Forgetting on Evolving Graphs for Accurate and Diverse Stream-Based Recommendation

Murilo F. L. Schmitt, Eduardo J. Spinosa

Universidade Federal do Paraná (UFPR), Brazil
{mflschmitt, spinosa}@inf.ufpr.br

Abstract. Stream-based recommender systems are an active research field, relying on incremental algorithms to update models by incorporating new data on a single pass, discarding such data after processing. A limitation of solely including new data is the accumulation of obsolete concepts, which eventually raises accuracy and scalability concerns. In this work, we propose a gradual forgetting technique for incremental neighborhood-based methods that locally forgets items based on recency and popularity, by decreasing importance of neighborhood of items for every incoming observation to emphasize more recent and reinforced ones. The technique includes parameters to increase diversity, by retaining less popular yet relevant items, and scalability, by pruning obsolete connections not reinforced by new data. Experiments conducted by extending a recent incremental graph-based approach highlight the effectiveness of the proposed technique, as its application improved scalability and diversity, outperforming baselines.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning

Keywords: recommender systems, data streams, forgetting, online learning

1. INTRODUCTION

Recommender systems present personalized sets of items to users based on their interests. When users interact with online systems, they provide feedback to the system, e.g., with click-through data and shopping behavior, which can be collected from all users and indirectly provides information related to user behavior, i.e., implicit feedback. Based on feedback provided by many users, models can be built to predict unknown user-item preferences through past behavior and recommend relevant items based on interests of similar users. Such approach is termed collaborative filtering (CF) and significant progress has been achieved through neighborhood-based methods [Ning et al. 2015] and matrix factorization [Koren 2009]. Despite their effectiveness, CF approaches mostly rely on batch processing. In real world systems, data is generated continuously at unpredictable rate, becoming unfeasible to retrain models as data is generated since it gets increasingly more expensive, which leads to poor recommendation performance as new information is disregarded between updates.

An alternative is to address recommendation as a data stream problem and design Stream-Based Recommender Systems (SBRS), where user feedback incomes continuously in real time at unpredictable rate and order, with new users and items, and an always-available model must process observations as fast as they arrive [Vinagre et al. 2019]. Incremental algorithms are capable of learning new concepts and updating models with incoming feedback from the data stream, discarding it after processing. Hence, SBRS rely on these algorithms, which are viable for dynamic scenarios, as they have the ability to evolve over time in scalable manner [Al-Ghossein et al. 2021].

A limitation of incremental SBRS in general is that they learn solely by incorporating new information. When learning from data streams, previously known concepts change over time and become

This work was partially funded by CAPES.

obsolete, i.e., concept drift [Gama et al. 2014]. Continuous incorporation of information leads to ever-growing models, which eventually causes the accumulation of obsolete information, raising two challenges [Vinagre and Jorge 2012; Matuszyk et al. 2018]: (1) it introduces noise in the learning procedure, negatively affecting recommendations, degrading predictive power; (2) Maintaining such noise in ever-growing models leads to scalability issues, as time and memory requirements increases.

In that sense, application of forgetting mechanisms to remove obsolete information from these models can lead to improvements in accuracy and scalability [Vinagre and Jorge 2012; Matuszyk et al. 2018]. However, the challenge of removing obsolete information is not trivial. The definition of obsolete is a problem in itself, as it differs from old [Matuszyk et al. 2018], e.g., a stable relation between two old movies is not necessarily obsolete, while new information from a user that is temporarily sharing her account may be. Thus, forgetting mechanisms must select and remove obsolete information during model update while avoiding more complexity in the process. Another consideration is sparsity, an issue related to CF, i.e., users interact with only a small number of items. Therefore, little information is available from each user. Removing information from learning should avoid aggravate this issue.

In this paper, we propose a forgetting technique that selects elements to be forgotten based on recency and popularity, where the importance of neighborhood of items decreases for every incoming observation to emphasize more recent ones. We introduce a diversity parameter to allow increases in likelihood of retaining less popular items, and a weight threshold to prune obsolete data and improve scalability. To evaluate our technique, we built on IGSI$_{\hat{\pi}^t}$ [Schmitt and Spinosa 2022], a recent graph-based SBRS that outperformed related models in accuracy with competitive scalability. Our results show that the proposed forgetting technique improves scalability, as the removal of obsolete data reduces time and memory requirements, diversity, by including less popular items in the recommendation lists, and may also improve accuracy, since obsolete data will no longer affect learning.

