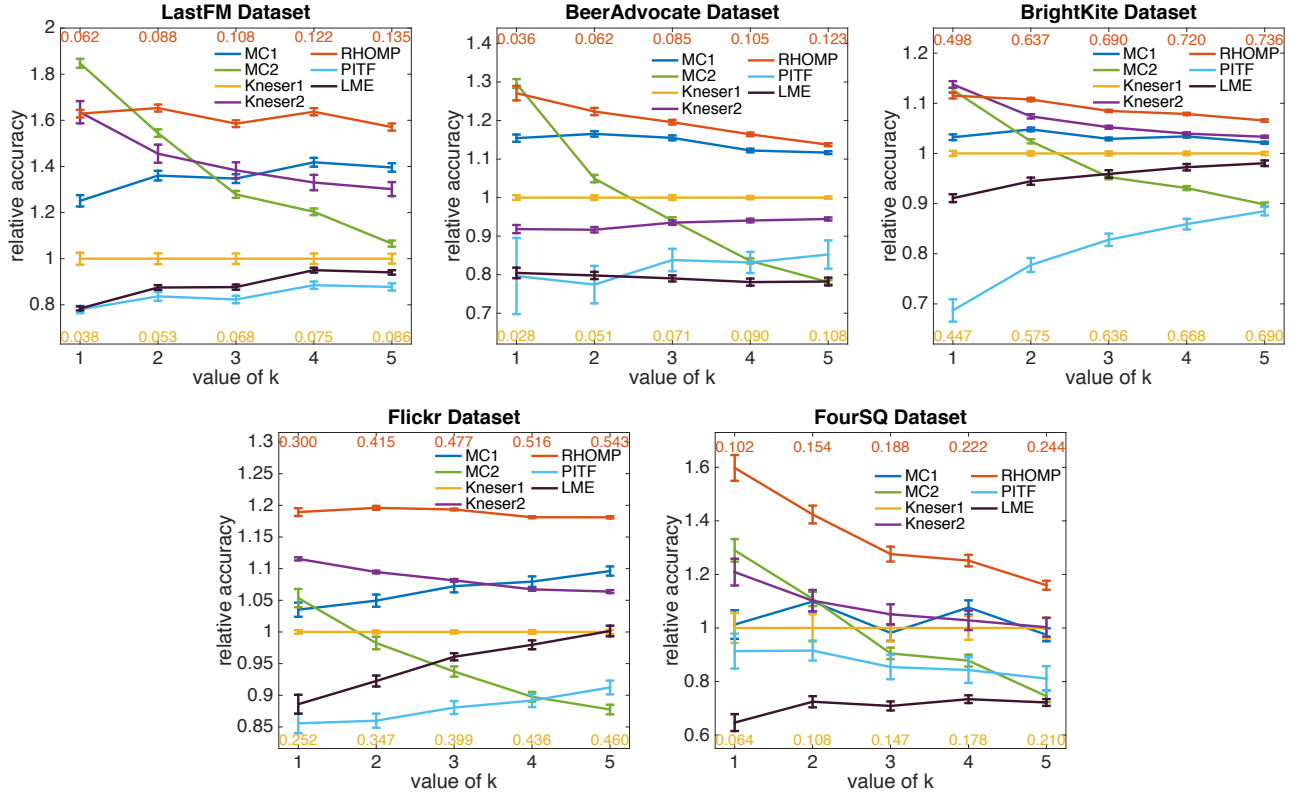
Figure 2: Relative accuracy results on all datasets with $k = 1, 2, 3, 4, 5$. We use Kneser1 as the baseline, and the relative accuracy is calculated as the accuracy ratio to that of Kneser1. The error bars in the figure are the standard deviations over 5 trials. The numbers in the bottom and the top of the figures denote the absolute accuracies for the Kneser1 and our RHOMP method respectively. We see that our RHOMP has noticeable improvements over other methods in most datasets.

5.2 General Results

First we compare our RHOMP ($m = 2$) with other baseline methods in terms of accuracy and MRR score.

