Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.5496
This work is licensed under a Creative Commons Attribution 4.0 International License.

A Novel Forgetting Technique with Random Walk Sampling for Scalable and Adaptive Stream-Based Recommender Systems

Murilo F. L. Schmitt [**State University of the Midwest of Paraná and Federal University of Paraná** | ***mschmitt@unicentro.br***]
Eduardo J. Spinosa [**Federal University of Paraná** | ***spinosa@inf.ufpr.br***]

Department of Computer Science, State University of the Midwest of Paraná (UNICENTRO), Alameda Élio Antonio Dalla Vecchia, 838, Guarapuava, Paraná, 85040-167, Brazil.

Received: 30 January 2025 • **Accepted:** 20 February 2026 • **Published:** 11 May 2026

Abstract. The explosion of user-generated data at fast rates in online services leads to the need for designing scalable recommender systems that are able to learn from data streams. Stream-based recommender systems are specifically devised for these scenarios, and have seen a recent increase in interest. These systems rely on incremental approaches that incorporate newly generated data on a single pass, resulting in a model that is always up-to-date. A known limitation of only incorporating data into a model is the presence and effect of old data, which negatively affects predictive performance and eventually raises scalability issues. Therefore, an explicit mechanism to forget such data and remove it from the model is required. In this work, we present a graph-based recommender system that recommends items based on random walk sampling, and simultaneously includes new information while also forgetting obsolete ones. Information obtained from random walk sampling is not only used to recommend relevant items, but also to capture structural information from the graph. We devise a forgetting function that prunes obsolete edges based on this information, and also on the recency, popularity and acceptance ratio of items. Our experiments highlight the importance of forgetting obsolete information and suggest the effectiveness of our method, which leads to scalability, accuracy and diversity improvements.

Keywords: Recommender Systems, Data Streams, Random Walks, Forgetting, Online Learning

1 Introduction

Recommender systems (RS) are a central component of several online interactive systems. Their task is to present personalized sets of items to users based on their interests in order to overcome information overload and improve user engagement [Aggarwal *et al*., 2016; Ricci *et al*., 2022]. When users interact with online systems, they provide feedback to it, e.g., with click-through data, shopping behavior and music/movie streaming. Such information can be easily collected from all users and indirectly provides information related to user preferences, i.e., implicit feedback [Jannach *et al*., 2018].

With sufficient feedback provided by several users, predictive models are built to infer unknown user-item preferences based on past user behavior, and recommend relevant items according to preferences of similar users. Approaches that follow this premise are termed collaborative filtering (CF) [Ricci *et al*., 2022], and significant progress through neighborhood-based methods [Nikolakopoulos *et al*., 2022] and matrix factorization [Koren *et al*., 2022] have been achieved.

Despite their effectiveness in predicting user preferences, collaborative filtering techniques usually rely on batch data processing, which assumes that large static datasets are always available for periodical updates. This assumption, however, does not always holds. In several real world systems, intrinsically sequential data is generated continuously at unpredictable rate and order, and it becomes unfeasible to retrain models as data is generated, as retraining gets increasingly more expensive, eventually hindering its application in online systems as it fails to keep up with the incoming data and stay up-to-date, leading to poor recommendation performance.

As opposed to batch-data processing, Stream-Based Recommender Systems (SBRS) [Vinagre *et al*., 2021; Al-Ghossein *et al*., 2021] addresses recommendation as a data stream problem. These assume that user feedback incomes continuously in real time at unpredictable rate and order, new users and items are constantly added to the system and an always-available model must process observations as fast as they arrive [Domingos and Hulten, 2001] and generate recommendations based on up-to-date information.

Incremental algorithms are well suited for these settings as they are able to learn new concepts and update models with continuous incoming feedback from the data stream, and discard data after processing. Hence, they are the basis of SBRS and viable methods for recommendation in dynamic scenarios as they are able to evolve over time in scalable manner, and have recently become an active research field [Al-Ghossein *et al*., 2021]. In fact, traditional CF methods have been adapted to work incrementally, and consistently outperform their batch counterparts in online scenarios [Lommatzsch and Albayrak, 2015; Frigó *et al*., 2017; Jugovac *et al*., 2018; Viniski *et al*., 2021].

Although incremental stream-based approaches are able to process user feedback and incorporate information into a model in real time following a chronological order, data stream learning is subject to *concept drift*, which refers to changes in the underlying concepts over time [Gama *et al*., 2014]. In the context of recommendation, concept drift is prevalent as users' preferences and items' dynamics change in no particular time or manner [Al-Ghossein *et al*., 2021]. For

instance, in online news domains, items have a very short shelf life, and lose relevance very quickly, while users' preferences adapt to current events [Das *et al*., 2007; Gama *et al*., 2014].

Therefore, besides updating the underlying predictive models with new information, learning under concept drift also requires the *forgetting* of obsolete information [Gama *et al*., 2014]. Besides the clear negative effect in predictive performance that results from considering outdated information in the recommendation process, as it does not necessarily reflect current users' interests [Matuszyk *et al*., 2018], the absence of an explicit mechanism to forget such information also leads to ever-growing models, specially in the case of neighborhood-based ones [Vinagre and Jorge, 2012], leading to increasing time and memory requirements and eventually raising scalability issues. Thus, introducing forgetting mechanisms in SBRS could lead to improvements in scalability and accuracy [Vinagre and Jorge, 2012; Gama *et al*., 2014; Matuszyk *et al*., 2018].

Nevertheless, the removal of obsolete concepts is not straightforward, as it differs from old information [Matuszyk *et al*., 2018]. For example, a stable relation between two old movies is not necessarily obsolete, even if not reinforced by new data, while information from a user that temporarily shares her account or buys a product for someone else is irrelevant to that profile even if the observations are new.

Also, a forgetting mechanism must select and remove obsolete information without introducing more complexity in the update process to avoid falling behind the data, and without aggravating sparsity issues intrinsically related to CF. As users interact with only a very small number of available items from the catalog, little information is available from each user. Thus, removing information from learning should avoid aggravating this issue.

In this work[1], we propose a forgetting mechanism for graph-based methods, Local Neighborhood Decay (LND), that fades edges based on recency, popularity and structural information from the underlying graph. Our proposal builds on a recently proposed graph-based SBRS, $\text{IGSI}_{\hat{\pi}^t}$ [Schmitt and Spinosa, 2020, 2022b], in order to overcome its limitations while also exploiting its advantages. $\text{IGSI}_{\hat{\pi}^t}$ incrementally incorporates information in an item-graph, and generates recommendations based on random walk sampling to allow scalable recommendations in data stream settings.

Clearly, its main limitation is that the incremental learning procedure results in an ever-growing graph prone to the issues previously discussed. Besides using random walk sampling to *recommend relevant items*, we propose its use to *also capture structural information from the graph*, and use it to forget obsolete edges. We design a forgetting function that decays edges based on such information, as well as on the recency, popularity and rejection ratio of items, where the relevance of neighborhood of items decreases to emphasize more recent ones.

Finally, we also introduce a weight threshold that prunes obsolete edges in order to improve scalability. The application of forgetting leads to a model capable of learning from incoming user feedback while also discarding outdated information. Our results suggest the effectiveness of our method, which improves scalability, as the removal of obsolete data reduces computational requirements, e.g., time and memory, diversity, by including less popular items in the recommendation lists, and accuracy, since it reduces the effect of obsolete data.

The novel contributions in this paper are threefold:

1. The design of a forgetting function that considers the relevance of items based not only on its recency and popularity, but also on *acceptance factors* and *structural information* of the graph. The proposed function includes parameters that: control the likelihood of retaining less popular items, leading to an improvement in the diversity of the recommendations; and allow faster pruning of edges that are not reinforced, and thus deemed as obsolete. Hence, scalability and diversity can be controlled according to the application;
2. The use of random walk sampling to infer structural information and use it to forget obsolete edges, resulting in smaller models. This procedure exploits the advantages of $\text{IGSI}_{\hat{\pi}^t}$ as it reuses samples that are originally generated for recommendation, and can be generalized to other neighborhood-based models;
3. Extensive experiments on several datasets from different domains, comparisons with other approaches, statistical significance analysis and discussion of the underlying results shed light on the difficulties of deploying forgetting and on how our proposal aims to overcome them.

The remainder of the paper is organized as follows. In Section 2 we review related work. We outline preliminary information in Section 3. Our proposed forgetting technique is defined in Section 4. We outline the experimental setup in Section 5 and present experiments and results in Section 6. In Section 7 we discuss limitations and possible improvements. Finally, we conclude our work in Section 8.

2 Background and related work

SBRS handle user generated data as a stream of observations that are expected to be received in real time at uncontrollable rates indefinitely. Hence a SBRS must deal with such continuous data stream while maintaining up-to-date information. Traditional collaborative filtering (CF) approaches such as neighborhood-based methods [Nikolakopoulos *et al*., 2022] and matrix factorization [Koren *et al*., 2022] have been adapted to work incrementally. A recent comprehensive review related to SBRS methods can be found in Al-Ghossein *et al*. [2021]. For comparative works between incremental and batch-based algorithms, see [Lommatzsch and Albayrak, 2015; Frigó *et al*., 2017; Jugovac *et al*., 2018; Viniski *et al*., 2021]. In this section we focus on contributions regarding forgetting mechanisms in RS.

[1] This paper is a significant extension to our previous work presented in Schmitt and Spinosa [2022a], where we proposed a forgetting function for graph-based methods in data stream settings that fades items based on recency and popularity. In this paper, we design a forgetting mechanism that relies not only on recency and popularity, but also on structural information from the underlying graph. Such information is inferred by random walk sampling, which is originally performed for recommendation purposes, and reused to forget obsolete edges, resulting in scalability, accuracy and diversity improvements. We also present extensive experiments on several datasets and comparisons with other forgetting approaches.

2.1 Forgetting in incremental neighborhood-based methods

Incremental neighborhood-based methods store similarities matrices between users or items based on a predefined metric. Neighborhood-based methods also include graph-based ones, where data is represented by a graph where nodes are users, items or both, and edges represent interactions or similarities between users and items [Nikolakopoulos *et al*., 2022]. These similarities are updated incrementally based on each observation, which ensures that the model always considers recent information. Computation of neighborhoods and ranking of candidate items are performed at recommendation time. Forgetting mechanisms aim to remove information that was previously stored during incremental updates and has since become obsolete in order to improve recommendation times. These mechanisms are typically categorized as *abrupt* or *gradual* [Gama *et al*., 2014].

Abrupt forgetting defines relevance of observations based on a sliding window, where only information in the window is considered and observations that falls outside of it are forgotten. Application of sliding windows require only setting its size. Sliding windows were explored both in user-based [Nasraoui *et al*., 2007; Siddiqui *et al*., 2014] and item-based [Vinagre and Jorge, 2012] methods. Predictive performance, reactivity to changes and scalability depends mainly on the size of the window: short windows likely react faster to changes, while large windows likely present better performance in periods of stability, but with higher computational costs. While the main advantages of abrupt forgetting are its straightforward application and reactivity to changes, it is prone to capturing noise [Gama *et al*., 2014] and fails to distinguish old data from obsolete. Past data is not necessarily obsolete [Matuszyk *et al*., 2018], but such information would be abruptly forgotten as defined by the window.

Gradual forgetting on the other hand relies on fading factors to gradually weight observations to reflect their age, in a way that observations are not abruptly dropped out of memory. The key concept in gradual forgetting is that the importance of observations should decrease with its age [Gama *et al*., 2014], thus recent observations are considered as more important than older ones. In Koychev [2000], a technique to assign higher weights to more recent observations in a content-based method is proposed, thus gradually decreasing the importance of observations over time. Such a technique results in faster adaptations to new user interests.

Similar approaches were explored in Ding and Li [2005] and Liu *et al*. [2010]. In Ding and Li [2005], an item-based CF algorithm is extended to predict new ratings based on time-weighted ratings. A time function is used to assign greater importance to recent data and less relevance to older data. In Liu *et al*. [2010], an online evolutionary CF framework based on an incremental item-based nearest neighbors method that also considers temporal relevance of ratings when generating recommendations is proposed, where a weighting function is used to increase the similarity of items that are rated around the same time. In Symeonidis *et al*. [2020], a Sigmoid-based function is used to assign weights to items according to their position in a session, such that recent items are given greater relevance. In general, generation of recommendations based on recent information results in accuracy improvements. A limitation of these approaches is that, while newer data is given greater importance in the recommendations, obsolete data is not removed from the models.

In Vinagre and Jorge [2012], the use of a decay function on similarities captured by user-based and item-based nearest neighbors methods to reduce relevance of older examples is explored. A positive fading factor $\alpha < 1$, which controls forgetting rate, multiplies the entire similarity matrices before each incremental update. This process decreases similarities continuously over time unless they are reinforced by new data. When these similarities reach a predefined threshold, they are assumed as zero, which improves scalability as a result of reduced models and lower computational requirements. The advantage of this method is its simplicity and application in a single scan, while a shortcoming is reduced effectiveness in the presence of subtle local changes, since forgetting is applied globally.

For recurring link prediction in graph streams, Tabassum *et al*. [2020] proposed a forgetting function that weights links based on frequency and recency. An exponential function is applied on every unique edge on the stream at predefined time intervals, where the weight of each edge at a previous timestamp is multiplied by $(1 - \alpha)$. Parameter $\alpha \in [0, 1]$ defines a bias rate: higher values of α forgets previous edge occurrences faster, biasing the weights according to their recency. The technique also includes a threshold parameter that prunes edges based on their weights at a given time interval, where higher values retains only strong and stable edges. Although their proposal improved performance of recurring link prediction in the context of music recommendation, its application is specific to this domain, where repeated interactions are expected to occur.

Verachtert *et al*. [2022] showed that using only a small and recent part of a dataset to train models results in substantial performance improvements in concept drift scenarios, as opposed to using the entire dataset. The study introduces a training data window δ, which is the maximal age of an event used in training, and evaluate its impact on several models, including the neighborhood-based approaches proposed in Ding and Li [2005] and Liu *et al*. [2010]. The reported results show that in highly dynamic datasets, particularly news domain ones where drifts are pervasive, δ can be set to the last few hours, resulting in significant accuracy improvements while using less data to train the models.

A related contribution by Verachtert *et al*. [2023] further showed that models quickly grow stale in news domain datasets, and benefit from rapid incremental updates. As these may result in increasing computational costs, the work proposes methods for scheduling updates, instead of updating with regular time intervals. Updates are scheduled by estimating information gain based on summary statistics from previous batches of data, hence detecting when they are more relevant. Experiments highlight that the proposed method increases accuracy with reduced resources.

The results of Verachtert *et al*. [2022, 2023] highlight both the importance of frequent incremental updates and of disregarding old information. In this work, we evaluate incremental learning under an even stricter setting, where models are updated incrementally based on each incoming observation

in a data stream.

Forgetting was also explored in the context of knowledge graph-based recommender systems [Burke, 2002; Lu *et al*., 2015]. A knowledge graph contains a set of triples, each represented as $\langle s, p, o \rangle$, where s, p, and o denote the subject, predicate, and object, respectively. Similar to link prediction, the goal of a knowledge graph-based recommender system is to predict missing relations between users and items. Wang and Brewster [2024] proposed an approach that integrates forgetting into knowledge graphs. Two types of forgetting were proposed: passive and intentional forgetting.