2. BACKGROUND

The top-N recommendation task consists of recommending personalized subsets of items to users based on their preferences [Cremonesi et al. 2010]. Let $U = \{u_1, u_2, ..., u_m\}$ denote the increasing set of users and $I = \{i_1, i_2, ..., i_n\}$ the increasing set of items. Denote $\langle u, i, t\rangle$ as an incoming observation in a continuous user feedback data stream, indicating that user u interacted with item i at time t. A SBRS model must update itself with each observation at least as fast as their arrival with a single pass and provide recommendations with up to date information in real time [Vinagre et al. 2019].

In [Schmitt and Spinosa 2020; 2022], we proposed IGSI$_{\hat{\pi}^t}$, a SBRS that incorporates interactions in an item-graph, where nodes represent items and directed edges represent sequential user interactions. Denote $G = (V, E, w)$ as a weighted directed graph, where $V = \{v_1, v_2, ..., v_n\} \subseteq I$ denotes the set of nodes and $E \subseteq V \times V$ the set of edges. Each edge e has an associated weight $w(e) \in R_+$. Denote A as the adjacency matrix of G, where $a_{ij} = w((i, j))$ if $(i, j) \in E$ and 0 otherwise, and D as the diagonal degree matrix of G, where $d_{ii} = \sum_{(i,k)\in E} a_{ik}$. For each user $u \in U$, a list of items interacted by u sorted according to time t is defined as $S_u = \{(v_1, t_1), ..., (v_n, t_n)\}$. We denote the last element of S_u as (li_u, lt_u), where li_u is the most recently interacted item by u and lt_u is the time of such interaction.

For each incoming $\langle u, i, t\rangle$, if u is an unknown user, a representation for u is created, i.e., $S_u = \emptyset$, (i, t) is included in S_u and $li_u \leftarrow i$. If u is a known user, feedback is included in the graph by an edge (li_u, i), and both S_u and li_u are updated. The relevance of edges is distinguished by their weights, and the relevance of each sequential interaction is reinforced based on the frequency with which those interactions are made. If i is unknown, A and V are updated to include node i, edge (li_u, i) is added to E and $w((li_u, i)) = 1$. If $i \in V$, the weight of edge (li_u, i) is updated by $w((li_u, i))_t = w((li_u, i))_{t-1} + 1$.

To generate recommendations to a user u, IGSI$_{\hat{\pi}^t}$ finds the most relevant items in relation to the last r items in the ordered list of interactions S_u, where r is a parameter. This filtering adjusts recommendations to short-term interests of u, improving accuracy. Candidate items are ranked through simulations of random walk with restart (RWR). A RWR infers which nodes are most relevant to a given node s by measuring the frequency with which a random walker visits nearby nodes. Starting from $s \in V$, with probability γ the walker moves from a node to one of its neighbors at random, where the probability of transitioning from a node i to j is $p_{ij} = a_{ij}/d_{ii}$, or returns to s with $(1-\gamma)$ probability. The stationary distribution π_s^t of the RWR starting at s provides a ranking of items based on s, where nodes close to s have more relevance as they will be visited more frequently.

Since it is impractical to continuously compute the stationary distribution in an ever changing graph, IGSI$_{\hat{\pi}^t}$ approximates π_s^t through random walk sampling, by running M independent t-step random walks starting from s, where for each node $v \in V$, $\hat{\pi}_{s,v}^t$ is defined by the number of visits to v by the M random walks multiplied by $\frac{(1-\gamma)}{M}$. Denote rS_u as the last r items in S_u. Recommendations to u are generated by computing and averaging $\hat{\pi}_s^t$ for each $s \in rS_u$. In [Schmitt and Spinosa 2022] we observed that short random walks ($t = 3$) are sufficient in providing accurate recommendations and that the number of samples M allows a trade-off between accuracy and scalability.

3. RELATED WORK

This section discusses the application of forgetting techniques to SBRS. A recent comprehensive review related to SBRS methods can be found in [Al-Ghossein et al. 2021].

Incremental neighborhood-based methods store similarities between users or items that are updated based on each received observation, where neighborhoods and ranking of items are computed before each recommendation. Forgetting strategies have been devised to disregard and remove obsolete information and reduce the size of models used for neighborhood computation in order to improve recommendation times. Such forgetting can be either *abrupt* or *gradual*. *Abrupt* forgetting relies on sliding windows, defined by a number of interactions or time intervals, where only observations included in the windows are considered, and observations outside of it are forgotten. Sliding windows have been applied to user and item-based methods [Vinagre and Jorge 2012]. Performance depends largely on the size of the window. Despite its simplicity and straightforward application, a limitation is that it does not distinguish old from obsolete and simply forgets as defined by the window.

