Learning Multi-granularity Dynamic Network Representations for Social Recommendation

Peng Liu*, Lemei Zhang, and Jon Atle Gulla

Department of Computer Science, NTNU, Trondheim, Norway
{peng.liu,lemei.zhang,jon.atle.gulla}@ntnu.no

Abstract. With the rapid proliferation of online social networks, personalized social recommendation has become an important means to help people discover useful information over time. However, the cold-start issue and the special properties of social networks, such as rich temporal dynamics, heterogeneous and complex structures with millions of nodes, render the most commonly used recommendation approaches (e.g. Collaborative Filtering) inefficient. In this paper, we propose a novel multi-granularity dynamic network embedding (m-DNE) model for the social recommendation which is capable of recommending relevant users and interested items. In order to support online recommendation, we construct a heterogeneous user-item (HUI) network and incrementally maintain it as the social network evolves. m-DNE jointly captures the temporal semantic effects, social relationships and user behavior sequential patterns in a unified way by embedding the HUI network into a shared low dimensional space. Meanwhile, multi-granularity proximities which include the second-order proximity and the community-aware high-order proximity of nodes, are introduced to learn more informative and robust network representations. Then, with an efficient search method, we use the encoded representation of temporal contexts to generate recommendations. Experimental results on several real large-scale datasets show its advantages over other state-of-the-art methods.

Keywords: Social recommendation, Heterogeneous social network, Network embedding, Temporal context, Community detection

1 Introduction

In the last few decades, the rapid development of Web 2.0 and smart mobile devices have resulted in the dramatic proliferation of online social networks. According to Twitter statistics, the number of users is estimated to have surpassed 300 million generating more than 500 million tweets per day[1]. Faced with the abundance of user generated content, a key issue of social networking services is how to help users find their potential friends or interested items that match

* Corresponding author.

[1] https://www.omnicoreagency.com/twitter-statistics/.

the users' preference as much as possible, by making use of both semantic information and social relationships. This is the problem of personalized social recommendation.

Collaborative filtering (CF) has been shown to be an effective approach to recommender systems. It makes predictions about user's interests based on preferences of other users. However, CF is generally designed for bipartite graphs which model interactions between users and items and thus cannot be easily applied over complex heterogeneous social networks. Besides, cold start issue becomes even more severe in online settings as the new users and new items will join in constantly over time. Many approaches [1, 2] have been proposed to alleviate this problem, but they are not designed specifically for online environment.

Recently, network representation learning (NRL) has attracted a considerable amount of interest from various domains, with recommender systems being no exception [3, 4]. The popularization of NRL in recommendation can be mainly attributed to the network embedding techniques which learn low-dimensional vertex representation by modelling vertex co-occurrence in individual user's interaction records, thus capturing the semantic relationships among vertices and boosting recommendation accuracy [4]. Cold start issues can be alleviated through mining the structure and relations among existing and newly arrived nodes. Despite these positive results, we argue that NRL for social recommendations still suffers from the following four challenges: 1) Different from widely used homogeneous networks, heterogeneous network which includes different-typed objects and links, is seldom studied but more commonly seen in real world. Besides, online networks often incorporate millions even billions of nodes and edges in real world, which brings more obstacles in dealing with them. 2) Most real-world networks are intrinsically dynamic with addition/deletion of edges and nodes. Meanwhile, similar as network structure, node attributes also change as new content patterns may emerge and outdated content patterns will fade. 3) So far, most previous network representation methods primarily preserve the local structure and content, such as the first- and second-order proximities of nodes, the global community structure, which is one of the most prominent features, is largely ignored. 4) Considering the online environment and frequently changing velocity of social networks, the scalability and updating complexity of learning algorithms should also play a pivotal role and be seriously reckoned. Recent researches only pay attention to several of the abovementioned challenges while still neglect one or more of them [5–9].