Passive forgetting reactively omits local knowledge based on shifts in users' preferences, while intentional forgetting proactively optimizes the entire knowledge graph by removing irrelevant information. Forgetting is performed by rule optimization, that excludes forgettable triples from the rule without altering its logical impact. This is done by assessing potential changes in the importance of entities through centrality measures, before and after the forgetting of a candidate triple [Wang and Brewster, 2024].

Experiments show that the approach is effective in reducing information from the knowledge graphs, while maintaining its predictive performance [Wang and Brewster, 2024]. However, as the approach constructs new graphs after the application of forgetting, and assess the impact of each triple individually, its application in data stream settings is limited.

2.2 Forgetting in incremental matrix factorization

Matrix factorization (MF) methods represent users and items in a common space of latent factors of low dimensionality. The affinity between users and items is measured by the inner product of their embedded vectors. Incremental MF approaches updates only the affected latent vectors based on each received observation [Takács *et al*., 2009; Vinagre *et al*., 2014]. If an observation includes new users or new items, these are randomly initialized and added to the model [Vinagre *et al*., 2014]. Modeling temporal information in latent vectors is known to improve predictive capabilities of MF methods significantly [Koren, 2009].

Several forgetting strategies for MF were developed by Matuszyk *et al*. [2015] and extended in Matuszyk *et al*. [2018]. These select obsolete information and remove their effect from the model, giving more importance to more representative observations, and are divided into two categories, *rating-based* and *latent factor-based* forgetting. Rating-based strategies operates directly on lists of ratings, selecting and discarding ratings from these through sliding windows, using a fixed size or time frame, or sensitivity analysis, e.g., by discarding ratings that would cause dramatic changes in user latent vector or by amplifying forgetting if a prediction model performs poorly.

Latent factor-based forgetting strategies adjust latent factors of users to reduce the impact of past observations, and are deployed for each incoming rating in a stream. These strategies include user fading factors, that reduces the importance of past preferences based on volatility and frequency, and forgetting popular or unpopular items. Experiments suggest that latent factor-based forgetting is successful both in predictive power and computational time, particularly user fading factor and forgetting unpopular items [Matuszyk *et al*., 2015].

Viniski *et al*. [2023] uses specialized optimizers to update latent factors and adjust individual hyperparameters for each user or item based on the observed performance of the model. The idea is to account for local changes in users preferences and items dynamics, which do not occur in the entire dataset, and only pertains to a few users or items. To that end, the proposal adjusts the latent factors considering a specific optimizer term learned specifically for the current user or item in the data stream. Results show that the proposed user-specialized variants increases accuracy and suggest that they are well suited for scenarios where fewer interactions per user are available, at the expense of an increased number of model parameters to store and update.

In Veloso *et al*. [2017], forgetting techniques based on individual, fixed size first in, first out (FIFO) queues were explored, where for each user a queue containing their last n ratings is maintained. With the arrival of new user ratings, these ratings are inserted in the user queue, the queue is shifted and the user latent vector is faded according to the ratings on the queue. Fading is performed based on four forgetting strategies: two time-based functions, that fade the ratings according to the timestamps of both the faded rating and the current rating, and two positional-based functions, that take into account the position of the rating on the queue. These strategies improve accuracy when compared to the forget unpopular technique proposed by Matuszyk *et al*. [2015].

Finally, a FIFO queue is also used in Vinagre *et al*. [2015], where a global queue of all items seen in the stream, ordered based on the recency of item occurrences, is deployed. For every new observation in the stream, a few older items from the queue are selected to be used as negative feedback to the current user, giving greater importance to more recent events, improving accuracy.

3 Preliminaries

The goal of the top-N recommendation problem is to recommend to users personalized subsets of items based on their interests [Cremonesi *et al*., 2010]. In this work we approach the top-N problem under a data stream setting. In such a setting, user feedback is continuously generated at unpredictable rate and order, is potentially unbounded, with new users and items entering the system and previously known concepts being subject to changes [Vinagre *et al*., 2021; Al-Ghossein *et al*., 2021].

Denote by $U = \{u_1, u_2, ..., u_m\}$ and $I = \{i_1, i_2, ..., i_n\}$ the increasing set of users and items, respectively. Denote $\langle u, i, t \rangle$ as an incoming interaction (observation) in a continuous user feedback data stream, which indicates that user u interacted with item i at time t. A stream-based model must update itself based on each $\langle u, i, t \rangle$, while also accounting for new users and items, at least as fast as the arrival of interactions, with a single pass in order to avoid falling behind the data [Domingos and Hulten, 2001; Vinagre *et al*., 2021], and generate recommendations based on up-to-date information in real time.

In Schmitt and Spinosa [2020, 2022b], the authors 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 a weighted directed graph by $G = (V, E, w)$, where $V = \{v_1, v_2, ..., v_n\} \subseteq I$ denotes the set of nodes and $E \subseteq V \times V$ denotes the set of edges. Each edge e has an associated weight $w(e) \in \mathbb{R}_+$. Let A denote the adjacency matrix of G, where $a_{ij} = w((i, j))$ if $(i, j) \in E$ and 0 otherwise, and D denote the diagonal degree matrix of G where: $d_{ii} = \sum_{(i,k)\in E} a_{ik}$.

A list of interactions ordered according to time t is defined for each user $u \in U$ as $S_u = \{(v_1, t_1), ..., (v_n, t_n)\}$. Denote the last element of S_u by (li_u, lt_u), such that $li_u \in I$ is the most recently interacted item by u and lt_u is the timestamp of the interaction. The graph is updated incrementally considering li_u and the current $\langle u, i, t\rangle$ on a data stream of implicit user feedback.

For each incoming observation $\langle u, i, t\rangle$, if u is unknown, a representation for u is created, i.e., $S_u = \emptyset$, then (i, t) is included in S_u and $li_u \leftarrow i$. If u is known, the feedback is included in the graph by an edge connecting li_u to i, and both S_u and li_u are updated accordingly. The relevance of edges is distinguished by their weights, where the relevance of each sequential interaction is reinforced in the weight of the edge based on the frequency with which those interactions are made by users. If item i is unknown, A and V are updated to include 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 as defined in Eq. (1):

$$w((li_u, i))_t = w((li_u, i))_{t-1} + 1 \tag{1}$$

Recommendations are made by ranking candidate items through simulations of random walks with restart. A random walk with restart (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 a source node $s \in V$, with probability γ the walker moves from a node to one of its neighbors at random, such that the probability of transitioning from a vertex i to j is $p_{ij} = a_{ij}/d_{ii}$, or returns back to the source s with $(1 - \gamma)$ probability. Denote a row vector with 1 in the column for node s and 0's elsewhere by e_s. The stationary distribution π_s^t of RWR starting at node s, which is a weighted sum of all landing probabilities, can be expressed by Eq. (2) [Andersen *et al*., 2008]:

$$\pi_s^t = (1 - \gamma)e_s \sum_{t=0}^{\infty} \gamma^t P^t \tag{2}$$

It is also possible to bias the probability distribution towards a set of source nodes instead of a single source node, an approach known as Personalized PageRank [Page *et al*., 1999; Jeh and Widom, 2002; Haveliwala, 2003].

While the stationary distribution π_s^t provides a ranking of items related to s, where nodes close to the source have more relevance as they tend to be visited more frequently by the walk, it is impractical to be continuously computed in dynamic models. Thus, IGSI$_{\hat{\pi}^t}$ approximates π_s^t through simulations of random walk, by running M t-step random walks starting from s. For each node $v \in V$, $\pi_{s,v}^t$ is defined by the number of visits in v by the M independent random walks multiplied by $\frac{(1-\gamma)}{M}$. The approximation $\hat{\pi}_s^t$ for a node v can be expressed by Eq. (3) [Schmitt and Spinosa, 2022b]:

$$\hat{\pi}_{s,v}^t = \frac{1 - \gamma}{M} \sum_{k=1}^{M} R_{s,t}^k(v) \tag{3}$$

where $R_{s,t}^k(v)$ is the number of visits to node v during the k-th run of a t-step random walk initiated at s, with t and M being hyperparameters.

Denote rS_u as the last r items in S_u, i.e., the r most recent items that user u has interacted with. To generate recommendations to a user u, IGSI$_{\hat{\pi}^t}$ computes and averages $\hat{\pi}_s^t$ for each $s \in rS_u$. The results obtained in Schmitt and Spinosa [2022b] suggest that filtering recommendations by a few recent items, defined by hyperparameter r, improves accuracy as it provides items related to the short-term interests of u. Also, that short random walks ($t = 3$) provide accurate recommendations with minor improvements afterwards, and finally that the number of independent random walks M offers a balance between accuracy and scalability.

4 Proposed method

4.1 Motivation

The original IGSI$_{\hat{\pi}^t}$ model [Schmitt and Spinosa, 2022b] performs two main operations: updating its underlying graph, and generating recommendations based on up-to-date information. The update procedure is performed on a single edge in the graph, depending on the previously collected sample. If the relation between two items as inferred by the most recent sample is new, an edge is created between these two items; otherwise, the edge connecting these two items already exists, and its weight is increased to reinforce its importance. The recommendation procedure then ranks candidate items through random walk sampling, such that nodes that are closer to the sources are assumed to be more relevant. While continuous inclusion of new information allows the model to stay up-to-date and immediately consider recent user preferences, which are likely to better reflect their present ones, such preferences are dynamic and change over time, hence subject to concept drift [Gama *et al*., 2014]. Concept drift refers to data distribution shifts over time, pervasive in non-stationary settings, thus requiring the design of adaptive models [Gama *et al*., 2014; Al-Ghossein *et al*., 2021].

As such, information that is included in incremental manner may become obsolete, which results in negative impact both on scalability and accuracy. Whereas scalability is impacted by the continuous growth of the underlying graph, predictive performance is affected by the inclusion of obsolete concepts in the recommendation procedure. The potential inclusion of new edges based on every incoming observation leads to a graph that grows linearly to the number of generated samples, which in turn raises the time required for the generation of recommendations. While degrading accuracy is obviously undesirable, scalability issues may actually hinder the applicability of models in data stream settings.

As discussed in Section 2.1, forgetting is typically done abruptly or gradually. In this work, we are interested in gradual forgetting, as it decreases the importance of observations

according to its age, instead of completely dropping it out of memory as done in abrupt forgetting. Gradual forgetting rely on fading factors to decay the importance of observations over time. Previous work has considered fading information globally, by decaying the entire similarity matrices [Vinagre and Jorge, 2012], and periodically, as defined by time intervals [Tabassum *et al*., 2020]. We propose the application of forgetting locally.

4.2 Local neighborhood decay

To avoid the aforementioned accuracy and scalability issues, we propose a novel forgetting mechanism, *local neighborhood decay* (LND), designed to exploit the advantages of $\text{IGSI}_{\hat{\pi}^t}$, and that could be generalized and extended to other graph-based approaches [Cooper *et al*., 2014; Christoffel *et al*., 2015] or neighborhood-based ones [Miranda and Jorge, 2009].

$\text{IGSI}_{\hat{\pi}^t}$ updates the edge connecting li_u, the last interacted item by user u, to i, the item of the new interaction. The most straightforward way to apply forgetting would be to simply fade all neighbors of li_u before updating edge (li_u, i), as defined by Eq.(4):

$$w((li_u, s))_t = \alpha \cdot w((li_u, s))_{t-1} \tag{4}$$

where $\alpha \in (0, 1)$ is a fading factor that controls the rate of forgetting and s is a neighbor node of li_u. This process, which is performed locally, ensures that recent data is emphasized over older information, and avoids aggravation of sparsity issues, as forgetting only occurs for items that are guaranteed to be updated with new information, i.e., concepts are replaced only in the presence of newer ones.

However, this forgetting mechanism does not further distinguish items and simply fades those that are not reinforced by the current observation, likely biasing the graph toward popular items. Hence, we propose an improved forgetting function that, besides recency, scores items based on popularity and structural information, fading them proportionally to a predefined relevance score, as defined by Eq.(5):

$$w((p, s))_t = \alpha^{(1-score)} \cdot w((p, s))_{t-1} \tag{5}$$

where p and s are adjacent nodes in the graph and $score$ is a relevance score assigned to s from p, such that items with low score are subject to faster forgetting.

We consider popularity information in order to increase the diversity of recommendation lists and boost the inclusion of items from the *long-tail*, a known issue in recommender systems [Celma and Cano, 2008] and seldom explored in SBRS [Al-Ghossein *et al*., 2021]. In recommender systems, the majority of interactions are related to a small fraction of the most popular items, while the remaining large portion of the catalog, the *long-tail* of the distribution, only account for a small fraction of interactions [Cremonesi *et al*., 2010].

We measure item popularity through node degree information. To boost the diversity of the recommendations lists, we are interested in items with both low indegree and high outdegree. Items with low indegree are likely to be either from the long-tail, or new ones, while items with high outdegree encourage exploration of random walks. As such, we propose to faster decay items with high indegree, which are popular items, and nodes with low outdegree and sink nodes, i.e., nodes without outgoing connections, likely to be noise.

We also score item popularity based on an acceptance factor, measured by the ratio between: the number of times an item was accepted (clicked) by users when recommended; and the number of times it was recommended. The premise behind it is that mainstream items that are constantly recommended, but ignored, can be assumed as irrelevant, as opposed to unknown to users [He *et al*., 2016; Matuszyk *et al*., 2018]. By decaying these very popular items, prevalent in the recommendation lists but with low acceptance, we reduce their impact on the model.

Hence we define the score for a given item in Eq.(6):

$$score = (\beta x_{s,p} + (1 - \beta) y_{s,p}) \cdot (z^{\tau}_{s,p}) \tag{6}$$

where $\beta \in [0, 1]$ is a diversity parameter and a convex combination that balances diversity and exploration, where $\beta = 0.5$ emphasizes items from the long-tail that increase the exploration of random walks, and $\tau \in [0, 1]$ is the acceptance parameter, which controls forgetting for popular but irrelevant items. $x_{s,p}$ and $y_{s,p}$ are the indegree factor and outdegree factor of node s normalized according to the degree distribution of the neighborhood of node p, respectively, and $z_{s,p}$ is the acceptance factor of item s normalized according to the acceptance factors of neighbors of node p. Indegree factor $x_{s,p}$ and outdegree factor $y_{s,p}$ are defined by Eq.(7) and Eq.(8), respectively:

$$x_{s,p} = \begin{cases} 0 & \text{if } \Delta^-(N^+(p)) = \delta^-(N^+(p)) \\ \frac{\Delta^-(N^+(p)) - deg^-(s)}{\Delta^-(N^+(p)) - \delta^-(N^+(p))} & \text{otherwise} \end{cases} \tag{7}$$

$$y_{s,p} = \begin{cases} 0 & \text{if } \Delta^+(N^+(p)) = \delta^+(N^+(p)) \\ \frac{deg^+(s) - \delta^+(N^+(p)}{\Delta^+(N^+(p)) - \delta^+(N^+(p))} & \text{otherwise} \end{cases} \tag{8}$$

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

4.3 Structural information through random walk sampling

Finally, we use random walk sampling *not only to provide recommendations but also to infer the structural information of the graph*. To that end, while performing random walk sampling in order to recommend relevant items, we store these samples to update the model *after* receiving the actual observation $\langle u, i, t \rangle$.