Alternatively, *gradual* forgetting relies on mechanisms to weight observations based on recency, by considering recent to be more important than older ones. Techniques to gradually decrease importance of observations over time and assign higher weights to more recent ones have been explored in neighborhood-based methods [Ding and Li 2005; Tabassum et al. 2020]. In general, recommendations based on recent information provide improved accuracy. [Vinagre and Jorge 2012] used a decay function to decrease similarities over time unless reinforced by new data, by multiplying the entire similarity matrices with a positive fading factor before each update. When these reach a low value, they are assumed to be zero, reducing model size and improving scalability. A limitation is that its global application presents reduced effectiveness in the presence of subtle local changes.

Matrix factorization (MF) methods maps users and items in a common latent feature space of low dimensionality, such that their affinity is given by the inner product of their embedded vectors. These methods obtain significant improvements in predictive capability by modeling temporal information [Koren 2009]. [Matuszyk et al. 2018] proposed several strategies to select and remove obsolete information from MF models, divided into two categories. *Rating-based* forgetting select and discard ratings from lists through sliding windows and sensitivity analysis. *Latent factor-based* forgetting adjust latent factors of users to reduce the impact of past observations, based on volatility, frequency and popularity. Experiments suggest that latent factor-based forgetting are successful in predictive power and computation time. Finally, forgetting based on a first in, first out (FIFO) queue that adjust user vectors was explored in [Vinagre et al. 2015], where a global queue of all items seen in the stream ordered based on recency of item occurrences, is deployed and used to select older items as negative feedback to the active user and give greater importance to more recent ones, improving accuracy.

4. PROPOSED APPROACH

The incremental nature of IGSI$_{\hat{\pi}^t}$ results in an ever-growing model that learns solely by incorporating new information. Although incremental models incorporate feedback in real time, users' preferences evolve over time and are subject to concept drift [Gama et al. 2014]. Hence, information retained in these models may become obsolete. The accumulation of obsolete information results in increasing time and memory requirements, raising scalability concerns. These concerns are related to all neighborhood-based methods, as ranking of items is performed at the time of recommendation based on similarities that are updated after every observation. The inclusion of obsolete information in this process also negatively impacts the predictive power of algorithms [Matuszyk et al. 2018]. The lack of mechanisms to remove obsolete information means that IGSI$_{\hat{\pi}^t}$ is susceptible to the aforementioned concerns.

We aim for a model capable of continuously incorporating user feedback, but also capable of removing outdated information while controlling its growth. To that end, we propose a gradual forgetting technique that selects information to be forgotten based on recency and popularity. IGSI$_{\hat{\pi}^t}$ is updated at every observation $\langle u, i, t\rangle$, where an edge connecting the last interaction by u, li_u, to the current item i is updated by increasing the weight of edge (li_u, i). In order to decrease the importance of older information and emphasize the most recent interaction, before the incremental update based on the current observation, we select the neighborhood of li_u to be forgotten. In other words, the relevance of successors to li_u is reduced before increasing the weight of the new connection. By forgetting information locally, we avoid aggravation of sparsity issues, since old and stable concepts that are not reinforced are retained and only forgotten in the presence of new concepts.

An important consideration is that the majority of interactions are related to the most popular items, which represents a small portion of the item catalog, while the set of less popular items that represent a large portion, i.e., the *long-tail* of the distribution [Cremonesi et al. 2010], accounts for a small fraction. The inclusion of items from the long-tail is important as it improves diversity of recommendation lists. However, the degrees distribution of the item-graph naturally follows the aforementioned distribution, and the recommendations generated by IGSI$_{\hat{\pi}^t}$ are affected by popularity.

Given that both items from the long-tail and new items have lower node degree, we propose a forgetting function where the fading factor of each item is proportional to its popularity. We introduce a diversity parameter that allows slower forgetting for less popular items. We measure the popularity of nodes through its degrees, and assume that nodes with low indegree represent both items from the long-tail and new items, i.e., candidate items for slower forgetting. To also allow exploration in random walks, we penalize sink nodes by considering the outdegree of nodes in the forgetting function.