To address the problems raised above, we propose a novel multi-granularity dynamic network embedding (m-DNE) model for online social recommendation. Specifically, we firstly construct a heterogeneous user-item (HUI) network which is incrementally maintained as the social network evolves. Then, a low complexity incremental learning algorithm is applied to embed HUI into low-dimensional representation space with the use of multi-granularity proximity information (second-order and community-aware high-order proximities) of each vertex. Afterwards, an efficient search method and a time-decay mechanism are adopted to conduct recommendation tasks. To the best of our knowledge, we

are the first to jointly model the temporal semantic effects, social relationships and user behavior sequential patterns in a unified way to address the issue of temporal dynamics, cold start and context awareness in an online social recommendation. Our experiments show that the proposed approach is superior to all baselines and state-of-the-art methods in social recommendation tasks.

In this paper, section 2 introduces the related work. In section 3, we define the key concepts and our problem. Section 4,5,6 present our model. We describe the experimental setup and results in section 7 and section 8 concludes the study.

2 Related Work

Social Recommender System In recent years, many studies have demonstrated the success of utilizing rich social network information to improve the recommendation performance [2, 4]. However, these efforts have not considered online updating or incremental processes. In order to capture the evolution of the recommender systems, Agarwal et al. [10] proposed a fast online bilinear factor model to learn item-specific factors through online regression by using a large amount of historical data to initialize the online models and thus reducing the dimensionality of the input features. Diaz-Aviles et al. [11] presented Stream Ranking Matrix Factorization, which utilizes a pairwise approach to matrix factorization for optimizing the personalized ranking of topics and follows a selective sampling strategy to perform incremental model updates based on active learning principles. Huang et al. [12] presented a practical scalable item-based collaborative filtering algorithm, with the characteristics such as robustness to implicit feedback problem. Subbian et al. [13] proposed a probabilistic neighbourhood-based algorithm for performing recommendations with streaming data. The recommendation strategies proposed by [12] and [13] focus on scalability and dynamic pruning in recommender systems. Our proposed framework considers the combination of the heterogeneous characteristics of social networks and graph-based updating schemes in online settings, and thus is substantially different from the above-mentioned systems.

Network Representation Learning Recently, network representation learning which aims to learn low-dimensional node embedding is attracting increasing attentions. DeepWalk [6] models the second-order proximity for node embedding with path sampling, whose complexity is $O(|V|log|V|)$. Node2Vec [14] extends DeepWalk with a controlled path sampling process, which requires $O(|V|log|V|+|V|a^2)$ where a is the average degree of the graph. LINE [5] preserves both first- and second-order proximity with complexity of $O(a|E|)$. Compared with our methods, the above works have a lower or comparable complexity, but they are neither worked for heterogeneous networks nor aware of community structure. Metapath2vec [15] extends the network embedding methods to heterogeneous network by introducing metapath based random walk with the complexity of $O(a|E||V|)$. PTE [16] utilizes labels of words and constructs a large-scale heterogeneous text network to learn predictive embedding vectors for words with com-

plexity of $O(a|E|)$. The above-mentioned approaches can model heterogeneous network but are still not community preserving. There is little work that tries to take into account community structure and dynamic environment. For example, Cavallari et al. [7] proposes a community embedding framework, ComE, which adopt global community structure to optimize node embedding results with relatively lower complexity of $O(|V|+|E|)$ but on homogeneous and static network. In [8], DANE performs network embedding in a dynamic environment also for homogeneous network with barely local structure of nodes, and thus ignores the importance of the high-order proximity. The online complexity of DANE is $O(|V|)$. M-NMF [9] constructs the modularity matrix, then applies non-negative matrix factorization to learn node embedding and community detection together with a higher complexity proportional to $O(|V|^2)$ based on static network. Our work is highly built upon LINE and DeepWalk. The novelty lies in the idea of adopting the network embedding methods into the dynamic environment for online recommendation with comparatively low complexity $O(|V|log(|V|))$ in worst cases. As far as we know, it is the first attempt to improve the representation learning with incorporating the temporal community structure into the dynamic network embedding method.

3 Problem Formulation

In this section, we define the key concepts and present the problem statement of this study before the detailed description of our m-DNE model.