Denote by $RW_{p,t} = \{RW^1_{p,t}, RW^2_{p,t}, ..., RW^M_{p,t}\}$ as the set of M t-step random walk samples starting from node p, where $RW^k_{p,t} = \langle v_0, v_1, ..., v_t \rangle$ is the random sequence of nodes generated by the k-th t-step random walk started from p. In the end, we are interested in nodes that are on paths from the M t-step random walks started from p that reached the desired item i, defined as $RW'_{p,t} = \{v \in RW^k_{p,t} : k = \{1, ..., M\}, i \in RW^k_{p,t}\}$. Items s not in any path that reached

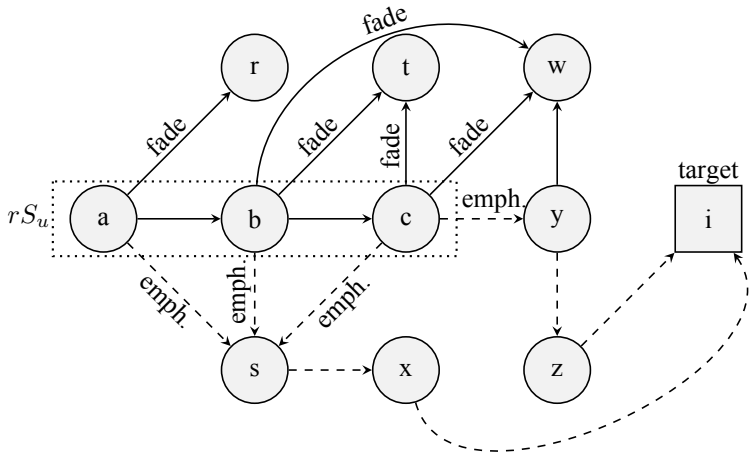


Figure 1. Application of our proposed forgetting mechanism on IGSI$_{\hat{\pi}^t}$. We omit edge weights for ease of visualization and instead notate where to emphasize and where to fade. The example considers a given user u whose most recent interactions are with items a, b and c, i.e., $rS_u =< a, b, c >$, as highlighted by a dotted black square. Thus, a recommendation would consist in performing random walks from these source nodes. Assuming that the next interaction from u is with item i, we wish to emphasize neighbors of nodes in rS_u that reached i as inferred by the samples with Eq.(9) and fade neighbors that did not reach i with Eq.(5). Assuming $t = 3$, paths to i from nodes in rS_u are highlighted with dashed edges. In this example i is reachable from a through s, thus edge (a, s) is emphasized and its remaining neighbor r is faded; as i is also reachable from b through s, edge (b, s) is emphasized and all its remaining neighbors t and w are faded; and as i is reachable from c through nodes s and y, edges (c, s) and (c, y) are emphasized, while the remaining neighbors t and w are faded.

i, i.e., $s \notin RW'_{p,t}$ are deemed irrelevant and are faded based on Eq.(5), while items in paths that reached i, i.e., $s \in RW'_{p,t}$ are considered relevant, and emphasized based on Eq.(9):

$$w((p, s))_t = \alpha^{-score} \cdot w((p, s))_{t-1} \tag{9}$$

In this way, we avoid aggravation of sparsity issues and ensure that relevant and recent information is kept in the graph, as perceived important connections are increased and irrelevant ones are faded as induced by the walks, by popularity factors and by recent observations from the data stream. The application of our proposed forgetting mechanism on IGSI$_{\hat{\pi}^t}$ is illustrated in Figure 1.

Consider the graph presented in Figure 1, and a user u whose most recent interactions are with items a, b and c, respectively, i.e., $rS_u =< a, b, c >$. A recommendation would consist in performing random walks from these source nodes. Assuming that the next interaction from u is with item i, LND emphasizes neighbors of nodes in rS_u that reached i as inferred by the samples with Eq.(9) and fade neighbors that did not reach i with Eq.(5). In the example, assuming 3-step random walks, paths to i from nodes in rS_u are highlighted with dashed edges: i is reachable from a through s, thus edge (a, s) is emphasized and its remaining neighbor r is faded; as i is also reachable from b through s, edge (b, s) is emphasized and all its remaining neighbors t and w are faded; and as i is reachable from c through nodes s and y, edges (c, s) and (c, y) are emphasized, while the remaining neighbors t and w are faded.

As this process ensures that irrelevant information is continuously faded unless reinforced by new data, hence ensuring that relevant information is retained in the neighborhood of nodes, we introduce a weight threshold $\phi = \alpha^x$ that removes obsolete edges proportionally to the fading factor α to account for the increasing scalability concerns related to ever-growing models, where x is a parameter that controls the weight threshold based on α, 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.

Thus, our forgetting technique, LND, requires four parameters: fading factor α, diversity parameter β, acceptance parameter τ and x that controls weight threshold ϕ. The online procedure of IGSI$_{\hat{\pi}^t}$ with our proposed forgetting mechanism is presented in Algorithm 1.

When a new observation $\langle u, i, t \rangle$ arrives from the data stream, we first produce a recommendation to u for evaluation purposes (Section 5), and also store the random walk samples generated in this process in $RW_{rS_u,t}$. Then, *after* evaluation, we apply forgetting over rS_u, the last r interacted items by user u that are used as sources in the random walks. Neighbors of nodes in rS_u that are in any path that reached i as inferred by the samples, $RW'_{p,t}$, are emphasized with Eq.(9), while nodes that are not in $RW'_{p,t}$ are faded based on Eq.(5). Then, obsolete edges are removed if necessary as defined by the weight threshold. Finally, we update the graph with the new observation, as described in Section 3.

5 Experimental setup

In this section, we describe the experimental setup used to evaluate our proposal on a simulated data stream setting. We first describe the datasets, then the prequential evaluation protocol, define the evaluation metrics and detail the evaluated baseline algorithms and hyperparameter tuning.

5.1 Datasets

Eight datasets from several domains were used, as summarized in Table 1. ML-1M and ML-10M are binary versions of MovieLens-1M and MovieLens-10M datasets [Harper and

Algorithm 1: IGSI$_{\tilde{\pi}^t}$ with proposed forgetting method LND

```
// Require:
D = {(< u, i, t >)_1, ...}: data stream;
r: recency parameter;
k: recommendation list size;
t: length of walks;
M: number of walks;
α: decay parameter;
β: diversity parameter;
τ: acceptance parameter;
x: parameter for edge removal;
// Procedure:
(V, E, w): weighted directed graph;
U: set of users;
I: set of items;
S_u: list of interactions by u;
li_u: last item interacted by u;
for < u, i, t >∈ D do
    top_n_items, RW_{rS_u,t} = recommendItems(S_u,
      r, k, V, E, t, M) [Schmitt and Spinosa, 2022b];
    evaluate(top_n_items);
    // Forgetting
    // last r items interacted by u
    for p ∈ rS_u do
        for s ∈ neighbors(p) do
            if s ∉ RW'_{p,t} then
                w((p,s))_t = α^{(1−score)} · w((p,s))_{t−1};
                if w((p,s)) < α^x then
                    E ← E \ (p,s);
            else
                w((p,s))_t = α^{−score} · w((p,s))_{t−1};
    U, I, S_u, li_u, lt_u, (V, E, w) =
      incrementalGraphUpdate() [Schmitt and
      Spinosa, 2022b];
```

Konstan, 2015], respectively. PLC-PL are logs of track additions to personal playlists and PLC-STR are logs of music listening events, both from social network Palco Principal [Vinagre *et al*., 2014]. LFM-1K are the first 25% of events observed from the Last.fm dataset [Celma, 2010] in chronological order. BOOK and ELEC are binary versions of categories *books* and *electronics* from the Amazon dataset [McAuley *et al*., 2015]. GLOBO is a news domain dataset extracted from news portal globo.com [de Souza Pereira Moreira *et al*., 2019].

Table 1. Dataset description.

Dataset	Domain	Events	Users	Items	Sparsity
ML-1M	Movie	226,310	6,014	3,232	98.84%
ML-10M	Movie	1,544,812	67,312	8,721	99.74%
PLC-PL	Music	111.942	10.392	26.117	99.96%
PLC-STR	Music	1,466,893	25,463	40,213	99.92%
LFM-1K	Music	4,234,033	546	399,171	99.46%
BOOK	E-commerce	6,278,141	1,827,029	734,918	99.99%
ELEC	E-commerce	618,560	451,486	60,842	99.99%
GLOBO	News	1,830,308	259,690	35,644	99.98%

5.2 Evaluation protocol

We adopt a prequential evaluation [Gama *et al*., 2009] for SBRS proposed by Vinagre *et al*. [2021]. Evaluation is performed for every incoming observation $\langle u, i, t\rangle$ in the stream in a test-then-learn manner, thus allowing continuous monitoring of several metrics, defined in 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 actual observed item i;
3. Update the model with $\langle u, i, t\rangle$;
4. Proceed to the next observation.

The evaluation process is illustrated in Figure 2. For all datasets, observations are ordered chronologically. For explicit feedback datasets ML-1M, ML-10M, BOOK and ELEC, we remove ratings below 5 to simulate implicit feedback. For dataset GLOBO, we disregard session information and only consider user information. After pre-processing, all datasets are composed of events with form $\langle u, i, t\rangle$, indicating that user u interacted with item i at time t. Then, we split the datasets following the evaluation procedure proposed by Matuszyk *et al*. [2018]: the first 25% are used to built initial models; the following 25% to optimize hyperparameters and the remaining 50% for prequential evaluation.

5.3 Metrics

We measure our proposal through three main factors: scalability, accuracy and diversity. We measure scalability through average update and recommendation times, the two main operations in our proposal, and also through average number of edges in the graph to assess the effect of ever-growing models in these operations. Accuracy is measured through HitRate (HR) and Discounted Cumulative Gain (DCG). HR@N returns 1 if item i is within the N first recommended items, and 0 otherwise. DCG@N considers the position of i in the recommendation list. For prequential evaluation, DCG@N is defined as [Frigó *et al*., 2017]:

$$DCG@N = \begin{cases} \frac{1}{\log_2(rank(i)+1)} & \text{if } rank(i) \leq N \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where i is the positive item and $rank(i)$ denotes the position of i in the recommendation list of size N. We set N = 20 for all our experiments, the same number set in previous work [Schmitt and Spinosa, 2020, 2022b,a].

We also evaluate diversity, through Intra-List Diversity (ILD) [Smyth and McClave, 2001], defined as the average pairwise distance of items in the list [Castells *et al*., 2015]:

$$ILD = \frac{1}{|R|(|R|-1)} \sum_{i \in R} \sum_{j \in R} d(i,j) \quad (11)$$

where R is the recommendation list and $d(i,j)$ is a given distance measure between items i and j, such as cosine distance:

$$d(i,j) = 1 - \frac{|U_i \cap U_j|}{\sqrt{|U_i|} \times \sqrt{|U_j|}} \quad (12)$$

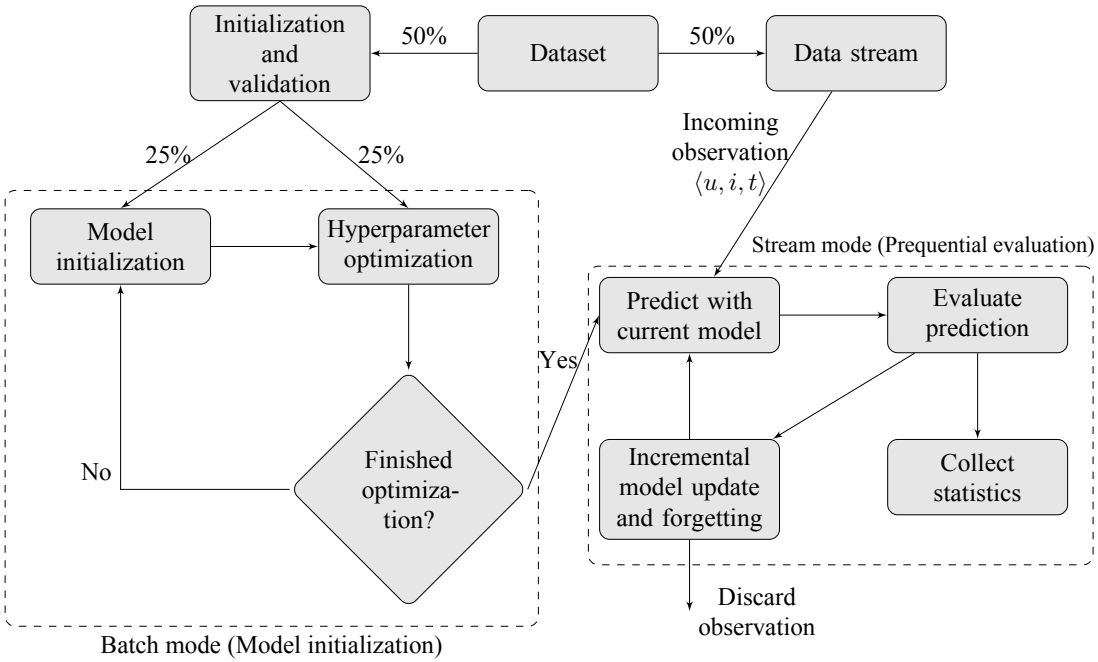


Figure 2. Evaluation protocol used in our experiments. Based on Matuszyk *et al*. [2018], Vinagre *et al*. [2021] and Viniski *et al*. [2023].

where U_i and U_j denote the set of users that interacted with items i and j, respectively.

Finally, statistical significance of the results are assessed through McNemar's test over sliding windows [Vinagre *et al*., 2021]. Throughout the prequential process, several metrics are produced, including a sequence of hit rate values. Each window produces a sequence of hit rates $\in \{0, 1\}$ of size n, and the test can be used in each window to compare the performance of two different algorithms. Given algorithms A and B, McNemar's test requires computing two quantities, the number of observations correctly classified by A and not by B, defined as n_{10}; and n_{01}, where the opposite situation happens. At every step of the prequential process, McNemar's test is performed by calculating the statistic:

$$Mcn = \frac{(n_{10} - n_{01})^2}{n_{10} + n_{01}} \tag{13}$$

where $Mcn \sim \chi^2_1$. For a significance level of $\alpha = 0.01$, the critical value is $Mcn = 6.635$.

5.4 Baselines

To assess the effectiveness of our proposed forgetting technique LND, we use IGSI$_{\hat{\pi}^t}$ as a base model and extend it with our proposed forgetting technique, and also with the following ones to allow for comparisons:

Sliding window. A traditional abrupt forgetting technique [Nasraoui *et al*., 2007]. We define sliding windows based on time intervals ω, e.g., 30 days, where only observations included in the windows are considered for the ranking of items, and observations outside of it are abruptly forgotten. IGSI$_{\hat{\pi}^t}$ with sliding window would consist in a graph composed only by interactions that occurred in the intervals defined by the window.