Denote by p a predecessor of an item i, and $\alpha \in (0, 1)$ a fading factor. To decrease the relevance of i, edges connecting nodes to i can be updated by $w((p,i))_t = \alpha \cdot w((p,i))_{t-1}$. To decrease the impact of forgetting for less popular items, we introduce a convex combination factor β, that adjusts α for each item based on their popularity, measured through the degrees of its node, normalized according to the degree distribution of the neighborhood of the predecessor item, as defined by Eq. (1):

$$w((p,i))_t = \alpha^{\beta X + (1-\beta)Y} \cdot w((p,i))_{t-1} \tag{1}$$

$$X = \begin{cases} 1 & \text{if } \Delta^-(N^+(p)) = \delta^-(N^+(p)) \\ \frac{deg^-(i) - \delta^-(N^+(p))}{\Delta^-(N^+(p)) - \delta^-(N^+(p))} & \text{otherwise} \end{cases} \tag{2}$$

$$Y = \begin{cases} 1 & \text{if } \Delta^+(N^+(p)) = \delta^+(N^+(p)) \\ \frac{\Delta^+(N^+(p)) - deg^+(i)}{\Delta^+(N^+(p)) - \delta^+(N^+(p))} & \text{otherwise} \end{cases} \tag{3}$$

where $deg^-(i)$ and $deg^+(i)$ are the indegree and outdegree of node i, $N^+(p)$ is the set of successors of node p, $\Delta^-(N^+(p))$ and $\delta^-(N^+(p))$ are the maximum and minimum indegree among successors of

Algorithm 1: Online local neighborhood decay forgetting

```
// Require:
D = {(< u, i, t >)_1, ...}: data stream, r: recency parameter, t: length of walks, M: number of walks, α: decay parameter, β: diversity
  parameter, x: parameter for edge removal, (V, E, w): weighted digraph, U: set of users, I: set of items, S_u: list of interactions by u;
for $< u, i, t > \in D$ do
    top_n_items = recommendItems($S_u$, $r$, $N$, $V$, $E$, $t$, $M$) [Schmitt and Spinosa 2022];
    evaluate(top_n_items);
    for $s \in successors(li_u)$ do
        $w((li_u, s))_t \leftarrow \alpha^{\beta X + (1-\beta) Y} \cdot w((li_u, s))_{t-1}$;
        if $w((li_u, s)) < \alpha^x$ then
            $E \leftarrow E \setminus (li_u, s)$
    $U$, $I$, $S_u$, $li_u$, $lt_u$, $(V, E, w)$ = incrementalGraphUpdate() [Schmitt and Spinosa 2022];
```

p, and $\Delta^+(N^+(p))$ and $\delta^+(N^+(p))$ are the maximum and minimum outdegree among successors of p, respectively. In essence, Eq. (1) introduces a balance between diversity and exploration that can be controlled through β, where slower forgetting is enforced on items with lower indegree and higher outdegree, i.e., items from the long-tail that increase the exploration possibilities of random walks.

Since Eq. (1) continuously reduces information over time unless reinforced by newly generated data, hence ensuring that recent information is retained in the neighborhood of nodes, we introduce a weight threshold $\phi = \alpha^x$ that removes obsolete edges to account for the increasing scalability concerns related to ever-growing models, where x is a parameter and can be seen as the number of forgetting interactions in the neighborhood without reinforcement. This way, the growth of the graph can be controlled parametrically, while also increasing the likelihood of retaining nodes from the long-tail.

Thus, our forgetting technique requires three parameters: fading factor α, diversity parameter β and x that controls weight threshold ϕ. We refer to this technique as *local neighborhood decay.* The online procedure of IGSI$_{\hat{\pi}^t}$ with forgetting is presented in Algorithm 1. When a new observation $\langle u, i, t \rangle$ arrives from the data stream, we first generate a recommendation to u for evaluation purposes. Then, we apply forgetting over the neighborhood of the last item interacted by u (li_u) and remove obsolete edges if necessary. Finally, the graph is updated based on $\langle u, i, t \rangle$.

5. EXPERIMENTS

Datasets. Four datasets from movie and music domains were used, as summarized in Table I. ML-1M and ML-10M are binary versions of the MovieLens datasets. To simulate positive implicit feedback, we discarded ratings below 5. PLC-STR consists of timestamped log of music listening events from social network Palco Principal. LFM-1K consists on the first 25% of events from the Last.fm dataset.

