Graph-Based Recommendation with Personalized Diffusions

Athanasios N. Nikolakopoulos
University of Minnesota

Dimitris Berberidis
University of Minnesota

George Karypis
University of Minnesota

Georgios B. Giannakis
University of Minnesota

ABSTRACT

The present work introduces PerDif; a novel framework for learning personalized diffusions over item-to-item graphs for top-n recommendation. PerDif learns the teleportation probabilities of a time-inhomogeneous random walk with restarts capturing a user-specific underlying item exploration process. Such approach can lead to significant improvements in recommendation accuracy, while also providing useful information about the users in the system. Per-user fitting can be performed in parallel and very efficiently even in large-scale settings. A comprehensive set of experiments on real-world datasets demonstrate the scalability as well as the qualitative merits of the proposed framework. PerDif achieves high recommendation accuracy, outperforming state-of-the-art competing approaches—including several recently proposed methods relying on deep neural networks.

KEYWORDS

Top-N Recommendation, Random Walks, Item Models

ACM Reference Format:
Athanasios N. Nikolakopoulos, Dimitris Berberidis, George Karypis, and Georgios B. Giannakis. 2019. Graph-Based Recommendation with Personalized Diffusions. In *Proceedings of International Workshop for Mining and Learning with Graphs (MLG'19).* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Top-n recommendation algorithms have become an indispensable component of most e-commerce applications as well as content delivery platforms. They work by providing ordered lists of items attuned to the particular tastes of the users, as depicted by their past interactions within the system. *Item-based methods* are among the most popular approaches for top-n recommendation. Such methods work by building a model that depicts the relations between the items, which is then used to recommend new items that are related to the ones each user has consumed in the past. Item-based methods are known to achieve high top-n recommendation accuracy while being scalable and easy to interpret [23]. The fact, however, that they typically rely only on direct item-to-item relations can impose limitations to their quality and make them brittle to the presence of sparsity—leading to poor itemspace coverage and substantial decay in performance [22]. A promising direction towards ameliorating such problems involves treating item models as graphs onto which random-walk-based techniques can then be applied to *diffuse* each user's historic preferences across the itemspace [21]. Combining random walks and item models allows for more effective use of the information captured in the item model; considering direct as well as transitive relations between the items, and also alleviating sparsity related problems. However, the adopted mechanism for propagating users' preferences along the edges of the item graph in existing approaches (i.e., the diffusion function, or the specific landing probability distribution to be used) is *the same* for all users in the system. This disregards potential variability in user behavior and can impose limitations in recommendation accuracy.

To address such limitations, we introduce **PerDif**; a novel framework for learning Personalized Diffusions over item models for top-n recommendation. PerDif introduces an item exploration stochastic process defined as a time-inhomogeneous random walk with *step-specific* teleportation probabilities adapted to *each user* directly from data. We show that by leveraging the properties of multidamping processes [17] the original formulation of the core optimization problem can be transformed to an equivalent form amenable to efficient solutions. Per-user fitting can be done in parallel and very efficiently even for large datasets, thereby enabling such task to be performed in real-time in practical settings[1]. A comprehensive set of experiments on real-world datasets illustrate the potential of the proposed methodology in providing a framework for improving the performance of item models. PerDif achieves high recommendation accuracy outperforming state-of-the-art competing approaches—including several recently proposed methods relying on deep neural networks. Besides its merits in terms of recommendation quality, the diffusions learned by PerDif can also facilitate further analysis of the system. As an example we show how one can use the learned diffusion coefficients to identify users for which the model will most likely lead to poor predictions, at training-time—thereby affording preemptive handling and interventions.

2 PERSONALIZED DIFFUSIONS

Definitions. Let $\mathcal{U} = \{1, \ldots, U\}$ be a set of *users* and $\mathcal{I} = \{1, \ldots, I\}$ a set of *items*. Let $\mathbf{R} \in \Re^{U \times I}$ be the *user-item interaction matrix*; i.e., the matrix whose ui-th element is 1 if user u has interacted with item i, and 0 otherwise. Each user $u \in \mathcal{U}$ is modeled by a vector $\mathbf{r}_u^\top \in \Re^I$, i.e., the corresponding row of $\mathbf{R}$; similarly, each item $i \in \mathcal{I}$ is modeled by a vector $\mathbf{r}_i \in \Re^U$ which coincides with the corresponding column of matrix $\mathbf{R}$. We use the term *item model* to refer to a matrix $\mathbf{W} \in \Re^{I \times I}$ whose ij-th element gives a measure of *proximity* between items i and j. Finally, we use the term *item graph* to refer to a graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ with adjacency matrix $\mathbf{W}$.

This work was supported by NSF 171141, NSF 1514056 and NSF 1500713.

MLG'19, August 2019, Alaska, USA
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] Parallel implementation of the method in C can be found *here*.

2.1 Motivation

Imagine of a random walker "jumping" from node to node on an item graph $\mathcal{G}$ with transition probabilities arising by the proximity scores of an underlying item model. If the initial distribution of this walker reflects the items consumed by a user u in the past, the probability the walker "lands" on different nodes after K steps provide an intuitive measure of proximity that can be used to rank the nodes and recommend new items to user u accordingly. Specifically, if $\mathbf{P}$ denotes the transition probability matrix of the walk, personalized recommendations for each user u can be produced e.g., by leveraging the *K-step landing probability distributions* of a walk rooted on the items consumed by u:

$$\boldsymbol{\pi}_u^\mathsf{T} \triangleq \boldsymbol{\omega}_u^\mathsf{T}\mathbf{P}^K, \qquad \boldsymbol{\omega}_u^\mathsf{T} \triangleq \frac{\mathbf{r}_u^\mathsf{T}}{\|\mathbf{r}_u^\mathsf{T}\|_1} \tag{1}$$

or by computing the limiting distribution of a *random walk with restarts* on $\mathbf{P}$, using $\boldsymbol{\omega}_u^\mathsf{T}$ as the restarting distribution. The latter approach gives rise to the well-known *personalized PageRank* [24] diffusion with teleportation vector $\boldsymbol{\omega}_u^\mathsf{T}$ and teleportation probability $1-p$. Its limiting distribution can be expressed as

$$\boldsymbol{\pi}_u^\mathsf{T} \triangleq \lim_{K\to\infty} \boldsymbol{\omega}_u^\mathsf{T}(p\mathbf{P} + (1-p)\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T})^K. \tag{2}$$

Such approaches were recently shown to increase recommendation accuracy with respect to simply using the item model directly [21]. Personalization of the recommendation vectors in both schemes comes from the use of $\boldsymbol{\omega}_u$; either as the 'root' distribution in (1), or as the restarting distribution in (2). However, the underlying mechanism for propagating user preferences, $\boldsymbol{\omega}_u$, across the itemspace (i.e., the adopted diffusion function, or the choice of the K-step distribution) is *fixed* for every user in the system. From a user modeling point of view this translates to the implicit assumption that every user explores the itemspace in *exactly the same* way—overlooking the fact that different users can have different behavioral patterns.