Time-decay. A gradual forgetting technique [Tabassum *et al*., 2020] that reduces the relevance of edges based on time intervals. We define a fixed time window ω, e.g., 30 days, and each time the predetermined window period elapses, edges are gradually forgotten based on Eq.(4), and removed if necessary, i.e., $w(e) < 1.0$. With this technique, forgetting is performed gradually and only based on the predefined time window, as opposed to a sliding window that abruptly forgets information.

Recency queue. An adaptation of the algorithm proposed by Vinagre *et al*. [2015] for incremental matrix factorization, designed to select negative feedback from the stream based on the frequency of items. In this forgetting technique, a global FIFO queue is maintained, containing all items seen in the stream, where the tail contains the most recent interacted items, and the head contains the items that have not been interacted with for a while. For an incoming interaction $\langle u, i, t \rangle$, item i is moved to the tail of the queue, and the item at the head of the queue j is selected to lose relevance, by reducing the weight of the edges connecting its predecessors to j, as defined in Eq.(4). If necessary, edges are removed from the graph based on threshold ϕ. Finally, j is reinserted at the tail of the queue. In essence, this approach penalizes infrequent items along the stream and values recent information as these will not be forgotten.

5.5 Hyperparameter tuning

The original IGSI$_{\hat{\pi}^t}$ has three hyperparameters: the number of steps in a random walk t, the number of random walk samples M and the number of most recent user interactions to use as random walk source r. Based on the optimal hyperparameters reported in Schmitt and Spinosa [2022b] and further optimizations based on grid searches, we set hyperparameters for IGSI$_{\hat{\pi}^t}$ as $t = 3$ and $M = 1000$ for all datasets, $r = 2$ for ML-10M and GLOBO, $r = 3$ for PLC-STR and ELEC, $r = 4$ for ML-1M and LFM-1K and $r = 5$ for PLC-PL and BOOK.

Table 2. Graph's properties per dataset. The reported properties are number of nodes ($|\mathcal{V}|$), number of edges ($|\mathcal{E}|$), density, average degree, average weighted degree, minimum weighted degree (Min), maximum weighted degree (Max) and number of low weighted degree nodes (#Low). For each dataset, cells from column (Min, Max, #Low) are split into two rows, where the first refer to indegree node information, and the second refers to outdegree node information.

Dataset	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	Density	Avg. degree	Avg. weighted degree	(Min, Max, #Low)
ML-1M	3,232	140,787	1.348	43.56	68.16	(0, 1813, 2473) (0, 1916, 2487)
ML-10M	8,721	648,368	0.852	74.34	169.40	(0, 15009, 7338) (0, 15745, 7356)
LFM	399,171	2,623,241	0.001	6.57	10.60	(0, 1698, 318626) (0, 1698, 318625)
PLC-PL	26,117	75,858	0.011	2.90	3.89	(0, 139, 19372) (0, 148, 19389)
PLC-STR	40,213	469,825	0.029	11.68	35.84	(0, 11932, 32732) (0, 11937, 32720)
BOOK	734,918	2,192,619	0.0004	2.98	6.05	(0, 2864, 644863) (0, 2863, 644820)
ELEC	60,842	133,369	0.003	2.19	2.74	(0, 920, 49797) (0, 802, 49496)
GLOBO	35,644	488,778	0.038	13.71	44.06	(0, 17516, 32689) (0, 17904, 32842)

For parameters related to forgetting, we tested values for $\alpha \in [0.8, 0.99]$ in steps of 0.01. Parameter α is the fading factor that controls the rate of forgetting, where higher values for α results in slower forgetting. We observed that higher values for α presented the best results for all techniques. Hence, we set $\alpha = 0.99$ for all forgetting approaches. Hyperparameters for baselines were optimized based on grid search following the same methodology used to optimize the hyperparameters of our proposal. Hyperparameters for all approaches are reported in Appendix A. Besides α, our proposal has three other hyperparameters, and we discuss their impact next, before performing comparisons with baselines.

6 Experiments

In this section we report experiments performed to evaluate our proposal forgetting technique, LND, on simulated data streams. First, we analyze the impact of each hyperparameter through different metrics. Then, we evaluate LND with the defined hyperparameters on the entire datasets and present overall results and comparisons with baseline techniques. Finally, we discuss the results and outline possible shortcomings and improvements.

Algorithms were implemented in Python 3 and ran on an Intel Core i7-4770 of 3.4 GHz with 16 GB RAM and Ubuntu 20.04. To complement the experimental analysis, we present in Table 2 properties of underlying graphs per dataset. These describe the obtained graphs through $\text{IGSI}_{\hat{\pi}^t}$, without the application of forgetting. We report general information, such as number of nodes, number of edges, density, average degree and minimum, maximum and average weighted degree. We also report the number of low degree nodes. We define a low degree node as one that is below the average degree.

6.1 Impact of diversity hyperparameter β

First we evaluate the impact of diversity parameter β, that acts as a convex combination of indegree and outdegree factors, $x_{s,p}$ and $y_{s,p}$, respectively. Nodes with low indegree are likely to be items from the long-tail, or new ones, while nodes with high outdegree increases the exploration of random walks. Thus, β introduces a balance between these two factors, and, based on Eq.(6), higher values of β should increase the diversity of recommendations, as we are including items from the long-tail in the recommendation lists. We tested values for $\beta \in [0, 1]$ in steps of 0.1. Results of these experiments, measured by HR@20 and ILD are presented in Tables 3 and 4.

As shown in Tables 3 and 4, for all datasets diversity increases with parameter β, even if slightly, as expected. The effect of β in accuracy depends on the dataset: for ML-1M, ML-10M and GLOBO, it tends to increase as β increases until reaching a certain threshold, after which it starts to decrease, but generally is still higher than $\text{IGSI}_{\hat{\pi}^t}$ without forgetting. For datasets PLC-PL and LFM-1K, β does not affect accuracy, while for PLC-STR, BOOK and ELEC it only slightly affects it.

Two interesting observations can be further made from these results. First, that for all datasets there is at least one value for β that increases diversity while maintaining accuracy, or even increasing it. Thus, β can be adjusted according to the application. Second, the impact of β on both accuracy and diversity is highly dependent on the dataset and directly related to the degree distribution on the graph.

The effect of the degree distribution can be seen when we compare results on ML-1M, ML-10M and GLOBO to the remaining datasets. Since the goal of β is to increase the exploration of random walks while also biasing it towards long-tail items, its impact depends on the amount of nodes available to explore during the walks. As such, there are greater changes in accuracy and diversity on these datasets as they have higher average degree, as seen in Table 2, hence

Table 3. Impact of parameter β on IGSI$_{\hat{\pi}^t}$ with our proposed forgetting technique for datasets ML-1M, ML-10M, PLC-PL and PLC-STR. Column β refers to the tested values for hyperparameter β, while HR@20 and ILD represents the HitRate@20 and intra-list diversity, respectively, grouped by dataset. The first row of the results refer to IGSI$_{\hat{\pi}^t}$ without forgetting. Values highlighted in bold indicate the highest value for β that is superior in accuracy with statistical significance in comparison to higher values.

	ML-1M		**ML-10M**		**PLC-PL**		**PLC-STR**	
β	**HR@20**	**ILD**	**HR@20**	**ILD**	**HR@20**	**ILD**	**HR@20**	**ILD**
-	0.234	0.813	0.198	0.829	0.396	0.826	0.590	0.817
1.0	0.232	0.839	0.199	0.874	**0.396**	**0.827**	0.591	0.819
0.9	0.235	0.833	0.212	0.863	0.396	0.827	0.591	0.819
0.8	0.235	0.826	0.220	0.854	0.396	0.827	0.591	0.818
0.7	**0.237**	**0.819**	0.224	0.844	0.396	0.826	0.591	0.818
0.6	0.236	0.812	**0.225**	**0.835**	0.397	0.826	**0.592**	**0.818**
0.5	0.236	0.805	0.222	0.826	0.397	0.826	0.592	0.817
0.4	0.233	0.798	0.216	0.821	0.397	0.826	0.592	0.817
0.3	0.232	0.792	0.209	0.819	0.397	0.826	0.592	0.817
0.2	0.226	0.786	0.205	0.818	0.397	0.826	0.592	0.817
0.1	0.226	0.782	0.198	0.816	0.397	0.826	0.592	0.816
0.0	0.221	0.778	0.191	0.815	0.397	0.826	0.592	0.816

Table 4. Impact of parameter β on IGSI$_{\hat{\pi}^t}$ with our proposed forgetting technique for datasets LFM-1K, BOOK, ELEC and GLOBO. Column β refers to the tested values for hyperparameter β, while HR@20 and ILD represents the HitRate@20 and intra-list diversity, respectively, grouped by dataset. The first row of the results refer to IGSI$_{\hat{\pi}^t}$ without forgetting. Values highlighted in bold indicate the highest value for β that is superior in accuracy with statistical significance in comparison to higher values.

	LFM-1K		**BOOK**		**ELEC**		**GLOBO**	
β	**HR@20**	**ILD**	**HR@20**	**ILD**	**HR@20**	**ILD**	**HR@20**	**ILD**
-	0.182	0.841	0.661	0.827	0.268	0.986	0.456	0.940
1.0	**0.182**	**0.842**	0.661	0.832	0.267	0.987	0.490	0.950
0.9	0.182	0.842	**0.662**	**0.831**	0.267	0.987	0.527	0.945
0.8	0.182	0.842	0.662	0.830	0.267	0.987	0.555	0.944
0.7	0.182	0.842	0.662	0.829	0.267	0.987	0.574	0.943
0.6	0.182	0.842	0.662	0.829	**0.268**	**0.987**	0.583	0.943
0.5	0.182	0.841	0.662	0.828	0.269	0.986	**0.594**	**0.942**
0.4	0.182	0.841	0.662	0.828	0.269	0.986	0.591	0.941
0.3	0.182	0.841	0.662	0.828	0.269	0.986	0.587	0.941
0.2	0.182	0.841	0.662	0.828	0.269	0.986	0.581	0.941
0.1	0.182	0.841	0.662	0.828	0.270	0.986	0.576	0.941
0.0	0.182	0.841	0.662	0.828	0.269	0.986	0.562	0.941

more nodes that can be visited through random walks.

On the other hand, on datasets with low average degree, such as LFM-1K, PLC-PL, BOOK and ELEC, the impact of β is minimal, as the possibilities for exploration are already very restrict.

For subsequent experiments, we select values for β that maximizes diversity without decreasing accuracy. To that end, we assess the statistical significance of the results. We select the maximum value for β whose difference to greater values of it results in statistically significant differences in accuracy. In other words, starting from $\beta = 1.0$, we decrease β until there are no statistically significant differences in accuracy. Thus, we set $\beta = 0.7$ for ML-1M, $\beta = 0.6$ for ML-10M, PLC-STR and ELEC, $\beta = 1.0$ for LFM-1K and PLC-PL, $\beta = 0.9$ for BOOK and $\beta = 0.5$ for GLOBO.

6.2 Impact of acceptance factor z and hyperparameter τ

The idea of acceptance factor $z_{s,p}$ in Eq.(6) is to further decrease the score of popular items, on the assumption that popular items which are constantly ignored are likely to be irrelevant as opposed to unknown to users [He *et al*., 2016]. As popular items are nodes in the graph with several connections, further penalizing them with $z_{s,p}$ should improve scalability when combined with the weight threshold ϕ, and should reduce the dominance of popular items in recommendation lists. Essentially, setting higher values for τ results in faster forgetting for popular items with low acceptance.

We tested values for $\tau \in [0, 1]$ in steps of 0.1, after first tuning β. Note that $\tau = 0.0$ ignores $z_{s,p}$ and only score nodes based on $x_{s,p}$ and $y_{s,p}$. Results of these experiments in accuracy, measured by HR@20 and DCG@20 are presented in Tables 5 and 6.

From Tables 5 and 6, we see that for all datasets, apart from LFM-1K and BOOK, there is a value for $\tau > 0$ that increases accuracy with statistical significance, and similarly to β, the effect of τ depends on the dataset. For dataset BOOK there are minor increases in DCG@20 that are not significant, while for LFM-1K there are no changes. Datasets ML-1M, ML-10M, PLC-STR and ELEC benefit from small values of τ. Since parameter β already penalizes popular items and increases the recommendation of long-tail items, the impact of τ is very limited on these datasets.

Conversely, dataset GLOBO is highly affected by the hyperparameter and benefits from higher values of it. These

Table 5. Impact of parameter τ on IGSI$_{\hat{\pi}^t}$ with our proposed forgetting technique for datasets ML-1M, ML-10M, PLC-PL and PLC-STR. Column τ refers to the tested values for hyperparameter τ, while HR@20 and DCG@20 represents the HitRate@20 and Discounted Cumulative Gain, respectively, grouped by dataset. Results for $\tau = 0$ refers to the best values for β obtained in Tables 3 and 4. Values highlighted in bold indicate the highest value for τ that is superior in accuracy with statistical significance in comparison to lower values.

	ML-1M		**ML-10M**		**PLC-PL**		**PLC-STR**	
τ	**HR@20**	**DCG@20**	**HR@20**	**DCG@20**	**HR@20**	**DCG@20**	**HR@20**	**DCG@20**
0	0.237	0.104	0.225	0.100	0.396	0.260	0.592	0.432
0.1	**0.238**	**0.105**	0.226	0.101	**0.397**	**0.261**	0.592	0.432
0.2	0.236	0.104	**0.227**	**0.102**	0.397	0.260	0.592	0.432
0.3	0.236	0.104	0.226	0.101	0.397	0.260	**0.593**	**0.433**
0.4	0.236	0.104	0.226	0.101	0.397	0.260	0.593	0.433
0.5	0.236	0.104	0.226	0.101	0.397	0.260	0.593	0.433
0.6	0.235	0.104	0.226	0.101	0.397	0.260	0.593	0.433
0.7	0.235	0.104	0.226	0.101	0.397	0.260	0.593	0.433
0.8	0.235	0.104	0.225	0.101	0.397	0.260	0.593	0.433
0.9	0.235	0.104	0.225	0.101	0.396	0.260	0.593	0.433
1.0	0.235	0.103	0.225	0.101	0.396	0.260	0.593	0.433

Table 6. Impact of parameter τ on IGSI$_{\hat{\pi}^t}$ with our proposed forgetting technique for datasets LFM-1K, BOOK, ELEC and GLOBO. Column τ refers to the tested values for hyperparameter τ, while HR@20 and DCG@20 represents the HitRate@20 and Discounted Cumulative Gain, respectively, grouped by dataset. Results for $\tau = 0$ refers to the best values for β obtained in Tables 3 and 4. Values highlighted in bold indicate the highest value for τ that is superior in accuracy with statistical significance in comparison to lower values.