Evaluation protocol. We use a prequential evaluation adopted by [Vinagre et al. 2019]. For each incoming event $\langle u, i, t \rangle$ in the stream, the model is first tested then updated based on four steps: (1) If u is a known user, use the current model to recommend N items to u, otherwise go to step 3; (2) Score the recommendation list given the observed item i; (3) Update the model with $\langle u, i, t \rangle$; (4) Proceed to the next observation. We measure accuracy through HitRate@N (HR@N) and DCG@N, as defined in [Frigó et al. 2017], with N = 20. We measure scalability through average update and recommendation times per sample, and average number of edges in the graph. Finally, we evaluate diversity through Intra-List Diversity (ILD) [Smyth and McClave 2001], defined as the average pairwise cosine distance of items in the recommendation list. We split datasets following the evaluation procedure proposed by [Matuszyk et al. 2018]: after sorting datasets chronologically, the first 20% are used to built models, the next 30% to optimize parameters and the remaining 50% for prequential evaluation.

Table I: Dataset description.

Dataset	Events	Users	Items	Sparsity	Source
ML-1M	226,310	6,014	3,232	98.84%	https://grouplens.org/datasets/movielens/1m/
ML-10M	1,544,812	67,312	8,721	99.74%	https://grouplens.org/datasets/movielens/10m/
PLC-STR	1,466,893	25,463	40,213	99.92%	https://rdm.inesctec.pt/dataset/cs-2017-003
LFM-1K	4,234,033	546	399,171	99.46%	http://ocelma.net/MusicRecommendationDataset/

Table II: Impact of parameter β on IGSI$_{\hat{\pi}t}$ with local neighborhood decay forgetting.

	ML-1M			ML-10M			LFM-1K			PLC-STR		
β	**HR**	**DCG**	**ILD**	**HR**	**DCG**	**ILD**	**HR**	**DCG**	**ILD**	**HR**	**DCG**	**ILD**
-	0.234	0.102	0.811	0.198	0.086	0.829	0.182	0.141	0.841	0.590	0.431	0.818
0.0	0.230	0.099	0.793	0.195	0.075	0.808	0.182	0.140	0.841	0.593	0.433	0.818
0.1	0.229	0.099	0.797	0.200	0.080	0.809	0.182	0.140	0.841	0.593	0.433	0.818
0.2	0.231	0.100	0.800	0.206	0.086	0.813	0.182	0.140	0.841	0.593	0.433	0.818
0.3	0.232	0.101	0.804	0.210	0.091	0.818	0.182	0.140	0.841	0.592	0.433	0.818
0.4	0.234	0.101	0.807	0.214	0.094	0.824	0.182	0.140	0.841	0.593	0.433	0.818
0.5	0.234	0.102	0.811	0.218	0.097	0.831	0.182	0.141	0.841	0.593	0.433	0.819
0.6	0.236	0.103	0.814	0.219	0.097	0.837	0.182	0.141	0.842	0.592	0.432	0.819
0.7	0.236	0.103	0.818	0.217	0.096	0.844	0.182	0.141	0.842	0.592	0.432	0.819
0.8	0.235	0.103	0.822	0.216	0.096	0.851	0.182	0.141	0.842	0.592	0.432	0.820
0.9	0.233	0.102	0.825	0.210	0.093	0.858	0.182	0.141	0.842	0.591	0.431	0.820
1.0	0.233	0.101	0.828	0.203	0.089	0.866	0.182	0.140	0.842	0.591	0.431	0.821

Baselines. We use IGSI$_{\hat{\pi}t}$ without forgetting, and also extend it with both our proposed forgetting technique and the following ones: *Sliding window* - A traditional abrupt technique. We define windows based on time intervals of size τ, where observations outside of the windows are forgotten; *Time-based decay* - A technique [Tabassum et al. 2020] that sporadically reduces relevance of edges based on time. We define a fixed period τ, and each time it elapses, edges are gradually forgotten by a factor α and removed if $w(e) < 1.0$; *Recency queue* - A global FIFO queue is maintained with all items seen in the stream, ordered based on recency. For each $\langle u, i, t\rangle$, j at the head is selected to lose relevance by a factor α. Edges are removed based on threshold ϕ, and j is reinserted at the tail [Vinagre et al. 2015].

Parameters. We use the optimal parameters for IGSI$_{\hat{\pi}t}$ reported in [Schmitt and Spinosa 2022]. Considering forgetting, we tested values for $\alpha \in [0.8, 0.99]$ in steps of 0.01, where higher values presented the best results, except for time-based decay, for which we set $\alpha = 0.9$. We set $\alpha = 0.99$ for the remaining ones. Baselines were optimized with grid search and values are reported in the results.