MRR score. Table 2 depicts the results on the MRR score. In all datasets, RHOMP has the highest score. From the table we see that MC1 outperforms the LME method. The LME has the advantage of embedding the states into Euclidean space for applications like visualization or clustering. However the embedding could potentially cause information loss, making the prediction less accurate. And we notice that MC2 has the lowest scores in many cases (i.e., Beer-Advocate, Flickr and FourSQ datasets), and the MRR scores drop

compared to MC1. The Kneser-Ney smoothing modification makes the MC2 estimate more robust, and in most cases outperforms the MC1, although such advantage is limited compared to that from our RHOMP method. The PITF method is also not competitive.

Accuracy score. Figure 2 shows the algorithms performances in terms of relative accuracy. Many of the observations from Table 2 on the MRR score also apply here. In addition we find MC2 is often able to provide one accurate output, so the relative accuracy ($k = 1$) is actually quite good in most cases. However as k increases the relative accuracy drops rapidly due to the fact that MC2 is not able to generate a few more reliable outputs. This limits the application

Table 3: Algorithm runtime (in minutes) for the three large datasets in terms of training time (left) and testing time (right). The experiments are run on a single-core of a 2.5Ghz Xeon CPU. Both MC1 and MC2 ran in under a minute.

	Kneser1	Kneser2	PITF	LME	RHOMP
LastFM	2/4	3/75	493/1980	3188/57	52/2
BrightKite	<1/1	<1/4	236/71	1153/22	3/1
Flickr	<1/1	1/8	168/97	764/11	6/1

of MC2 because in the task of recommendation, it is important for the algorithm to generate a few instead of one unique candidate state. Another observation is that the results of PITF over different trials are often more volatile because of its underlying stochastic gradient descent solver. We also find that for some datasets (e.g., BeerAdvocate and FourSQ) the relative accuracies of our RHOMP decrease as k increases. The reason is that as k increases, the prediction task itself becomes easier, so it is hard to maintain the same advantage (i.e., constant relative accuracy). This also explains why methods like LME and PITF can catch up as k increases.

Algorithm Runtime. Table 3 shows the runtime for each method. The RHOMP approach takes slightly more time to train than Kneser-Ney methods, but has faster prediction and testing. It is slower than the pure MC methods, but much faster than PITF, LME.

5.3 Analysis on Overfitting

One of the reasons we propose the RHOMP method is to improve the higher-order Markov chain method in the aspect of overfitting. In this section we analyze the results in detail and give an explanation on the performances of different methods.

First we show the comparison between training and testing performance in Table 4. We present the result using accuracy with $k = 3$ as it is representative of the remaining results. Both PITF and LME had the least overfitting effect as the testing and training accuracies are very close. However, their testing accuracies are also low. The training accuracy of MC2 is the highest for all datasets. But these are often more than 10 times of the corresponding testing accuracies. So MC2 is a highly overfitting method. Kneser2 also has comparatively high training accuracy since it is a second-order method and tends to fit the training data well. But the performance on testing set is better than MC2 as it uses lower-order information to smooth the output. The methods MC1, Kneser1 and RHOMP have a good training and testing balance, and among them, our RHOMP has superior testing performances.

Next we analyze the performance on individual states to help understand the behaviors of different algorithms. We sort all the states from high to low based on the total number of counts of each state in the training set. Our aim is to look at how testing accuracy correlates with these counts. Figure 3 shows the accuracy ($k = 3$) comparisons (i.e., MC1 vs MC2 vs RHOMP and Kneser1 vs Kneser2 vs RHOMP) on the Flickr dataset based on counts of the states. We aggregate small sets of states based on their counts into baskets of at least 1000 transitions and 5 states. We find that all methods show accuracy drops when predicting infrequent states, with MC2

being affected most. Here, RHOMP does the best out of all methods, which reflects its ability to avoid overfitting.

5.4 Analysis on Higher-order Approaches

In the previous sections, we analyze the results for first and second-order approaches. Now we study the behavior as the order varies. Figure 4 shows change in performance as the order increases for the three frameworks: MC, Kneser-Ney smoothing and RHOMP. For the cases when the history states length is smaller than the order, we use the approach with the correct order to generate the prediction.