The fundamental premise behind this work is that the latent item exploration behavior of the users can be captured better by *user-specific* preference propagation mechanisms; thus, leading to better recommendations. Motivated by this, we introduce PerDif.

2.2 The PerDIF Item Discovery Process

Consider a random walker carrying a bag of K biased coins. The coins are labeled with consecutive integers from 1 to K. Initially, the random walker occupies the nodes of graph $\mathcal{G}$ according to distribution $\boldsymbol{\omega}$. She then flips the 1st coin: if it turns heads (with probability μ_1), she jumps to a different node in the graph abiding by the probability matrix $\mathbf{P}$; if it turns tails (with probability $1-\mu_1$), she jumps to a node according to the probability distribution $\boldsymbol{\omega}$. She then flips the 2nd coin and she either follows $\mathbf{P}$ with probability μ_2 or 'restarts' to $\boldsymbol{\omega}$ with probability $(1-\mu_2)$. The walk continues until she has used all her K coins. At the k-th step the transitions of the random walker are completely determined by the probability the k-th coin turning heads (μ_k), the transition matrix $\mathbf{P}$, and the restarting distribution $\boldsymbol{\omega}$. Thus, the stochastic process that governs the position of the random walker over time is a time-inhomogeneous Markov chain with state space the nodes of the graph, and transition matrix at time k given by

$$\mathbf{G}(\mu_k) \triangleq \mu_k\mathbf{P} + (1-\mu_k)\mathbf{1}\boldsymbol{\omega}^\mathsf{T}. \tag{3}$$

The node occupation distribution of the random walker after the last transition can therefore be expressed as

$$\boldsymbol{\pi}^\mathsf{T} \triangleq \boldsymbol{\omega}^\mathsf{T}\mathbf{G}(\mu_1)\mathbf{G}(\mu_2)\cdots\mathbf{G}(\mu_K). \tag{4}$$

The above stochastic process aims to provide a simple model of the item exploration procedure taking place when users interact with a real system. At each step the users might either decide to go forth and discover items related to the ones they are currently considering, or return to their base and possibly go down alternative paths. Different users, might explore the itemspace in different ways; and their behavior might change throughout the exploration session.

Given an item transition probability matrix $\mathbf{P}$ and a user-specific restarting distribution $\boldsymbol{\omega}_u$, our goal is to find a set of probabilities $\boldsymbol{\mu}_u \triangleq \left[\mu_1,\ldots,\mu_K\right]$ so that the outcome of the aforementioned item exploration process yields a meaningful distribution over the items that can be used for recommendation.

2.3 Learning the diffusions

For each user u we randomly sample one item she has interacted with (henceforth referred to as the 'target' item) alongside τ_{neg} unseen items, and we fit $\boldsymbol{\mu}_u$ so that the node occupancy distribution after a K-step item exploration process rooted on $\boldsymbol{\omega}_u$ (cf (4)) yields high probability to the target item while keeping the probabilities of the negative items low. Concretely, upon defining a vector $\mathbf{h}_u \in \Re^{\tau_{neg}+1}$ which contains the value 1 for the target item and zeros for the negative items, we learn $\boldsymbol{\mu}_u$ by solving

$$\begin{aligned} \underset{\boldsymbol{\mu}_u\in\Re^K}{\text{minimize}} \quad & \|\boldsymbol{\omega}_u^\mathsf{T}\mathbf{G}(\mu_1)\cdots\mathbf{G}(\mu_K)\mathbf{E}_u - \mathbf{h}_u^\mathsf{T}\|_2^2 \\ \text{subject to} \quad & \mu_i \in (0,1), \quad \forall i\in[K] \end{aligned} \tag{5}$$

where $\mu_i \triangleq [\boldsymbol{\mu}_u]_i, \forall i$, and $\mathbf{E}_u$ is a $(I\times(\tau_{neg}+1))$ matrix designed to select and rearrange the elements of the vector $\boldsymbol{\omega}_u^\mathsf{T}\mathbf{G}(\mu_1)\cdots\mathbf{G}(\mu_K)$ according to the sequence of items comprising $\mathbf{h}_u$. Upon obtaining $\boldsymbol{\mu}_u$, personalized recommendations for user u can be computed as

$$\boldsymbol{\pi}_u^\mathsf{T} \triangleq \boldsymbol{\omega}_u^\mathsf{T}\mathbf{G}(\mu_1)\cdots\mathbf{G}(\mu_K). \tag{6}$$

Due to the multiplicative dependencies among the optimization variables, problem (5) appears to be difficult to tackle—especially as the value of K gets larger. Nevertheless, by leveraging the properties of $\mathbf{G}$ the above non-linear optimization problem can be solved efficiently.

Theorem 2.1. *The optimization problem* (5) *is equivalent to*

$$\underset{\boldsymbol{\phi}(\boldsymbol{\mu}_u)\in\Delta_{++}^{K+1}}{\text{minimize}} \quad \|\boldsymbol{\phi}(\boldsymbol{\mu}_u)^\mathsf{T}\mathbf{S}_u\mathbf{E}_u - \mathbf{h}_u^\mathsf{T}\|_2^2$$

where $\Delta_{++}^{K+1} = \{\mathbf{x} : \mathbf{x}^\mathsf{T}\mathbf{1} = 1, \mathbf{x} > \mathbf{0}\}$ *and*

$$\mathbf{S}_u \triangleq \begin{bmatrix} \boldsymbol{\omega}_u^\mathsf{T} \\ \boldsymbol{\omega}_u^\mathsf{T}\mathbf{P} \\ \boldsymbol{\omega}_u^\mathsf{T}\mathbf{P}^2 \\ \vdots \\ \boldsymbol{\omega}_u^\mathsf{T}\mathbf{P}^K \end{bmatrix}, \boldsymbol{\phi}_u \equiv \boldsymbol{\phi}(\boldsymbol{\mu}_u) \triangleq \begin{bmatrix} 1-\mu_K \\ \mu_K(1-\mu_{K-1}) \\ \mu_K\mu_{K-1}(1-\mu_{K-2}) \\ \vdots \\ \mu_K\cdots\mu_2(1-\mu_1) \\ \mu_K\cdots\mu_2\mu_1 \end{bmatrix}.$$

Proof. We will make use of the following identity the proof of which can be found in the Appendix. To simplify the notation we exploit that $\prod\emptyset = 1$ by definition.