Impact of diversity parameter β. We tested values for $\beta \in [0, 1]$ in steps of 0.1. Results of these experiments are presented in Table II. As shown in Table II, diversity increases with parameter β. Accuracy tends to increase with β until a certain threshold, after which it starts to decrease. An interesting observation is that the impact of β is highly dependent on the dataset and directly related to the degree distribution on the graph, as seen when comparing results on ML-1M and ML-10M to LFM-1K and PLC-STR, where there are greater changes on datasets with higher average neighborhood size. Thus, β can be adjusted according to the application. In subsequent experiments, we select values for β that produce the highest DCG@20, followed by highest ILD.

Impact of weight threshold ϕ. We next evaluate the impact of ϕ. Results are presented in Table III. From Table III, ϕ can considerably reduce the number of edges, thus reducing memory requirements, while also obtaining major improvements in accuracy for ML-1M and ML-10M, and slight ones for PLC-STR. For LFM-1K, it reduces the size of the model without decreases in accuracy. It is beneficial to set a threshold, as accuracy starts to decrease after a certain value of ϕ, suggesting an accumulation of obsolete edges, which directly impacts accuracy and scalability. We select values for x that produce the highest DCG@20, followed by highest decrease in average number of edges.

Overall results. Results are shown in Table IV. For all datasets, the proposed forgetting technique, local neighborhood decay (LND), obtained the best results in accuracy. The application of the remain-

Table III: Impact of parameter x on IGSI$_{\hat{\pi}t}$ with local neighborhood decay forgetting.

	ML-1M			ML-10M			LFM-1K			PLC-STR		
x	**HR**	**DCG**	**#Edges**	**HR**	**DCG**	**#Edges**	**HR**	**DCG**	**#Edges**	**HR**	**DCG**	**#Edges**
-	0.234	0.102	61,939	0.198	0.086	224,781	0.182	0.141	1,019,439	0.590	0.431	185,060
0	0.228	0.098	9,928	0.190	0.084	36,948	0.155	0.125	421,626	0.448	0.341	57,811
5	0.241	0.106	33,103	0.217	0.097	75,305	0.182	0.141	881,605	0.589	0.429	116,502
10	0.243	0.107	44,010	0.223	0.100	96,784	0.182	0.141	955,763	0.591	0.431	134,709
15	0.241	0.107	49,671	0.225	0.101	112,532	0.182	0.141	983,056	0.593	0.432	145,352
20	0.240	0.105	52,922	0.226	0.101	124,921	0.182	0.141	996,639	0.593	0.432	152,509
25	0.240	0.105	54,987	0.225	0.100	134,839	0.182	0.141	1,004,021	0.593	0.432	157,571
30	0.239	0.105	56,384	0.224	0.100	143,123	0.182	0.141	1,008,564	0.593	0.432	161,409
35	0.237	0.103	57,403	0.224	0.099	150,101	0.182	0.141	1,011,426	0.593	0.433	164,415
40	0.237	0.104	58,245	0.224	0.099	156,059	0.182	0.141	1,013,395	0.592	0.433	166,898
45	0.237	0.104	58,916	0.222	0.099	161,240	0.182	0.141	1,014,750	0.592	0.432	168,921
50	0.237	0.103	59,445	0.222	0.099	165,814	0.182	0.141	1,015,746	0.593	0.433	170,607

Table IV: Overall results for all techniques. Best results are highlighted in bold.