For the MC framework, higher-order approaches make the prediction less accurate. This occurs because these methods overfit the training data and there are more ways to overfit for a higher-order chain. For the Kneser-Ney smoothing approaches, in most cases (except BeerAdvocate dataset) there are improvements moving from first-order to second-order. However the improvements are slight. For order > 2, there are usually either no clear improvements or small performance dips. The reason is that as the order increase, the higher-order transition become very sparse, and could easily encounter an unseen higher-order state. So in this case the algorithm will frequently seek the prediction from a lower-order approach.

For the RHOMP framework, there are improvements for each dataset when moving from MC1 to RHOMP with order = 2, and for order > 3, the results further improve. Compared to MC and Kneser-Ney smoothing frameworks, The RHOMP is more robust in terms of not decreasing the accuracy as order increases, with the exception of BrightKite dataset. In BrightKite, the average trail length is around 3, so there is insufficient information to train higher-order models and we lack the lower-order fallback in Kneser-Ney.

6 SUMMARY AND FUTURE WORK

In this paper we study the problem of modeling user trails, which encode useful information for the downstream applications of user experiences, recommendations and advertising. We propose a new class of structured higher-order Markov chains which we call the retrospective higher-order Markov process (RHOMP). This model preserves the higher-order nature of user trails without risks of overfitting the data. A RHOMP can be estimated from data via maximum likelihood estimation (MLE) using a projected gradient descent algorithm. In the experiments, we find that RHOMPs are superior in terms of accuracy and mean reciprocal rank compared to other methods. Also RHOMPs are robust for higher-order chains when there is data available.

There are several directions to extend this work. First it would be interesting to explore other forms of retrospection that allow more interaction between the history states. (Note that the current approach in this paper selects a single state during the retrospective process). This will allow to model the case when certain combined history states have strong evidence in terms of transition patterns. Second it would also be useful to extend this framework in terms of personalization. This can be achieved by a tensor factorization approach or a collaborative filtering method. Lastly we also would like to embed time information into our prediction either by modeling the event time directly or using it as a side information to

Table 4: accuracy (k=3) results for testing set (the left number) vs train set (the right number) that we use to estimate overfitting. Bold denotes the highest testing result. We judge the overfitting effects as {MC2, Kneser2} ≫ {MC1, Kneser1, RHOMP} > {LME, PITF}. But LME and PITF have poor test performances.