	LFM-1K		**BOOK**		**ELEC**		**GLOBO**	
τ	**HR@20**	**DCG@20**	**HR@20**	**DCG@20**	**HR@20**	**DCG@20**	**HR@20**	**DCG@20**
0	**0.182**	**0.141**	**0.662**	**0.452**	0.269	0.215	0.591	0.294
0.1	0.182	0.141	0.662	0.455	0.269	0.215	0.610	0.307
0.2	0.182	0.141	0.662	0.455	**0.270**	**0.215**	0.615	0.312
0.3	0.182	0.141	0.662	0.455	0.269	0.215	0.619	0.314
0.4	0.182	0.141	0.662	0.455	0.269	0.215	0.621	0.314
0.5	0.182	0.141	0.662	0.455	0.268	0.214	0.624	0.318
0.6	0.182	0.141	0.662	0.456	0.268	0.214	0.625	0.318
0.7	0.182	0.141	0.662	0.456	0.268	0.214	0.626	0.318
0.8	0.182	0.141	0.662	0.456	0.268	0.214	**0.627**	**0.320**
0.9	0.182	0.141	0.662	0.456	0.268	0.214	0.627	0.320
1.0	0.182	0.141	0.662	0.456	0.268	0.214	0.628	0.320

improvements are related to the characteristics of the dataset. GLOBO is a news domain dataset, with a dynamic set of items, where concept drift is prevalent. New items are constantly added to the set, and these items have a very short lifespan. Also, users' preferences adjust to current events. Thus, faster forgetting of popular items is beneficial. On the other hand, datasets from domains characterized by stable relations, e.g., movies, benefit from slower forgetting, as these relations remain relevant even if not reinforced by new data.

In subsequent experiments, we set $\tau = 0.0$ for LFM-1K and BOOK, $\tau = 0.1$ for ML-1M and PLC-PL, $\tau = 0.2$ for ML-10M and ELEC, $\tau = 0.3$ for PLC-STR and $\tau = 0.8$ for GLOBO.

6.3 Impact of weight threshold ϕ

Next, based on the optimal values for β and τ, we evaluate the impact of weight threshold ϕ on the performance of IGSI$_{\hat{\pi}^t}$ with the proposed forgetting technique, where ϕ removes obsolete edges from the graph based on parameters α and x. Results are presented in Tables 7 and 8, with accuracy measured through HR@20. We also report the average number of edges in the graph.

From Tables 7 and 8, when comparing its first row, which is IGSI$_{\hat{\pi}^t}$ without forgetting, to increasing values of x, we can see that LND is able to considerably reduce the number of edges on the graph, thus reducing memory requirements, while also obtaining major improvements in accuracy for ML-1M, ML-10M and GLOBO, and slight improvements for PLC-STR. For LFM-1K, PLC-PL, BOOK and ELEC, it reduces the size of the model without decreases in accuracy. This behavior relates to the degree distribution of the graphs, presented in Table 2: the accumulation of obsolete information is more substantial on datasets with higher average degree, and forgetting plays a greater role on these scenarios.

We note two main observations from these results. First, it is beneficial to set a threshold to remove edges, as accuracy stagnates, or starts to decrease, after a certain value of ϕ, which suggests an accumulation of obsolete (or unnecessary) information on the graph. The removal of these connections impacts directly accuracy and scalability.

Second, besides parameter x, the rate of forgetting is affected by hyperparameter τ. Lower values for τ reduces the discrepancies between scores of items, i.e., it increases the values for acceptance factor $z_{s,p}$ in Eq.(6) and thus it reduces its effect, which is to penalize popular items. On the other hand, higher values for τ increases the effect of $z_{s,p}$, which results in faster forgetting. Hence, we note here that if the objective would be to obtain the smallest possible model, this could be achieved by setting $\tau = 1.0$.

Table 7. Impact of parameter x on IGSI$_{\hat{\pi}^t}$ with local neighborhood decay forgetting for datasets ML-1M, ML-10M, PLC-PL and PLC-STR. Column x refers to the tested values for hyperparameter x, while HR@20 and $|\mathcal{E}|$ represents the HitRate@20 and the number of edges on the underlying graph, respectively, grouped by dataset. The first row of the results refer to IGSI$_{\hat{\pi}^t}$ without forgetting. Values highlighted in bold indicate the smallest value for x that is superior in accuracy with statistical significance in comparison to lower values.

	ML-1M		**ML-10M**		**PLC-PL**		**PLC-STR**	
x	**HR@20**	$\|\mathcal{E}\|$	**HR@20**	$\|\mathcal{E}\|$	**HR@20**	$\|\mathcal{E}\|$	**HR@20**	$\|\mathcal{E}\|$
-	0.234	61,939	0.198	224,781	0.396	28,644	0.590	185,060
5	0.238	21,288	0.221	53,231	0.395	26,338	0.591	121,592
10	0.242	27,832	0.226	66,854	**0.398**	**27,646**	0.592	140,580
15	**0.244**	**32,864**	0.230	77,676	0.397	28,196	**0.593**	**151,412**
20	0.244	36,793	0.232	86,481	0.397	28,430	0.593	158,420
25	0.243	39,434	**0.233**	**94,019**	0.397	28,539	0.593	163,266

Table 8. Impact of parameter x on IGSI$_{\hat{\pi}^t}$ with local neighborhood decay forgetting for datasets LFM-1K, BOOK, ELEC and GLOBO. Column x refers to the tested values for hyperparameter x, while HR@20 and $|\mathcal{E}|$ represents the HitRate@20 and the number of edges on the underlying graph, respectively, grouped by dataset. The first row of the results refer to IGSI$_{\hat{\pi}^t}$ without forgetting. Values highlighted in bold indicate the smallest value for x that is superior in accuracy with statistical significance in comparison to lower values.

	LFM-1K		**BOOK**		**ELEC**		**GLOBO**	
x	**HR@20**	$\|\mathcal{E}\|$	**HR@20**	$\|\mathcal{E}\|$	**HR@20**	$\|\mathcal{E}\|$	**HR@20**	$\|\mathcal{E}\|$
-	0.182	1,019,439	0.661	784,214	0.268	36,401	0.456	143,262
5	**0.182**	**1,003,435**	0.660	670,291	0.268	29,715	0.614	63,163
10	0.182	1,015,856	**0.661**	**713,342**	**0.269**	**32,022**	0.622	73,394
15	0.182	1,018,308	0.661	732,023	0.269	33,315	0.625	79,956
20	0.182	1,019,023	0.661	743,387	0.269	34,037	**0.626**	**84,744**
25	0.182	1,019,262	0.662	750,876	0.269	34,539	0.626	88,451

For subsequent experiments, we select values for x that results in the smallest possible models without losses in accuracy. To that end, we assess the statistical significance of the results. We increase x until there are no significant difference in comparison to smaller values. We set $x = 5$ for LFM-1K, $x = 10$ for PLC-PL, BOOK and ELEC, $x = 15$ for ML-1M and PLC-STR, $x = 20$ for GLOBO and $x = 25$ for ML-10M.

6.4 Results

Tables 9 and 10 present overall results for IGSI$_{\hat{\pi}^t}$ without forgetting, IGSI$_{\hat{\pi}^t}$ with baseline forgetting techniques and IGSI$_{\hat{\pi}^t}$ with our proposed technique. Accuracy is measured through HR@20 and DCG@20, diversity through ILD, and scalability measured through average update and recommendation times and number of edges in the graph. The best results are highlighted in bold. We also show the relative performance gain of every approach over IGSI$_{\hat{\pi}^t}$ without forgetting.

The first main observation is that the impact on the results is highly related to the datasets. For the two movie-domain datasets, ML-1M and ML-10M, the proposed forgetting technique LND obtained the best results in accuracy. Considering dataset ML-1M, the application of the remaining ones generally decreased accuracy, with the exception of the time-decay approach. We note, however, that all forgetting approaches significantly decrease the average number of edges in the graph, which consequently impacts the recommendation time. For ML-10M, the application of forgetting overall improved both accuracy and scalability. With the exception of recency queue approach, the remaining ones improved accuracy with smaller models.

For the three music-domain datasets, LFM-1K, PLC-PL, PLC-STR, the impact of forgetting varies with the approach. Considering LFM-1K, we see that forgetting reduces the number of edges, at the expense of accuracy, specially for both time related approaches. The exception is LND, which is capable of reducing the number of edges, while maintaining its original accuracy.

For PLC-PL, the application of forgetting does not yield relevant results. We argue that this relates to the characteristics of the dataset. It is composed of about 100,000 interactions spread across close to four years of activity. Thus, the rate of data is very low, affecting the behavior of all forgetting techniques. For the sliding window, it is difficult to set an ideal size, as a smaller one does not include sufficient information, while a higher one does not actually perform any forgetting. The time-decay approach suffers from the same issue. For the recency queue and LND, there are not sufficient interactions with the same items several times in order for these approaches to detect what has become irrelevant. This is seen when we compare accuracy and number of edges: these approaches are unable to perceive obsolete edges, hence not providing any change in accuracy when compared to no forgetting.

PLC-STR, on the other hand, is composed of about 1,400,000 interactions also spread across close to four years of activity. While the rate of data is not high, it has more information and we can better observe the impact of forgetting. LND manages to reduce the number of edges in the model without losses in accuracy. The same goes for the time-decay approach. Sliding window still suffer from the same issue present in PLC-PL, where it is difficult to set its appropriate size.

Considering the e-commerce domain, the impact of forgetting is very similar on both its datasets BOOK and ELEC. While recency queue fails to detect obsolete edges, LND removes it without losses in accuracy. We can also see some

Table 9. Overall results for all techniques grouped by datasets ML-1M, ML-10M, PLC-PL and PLC-STR, including relative performance gain of every approach over $IGSI_{\hat{\pi}^t}$ without forgetting. Best results are highlighted in bold.

Technique	HR	DCG	ILD	$\lvert\mathcal{E}\rvert$	Time(ms)		
					Upd.	Rec.	Tot.
ML-1M							
$IGSI_{\hat{\pi}^t}$	0.233	0.103	0.788	109,901	0.2	5.3	5.5
LND	**0.248 (+6.4%)**	**0.110 (+6.8%)**	0.817 (+3.7%)	52,735 (-52%)	0.5	4.6	5.1 (-7.3%)
recency_queue	0.231 (-0.9%)	0.102 (-1.0%)	0.771 (-2.2%)	68,373 (-37.8%)	0.7	4.9	5.6 (+1.8%)
time_decay	0.238 (+2.1%)	0.104 (+1.0%)	0.799 (+1.4%)	52,040 (-52.6%)	0.2	4.6	4.8 (-12.7%)
sliding_window	0.204 (-12.4%)	0.090 (-12.6%)	**0.821 (+4.2%)**	**46,087 (-58.1%)**	0.2	4.4	**4.6 (-16.4%)**
ML-10M							
$IGSI_{\hat{\pi}^t}$	0.192	0.087	0.837	476,448	0.5	5.9	6.4
LND	**0.222 (+15.6%)**	**0.104 (+19.5%)**	0.860 (+2.7%)	169,457 (-64.6%)	0.5	5.0	**5.5 (-14.1%)**
recency_queue	0.194 (+1.0%)	0.087 (0.0%)	0.820 (-2.0%)	217,519 (-54.3%)	2.8	5.8	8.6 (+34.4%)
time_decay	0.203 (+5.7%)	0.093 (+6.9%)	0.840 (+0.4%)	**118,758 (-75.1%)**	0.6	5.1	5.7 (-10.9%)
sliding_window	0.212 (+10.4%)	0.098 (+12.6%)	**0.862 (+3.0%)**	127,683 (-73.2%)	0.3	5.3	5.8 (-9.4%)
PLC-PL							
$IGSI_{\hat{\pi}^t}$	0.402	0.249	0.838	56,922	0.02	4.2	4.2
LND	**0.403 (+0.2%)**	**0.250 (+0.4%)**	0.840 (+0.2%)	53,749 (-5.6%)	0.06	4.1	4.2 (0.0%)
recency_queue	0.402 (0.0%)	0.249 (0.0%)	0.838 (0.0%)	56,922 (0.0%)	0.3	4.3	4.6 (+9.5%)
time_decay	0.384 (-4.5%)	0.240 (-3.6%)	0.835 (-0.4%)	32,841 (-42.3%)	0.07	4.1	4.2 (0.0%)
sliding_window	0.355 (-11.7%)	0.221 (-11.2%)	**0.853 (+1.8%)**	**25,365 (-55.4%)**	0.05	4.0	**4.1 (-2.4%)**
PLC-STR							
$IGSI_{\hat{\pi}^t}$	0.597	0.430	0.784	355,926	0.1	4.7	4.8
LND	**0.601 (+0.7%)**	**0.432 (+0.5%)**	0.788 (+0.4%)	276,382 (-22.3%)	0.1	4.6	4.7 (-2.1%)
recency_queue	0.595 (-0.3%)	0.429 (-0.2%)	0.786 (+0.3%)	322,699 (-9.3%)	1.0	4.9	5.9 (+22.9%)
time_decay	0.595 (-0.3%)	0.427 (-0.7%)	0.784 (0.0%)	207,944 (-41.6%)	0.1	4.6	4.7 (-2.1%)
sliding_window	0.570 (-4.5%)	0.416 (-3.3%)	**0.800 (+2.0%)**	**118,604 (-66.7%)**	0.1	4.5	**4.6 (-4.2%)**

limitations of time-based approaches in some scenarios. In this case, long-term connections that are not co-occurring again in the stream, and infrequent items that are seldom interacted by any user are abruptly forgotten, even thought they are not obsolete. On the other hand, LND only removes connections from edges when newer ones are available, or are updated again.

Finally, for the news-domain dataset GLOBO, forgetting has a crucial role. A news-domain dataset is the exact opposite of what we have in the aforementioned ones. The set of items goes through constant churn, new items are continuously added to it and old ones are dropped. In other words, in this domain items have a very short shelf life, and become old very quickly [Das *et al*., 2007]. Also, the user-item matrix is very sparse, and are prone to popularity factors, such as breaking news or popular topics [Lommatzsch and Albayrak, 2015]. Hence, news-domain datasets seems to benefit from forgetting, and we confirm that on the obtained results. All techniques considerably reduce the number of edges from its graphs, without compromising accuracy in the case of recency queue and time-decay. LND and sliding window, on the other hand, significantly increased accuracy, with LND obtaining the best results while also reducing the number of edges by about half of the original. An important consideration is the effectiveness of sliding window, as its advantages align with the properties of the news-domain, and thus improved accuracy, with the smallest model by a wide margin.

Table 10. Overall results for all techniques grouped by datasets LFM-1K, BOOK, ELEC and GLOBO, including relative performance gain of every approach over $IGSI_{\tilde{\pi}^t}$ without forgetting. Best results are highlighted in bold.