Lemma 2.2. *For all $N \in \mathbb{Z}_+$ and for $\{\mu_i\}_{i\in N} \in (0,1)$ it holds*

$$\sum_{i=1}^{N}\left((1-\mu_i)\left(\prod_{\ell=i+1}^{N}\mu_\ell\right)\right)+\left(\prod_{\ell=1}^{N}\mu_\ell\right)=1 \tag{7}$$

First we show that $\prod_{j=1}^{K}\mathrm{G}(\mu_j)$ can be expressed as

$$\overbrace{\left(\prod_{\ell=1}^{K}\mu_\ell\right)\mathbf{P}^K}^{\text{Term A}}+\overbrace{\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T}\sum_{i=1}^{K}\left((1-\mu_i)\left(\prod_{\ell=i+1}^{K}\mu_\ell\right)\mathbf{P}^{K-i}\right)}^{\text{Term B}}.$$

This can be proved by induction. Indeed, for $K=1$ the statement is trivially true. Assuming it is true for $K=M-1$, multiplication with $\mathrm{G}(\mu_M)$

$$(\mathtt{A}+\mathtt{B})(\mu_M\mathbf{P}\overbrace{\quad}^{\mathtt{Term\ C}}+\overbrace{(1-\mu_M)\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T}}^{\mathtt{Term\ D}})$$

can be computed as

$$\mathtt{A}\times\mathtt{C}=\left(\prod_{\ell=1}^{M}\mu_\ell\right)\mathbf{P}^M \tag{8}$$

$$\mathtt{A}\times\mathtt{D}=\left(\prod_{\ell=1}^{M-1}\mu_\ell\right)\mathbf{P}^{M-1}(1-\mu_M)\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T}$$

$$=(1-\mu_M)\left(\prod_{\ell=1}^{M-1}\mu_\ell\right)\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T} \tag{9}$$

$$\mathtt{B}\times\mathtt{C}=\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T}\sum_{i=1}^{M-1}\left((1-\mu_i)\left(\prod_{\ell=i+1}^{M}\mu_\ell\right)\mathbf{P}^{M-i}\right) \tag{10}$$

$$\mathtt{B}\times\mathtt{D}=\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T}\sum_{i=1}^{M-1}\left((1-\mu_i)\left(\prod_{\ell=i+1}^{M-1}\mu_\ell\right)\mathbf{P}^{M-1-i}\right)(1-\mu_M)\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T}$$

$$=\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T}(1-\mu_M)\sum_{i=1}^{M-1}\left((1-\mu_i)\left(\prod_{\ell=i+1}^{M-1}\mu_\ell\right)\right) \tag{11}$$

where (9) and (11) follow from the stochasticity of $\mathbf{P}$. Adding (9) and (11) (also using Lemma 2.2) yields

$$(9)+(11)=(1-\mu_M)\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T}. \tag{12}$$

Then adding (12), (8) and (10) establishes the induction step[2]

$$\left(\prod_{\ell=1}^{M}\mu_\ell\right)\mathbf{P}^M+\mathbf{1}\boldsymbol{\omega}_u^\mathsf{T}\sum_{i=1}^{M}\left((1-\mu_i)\left(\prod_{\ell=i+1}^{M}\mu_\ell\right)\mathbf{P}^{M-i}\right).$$

Therefore by carefully rearranging the terms we can express the minimization objective of problem (5) as

$$\|\boldsymbol{\phi}(\boldsymbol{\mu}_u)^\mathsf{T}\mathbf{S}_u\mathbf{E}_u-\mathbf{h}_u^\mathsf{T}\|_2^2$$

with $\mathbf{S}_u$ and $\boldsymbol{\phi}(\boldsymbol{\mu}_u)$ defined as in the statement of the theorem. To formally establish the equivalence of the two problems it suffices to show that $\boldsymbol{\phi}:\mathfrak{R}^K_{(0,1)}\mapsto\Delta^{K+1}_{++}$ is one-to-one and onto [4].

One-to-one: Let $\boldsymbol{\mu},\boldsymbol{\mu}'\in\mathfrak{R}^K_{(0,1)}$. It suffices to show that if $\boldsymbol{\phi}(\boldsymbol{\mu})=\boldsymbol{\phi}(\boldsymbol{\mu}')$ then $\boldsymbol{\mu}=\boldsymbol{\mu}'$. When $\boldsymbol{\phi}(\boldsymbol{\mu})=\boldsymbol{\phi}(\boldsymbol{\mu}')$, we have that $\phi_1=\phi_1'$ from which we directly establish that

$$1-\mu_K=1-\mu_K'\Rightarrow\mu_K=\mu_K'. \tag{13}$$

From $\phi_2=\phi_2'$ together with (13) we get

$$\mu_K(1-\mu_{K-1})=\mu_K'(1-\mu_{K-1}')\overset{\mu_K>0}{\Longrightarrow}\mu_{K-1}=\mu_{K-1}'.$$

Following similar arguments we establish $\mu_{K-2}=\mu_{K-2}',\mu_{K-3}=\mu_{K-3}',\cdots,\mu_1=\mu_1'$. Therefore, $\boldsymbol{\mu}=\boldsymbol{\mu}'$ as needed.

Onto: It suffices to show that for every $\boldsymbol{\chi}\in\Delta^{K+1}_{++}$, there exists a $\boldsymbol{\mu}\in\mathfrak{R}^K_{(0,1)}$ such that $\boldsymbol{\phi}(\boldsymbol{\mu})=\boldsymbol{\chi}$. Let $\boldsymbol{\chi}=\left[\chi_1,\chi_2,\ldots,\chi_{K+1}\right]$ be an arbitrary vector in Δ^{K+1}_{++}. We have

$$(1-\mu_K)=\chi_1\overset{\chi_1\in(0,1)}{\Longrightarrow}\mu_K=1-\chi_1\in(0,1).$$

Now

$$\mu_K(1-\mu_{K-1})=\chi_2$$

$$(1-\chi_1)(1-\mu_{K-1})=\chi_2\overset{\chi_1\neq1}{\Longrightarrow}\mu_{K-1}=\frac{1-\chi_1-\chi_2}{1-\chi_1}$$

Since $\boldsymbol{\chi}\in\Delta^{K+1}_{++}$, we know that $\chi_1+\chi_2<1$. Therefore $\chi_2<1-\chi_1$, which means that $\mu_{K-1}\in(0,1)$ as needed. Similarly

$$\mu_K\mu_{K-1}(1-\mu_{K-2})=\chi_3$$

$$\mu_{K-2}=\frac{1-\chi_1-\chi_2-\chi_3}{1-\chi_1-\chi_2}$$

$$\vdots$$

$$\mu_1=\frac{1-\chi_1-\cdots-\chi_K}{1-\chi_1-\cdots-\chi_{K-1}} \tag{14}$$

with $\mu_i\in(0,1),\forall i$. Thus, starting from an arbitrary $\boldsymbol{\chi}\in\Delta^{K+1}_{++}$ we can always get a $\boldsymbol{\mu}\in\mathfrak{R}^K_{(0,1)}$ such that $\boldsymbol{\phi}(\boldsymbol{\mu})=\boldsymbol{\chi}$. Hence, both properties of the map $\boldsymbol{\phi}$ have been established and the proof is complete. □