Dataset	Technique	Param.	HR	DCG	ILD	#Edges	Time(ms)		
							Upd.	Rec.	Tot.
ML-1M	$\text{IGSI}_{\hat{\pi}^t}$	-	0.246	0.109	0.779	109,901	**0.2**	20.9	21.1
	local_n_decay	$\beta = 0.6, \phi = \alpha^{10}$	**0.255**	**0.114**	0.795	65,975	0.3	20.2	20.5
	recency_queue	$\phi = \alpha^5$	0.239	0.106	0.765	68,373	2.0	20.5	22.5
	time_based	$\tau = 30$	0.246	0.108	0.794	50,766	0.4	**19.3**	19.7
	sliding_window	$\tau = 90$	0.214	0.094	**0.816**	46,086	0.3	**19.3**	**19.6**
ML-10M	$\text{IGSI}_{\hat{\pi}^t}$	-	0.200	0.091	0.830	476,448	**0.5**	24.8	25.3
	local_n_decay	$\beta = 0.6, \phi = \alpha^{20}$	**0.221**	**0.103**	**0.867**	238,129	0.5	21.9	22.4
	recency_queue	$\phi = \alpha^5$	0.200	0.090	0.816	217,519	7.5	23.3	30.8
	time_based	$\tau = 14$	0.205	0.096	0.865	59,152	0.6	**19.2**	**19.8**
	sliding_window	$\tau = 540$	0.220	0.102	0.857	129,411	0.9	21.7	22.6
LFM-1K	$\text{IGSI}_{\hat{\pi}^t}$	-	**0.233**	0.180	0.834	1,990,079	**0.1**	20.8	20.9
	local_n_decay	$\beta = 0.9, \phi = \alpha^5$	**0.233**	**0.181**	0.837	1,579,646	0.3	19.6	19.9
	recency_queue	$\phi = \alpha^0$	0.210	0.166	0.822	1,010,565	55.7	20.3	76.9
	time_based	$\tau = 60$	0.223	0.175	0.832	1,068,100	0.2	**19.2**	**19.4**
	sliding_window	$\tau = 180$	0.203	0.157	**0.845**	1,162,517	0.2	20.0	20.2
PLC-STR	$\text{IGSI}_{\hat{\pi}^t}$	-	0.600	0.431	0.784	355,926	**0.01**	19.5	19.5
	local_n_decay	$\beta = 0.2, \phi = \alpha^{20}$	**0.603**	**0.434**	0.788	277,396	0.03	18.7	**18.7**
	recency_queue	$\phi = \alpha^{10}$	0.596	0.429	0.782	258,691	2.9	19.8	22.7
	time_based	$\tau = 90$	0.597	0.429	0.782	207,944	0.2	18.9	19.1
	sliding_window	$\tau = 180$	0.573	0.417	**0.798**	118,494	0.2	**18.5**	**18.7**

ing techniques generally decreased accuracy when compared to $\text{IGSI}_{\hat{\pi}^t}$ without forgetting, except for ML-10M. For diversity, LND improves performance compared to no forgetting, and was best or second best for all datasets. While the sliding window obtained the best diversity for two datasets, it was also less accurate, as these are conflicting objectives. For LND, parameter β allows for an increase in diversity. Considering update time, $\text{IGSI}_{\hat{\pi}^t}$ without forgetting presents better results, since the update is made solely on a single edge in the graph. Update time significantly increases for the recency queue technique, as the queue grows linearly to the number of items. The increase is less significant for the remaining techniques. Although update time is increased for all forgetting techniques, a consequence of their application is the removal of edges in the graph, reducing time and memory requirements and improving recommendation time. Thus, the trade-off between accuracy and scalability can be controlled parametrically. Comparing the average processing time per sample, i.e., both update and recommendation times, our proposed technique is outperformed only by the time-based technique.

For further evaluation, we plot moving averages for HR@20, number of edges and processing time per sample over time for dataset ML-10M, as presented in Fig.1. We limit the plots to one dataset due to space restrictions. From Fig.1a, LND generally outperforms other techniques throughout most of the time in accuracy. The increase obtained by our technique results from the manner in which items are selected to lose relevance. Since it is applied locally, it ensures that concepts are only forgotten in the presence of newer concepts and when not reinforced by new data. In Fig.1b, for $\text{IGSI}_{\hat{\pi}^t}$ without forgetting, the average number of edges grows proportionally to new samples over time. For LND, following a decrease resulting from the deletion of obsolete edges, it grows steadily, as insertions and deletions occur continuously, adding new information and removing outdated ones based on recency. Finally, Fig.1c shows the effect of reduced models on the average processing time per sample. For $\text{IGSI}_{\hat{\pi}^t}$ without forgetting, average time increases with the size of the model. With forgetting, reduction of edges decreases recommendation time. Although there is an increase in update time (Table IV), forgetting eventually reduces average processing time per sample, and such reduction depends on the technique. Overall, our proposed forgetting technique improved diversity and processing time, allowing a balance between these metrics through parameter β and threshold ϕ.