Definition 1. Heterogeneous User-Item (HUI) Network *A heterogeneous user-item network can be represented by $G_{mix} = G_{uu} \cup G_{pp} \cup G_{up}$, which consists of the user-user relationship network $G_{uu} = (\mathcal{U}, \varepsilon_{uu})$, the item-item relationship network $G_{pp} = (\mathcal{P}, \varepsilon_{pp})$ and the user-item interaction network $G_{up} = (\mathcal{U} \cup \mathcal{P}, \varepsilon_{up})$. Among this, $\mathcal{U} = \{u_1, u_2, ...u_n\}$ is the set of users, where u_i is the user profile represented with a three tuple $(uId, \mathcal{L}, \mathcal{D})$, which indicates userID, user social links and a set of items associated with u_i. $\mathcal{P} = \{p_1, p_2, ...p_n\}$ is the set of items, where p_i is the item profile with a five tuple $(iId, \mathcal{M}, \mathcal{H}, \mathcal{W}, \rho)$, representing itemID, named entity, hashtag/category, content, create time respectively. ε_{uu}, ε_{pp} and ε_{up} are the sets of edges, which indicate different relation types.*

Definition 2. Community *A community c is a group of vertices, including both users and items, in G_{mix}, and all vertices can be grouped into $\mathcal{K}$ communities $\mathcal{C} = \{c_1, c_2, ..., c_{\mathcal{K}}\}$. The communities can be overlapping, which is to say each vertex $v \in \mathcal{U} \cup \mathcal{P}$, can belong to different c to different degree.*

Finally, we formally define the problem investigated in our work. Given a time-stamped heterogeneous user-item network, we aim to provide online social recommendations stated as follows.

Problem 1 (Online Social Recommendation) *Given a heterogeneous user-item network G_{mix} at timestamp t and a querying user $u \in \mathcal{U}$, the task is to generate a ranked list of user or item recommendations that u would be interested in.*

4 Heterogeneous User-Item Network

4.1 HUI Network Construction

For notational simplicity, we ignore the time-subscript in this subsection. Given a set of users $\mathcal{U} = \{u_1, u_2, ...u_m\}$ and a set of items $\mathcal{P} = \{p_1, p_2, ...p_n\}$, to integrate the semantic effects, social relationships and the user behavior sequential patterns simultaneously, we construct a heterogeneous user-item network comprising two types of nodes and three types of edges, as shown in Fig. 1. The two types of nodes which consist of user and item nodes are formed by projecting the user set and item set respectively. The three types of edges are defined as follows: 1) Each user node u_i and each item node p_j are connected if user u_i shows an interest on item p_j. In the HUI network, such an edge is indicated by yellow solid lines. The associated item nodes of the user node u_i are denoted as $\mathcal{I}_p(u_i)$, the associated user nodes of the item node p_j are denoted as $\mathcal{I}_u(p_j)$. 2) Two user nodes u_i and u_j are connected with the property of user similarity $sim_u(u_i, u_j)$ if they have a social link, such as follower or followee. In the HUI network, such edge is indicated by grey dash lines. The adjacent user nodes of the user node u_i are denoted as $\mathcal{A}_u(u_i)$. 3) Two item nodes p_i and p_j are connected with the item similarity $sim_p(p_i, p_j)$ if they have a semantic link such as Named Entity or Hashtag. In the HUI network, such edge is indicated by orange dash lines. The adjacent item nodes of the item node p_i are denoted as $\mathcal{A}_p(p_i)$.

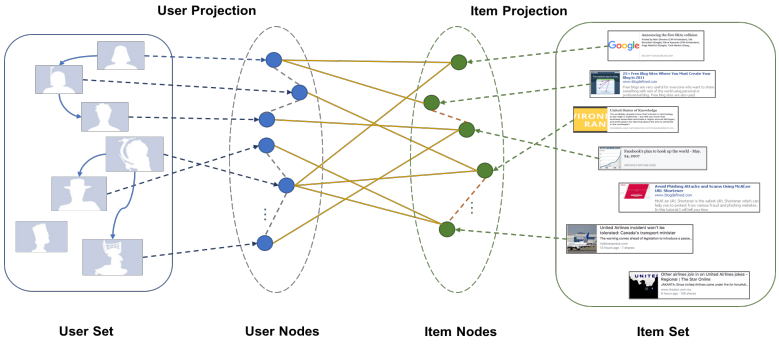


Fig. 1: The Heterogeneous User-Item (HUI) Network.