Theorem 2.1 simplifies learning $\boldsymbol{\mu}_u$ significantly. Indeed, by exploiting the intrinsic connections between functional rankings [1] and multidamping processes [17], the task reduces to finding diffusion coefficients $\boldsymbol{\phi}_u$ over the space of the first K landing probabilities of a walk rooted on $\boldsymbol{\omega}_u$ (see definition of $\mathbf{S}_u$). Afterwards $\boldsymbol{\mu}_u$ can be obtained in linear time from $\boldsymbol{\phi}_u$ upon solving the forward recurrence (14). Taking into account the fact that in recommendation settings K will typically be small[3] and $\boldsymbol{\omega}_u,\mathbf{P}$ sparse, building 'on-the-fly' $\mathbf{S}_u\mathbf{E}_u$ row-by-row, and solving the $(K+1)$-dimensional convex quadratic problem

$$\boxed{\textsc{PerDif}^{\textsc{free}}:\quad\underset{\boldsymbol{\phi}_u\in\Delta^{K+1}_{++}}{\text{minimize}}\quad\|\boldsymbol{\phi}_u^\mathsf{T}\mathbf{S}_u\mathbf{E}_u-\mathbf{h}_u^\mathsf{T}\|_2^2} \tag{15}$$

can be performed very efficiently (typically in a matter of milliseconds even in large scale settings).

Moreover, working on the space of landing probabilities can also facilitate parametrising the diffusion coefficients within a family of known diffusions. This motivates the parameterized variant of PerDif

$$\boxed{\textsc{PerDif}^{\textsc{par}}:\quad\underset{\boldsymbol{\gamma}_u\in\Delta^{L}_{+}}{\text{minimize}}\quad\|\boldsymbol{\gamma}_u^\mathsf{T}\mathbf{D}\mathbf{S}_u\mathbf{E}_u-\mathbf{h}_u^\mathsf{T}\|_2^2} \tag{16}$$

[2] For an alternative induction-based proof of this the reader can see [17].

[3] Note that as K gets large the usefulness of these vectors in terms of recommendation will start to decay since more and more probability mass will get concentrated to the popular items in the system.

with $\Delta_+^L \triangleq \{\mathbf{y} : \mathbf{y}^\top \mathbf{1} = 1, \mathbf{y} \geq \mathbf{0}\}$ and $\mathbf{D} \in \Re^{L\times(K+1)}$ defined such that its rows contain preselected diffusion coefficients per step, normalized to sum to one. Upon obtaining $\boldsymbol{\gamma}_u$, vector $\boldsymbol{\phi}_u$ can be computed as $\boldsymbol{\phi}_u^\top = \boldsymbol{\gamma}_u^\top \mathbf{D}$. For the definition of matrix $\mathbf{D}$ here we consider two well-known diffusions, namely the personalized PageRank [24] and the heat kernel [6] and we build $\mathbf{D}$ using ℓ_{PR} values for pagerank's *damping factor* $(\alpha_1, \ldots, \alpha_{\ell_{\text{PR}}})$ and ℓ_{HK} values for heat kernel's *temperature* $(t_1, \ldots, t_{\ell_{\text{HK}}})$, equally spaced in [0,1] and [1,10] respectively. Concretely, matrix $\mathbf{D}$ is defined as

$$\mathbf{D} \triangleq \begin{bmatrix} \mathbf{C}_{\text{PR}} \\ \mathbf{C}_{\text{HK}} \end{bmatrix} \tag{17}$$

where $\mathbf{C}_{\text{PR}} \in \Re^{\ell_{\text{PR}}\times(K+1)}$ is the row-normalized version of matrix $\mathbf{X}_{\text{PR}}$ defined by

$$[\mathbf{X}_{\text{PR}}]_{ij} \triangleq (1-\alpha_i)\alpha_i^{j-1}, \quad i = 1, \ldots, \ell_{\text{PR}} \tag{18}$$

and $\mathbf{C}_{\text{HK}} \in \Re^{\ell_{\text{HK}}\times(K+1)}$ is the row-normalized version of matrix $\mathbf{X}_{\text{HK}}$ defined such that

$$[\mathbf{X}_{\text{HK}}]_{ij} = e^{-t_i}\frac{t_i^{j-1}}{(j-1)!}, \quad i = 1, \ldots, \ell_{\text{HK}}. \tag{19}$$

While PerDif$^{\text{FREE}}$ learns $\boldsymbol{\phi}_u$ by weighing the contributions of the landing probabilities directly, PerDif$^{\text{PAR}}$ constrains $\boldsymbol{\phi}_u$ to comprise a user-specific mixture of predetermined such weights (i.e., the rows of $\mathbf{D}$), thus allowing one to endow $\boldsymbol{\phi}_u$ with desired properties, relevant to the specific recommendation task at hand. We also note that the use of matrix $\mathbf{D}$ can improve the robustness of the personalized diffusions in settings where the recommendation quality of the individual landing distributions comprising $\mathbf{S}_u$ is uneven across the K steps considered (we will report such case in the experimental section of this paper).

2.4 The Item Transition Matrix P

Our discussion so far, applies to any item transition probability matrix $\mathbf{P} \in \Re^{I\times I}$ arising by an underlying item model $\mathbf{W}$. In practice, the construction of $\mathbf{W}$ and $\mathbf{P}$ can be approached in several different ways depending on the available information and the underlying recommendation task at hand. Here, aiming to promote sparsity to $\mathbf{P}$, we construct $\mathbf{W}$ using a locally restricted variant of the well-known SLIM method [23] that is forced to consider only fixed-size neighborhoods when learning relations between the items. Specifically, for each item i we consider its C nearest neighbors (in terms of cosine similarity between their vector representations) and we form a matrix $\mathbf{N}_i \in \Re^{I\times C}$, by selecting the corresponding columns of $\mathbf{R}$. We then solve for each item the optimization problem

$$\begin{aligned} \underset{\mathbf{x}\in\Re^C}{\text{minimize}} \quad & \tfrac{1}{2}\|\mathbf{r}_i - \mathbf{N}_i\mathbf{x}\|_2^2 + \gamma_1\|\mathbf{x}\|_1 + \tfrac{1}{2}\gamma_2\|\mathbf{x}\|_2^2 \\ \text{subject to} \quad & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{20}$$

and we obtain the corresponding proximity scores for the i-th column of matrix $\mathbf{W}$. We then define $\mathbf{P}$ as

$$\mathbf{P} \triangleq \frac{1}{\|\mathbf{W}\|_\infty}\mathbf{W} + \text{Diag}(\mathbf{1} - \frac{1}{\|\mathbf{W}\|_\infty}\mathbf{W}\mathbf{1}) \tag{21}$$

in order to prevent items that are loosely related to the rest of the itemspace to disproportionately influence the item-to-item transitions, as well as to promote slow mixing properties to the random walk [21].

The overall procedure for constructing $\mathbf{P}$ and obtaining $\boldsymbol{\mu}_u$ for every user in the system is given in Algorithm 1.