|	MC1	MC2	Kneser1	Kneser2	PITF	LME	RHOMP
LastFM	0.092/0.216	0.087/0.961	0.068/0.109	0.094/0.792	0.056/0.061	0.062/0.083	**0.108**/0.218
BeerAdvocate	0.082/0.115	0.067/0.777	0.071/0.074	0.066/0.490	0.059/0.061	0.056/0.600	**0.085**/0.109
BrightKite	0.654/0.782	0.606/0.940	0.636/0.729	0.669/0.868	0.526/0.584	0.610/0.665	**0.690**/0.796
Flickr	0.428/0.496	0.374/0.832	0.399/0.440	0.432/0.710	0.352/0.373	0.384/0.401	**0.477**/0.530
FourSQ	0.145/0.199	0.133/0.778	0.147/0.174	0.155/0.524	0.126/0.140	0.104/0.137	**0.188**/0.241
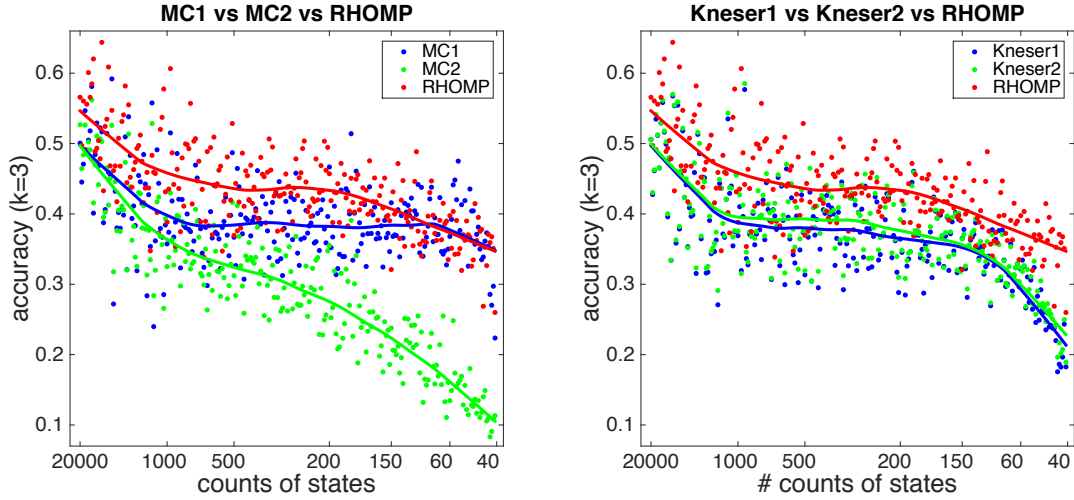
Figure 3: State-wise accuracy ($k = 3$) comparison on MC1 vs MC2 vs RHOMP (left figure) and Kneser1 vs Kneser2 vs RHOMP (right figure) on the Flickr dataset. Each marker represents the average accuracy over a group of states. The curves are fit from the scatter points based on Locally Weighted Scatterplot Smoothing (LOWESS).
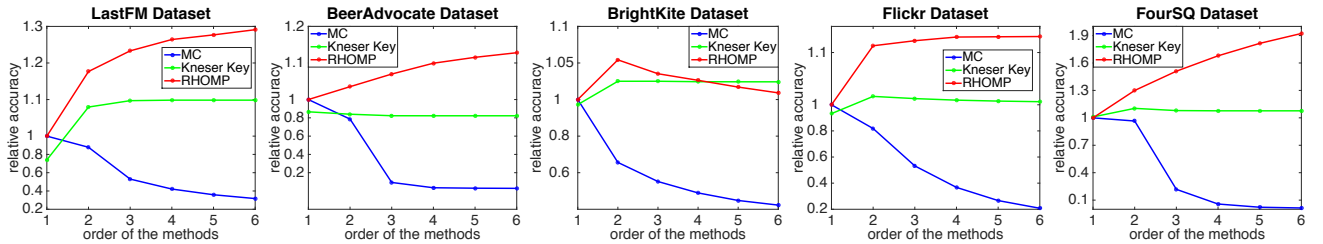

Figure 4: Relative accuracy ($k = 3$) vs order of the methods: MC, Kneser-Ney smoothing and RHOMP. The relative accuracy is the accuracy ratio to that from MC1 of the corresponding datasets. Note that the y-axis may not be scaled linearly to make the figures more clear.

help generate a non-stationary process where the random walk behavior could change overtime.

REFERENCES
[1] M. A. Awad and I. Khalil. Prediction of user's web-browsing behavior: Application of Markov model. *IEEE T. Syst. Man Cy. B*, 42(4):1131–1142, 2012.

[2] A. Benson, D. F. Gleich, and J. Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

[3] A. R. Benson, D. F. Gleich, and J. Leskovec. Tensor spectral clustering for partitioning higher-order network structures. In *SDM*, pages 118–126, 2015.

[4] A. R. Benson, D. F. Gleich, and L.-H. Lim. The spacey random walk: A stochastic process for higher-order data. *SIAM Rev.*, page To appear, 2017.

[5] J. Borges and M. Levene. Evaluating variable-length Markov chain models for analysis of user web navigation sessions. *IEEE T. Knowl. Data En.*, 19(4), 2007.

[6] P. Bühlmann, A. J. Wyner, et al. Variable length Markov chains. *The Annals of Statistics*, 27(2):480–513, 1999.

[7] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Model-based clustering and visualization of navigation patterns on a web site. *Data Mining and Knowledge Discovery*, 7(4):399–424, 2003.

[8] Ò. Celma Herrada. *Music recommendation and discovery in the long tail.* PhD thesis, Universitat Pompeu Fabra, 2009.

[9] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims. Playlist prediction via metric embedding. In *KDD*, pages 714–722, 2012.