We assume that $\mathcal{R}_i$ is a r-dimensional vector representing the social links of user u_i, where r is the total number of users, and the k-th dimension of vector $\mathcal{R}_i$ equals 1 only if there is an edge between u_i and u_k, otherwise 0. The user similarity $sim_u(u_i, u_j)$ between user u_i and user u_j can be defined as the cosine similarity between the two vectors. Likewise, we use the cosine similarity $sim_p(p_i, p_j)$ to measure the similarity between two item nodes p_i and p_j.

Directly applying random walk to the HUI network does not work due to different edge types, leading to a challenging problem. To this end, we propose a novel way to capture the different edge type characteristic into the transition probability matrix P, where three parameters α, β, γ with $\alpha + \beta + \gamma = 1$ are

used to respectively control the relative importance of user behavior sequential patterns, social relationships and semantic effects. The values of α, β and γ will be varied depending on different datasets[2].

Definition 3. *A transition probability matrix $P \in \mathbb{R}^{(m+n)\times(m+n)}$ is constructed for the HUI network,*

$$P = \begin{pmatrix} P_u & P_{up} \\ P_{pu} & P_p \end{pmatrix} \tag{1}$$

which comprises four matrix blocks $P_u \in \mathbb{R}^{m\times m}$, $P_{up} \in \mathbb{R}^{m\times n}$, $P_{pu} \in \mathbb{R}^{n\times m}$ and $P_p \in \mathbb{R}^{n\times n}$ respectively representing the transition probabilities of random walks between user nodes, from user nodes to item nodes, from item nodes to user nodes and between item nodes. That is

$$\begin{aligned} P_{i,j} &= Prob(u_j|u_i), \qquad i < m, j < m \\ &= \frac{\beta}{\alpha+\beta} \times \frac{sim_u(u_i,u_j)}{\sum_{u_k\in\mathcal{A}_u(u_i)} sim_u(u_i,u_k)},\ if\ u_j \in \mathcal{A}_u(u_i),\ otherwise\ 0. \end{aligned} \tag{2}$$

$$\begin{aligned} P_{i,m+j} &= Prob(p_j|u_i), \qquad i < m, j < n \\ &= \frac{\alpha}{\alpha+\beta} \times \frac{w_{ij}}{\sum_{p_k\in\mathcal{I}_p(u_i)} w_{ik}},\ if\ p_j \in \mathcal{I}_p(u_i),\ otherwise\ 0. \end{aligned} \tag{3}$$

$$\begin{aligned} P_{m+i,j} &= Prob(u_j|p_i), \qquad i < n, j < m \\ &= \frac{\alpha}{\alpha+\gamma} \times \frac{w_{ji}}{\sum_{u_k\in\mathcal{I}_u(p_i)} w_{ki}},\ if\ u_j \in \mathcal{I}_u(p_i),\ otherwise\ 0. \end{aligned} \tag{4}$$

$$\begin{aligned} P_{m+i,m+j} &= Prob(p_j|p_i), \qquad i < n, j < n \\ &= \frac{\gamma}{\alpha+\gamma} \times \frac{sim_p(p_i,p_j)}{\sum_{p_k\in\mathcal{A}_p(p_i)} sim_p(p_i,p_k)},\ if\ p_j \in \mathcal{A}_p(p_i),\ otherwise\ 0. \end{aligned} \tag{5}$$

In the above definition, to incorporate user bias in our algorithm, we introduce w_{ij} which denotes the rating score that the user u_i assigns to item p_j, and it has different rating scales. For example, in the movie recommendation case, w_{ij} might correspond to an explicit rating given by user u_i to movie p_j or, in the case of twitter/music recommendation, w_{ij} is implicitly derived from user's interaction patterns, e.g., how many times user u_i has clicked/listened item p_j.