Algorithm 1 PerDif

Input: Matrix $\mathbf{R}$, [item model $\mathbf{W}$]
Parameters: K, γ_1, γ_2, C, L, `mode` $\in$ {`Par`, `Free`}
Output: User-personalized parameters: $\{\boldsymbol{\mu}_u\}_{u\in\mathcal{U}}$
if $\mathbf{W}$ is not provided **then**
  Obtain $\mathbf{W}$ by solving (20)
end if
$\mathbf{P} \leftarrow \frac{1}{\|\mathbf{W}\|_\infty}\mathbf{W} + \text{Diag}(\mathbf{1} - \frac{1}{\|\mathbf{W}\|_\infty}\mathbf{W}\mathbf{1})$
parfor $u \in \mathcal{U}$ **do**
  Collect $\mathbf{S}_u\mathbf{E}_u$ on the fly, row-by-row by successive matvecs with $\mathbf{P}$ starting from $\boldsymbol{\omega}_u$
  if `mode` == `Par` **then**
    Build $\mathbf{D}$
    Obtain $\boldsymbol{\gamma}_u$ by solving (16)
    $\boldsymbol{\phi}_u^\top \leftarrow \boldsymbol{\gamma}_u^\top\mathbf{D}$
  else
    Obtain $\boldsymbol{\phi}_u$ by solving (15) (since Δ_{++}^{K+1} is open in practice we constrain $\boldsymbol{\phi}_u$ to be larger than or equal to $eps \times \mathbf{1}$)
  end if
  Obtain $\boldsymbol{\mu}_u$ from $\boldsymbol{\phi}_u$ by the forward recurrence (14)
end parfor

3 PRIOR ART

Over the past decade a vast number of algorithms have been proposed to tackle the top-n recommendation task. These include neighborhood-based methods [15, 22, 23]; latent-space methods [8, 13, 18, 20, 28]; graph-based methods [5, 7, 9, 10, 16, 21]; and more recently methods relying on deep neural networks [12, 19, 29]. PerDif brings together item-models with random walks, and thus lies at the intersection between neighborhood- and graph-based methods; the item transition component captures neighborhood information of the items which is then integrated in a random-walk-based framework to learn personalized diffusions for top-n recommendations.

Mathematically, the per-user item exploration processes produced by PerDIF are members of the family of multidamping processes introduced by Kollias *et al.* in [17]. Their paper explores meticulously how different damping weighing schemes can be chosen to approximate known functional rankings, such as linear rank and generalized hyperbolic rank [1]. Here, instead of fixing the damping weights to abide by target functional rankings, we provide a framework for learning user-specific and item-model-aware multidamping mechanisms. In this sense, PerDif undertakes statistical learning in the space of multidamping processes, tailored to the underlying top-n recommendation task. Related adaptive-diffusion models have recently been considered for other learning tasks over graphs, most notably for graph-signal filtering [27], semi-supervised node classification [2, 3], community detection [11], as well as protein function prediction in biological networks [14].

4 EXPERIMENTAL SETTING

4.1 Datasets

Our experimental evaluation is based on five real-world publicly available datasets. Namely the widely-used benchmark *movielens* dataset; the song recommendation dataset *yahoo*; the Amazon datasets *movies&tv* and *books*; as well as the *netflix* prize dataset. Basic characteristics of the datasets can be found in Table 1. For all datasets we consider only implicit feedback.

Table 1: Dataset characteristics.

Name	#users	#items	#interactions	density
movielens	6,040	3,706	1,000,029	0.0447
yahoo	7,307	3,312	404,745	0.0167
movies&tv	10,039	5,400	437,763	0.0081
books	43,550	24,811	1,777,072	0.0016
netflix	480,189	17,770	100,480,507	0.0118

4.2 Evaluation Methodology and Metrics

To evaluate the top-n recommendation performance, we adopted the widely used *leave-one-out* evaluation protocol [8, 12, 20, 25]. In particular, for each user we randomly select one liked[4] item and we create a test set $\mathcal{T}$. The rest of the dataset is used for training the models. For model selection we repeat the same procedure on the training data and we create a validation set $\mathcal{V}$; and for each method considered we explore the hyperparameter space to find the model that yields the best performance in recommending the items in $\mathcal{V}$, and then we evaluate its out-of-sample performance based on the held-out items in $\mathcal{T}$. For the evaluation we consider for each user her corresponding test item alongside 999 randomly selected unseen items and we rank the item lists based on the recommendation scores produced by each method.

The evaluation of the top-n recommendation performance is based on three widely used ranking-based metrics; namely the *hit ratio* (HR@n), the *average reciprocal hit rank* (ARHR@n), and the truncated *normalized discounted cumulative gain* (NDCG@n) over the set of users (due to space constraints we refer the reader to eg. [26] for a detailed definition). While HR@n gives a perfect score if the held-out item is ranked within the first n, ARHR@n and NDCG@n use a monotonically decreasing reward to emphasize the importance of the actual position of the held-out item in the top-n recommendation list.

5 EXPERIMENTAL RESULTS

5.1 Personalized Diffusions as a Framework

In the definition of the item transition probability matrix $\mathbf{P}$ we adopted a particular strategy for constructing matrix $\mathbf{W}$ designed to promote locality on the direct item-to-item transitions. Instead of this particular matrix $\mathbf{W}$ one could use any model that captures item-to-item relations. But does the proposed approach offer any benefit with respect to using the underlying item models directly? Can user-specific diffusions boost recommendation accuracy beyond the levels reached by using the same assumptions for all users?

In order to explore these questions here we empirically evaluate PerDif's performance on the movielens data using the proposed item model $\mathbf{W}$ (cf. (20)) as well as other two commonly used item models. Namely:

(1) COS: a cosine similarity model defined such that its ij−th element is given by $\mathbf{r}_i^{\mathsf{T}}\mathbf{r}_j/(\|\mathbf{r}_i\|\|\mathbf{r}_j\|)$;
(2) SLIM: which learns a sparse item model by solving an ℓ_1, ℓ_2 regularized optimization problem (see [23] for details).

We build the respective item models and we create the corresponding item transition matrices $\mathbf{P}$ as in (21). We then run both variations of our method for $K \in [1, \ldots, 10]$ and in Figure 1 we report recommendation accuracy per K in terms of NDCG@n as well as the accuracy achieved by applying the base item model directly.