[10] S. Chen, J. Xu, and T. Joachims. Multi-space probabilistic sequence modeling. In *KDD*, pages 865–873, 2013.

[11] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *ACL*, pages 310–318, 1996.

[12] F. Chierichetti, R. Kumar, P. Raghavan, and T. Sarlos. Are web users really Markovian? In *WWW*, pages 609–618, 2012.

[13] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *KDD*, pages 1082–1090, 2011.

[14] M. Deshpande and G. Karypis. Selective Markov models for predicting web page accesses. *ACM T. Internet Techno.*, 4(2):163–184, 2004.

[15] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*, pages 1555–1564, 2016.

[16] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *ICML*, pages 272–279, 2008.

[17] S. R. Eddy. Hidden Markov models. *Current opinion in structural biology*, 6(3):361–365, 1996.

[18] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan. Personalized ranking metric embedding for next new POI recommendation. In *IJCAI*, pages 2069–2075, 2015.

[19] F. Figueiredo, B. Ribeiro, J. M. Almeida, and C. Faloutsos. TribeFlow: Mining & predicting user trajectories. In *WWW*, pages 695–706, 2016.

[20] R. Gupta, R. Kumar, and S. Vassilvitskii. On mixtures of Markov chains. In *NIPS*, pages 3441–3449, 2016.

[21] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009.

[22] R. Kumar, M. Raghu, T. Sarlós, and A. Tomkins. Linear additive markov processes. In *WWW*, pages 411–419, 2017.

[23] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*, pages 931–940, 2008.

[24] A. Markov. Extension of the limit theorems of probability theory to a sum of variables connected in a chain. 1971.

[25] J. J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*, pages 897–908, 2013.

[26] S. Melnyk, O. Usatenko, and V. Yampol'skii. Memory functions of the additive markov chains: applications to complex dynamic systems. *Physica A: Statistical Mechanics and its Applications*, 361(2):405–415, 2006.

[27] Y. Peres and P. Shields. Two new Markov order estimators. *arXiv preprint math/0506080*, 2005.

[28] P. L. Pirolli and J. E. Pitkow. Distributions of surfers' paths through the world wide web: Empirical characterizations. *World Wide Web*, 2(1-2):29–45, 1999.

[29] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*, pages 811–820, 2010.

[30] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90, 2010.

[31] D. Ron, Y. Singer, and N. Tishby. Learning probabilistic automata with variable memory length. In *COLT*, pages 35–46. ACM, 1994.

[32] M. Rosvall, A. V. Esquivel, A. Lancichinetti, J. D. West, and R. Lambiotte. Memory in network flows and its effects on spreading dynamics and community detection. *Nature communications*, 5, 2014.

[33] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *ICML*, pages 880–887. ACM, 2008.

[34] L. K. Saul and M. I. Jordan. Mixed memory Markov models: Decomposing complex stochastic processes as mixtures of simpler ones. *Machine learning*, 37(1):75–87, 1999.

[35] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):3, 2014.

[36] P. Singer, D. Helic, A. Hotho, and M. Strohmaier. Hyptrails: A Bayesian approach for comparing hypotheses about human trails on the web. In *WWW*, pages 1003–1013, 2015.

[37] H. M. Taylor and S. Karlin. *An Introduction to Stochastic Modeling.* Academic press, 2014.

[38] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 1(8), 2015.

[39] O. Usatenko. *Random finite-valued dynamical systems: additive Markov chain approach.* Cambridge Scientific Publishers, 2009.

[40] T. Wu, A. R. Benson, and D. F. Gleich. General tensor spectral co-clustering for higher-order data. In *NIPS*, pages 2559–2567, 2016.

[41] Z. Xing, J. Pei, and E. Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, 12(1):40–48, 2010.

[42] D. Yang, D. Zhang, Z. Yu, and Z. Yu. Fine-grained preference-aware location search leveraging crowdsourced digital footprints from LBSNs. In *UbiComp*, pages 479–488, 2013.

[43] J.-D. Zhang and C.-Y. Chow. Spatiotemporal sequential influence modeling for location recommendations: A gravity-based approach. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(1):11, 2015.