Technique	HR	DCG	ILD	$\lvert\mathcal{E}\rvert$	Time(ms)		
					Upd.	Rec.	Tot.
LFM-1K							
$IGSI_{\tilde{\pi}^t}$	0.230	0.179	0.836	1,990,079	0.1	5.8	5.9
LND	**0.232 (+0.9%)**	**0.180 (+0.6%)**	0.836 (0.0%)	1,699,049 (-14.6%)	0.1	5.5	**5.6 (-5.1%)**
recency_queue	0.220 (-4.3%)	0.172 (-3.9%)	0.834 (-0.2%)	1,581,937 (-20.5%)	15.6	5.6	21.2 (+259.3%)
time_decay	0.221 (-3.9%)	0.174 (-2.8%)	0.830 (-64.1%)	**1,068,103 (-46.3%)**	0.1	5.6	5.7 (-3.4%)
sliding_window	0.202 (-12.2%)	0.156 (-12.8%)	**0.846 (+1.2%)**	1,162,618 (-41.6%)	0.1	5.5	**5.6 (-5.1%)**
BOOK							
$IGSI_{\tilde{\pi}^t}$	0.647	0.411	0.736	1,790,105	0.05	4.7	4.7
LND	**0.652 (+0.8%)**	**0.417 (+1.5%)**	**0.745 (+1.2%)**	1,554,975 (-13.1%)	0.1	4.5	4.6 (-2.1%)
recency_queue	0.647 (0.0%)	0.411 (0.0%)	0.736 (0.0%)	1,783,292 (-0.4%)	30.1	4.7	34.8 (+640.4%)
time_decay	0.621 (-4.0%)	0.394 (-4.1%)	0.686 (-6.8%)	587,480 (-67.2%)	0.1	2.6	**2.7 (-42.6%)**
sliding_window	0.559 (-7.4%)	0.346 (-15.8%)	0.680 (-7.6%)	**259,018 (-85.5%)**	0.1	3.0	3.1 (-34.0%)
ELEC							
$IGSI_{\tilde{\pi}^t}$	0.237	0.187	0.980	102,098	0.01	3.9	3.9
LND	**0.238 (+0.4%)**	**0.188 (+0.5%)**	0.980 (0.0%)	85,720 (-16.0%)	0.01	3.8	3.8 (-2.6%)
recency_queue	0.236 (-0.4%)	0.187 (0.0%)	0.980 (0.0%)	101,789 (-0.3%)	1.0	4.1	5.1 (+30.8%)
time_decay	0.230 (-3.0%)	0.183 (-2.1%)	0.981 (+0.1%)	35,569 (-65.2%)	0.01	3.2	3.2 (-17.9%)
sliding_window	0.209 (-11.8%)	0.166 (-11.2%)	**0.984 (+0.4%)**	**17,143 (-83.2%)**	0.01	2.9	**2.9 (-25.6%)**
GLOBO							
$IGSI_{\tilde{\pi}^t}$	0.443	0.228	0.938	340,327	0.2	5.0	5.2
LND	**0.628 (+41.8%)**	**0.327 (+43.4%)**	**0.941 (+0.3%)**	195,755 (-42.5%)	0.2	4.6	4.8 (-7.7%)
recency_queue	0.442 (-0.2%)	0.228 (0.0%)	0.938 (0.0%)	211,555 (-37.8%)	0.2	5.0	5.2 (0.0%)
time_decay	0.443 (0.0%)	0.227 (-0.4%)	0.938 (0.0%)	150,933 (-55.7%)	0.2	4.9	5.1 (-1.9%)
sliding_window	0.592 (+33.6%)	0.297 (+30.3%)	0.940 (+0.2%)	**20,189 (-94.1%)**	0.1	4.4	**4.5 (-13.5%)**

6.5 Comparisons between forgetting techniques

We now present more general observations based on all obtained results, considering the evaluated metrics and the competing forgetting techniques. In terms of recommendation quality, measured by HR@20 and DCG, the proposed forgetting technique, LND, obtained the best results. When comparing LND to $IGSI_{\tilde{\pi}^t}$ without forgetting, we note that LND always reduces the number of edges from the models without compromising accuracy. In fact, accuracy always increased, even if the improvements were minor. The effect of forgetting largely depends on the properties of the underlying graph, which can be seen in Table 2. Datasets with higher density and average degrees, ML-1M, ML-10M, PLC-STR and GLOBO were better affected by forgetting. This highlights the difficulty of deploying forgetting and removing obsolete information from already very sparse graphs.

When assessing diversity, we can see that increases in ILD generally result in decreases in accuracy. This is justified by the evaluated metric, ILD, which measures the average distance between items in the recommendation list, by cosine distance. Thus, these are in a sense conflicting objectives, as $IGSI_{\tilde{\pi}^t}$ is an item-based recommendation approach. Ideally, the desirable result would be to increase both accuracy and diversity, or at least one in a controlled manner. This is achieved with LND, where parameter β specifically allows for an increase in diversity. Also, when comparing LND to no forgetting, we see that we never have the decrease of either accuracy or diversity, and that in some cases there was an increase in both. We note that while the remaining forgetting

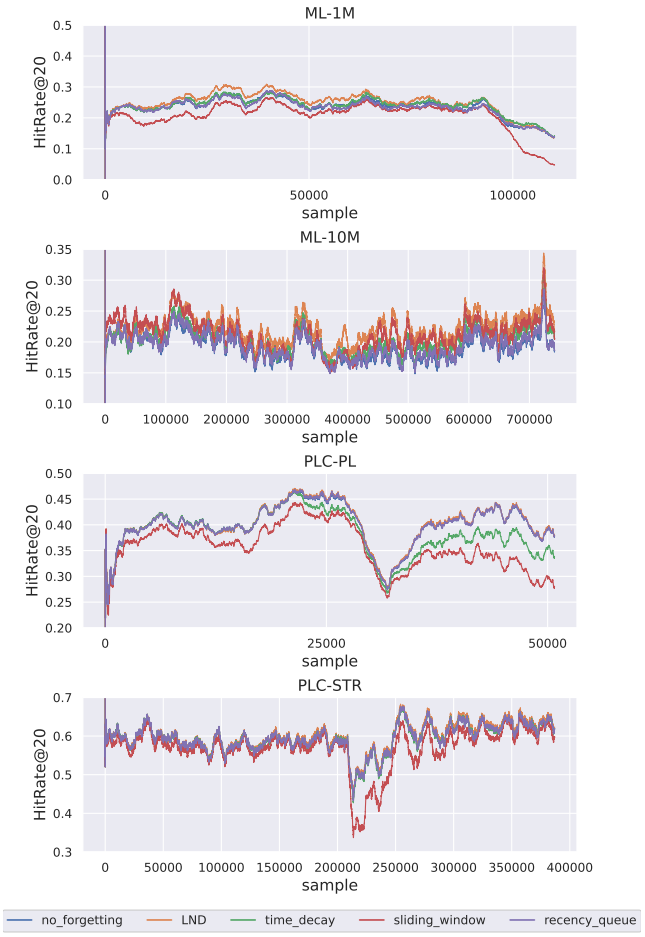


Figure 3. Evolution of HitRate@20 with window size $n = 5000$ for datasets ML-1M, ML-10M, PLC-PL and PLC-STR.

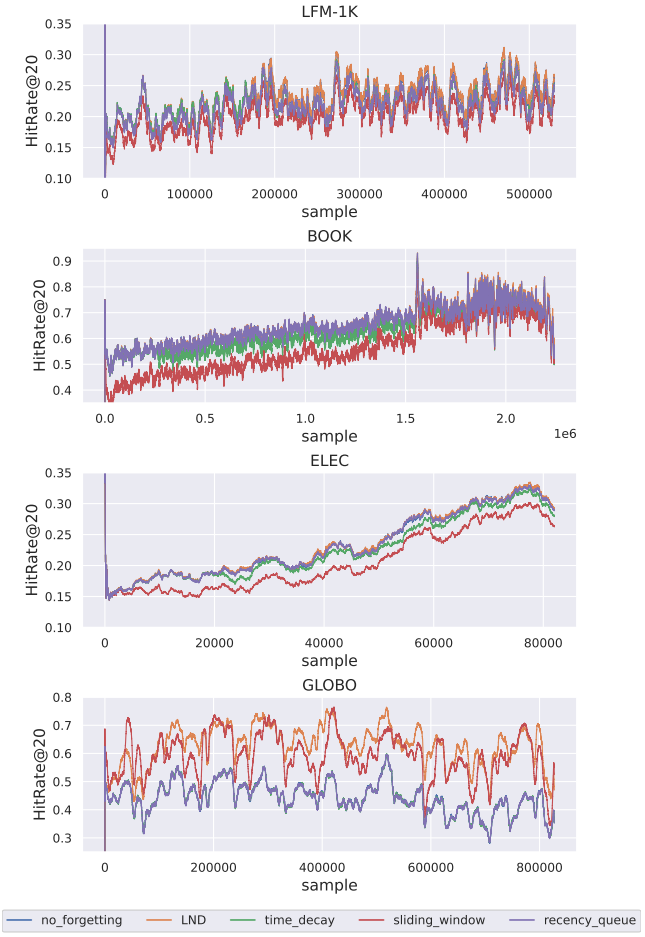


Figure 4. Evolution of HitRate@20 with window size $n = 5000$ for datasets LFM-1K, BOOK, ELEC and GLOBO.

techniques did obtained greater diversity in some datasets, they only did so at the expense of accuracy.

Besides parameter β, the effect of LND on diversity is clearly correlated with the degree distribution of the graph. Since nodes are scored based on popularity, measured by both indegree and outdegree factors (Eqs. (7) and (8)), the increases in diversity are more substantial in graphs with higher density and higher average degree. This can be seen when comparing the differences in diversity for ML-1M and ML-10 with BOOK and ELEC. For BOOK and ELEC, the two datasets with smallest average degree, there are no differences in diversity. This is because a large portion of nodes have low indegree and outdegree, and thus the options for random walk exploration are very limited. Consequently, LND is unable to increase the diversity. On the other hand, for ML-1M and ML-10M, which have the two highest average degree of all datasets, LND improved diversity. As the average degree distribution is higher, resulting in a larger number of neighbors, LND increases the likelihood of exploring less popular nodes in the random walk simulation, which impacts both diversity and accuracy.

Considering update time, it changes based on the technique at use, and generally it incurs only a minor addition compared to no forgetting. The exception would be the recency queue technique, which presents high update time for datasets with a large number of items. We note that recency queue and LND are applied for each incoming observation in the stream, while time-decay and sliding window are only deployed based on the predefined time window hyperparameter, hence they have lower average update times of all forgetting 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. Hence, recommendation times are improved, and the trade-off between accuracy and scalability can be controlled parametrically based on weight threshold ϕ. As the recommendation procedure accounts for the majority of processing time per sample, the total processing time per sample (update + recommendation times) is reduced with all techniques but recency queue. Even if there is a minor overhead in update time with forgetting, its deployment is beneficial in the end as the reduction in recommendation times reduces the overall processing time, which is associated with the size of the graph. As is the case for accuracy and diversity, the influence of LND on the number of edges and recommendation time depends on the density of the dataset, where the reduction is more considerable in denser ones.

6.6 Statistical analysis

We note that these results are averages from the entire simulated stream process. To better visualize the behavior of all algorithms, we plot moving averages of size 5000 for HR@20, the number of edges and processing time per sample over time for all datasets. These are presented in Figures 3

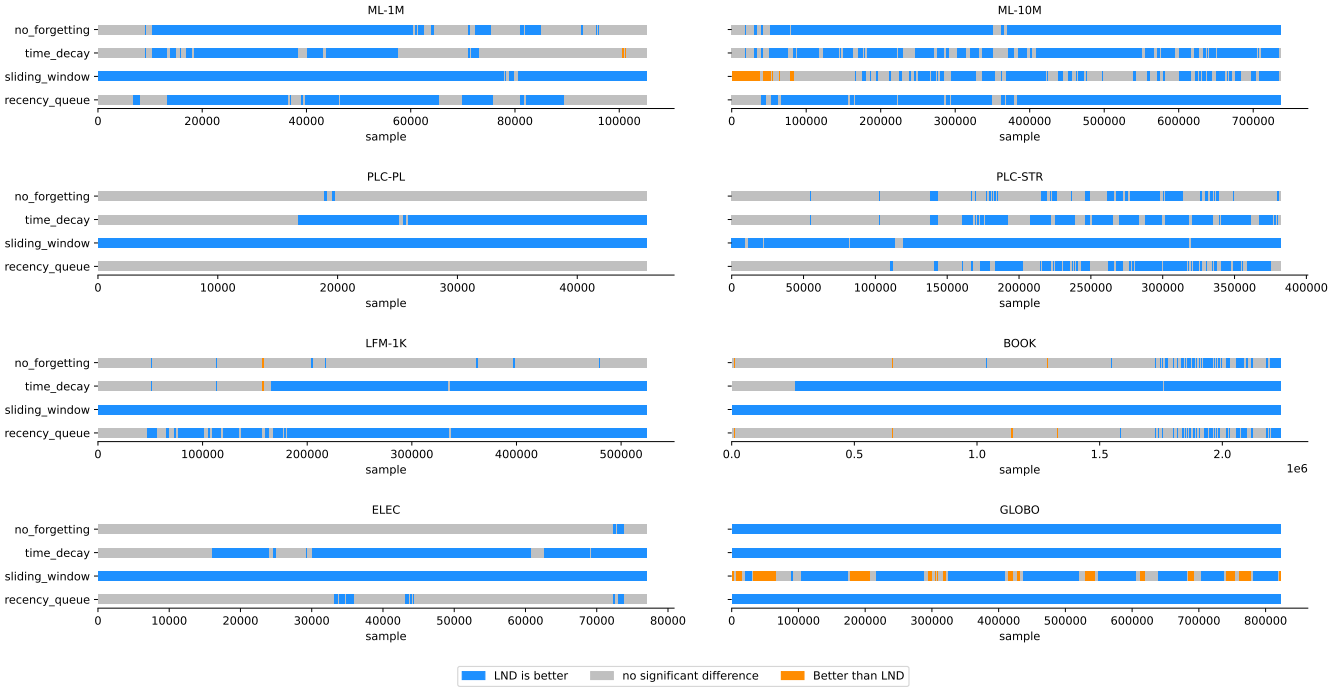


Figure 5. McNemar's pairwise test results between LND and other forgetting approaches for all datasets with a confidence level of 99%.

and 4, Figures 6 and 7, and Figures 8 and 9, respectively.

From Figures 3 and 4, 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 they are not reinforced by new data. The recency queue technique presents similar accuracy to IGSI$_{\hat{\pi}^t}$ without forgetting as it punishes infrequent items that do not seem to directly reflect the current interests of users. Conversely, IGSI$_{\hat{\pi}^t}$ without forgetting generally outperforms both time-decay and sliding window, except for ML-10M and GLOBO. These techniques do not distinguish the relevance of edges and simply forget information over time based on intervals.

We assess the statistical significance of these results through McNemar's tests between our proposal LND and every other competing approach, for each dataset using sliding windows with size $n = 5000$, the same size set for moving averages presented in Figures 3 and 4. These assessments are shown in Figure 5.

The statistical significance tests show that LND is rarely outperformed by any other baseline. Some are clearly visible in Figures 3 and 4. For example, LND is outperformed by sliding window in some parts of the prequential process for dataset GLOBO. Conversely, it generally outperforms the remaining approaches throughout the prequential process, confirming some of the visible superiority in Figures 3 and 4, and there is mostly no statistical significant differences where there is overlap between accuracies. Even if there are instances without statistically significant differences in accuracy, there are still other advantages in its deployment, such as potential increases in diversity and reduction in the size of the underlying models.