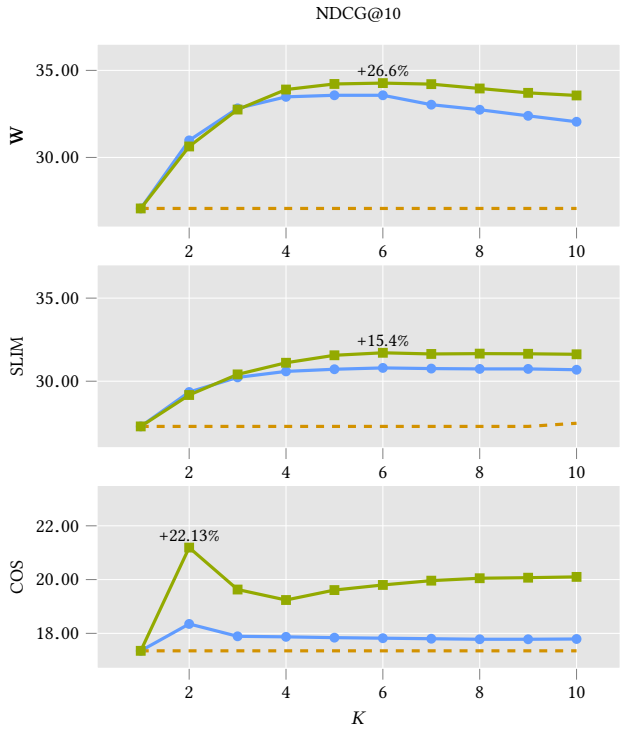

Figure 1: Recommendation accuracy of PerDif$^{\text{PAR}}$ (green line) and PerDif$^{\text{FREE}}$ (blue line) applied to W as well as two other item models. For reference we also report the performance achieved by using the base item-models directly (dashed orange line).

We see that PerDif increases the accuracy of all item models significantly with PerDif$^{\text{PAR}}$ performing relatively better than PerDif$^{\text{FREE}}$ for all models. Furthermore, note that the difference between the two variants of PerDif is considerably larger in case of COS. To gain more insight of this behavior we also report the NDCG@n yielded by using the individual k−step landing probability distributions each item model (for up to 10 steps), along with the spectra of the corresponding transition probability matrices

[4] When the original data contain ratings, the per-user target item is randomly sampled among the highest rated items of each particular user in order to ensure that it indeed denotes an item that the user liked. When no such information is available every user-item interaction is considered a 'like'.

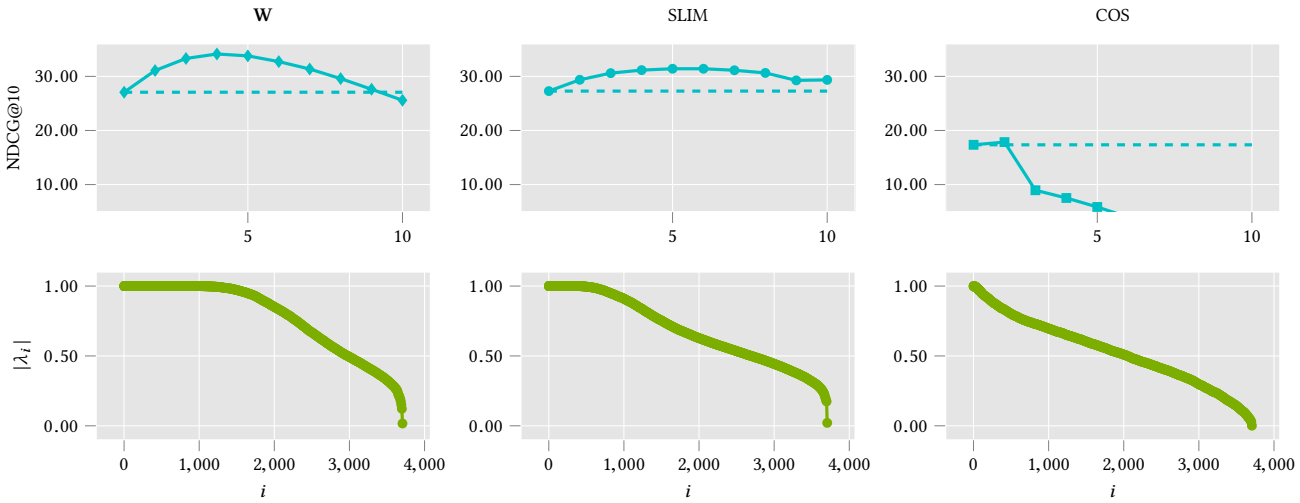

Figure 2: Recommendation accuracy per step (1st row) and plots of the spectrum of P (2nd row) for all item models.

(Figure 2). Observe that in case of COS after the first few steps of the walk there is a rapid decay in performance; with NDCG@n plummeting below the levels of the base item model. Examination of the spectra of the transition matrices reveals that the subdominant eigenvalues in case of COS cause the walks to mix quickly; thereby forcing the $k-$step distributions to rapidly 'forget' the user-specific starting distribution and succumb to the 'pull' of the popular items in the system. Therefore, freely learning personalized diffusions using these landing probabilities as features (as is the case for PerDif$^{\text{FREE}}$) together with the sparsity promoting properties of the simplex constraint, carries the risk of 'trapping' the diffusions in exploiting suboptimal landing distributions for recommendation. PerDif$^{\text{PAR}}$ on the other hand, by s the mixtures proves more robust to such problems.

On the contrary, for the other two models recommendation accuracy remains high throughout the range of $k-$step landing probabilities considered; this is directly reflected on the performance of PerDif$^{\text{FREE}}$ which follows closely that of PerDif$^{\text{PAR}}$. For both item models there appears to be a large number of eigenvalues clustered around the value 1, which delay the convergence of the successive landing probability distributions towards equilibrium, thereby increasing the number of features that are still useful for recommendation.

Table 2 reports the performance of PerDif with 2, 4, 6, 8 steps compared to the best fixed diffusion (cf (2)) or the best fixed step (cf (1)) per item model. Even though adopting the same preference propagation mechanism for all users can too provide significant benefits over using the respective item models directly, personalizing the diffusions to each user can improve recommendation accuracy even further, by 5.64% for our base model, 6.50% for SLIM, and 4.85% for COS.

5.2 Performance vs Competing Approaches

We evaluate the top-n recommendation accuracy of PerDif$^{\text{FREE}}$ and PerDif$^{\text{PAR}}$ against competing approaches.

Competing Baselines. We compare against six state-of-the-art baselines; namely (i) the well-known PureSVD method [8], which produces recommendations based on the truncated SVD of **R**; (ii) the item-based method SLIM [23] which learns a sparse item model by solving an ℓ_1, ℓ_2-regularized optimization problem; (iii) the EigenRec method [20], which builds a factored item model based on a scaled cosine similarity matrix; as well as the recently proposed deep learning methods (iv - v) Mult-VAE and Mult-DAE [19] which extend variational and denoising autoencoders to collaborative filtering using a multinomial likelihood; and (vi) NAIS [12], which generalizes factored item similarity models [15] employing an attention mechanism.

Table 2: Performance comparison between personalized and fixed preference propagation strategies.

Method	**W**	SLIM	COS
Base Model	27.07	27.28	17.35
FixedDif	32.47	29.70	20.19
FixedK	32.73	27.28	17.85
PerDif 2-steps	30.63	29.18	**21.17**
PerDif 4-steps	33.82	31.10	19.27
PerDif 6-steps	**34.30**	**31.53**	19.76
PerDif 8-steps	34.02	31.63	20.02

Hyperparameters: FixedDif: $p \in \{0.5, 0.55, \ldots, 0.9\}$; FixedK: $K \in \{1, \ldots, 10\}$.