6. CONCLUSION

In this work, we proposed a gradual forgetting technique that locally forgets information based on recency and popularity in order to overcome concerns resulting from the accumulation of obsolete information in incremental neighborhood-based models. The proposed technique decreases the importance of neighborhood of items for every incoming observation to emphasize more recent ones. To increase diversity, parameter β balances the impact of neighborhood degree distribution to increase the likelihood of retaining less popular items. Scalability is controlled by a weight threshold ϕ that

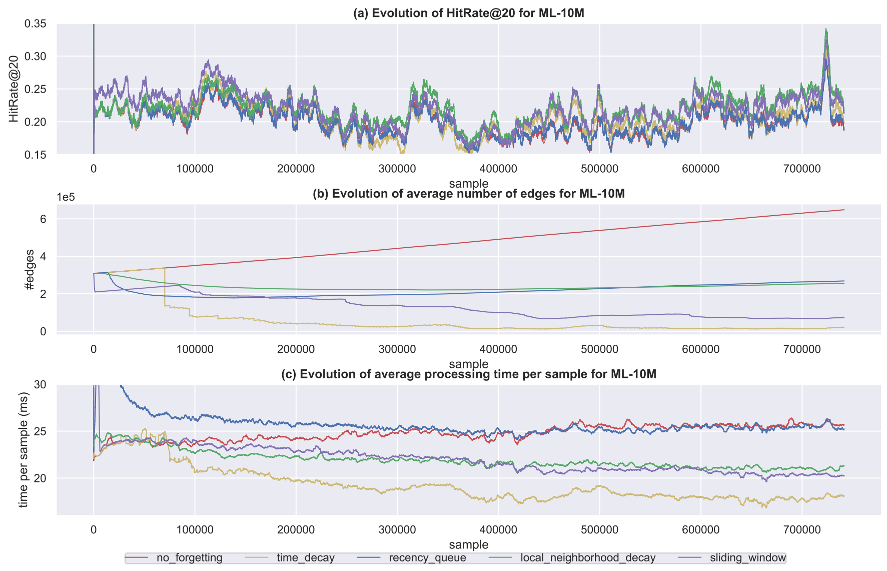

Fig. 1: Evolution of several metrics for dataset ML-10M with window size $n = 5000$.

prunes edges that are not reinforced. We evaluated our proposal on a recent graph-based approach IGSI$_{\hat{\pi}^t}$. Experiments showed that our proposal was able to reduce the size of the graph, improving scalability, while also generally improving accuracy. In future work, we aim to experiment on larger datasets and evaluate its impact on other incremental neighborhood-based methods.

REFERENCES

Al-Ghossein, M., Abdessalem, T., and Barre, A. A survey on stream-based recommender systems. *ACM Computing Surveys (CSUR)* 54 (5): 1–36, 2021.

Cremonesi, P., Koren, Y., and Turrin, R. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM conference on Recommender systems*. pp. 39–46, 2010.

Ding, Y. and Li, X. Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, 2005.

Frigó, E., Pálovics, R., Kelen, D., Kocsis, L., and Benczúr, A. Online ranking prediction in non-stationary environments. In *Proceedings of the 1st Workshop on Temporal Reasoning in Recommender Systems*. ACM, 2017.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. A survey on concept drift adaptation. *ACM computing surveys (CSUR)* 46 (4): 1–37, 2014.

Koren, Y. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD*, 2009.

Matuszyk, P., Vinagre, J., Spiliopoulou, M., Jorge, A. M., and Gama, J. Forgetting techniques for stream-based matrix factorization in recommender systems. *Knowledge and Information Systems* 55 (2): 275–304, 2018.

Ning, X., Desrosiers, C., and Karypis, G. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, 2015.

Schmitt, M. F. and Spinosa, E. J. Scalable stream-based recommendations with random walks on incremental graph of sequential interactions with implicit feedback. *User Modeling and User-Adapted Interaction* 32 (4): 543–573, 2022.

Schmitt, M. F. L. and Spinosa, E. J. Incremental graph of sequential interactions for online recommendation with implicit feedback. In *3rd Workshop on Online Recommender Systems and User Modeling*, 2020.

Smyth, B. and McClave, P. Similarity vs. diversity. In *ICCBR*. Springer, pp. 347–361, 2001.

Tabassum, S., Veloso, B., and Gama, J. On fast and scalable recurring link's prediction in evolving multi-graph streams. *Network Science* 8 (S1): S65–S81, 2020.

Vinagre, J. and Jorge, A. M. Forgetting mechanisms for scalable collaborative filtering. *Journal of the Brazilian Computer Society* 18 (4): 271–282, 2012.

Vinagre, J., Jorge, A. M., and Gama, J. Collaborative filtering with recency-based negative feedback. In *Proceedings of the 30th Annual ACM SAC*. Spain, pp. 963–965, 2015.

Vinagre, J., Jorge, A. M., Rocha, C., and Gama, J. Statistically robust evaluation of stream-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* 33 (7): 2971–2982, 2019.