In Figures 6 and 7 we can see how the underlying graph grows based on the deployed forgetting technique. For IGSI$_{\hat{\pi}^t}$ without forgetting, the average number of edges tends to grow proportionally to new samples over time. For LND and recency queue, following an initial decrease resulting from the deletion of obsolete edges, it grows steadily, as the insertions and deletions occur continuously, adding new information and removing outdated ones based on recency. For time-decay and sliding window, there is usually a major decrease based on the time interval, followed by increases proportional to the new data. A limitation of these techniques is illustrated on the final samples of ML-1M, where the rate of forgetting is faster than the arrival of new data. This could be corrected by using windows with size based on the number of interactions instead of time intervals.

Finally, Figures 8 and 9 shows the effect of removing edges on the average processing time per sample. For IGSI$_{\hat{\pi}^t}$ without forgetting, average time increases with the size of the model. With forgetting, the reduction of edges decreases average recommendation time, as the complexity of the recommendation procedure depends on the number of random walk samples and the size of neighborhoods. Although there is a minor increase in average update time, as shown in Tables 9 and 10, with exception of recency queue for which is exceptionally high and omitted from some plots, forgetting eventually reduces the average processing time per sample. The reduction is directly related to the technique and dataset.

Overall, extending IGSI$_{\hat{\pi}^t}$ with LND improved accuracy, diversity and average processing time per sample. We present in Appendix B the properties of underlying graphs per dataset *after* the application of LND. As forgetting prunes edges from the graph, LND impacts density and the degree distribution, more significantly the outdegree distribution. Datasets with a greater reduction in these properties are the ones that were better affected by LND in accuracy, diversity and processing time per sample, e.g., ML-1M, ML-10M, and GLOBO. We argue that a balance between the relevant evaluated metrics can be obtained through parameter β, which increases accuracy

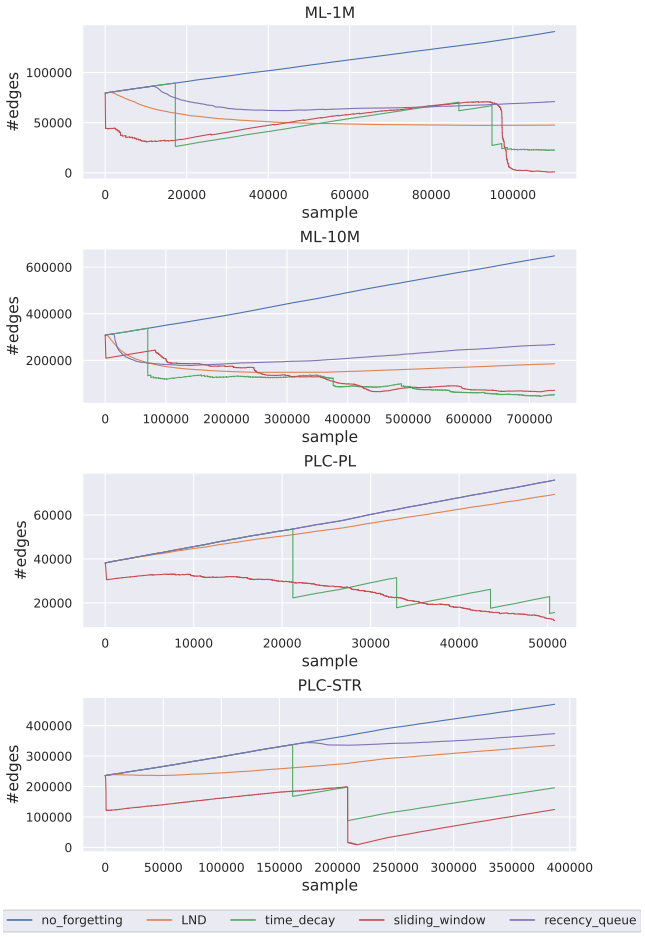


Figure 6. Evolution of the number of edges with window size $n = 5000$ for datasets ML-1M, ML-10M, PLC-PL and PLC-STR.

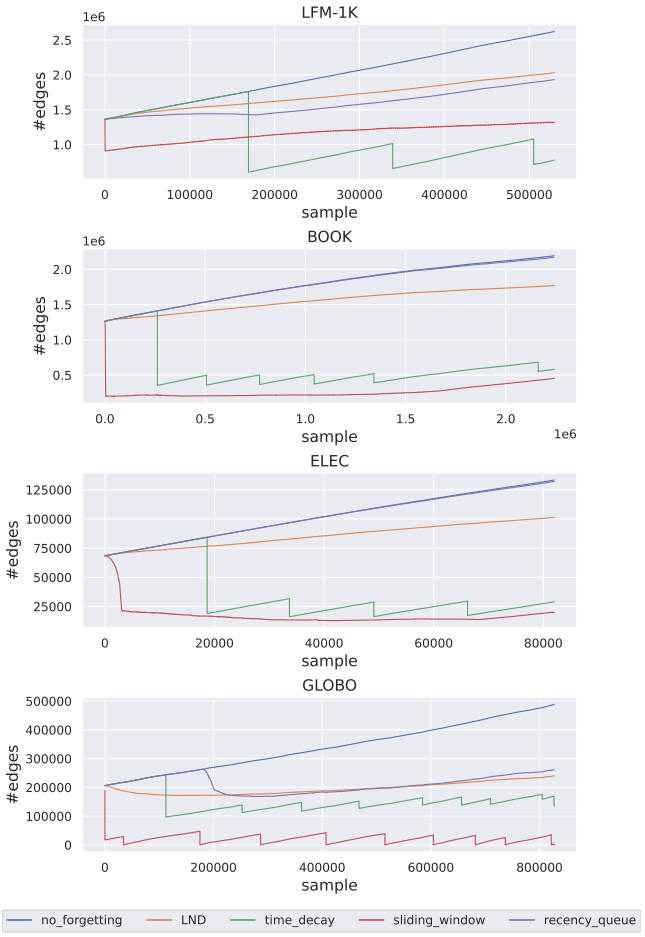


Figure 7. Evolution of the number of edges with window size $n = 5000$ for datasets LFM-1K, BOOK, ELEC and GLOBO.

and diversity, τ that may increase accuracy, and the weight threshold ϕ that allows controlling the growth of the graph, which tends to decrease the recommendation time, at the expense of a minor increase in update time. Such decrease may allow the usage of a higher number of random walk samples, which would likely improve accuracy.

7 Discussion

In the previous section, we reported results that suggest the potential of our proposed forgetting technique, and highlight the benefits of forgetting in general when deployed on data streams scenarios. We now discuss some shortcomings of our approach and the experimental analysis, and outline possible improvements.

Although we have analyzed each LND hyperparameter individually, we have not included an isolated study comparing the effect of structural information inferred through random walk sampling, to simple recency-based forgetting. Experiments that evaluate recency-based forgetting are reported in a preliminary study [Schmitt and Spinosa, 2022a]. Comparing those results to the present ones, we note that the main advantage of random walk-based forgetting is *scalability*.

While simple recency-based forgetting only fades outgoing edges from the users' last interacted item, our random walk-based forgetting technique considers all outgoing edges from the most recent items interacted by the active user. Edges are faded if they are not included in paths that lead to the item of the current interaction, as inferred from the random walk samples. Thus, more edges may be considered irrelevant and faded in the process. Fading more edges leads to smaller models, which then results in faster recommendation times and scalability gains.

Regarding evaluation, although we have included several datasets from different domains that are commonly used to evaluate SBRS [Al-Ghossein *et al.*, 2021], and obtained positive results, we note that some datasets, such as ML-1M, ML-10M, ELEC and BOOK, do not fully represent the dynamics of concept drift, as these span several years of activity, and users' interactions may be spread over time.

Conversely, we highlight the substantial improvements obtained with forgetting on dataset GLOBO, particularly with our proposed approach, LND. GLOBO is a dynamic dataset that accurately represents data streams with concept drift: new items are continuously included in the catalog, have a very short shelf life and quickly lose relevance, affecting users' preferences, as they adjust to current events [Lommatzsch and Albayrak, 2015]. These results emphasize that adaptive learning in data streams requires both incremental learning and mechanisms to forget obsolete data.

Another consideration is that in online systems, new users regularly arrive to use the system for the first time, and new items are constantly added to the catalog. How to generate relevant recommendations to these new users, or recommend

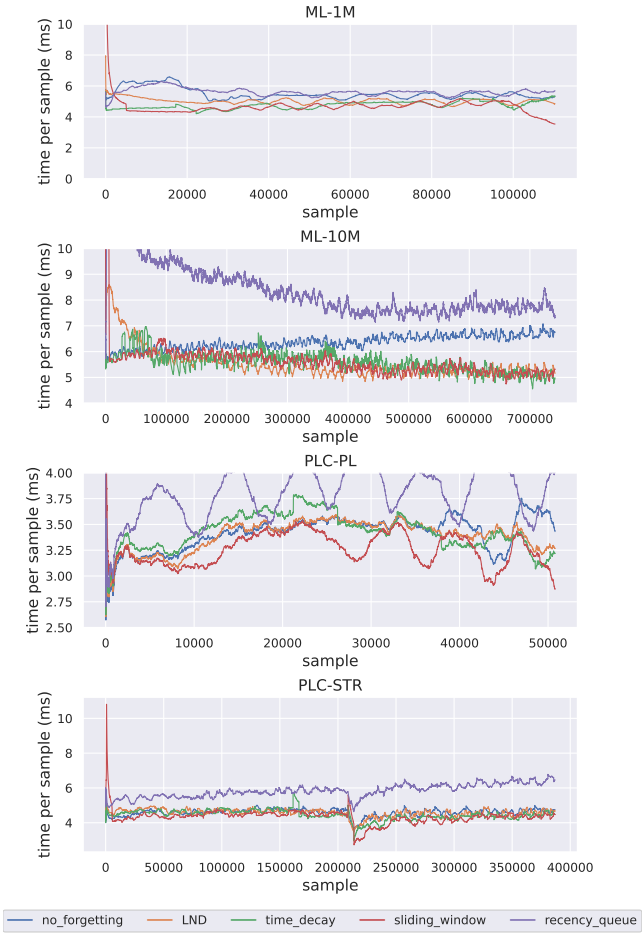


Figure 8. Evolution of processing time per sample with window size $n = 5000$ for datasets ML-1M, ML-10M, PLC-PL and PLC-STR.

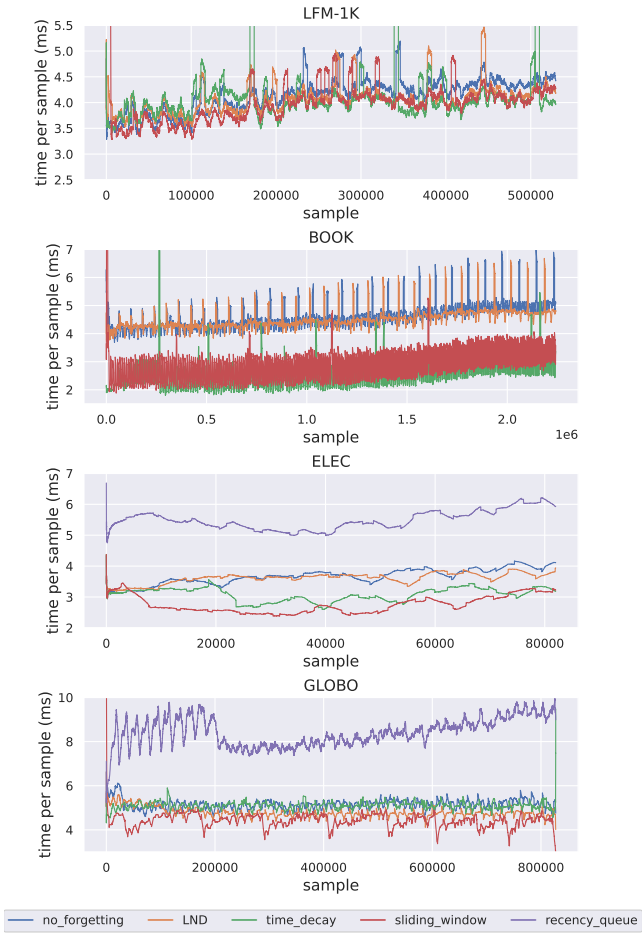


Figure 9. Evolution of processing time per sample with window size $n = 5000$ for datasets LFM-1K, BOOK, ELEC and GLOBO.

new items, given the limited availability of information about them is a challenging problem, known as *cold-start* [Schein *et al.*, 2002]. While incremental approaches are able to incorporate new users and items into a model, and prequential evaluation is well suited to assess the impact of cold-start, most of the works on SBRS typically do not focus on this issue [Al-Ghossein *et al.*, 2021]. We note that addressing the cold-start problem is not the focus of the present work, although we intend to investigate it in future work.

Finally, we have not explicitly considered users' sessions information in our evaluation. For instance, GLOBO dataset includes users' sessions information that could be used in the future to better assess drifts in users' preferences. Time information could be further exploited by implementing models as both SBRS and session-aware [Quadrana *et al.*, 2018], by explicitly considering the start and end of users' sessions. In this way, short-term interest would be based on the most current user session, while long-term interests would consider all user sessions.

8 Conclusion

In this work, we proposed a gradual forgetting technique for graph-based methods that locally forgets information based on recency, popularity and structural information in order to overcome concerns resulting from the accumulation of obsolete information in incremental neighborhood-based models. The proposed technique, Local Neighborhood Decay (LND), decreases the importance of neighborhood of items for every incoming observation to emphasize more recent ones, based on the aforementioned factors. LND reuses random walk samples originally generated for recommendation purposes in order to infer structural information from the graph, and determines relevant connections between items based on such information. Items are also scored based on popularity and acceptance factors to boost the inclusion of items from the long-tail in the recommendation lists and increase diversity.

Parameter β balances the impact of neighborhood degree distribution to increase the likelihood of retaining less popular items, which allows increases in diversity, while parameter τ controls forgetting for popular items, allowing fading them faster. Scalability is controlled by a weight threshold ϕ that prunes edges that are not reinforced, and thus deemed as obsolete.

We evaluated our proposal by extending a recent graph-based approach $\text{IGSI}_{\hat{\pi}^t}$. Experiments showed that the proposed technique was able to reduce the size of the graph, reducing average recommendation times and improving scalability, while also improving accuracy when compared to both $\text{IGSI}_{\hat{\pi}^t}$ without forgetting and with baseline forgetting techniques. Increases in diversity can also be obtained by adjusting parameter β appropriately. In future work we intend to explore automatic parameter adjustment and evaluate algorithms on scenarios where concept drift and cold start are

prevalent.

Declarations

Authors' Contributions

Murilo F. L. Schmitt: Conceptualization, Investigation, Methodology, Validation, Data Curation, Software, Writing - Original Draft. **Eduardo J. Spinosa**: Conceptualization, Methodology, Validation, Supervision, Writing - Original Draft.

Competing interests

The authors declare that they have no competing interests.

Funding

This work was partially funded by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES).

Availability of data and materials

Data can be made available upon request.

References

Aggarwal, C. C. *et al*. (2016). *Recommender systems: the textbook*, volume 1. Springer. Available at: `https://pzs.dstu.dp.ua/DataMining/recom/bibl/1aggarwal_c_c_recommender_systems_the_textbook.pdf`.