Results. Table 3 reports top-n recommendation performance of the competing approaches. The performance was measured in terms of HR@n, ARHR@n and NDCG@n, focusing on $n = 10$. Model selection was performed for each dataset following the procedure detailed in Section 4.2 and considering for each method the hyperparameters reported on Table 3. We see that both variants of PerDif outperform all other methods considered on every metric and for all datasets[5].

[5]Netflix dataset is not included in Table 3 because due to the computational requirements of certain of the competing approaches we were unable to search the hyperparameter space for cross-validation and train them in a reasonable amount of time. Testing the performance of PerDif, against the efficient methods Mult-VAE

Table 3: Top-n recommendation quality of the competing approaches in terms of HR@10, ARHR@10 and NDCG@10.

	movielens			*yahoo*			*movies&tv*			*books*		
Method	HR	ARHR	NDCG	HR	ARHR	NDCG	HR	ARHR	NDCG	HR	ARHR	NDCG
PureSVD	44.14	19.33	25.36	38.68	18.30	22.62	22.58	09.88	12.86	46.84	25.84	30.81
SLIM	46.34	21.39	27.28	52.24	23.21	30.03	27.26	12.95	16.37	56.72	34.50	39.67
EigenRec	45.21	20.44	26.35	48.12	23.30	29.23	25.22	11.44	14.66	52.89	29.36	34.93
Mult-DAE	44.06	18.97	24.83	45.37	21.46	27.07	27.10	11.96	15.50	54.66	29.75	35.60
Mult-VAE	44.35	19.50	25.31	45.09	21.22	26.80	26.72	12.05	15.40	53.85	29.24	35.08
NAIS	46.36	20.65	26.68	50.53	23.64	29.91	24.35	10.87	13.99	51.18	28.36	33.65
PerDif$^{\text{PAR}}$	**53.13**	**28.56**	**34.30**	**55.29**	**29.01**	**35.21**	**28.17**	**13.04**	**16.58**	**57.30**	**34.73**	**40.10**
PerDif$^{\text{FREE}}$	52.52	27.74	33.57	54.78	28.71	34.87	27.96	12.98	16.49	57.20	34.66	40.02

Hyperparameters: PerDif: $C = \{2.5\%, 5\%, 7.5\%, 10\%, 20\%\}$ of $|\mathcal{I}|$, $\gamma_1 \in \{1, 3, 5, 10\}$, $\gamma_2 \in \{0.1, 0.5, 1, 3, 5, 7, 9, 11, 15\}$ and $K \in \{1, \ldots, 10\}$; τ_{neg} was set to 999; for building the diffusion dictionary, ℓ_{HK}, ℓ_{PR} were set to 15; PureSVD: $f \in \{10, 20, \ldots, 1000\}$. SLIM: $\lambda, \beta \in \{0.1, 0.5, 1, 3, 5, 10, 20\}$. EigenRec: $f \in \{10, 20, \ldots, 1000\}$, $d \in \{-2, -1.95, \ldots, 2\}$. Mult-DAE- Mult-VAE: we used the hyperparameter tuning approach provided by the authors in their publicly available implementation; we considered both architectures proposed in [19]; namely $[I - 200 - I]$ and $[I - 600 - 200 - 600 - I]$. NAIS: We used the implementation provided by [12], considering the parametric ranges discussed in the paper, namely α, $k \in \{8, 16, 32, 64\}$, $\beta = 0.5$ and regularization parameters $\lambda \in \{10^{-6}, \ldots, 1\}$.

5.3 Apriori Identification of Users that will Tend to Receive Poor Recommendations

Personalizing the diffusions within the proposed framework can also provide useful information arising from the analysis of the learned diffusion coefficients. Here we showcase a particular such example. Consider a user u for whom the learned diffusion coefficient ϕ_1 has a large value. From a modeling point of view, this translates to a somewhat 'uneventful' itemspace exploration path: regardless of where the user is after $K-1$ steps, his last coin with high probability takes him back exactly where he started. Due to the form of our optimization objective, such outcome would indicate that it was not possible to find suitable diffusion coefficients yielding high recommendation score for the target item i while keeping the scores of the negative samples low[6]. This renders ϕ_1 a potentially informative indicator of user/model mismatch at *training-time*. To test this idea we run PerDif on every dataset and we compare the average hit rate of the problematic users (i.e., users with $\phi_1 > \tau$) with the rest of the users in the system.

Table 4: Percentage drop in hit rate of the problematic users ($\phi_1 > \tau$) with respect to the rest of the users in the system.

	$\phi_1 > 0.10$		$\phi_1 > 0.15$	
Dataset	# users	drop	# users	drop
movielens	237	19.18%	117	20.29%
yahoo	530	19.24%	472	20.60%
movies&tv	1362	14.11%	1296	13.92%
books	1614	12.94%	1533	12.62%
netflix	1059	5.65%	985	5.63%

Mult-DAE, produces the same relative ordering in performance, verifying the same performance trends observed in the rest of the datasets.

[6]Notice that in such case, 'falling-back' to concentrating probability mass on ϕ_1 is aligned with the goal of minimizing our loss since by definition $\boldsymbol{\omega}_u$ has zero values in every position corresponding to negative samples.

The out-of-sample performance of the problematic users is indeed lower for every dataset considered, with the results being statistically significant for all reported cases. Note that this interesting side information of PerDif can prove particularly useful in practical settings; e.g., allow for preemptive interventions to handle such cases appropriately.

5.4 Runtimes

Table 5 reports the wall-clock timings for computing the item transition matrix $\mathbf{P}$ as well as learning personalized diffusions with PerDif$^{\text{FREE}}$ and PerDif$^{\text{PAR}}$ (average runtimes over 10 runs). PerDif$^{\text{FREE}}$ typically runs faster than PerDif$^{\text{PAR}}$. When the number of steps of the underlying random walk is small, fitting-time differences between the two variants are minor (as was the case for books and netflix); when more steps are needed such differences become more pronounced (e.g., for movielens and yahoo). For the larger datasets the computational bottleneck of the method is the construction of matrix $\mathbf{P}$. In practice updates of $\mathbf{P}$ will be typically performed periodically and offline. On the contrary, notice that the per-user fitting costs are of the order of milliseconds for all datasets. Taking into account the fact that given $\mathbf{P}$, PerDif can be performed separately for each user, the modest computational requirements for personalizing the diffusions can enable in real applications online fitting as well as 'on-the-fly' adaptation, to track the current user session in the system.