Al-Ghossein, M., Abdessalem, T., and Barre, A. (2021). A survey on stream-based recommender systems. *ACM Computing Surveys (CSUR)*, 54(5):1–36. DOI: 10.1145/3453443.

Andersen, R., Borgs, C., Chayes, J., Hopcroft, J., Mirrokni, V., and Teng, S.-H. (2008). Local computation of pagerank contributions. *Internet Mathematics*, 5(1-2):23–45. DOI: 10.1007/978-3-540-77004-6_12.

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370. DOI: 10.1023/A:1021240730564.

Castells, P., Hurley, N. J., and Vargas, S. (2015). Novelty and diversity in recommender systems. In *Recommender systems handbook*, pages 881–918. Springer, Boston, MA. DOI: 10.1007/978-1-4899-7637-6_26.

Celma, Ò. (2010). Music recommendation. In *Music recommendation and discovery*, pages 43–85. Springer, Boston, MA. DOI: 10.1007/978-3-642-13287-2.

Celma, Ò. and Cano, P. (2008). From hits to niches? or how popular artists can bias music recommendation and discovery. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–8. DOI: 10.1145/1722149.1722154.

Christoffel, F., Paudel, B., Newell, C., and Bernstein, A. (2015). Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 163–170. DOI: 10.1145/2792838.2800180.

Cooper, C., Lee, S. H., Radzik, T., and Siantos, Y. (2014). Random walks in recommender systems: exact computation and simulations. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 811–816. DOI: 10.1145/2567948.2579244.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. DOI: 10.1145/1864708.1864721.

Das, A. S., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. DOI: 10.1145/1242572.1242610.

de Souza Pereira Moreira, G., Jannach, D., and Da Cunha, A. M. (2019). Contextual hybrid session-based news recommendation with recurrent neural networks. *IEEE Access*, 7:169185–169203. DOI: 10.1109/ACCESS.2019.2954957.

Ding, Y. and Li, X. (2005). Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 485–492. DOI: 10.1145/1099554.1099689.

Domingos, P. M. and Hulten, G. (2001). Catching up with the data: Research issues in mining data streams. In *DMKD*. Available at: `https://www.cs.cornell.edu/johannes/papers/dmkd2001-papers/p8_domingos.pdf`.

Frigó, E., Pálovics, R., Kelen, D., Kocsis, L., and Benczúr, A. (2017). Online ranking prediction in non-stationary environments. In *Proceedings of the 1st Workshop on Temporal Reasoning in Recommender Systems co-located with RecSys '17 (RecTemp '17).*, pages 28–34, CEUR-WS.org. Available at: `https://ceur-ws.org/Vol-1922/paper6.pdf`.

Gama, J., Sebastião, R., and Rodrigues, P. P. (2009). Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 329–338. DOI: 10.1145/1557019.1557060.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37. DOI: 10.1145/2523813.

Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19. DOI: 10.1145/2827872.

Haveliwala, T. H. (2003). Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering*, 15(4):784–796. DOI: 10.1145/511446.511513.

He, X., Zhang, H., Kan, M.-Y., and Chua, T.-S. (2016). Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558. DOI: 10.1145/2911451.2911489.

Jannach, D., Lerche, L., and Zanker, M. (2018). *Recommend-*

ing Based on Implicit Feedback, pages 510–569. Springer International Publishing, Cham. DOI: 10.1007/978-3-319-90092-6_14.

Jeh, G. and Widom, J. (2002). Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. DOI: 10.1145/775047.775126.

Jugovac, M., Jannach, D., and Karimi, M. (2018). Streamingrec: a framework for benchmarking stream-based news recommenders. In *Proceedings of the 12th ACM conference on recommender systems*, pages 269–273. DOI: 10.1145/3240323.3240384.

Koren, Y. (2009). Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456. DOI: 10.1145/1557019.1557072.

Koren, Y., Rendle, S., and Bell, R. (2022). Advances in collaborative filtering. *Recommender systems handbook*, pages 91–142. DOI: 10.1007/978-1-0716-2197-4_3.

Koychev, I. (2000). Gradual forgetting for adaptation to concept drift. In *Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning*. Available at: `https://research.uni-sofia.bg/bitstream/10506/57/1/ECAI2000_WSTR.pdf`.

Liu, N. N., Zhao, M., Xiang, E., and Yang, Q. (2010). Online evolutionary collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 95–102. DOI: 10.1145/1864708.1864729.

Lommatzsch, A. and Albayrak, S. (2015). Real-time recommendations for user-item streams. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1039–1046. DOI: 10.1145/2695664.2695678.

Lu, J., Wu, D., Mao, M., Wang, W., and Zhang, G. (2015). Recommender system application developments. *Decis. Support Syst.*, 74(C):12–32. DOI: 10.1016/j.dss.2015.03.008.

Matuszyk, P., Vinagre, J., Spiliopoulou, M., Jorge, A. M., and Gama, J. (2015). Forgetting methods for incremental matrix factorization in recommender systems. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 947–953. DOI: 10.1145/2695664.2695820.

Matuszyk, P., Vinagre, J., Spiliopoulou, M., Jorge, A. M., and Gama, J. (2018). Forgetting techniques for stream-based matrix factorization in recommender systems. *Knowledge and Information Systems*, 55(2):275–304. DOI: 10.1007/s10115-017-1091-8.

McAuley, J., Pandey, R., and Leskovec, J. (2015). Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794. DOI: 10.1145/2783258.2783381.

Miranda, C. and Jorge, A. M. (2009). Item-based and user-based incremental collaborative filtering for web recommendations. In *Portuguese Conference on Artificial Intelligence*, pages 673–684. Springer. DOI: 10.1007/978-3-642-04686-5.

Nasraoui, O., Cerwinske, J., Rojas, C., and Gonzalez, F. (2007). Performance of recommendation systems in dynamic streaming environments. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 569–574. SIAM. DOI: 10.1137/1.9781611972771.63.

Nikolakopoulos, A. N., Ning, X., Desrosiers, C., and Karypis, G. (2022). *Trust Your Neighbors: A Comprehensive Survey of Neighborhood-Based Methods for Recommender Systems*, pages 39–89. Springer US, New York, NY. DOI: 10.1007/978-1-0716-2197-4_2.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab. Available at: `https://api.semanticscholar.org/CorpusID:1508503`.

Quadrana, M., Cremonesi, P., and Jannach, D. (2018). Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4):1–36. DOI: 10.1145/3190616.

Ricci, F., Rokach, L., and Shapira, B. (2022). *Recommender Systems: Techniques, Applications, and Challenges*, pages 1–35. Springer US, New York, NY. DOI: 10.1007/978-1-0716-2197-4_1.

Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. DOI: 10.1145/564376.564421.

Schmitt, M. F. L. and Spinosa, E. J. (2020). Incremental graph of sequential interactions for online recommendation with implicit feedback. In *3rd Workshop on Online Recommender Systems and User Modeling*. Available at: `https://ceur-ws.org/Vol-2715/paper10.pdf`.

Schmitt, M. F. L. and Spinosa, E. J. (2022a). Forgetting on evolving graphs for accurate and diverse stream-based recommendation. In *Anais do X Symposium on Knowledge Discovery, Mining and Learning*, pages 138–145, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/kdmile.2022.227804.

Schmitt, M. F. L. and Spinosa, E. J. (2022b). 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. DOI: 10.1007/s11257-021-09315-6.

Siddiqui, Z. F., Tiakas, E., Symeonidis, P., Spiliopoulou, M., and Manolopoulos, Y. (2014). xstreams: Recommending items to users with time-evolving preferences. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, pages 1–12. DOI: 10.1145/2611040.2611051.

Smyth, B. and McClave, P. (2001). Similarity vs. diversity. In *International conference on case-based reasoning*, pages 347–361. Springer. DOI: 10.1007/3-540-44593-5$_2$5.

Symeonidis, P., Kirjackaja, L., and Zanker, M. (2020). Session-aware news recommendations using random walks on time-evolving heterogeneous information networks. *User Modeling and User-Adapted Interaction*, pages 1–29. DOI: 10.1007/s11257-020-09261-9.

Tabassum, S., Veloso, B., and Gama, J. (2020). On fast and scalable recurring link's prediction in evolving multigraph streams. *Network Science*, 8(S1):S65–S81. DOI: 10.1017/nws.2019.64.

Takács, G., Pilászy, I., Németh, B., and Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10:623–656. Available at: `https://jmlr.org/papers/v10/takacs09a.html`.

Veloso, B., Malheiro, B., Burguillo, J. C., and Foss, J. (2017). Personalised fading for stream data. In *Proceedings of the Symposium on Applied Computing*, pages 870–872. DOI: 10.1145/3019612.3019868.

Verachtert, R., Jeunen, O., and Goethals, B. (2023). Scheduling on a budget: Avoiding stale recommendations with timely updates. *Machine Learning with Applications*, 11:100455. DOI: 10.1016/j.mlwa.2023.100455.

Verachtert, R., Michiels, L., and Goethals, B. (2022). Are we forgetting something? correctly evaluate a recommender system with an optimal training window. In *PERSPECTIVES 2022: Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop 2022, September 22, 2022, Seatle, USA*, volume 3228, pages 1–15. Available at: `https://ceur-ws.org/Vol-3228/paper1.pdf`.

Vinagre, J. and Jorge, A. M. (2012). Forgetting mechanisms for scalable collaborative filtering. *Journal of the Brazilian Computer Society*, 18(4):271–282. DOI: 10.1007/s13173-012-0077-3.

Vinagre, J., Jorge, A. M., and Gama, J. (2014). Fast incremental matrix factorization for recommendation with positive-only feedback. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 459–470. Springer. DOI: 10.1007/978-3-319-08786-3_41.

Vinagre, J., Jorge, A. M., and Gama, J. (2015). Collaborative filtering with recency-based negative feedback. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 963–965. DOI: 10.1145/2695664.2695998.

Vinagre, J., Jorge, A. M., Rocha, C., and Gama, J. (2021). Statistically robust evaluation of stream-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 33(7):2971–2982. DOI: 10.1109/TKDE.2019.2960216.

Viniski, A. D., Barddal, J. P., de Souza Britto Jr, A., and de Campos, H. V. A. (2023). Incremental specialized and specialized-generalized matrix factorization models based on adaptive learning rate optimizers. *Neurocomputing*, 552:126515. DOI: 10.1016/j.neucom.2023.126515.

Viniski, A. D., Barddal, J. P., de Souza Britto Jr, A., Enembreck, F., and de Campos, H. V. A. (2021). A case study of batch and incremental recommender systems in supermarket data under concept drifts and cold start. *Expert Systems with Applications*, 176:114890. DOI: 10.1016/j.eswa.2021.114890.

Wang, X. and Brewster, C. (2024). Forgetting in knowledge graph based recommender systems. In *Proceedings of the 13th International Conference on Data Science, Technology and Applications*, pages 309–317. DOI: 10.5220/0012757300003756.

A Appendix - Optimal hyperparameters

As in Section 5, hyperparameters for baselines were optimized based on grid search following the same methodology used to optimize the hyperparameters of our proposal. Optimal hyperparameters values used for each approach grouped by dataset are reported in Table 11.

Table 11. Optimal hyperparameters per algorithm grouped by dataset.

Dataset	Technique	Param.
ML-1M	**LND**	$\beta = 0.7, \tau = 0.1, \phi = \alpha^{15}$
	recency_queue	$\phi = \alpha^{5}$
	time_decay	$\omega = 30$
	sliding_window	$\omega = 90$
ML-10M	**LND**	$\beta = 0.6, \tau = 0.2, \phi = \alpha^{25}$
	recency_queue	$\phi = \alpha^{5}$
	time_decay	$\omega = 14$
	sliding_window	$\omega = 365$
PLC-PL	**LND**	$\beta = 1.0, \tau = 0.1, \phi = \alpha^{10}$
	recency_queue	$\phi = \alpha^{10}$
	time_decay	$\omega = 210$
	sliding_window	$\omega = 365$
PLC-STR	**LND**	$\beta = 0.6, \tau = 0.3, \phi = \alpha^{15}$
	recency_queue	$\phi = \alpha^{10}$
	time_decay	$\omega = 90$
	sliding_window	$\omega = 180$
LFM-1K	**LND**	$\beta = 1.0, \tau = 0.0, \phi = \alpha^{5}$
	recency_queue	$\phi = \alpha^{0}$
	time_decay	$\omega = 60$
	sliding_window	$\omega = 180$
BOOK	**LND**	$\beta = 0.9, \tau = 0.0, \phi = \alpha^{10}$
	recency_queue	$\phi = \alpha^{5}$
	time_decay	$\omega = 365$
	sliding_window	$\omega = 365$
ELEC	**LND**	$\beta = 0.6, \tau = 0.2, \phi = \alpha^{10}$
	recency_queue	$\phi = \alpha^{5}$
	time_decay	$\omega = 365$
	sliding_window	$\omega = 365$
GLOBO	**LND**	$\beta = 0.5, \tau = 0.8, \phi = \alpha^{20}$
	recency_queue	$\phi = \alpha^{10}$
	time_decay	$\omega = 1$
	sliding_window	$\omega = 1$

B Appendix - Graph's properties with LND

We present in Table 12 the properties of the underlying graphs per dataset after the application of LND. Comparing to the properties of the original graphs presented in Table 2, LND impacts density and the degree distribution, as it removes edges from the graph. Reductions are more significant for the outdegree distribution, since LND when necessary prunes outgoing edges from a single source. Datasets with a greater reduction in these properties are the ones that were better affected by LND in accuracy, diversity and processing time per sample, e.g., ML-1M, ML-10M, PLC-STR and GLOBO.

Table 12. Graph's properties per dataset with LND. The reported properties are number of nodes ($|\mathcal{V}|$), number of edges ($|\mathcal{E}|$), density, average degree, average weighted degree, minimum weighted degree (Min), maximum weighted degree (Max) and number of low weighted degree nodes (#Low). For each dataset, cells from column (Min, Max, #Low) are split into two rows, where the first refer to indegree node information, and the second refers to outdegree node information.

Dataset	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	Density	Avg. degree	Avg. weighted degree	(Min, Max, #Low)
ML-1M	3,232	47,540	0.455	14.70	19.21	(0, 490.51, 2307) (0, 422.38, 1913)
ML-10M	8,721	185,777	0.244	21.30	25.65	(0, 1118.99, 7056) (0, 609.92, 5584)
LFM	399,171	2,030,271	0.001	5.08	8.77	(0, 1352.94, 312460) (0, 1313.98, 303074)
PLC-PL	26,117	73,028	0.010	2.79	3.68	(0, 132.0, 19374) (0, 148.0, 19389)
PLC-STR	40,213	335,214	0.020	8.33	27.38	(0, 11655.94, 32141) (0, 11778.08, 31469)
BOOK	734,918	1,770,607	0.0003	2.40	3.97	(0, 555.06, 613482) (0, 422.83, 598751)
ELEC	60,842	101,324	0.003	1.66	2.03	(0, 673.71, 51709) (0, 314.41, 49552)
GLOBO	35,644	240,216	0.018	6.73	9.95	(0, 3894.54, 31556) (0, 573.17, 28853)