6 CONCLUSIONS AND FUTURE DIRECTIONS

In this work we introduce PerDif. Starting from an intuitive time-inhomogeneous random-walk-with-restarts over an item-to-item graph, PerDif learns user-specific teleportation probabilities across time, that translate to an interpretable exploration process of the underlying itemspace, particular to each user. Personalization can be performed in parallel and very efficiently, making PerDif applicable even in large scale settings. A comprehensive set of experiments on several real-world datasets against state-of-the-art competing

Table 5: Runtimes for building P as well as fitting the diffusions with both variants of our model.

		PerDif$^{\text{FREE}}$		PerDif$^{\text{PAR}}$	
Dataset	**P**	All	Per user	All	Per user
movielens	1.4s	0.6s	0.1ms	14s	2.4ms
yahoo	1.3s	0.4s	0.1ms	25s	3.4ms
movies&tv	2.0s	1.3s	0.1ms	15s	1.5ms
books	40s	55s	1.3ms	1.1m	1.5ms
netflix	87m	5m	0.9ms	6.1m	1.1ms

All experiments are ran on a single Intel Xeon Gold 6148 CPU @ 2.40GHz Machine with 20 cores and 64Gb DDR4 RAM. Column **P** reports the average runtimes over all base item models built during cross-validation.

baselines, showcase the potential of the proposed methodology in achieving high top-n recommendation accuracy.

A very interesting direction we are currently exploring, includes expanding our optimization objective to consider more than one target items per user, or averaging the per-user diffusions over multiple single-target runs, in order to further improve the robustness of the learned diffusions. Here, opting for simplicity and clarity of exposition, we adopted the single-target formulation, which was empirically shown to perform very well for all datasets. Another interesting path that remains to be explored involves extending our framework to tackle sequential as well as session-based recommendation settings.

A PROOF OF LEMMA 2.2

Proof. For $N = 1$ the identity holds trivially. Let us assume that it holds for n. We will show that it holds for $n+1$ as well. We have

$$\begin{aligned}
&\sum_{i=1}^{n+1}\left((1-\mu_i)\left(\prod_{\ell=i+1}^{n+1}\mu_\ell\right)\right)+\left(\prod_{\ell=1}^{n+1}\mu_\ell\right)= \\
&= 1-\mu_{n+1}+\sum_{i=1}^{n}\left((1-\mu_i)\prod_{\ell=i+1}^{n+1}\mu_\ell\right)+\prod_{\ell=1}^{n+1}\mu_\ell \\
&= 1-\mu_{n+1}+\mu_{n+1}\left(\sum_{i=1}^{n}(1-\mu_i)\prod_{\ell=i+1}^{n}\mu_\ell+\prod_{\ell=1}^{n}\mu_\ell\right) \\
&= 1-\mu_{n+1}+\mu_{n+1}=1
\end{aligned}$$

Therefore, the induction step is established and the proof is complete. □

REFERENCES

[1] Ricardo Baeza-Yates, Paolo Boldi, and Carlos Castillo. 2006. Generic damping functions for propagating importance in link-based ranking. *Internet Mathematics* 3, 4 (2006), 445–478.

[2] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis. 2018. AdaDIF: Adaptive Diffusions for Efficient Semi-supervised Learning over Graphs. *Proc. of IEEE Int. Conf. on Big Data*, 92–99.

[3] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis. 2019. Adaptive Diffusions for Scalable Learning Over Graphs. *IEEE Transactions on Signal Processing* 67, 5 (March 2019), 1307–1321. https://doi.org/10.1109/TSP.2018.2889984

[4] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[5] Fabian Christoffel, Bibek Paudel, Chris Newell, and Abraham Bernstein. 2015. Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 163–170.

[6] Fan Chung. 2007. The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences* 104, 50 (2007), 19735–19740.

[7] Colin Cooper, Sang Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. 2014. Random Walks in Recommender Systems: Exact Computation and Simulations. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion)*. ACM, New York, NY, USA, 811–816. https://doi.org/10.1145/2567948.2579244

[8] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems (RecSys '10)*. ACM, 39–46.

[9] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1775–1784. https://doi.org/10.1145/3178876.3186183

[10] Ashish Goel, Pankaj Gupta, John Sirois, Dong Wang, Aneesh Sharma, and Siva Gurumurthy. 2015. The Who-To-Follow System at Twitter: Strategy, Algorithms, and Revenue Impact. *Interfaces* 45, 1 (Feb. 2015), 98–107. https://doi.org/10.1287/inte.2014.0784

[11] K. He, P. Shi, J. E. Hopcroft, and D. Bindel. 2016. Local Spectral Diffusion for Robust Community Detection. *Proc. of the SIGKDD workshop*.

[12] X. He, Z. He, J. Song, Z. Liu, Y. Jiang, and T. Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (Dec 2018), 2354–2366. https://doi.org/10.1109/TKDE.2018.2831682

[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

[14] B. Jiang, K. Kloster, D. F. Gleich, and M. Gribskov. 2017. AptRank: an adaptive PageRank model for protein function prediction on bi-relational graphs. *Bioinformatics* 33, 12 (2017), 1829–1836.

[15] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored Item Similarity Models for top-N Recommender Systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. ACM, New York, NY, USA, 659–667. https://doi.org/10.1145/2487575.2487589

[16] Zhao Kang, Chong Peng, Ming Yang, and Qiang Cheng. 2016. Top-n recommendation on graphs. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2101–2106.

[17] G. Kollias, E. Gallopoulos, and A. Grama. 2014. Surfing the Network for Ranking by Multidamping. *IEEE Transactions on Knowledge and Data Engineering* 26, 9 (Sep. 2014), 2323–2336. https://doi.org/10.1109/TKDE.2013.15

[18] Yehuda Koren. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. on Knowledge Discovery from Data (TKDD)* 4, 1 (2010), 1.

[19] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 689–698.

[20] Athanasios N. Nikolakopoulos, Vassilis Kalantzis, Efstratios Gallopoulos, and John D. Garofalakis. 2018. EigenRec: generalizing PureSVD for effective and efficient top-N recommendations. *Knowledge and Information Systems* (03 May 2018). https://doi.org/10.1007/s10115-018-1197-7

[21] Athanasios N. Nikolakopoulos and George Karypis. 2019. RecWalk: Nearly Uncoupled Random Walks for Top-N Recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19)*. ACM, New York, NY, USA, 9. https://doi.org/10.1145/3289600.3291016

[22] Xia Ning, Christian Desrosiers, and George Karypis. 2015. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*. Springer, 37–76.

[23] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 11th IEEE International Conference on Data Mining*. IEEE, 497–506.

[24] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report 1999-66. Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[26] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. Recommender systems: introduction and challenges. In *Recommender systems handbook*. Springer, 1–34.

[27] A. Sandryhaila and J. M. F. Moura. 2013. Discrete Signal Processing on Graphs. *IEEE Transactions on Signal Processing* 61, 7 (April 2013), 1644–1656.

[28] Jason Weston, Ron J Weiss, and Hector Yee. 2013. Nonlinear latent factorization by embedding multiple user interests. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 65–68.

[29] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 153–162.