4.2 HUI Network Update

Assume at timestamp t, the current HUI network $G_{mix,t} = (\mathcal{V}_t, \varepsilon_t) = (\mathcal{U}_t, \varepsilon_{uu,t}, \mathcal{P}_t, \varepsilon_{pp,t}, \varepsilon_{up,t})$ contains the user node set $\mathcal{U}_t$, item node set $\mathcal{P}_t$ and their related edge sets $\varepsilon_{uu,t}$, $\varepsilon_{pp,t}$ and $\varepsilon_{up,t}$. Due to the evolution of the network, $\mathcal{U}_t$ and $\mathcal{P}_t$ will contain the sets of the newly attached nodes, denoted as $\Delta\mathcal{U}_t$ and $\Delta\mathcal{P}_t$

[2] In our experiments, we set $\alpha = \beta = \gamma = 1/3$ by grid-search over $\{1/9, 3/9, 4/9, 7/9\}$ which achieves the best recommendation performance for all datasets.

respectively, while there exists another subsets of $\mathcal{U}_t$ and $\mathcal{P}_t$ containing the nodes that have user or item profile changed at the current timestamp, which are denoted as $\Theta\mathcal{U}_t$ and $\Theta\mathcal{P}_t$. Similarly, subsets of $\varepsilon_{uu,t}$, $\varepsilon_{pp,t}$ and $\varepsilon_{up,t}$ contain the newly attached edges, separately denoted as $\Delta\varepsilon_{uu,t}$, $\Delta\varepsilon_{pp,t}$ and $\Delta\varepsilon_{up,t}$, while the edges with changed similarities or rating scores within $\varepsilon_{uu,t}$, $\varepsilon_{pp,t}$ and $\varepsilon_{up,t}$ at timestamp t are denoted as $\Theta\varepsilon_{uu,t}$, $\Theta\varepsilon_{pp,t}$ and $\Theta\varepsilon_{up,t}$.

It is necessary to update the HUI network from timestamp $t-1$ to t according to the evolving nodes ($\Delta\mathcal{U}_t \cup \Theta\mathcal{U}_t, \Delta\mathcal{P}_t \cup \Theta\mathcal{P}_t$) and edges ($\Delta\varepsilon_{uu,t} \cup \Theta\varepsilon_{uu,t}, \Delta\varepsilon_{pp,t} \cup \Theta\varepsilon_{pp,t}, \Delta\varepsilon_{up,t} \cup \Theta\varepsilon_{pp,t}$). This can be achieved by updating the two types of nodes and three types of edges in HUI network. Therefore, the active nodes at timestamp t (denoted as $\widetilde{\mathcal{V}}_t$ and nodes in $\widetilde{\mathcal{V}}_t$ are unique) are defined as follows:

$$\begin{aligned}\widetilde{\mathcal{V}}_t = &\Delta\mathcal{U}_t \cup \Theta\mathcal{U}_t \cup \Delta\mathcal{P}_t \cup \Theta\mathcal{P}_t \cup \{u_i | \exists e_u \in \Delta\varepsilon_{uu,t} \cup \Theta\varepsilon_{uu,t}, e_u = (u_i, u_j)\} \\ &\cup \{p_i | \exists e_p \in \Delta\varepsilon_{pp,t} \cup \Theta\varepsilon_{pp,t}, e_p = (p_i, p_j)\} \\ &\cup \{u_k, p_f | \exists e_{up} \in \Delta\varepsilon_{up,t} \cup \Theta\varepsilon_{up,t}, e_{up} = (u_k, p_f)\}\}\end{aligned} \tag{6}$$

The underlying principle of constructing the network and updating process can be analogous to the case of adopting sliding window schema to manage continuous data streams. The construction process of HUI network are based on the historical records, and the updating course of the network can be conducted only within several timestamps like a certain length sliding window. The worst case happens only when all nodes $\{v_i | v_i \in \mathcal{V}_t\}$ have changed within timestamp t. In such case, the retraining process of the whole HUI network is inevitable.

5 Multi-granularity Dynamic Network Embedding

Inspired by DeepWalk [6] and the idea of modelling document [17] in natural language processing, our model contains three main stages as shown in Fig. 2: heterogeneous random walk, community integration and model learning process, based on which, vertex representations will evolve after incremental learning. Given the length of random walk as h and the total number of random walks as l, the starting step will be performed at each of the active node $\widetilde{\mathcal{V}}_t$ at timestamp t. Based on the updated transition probability matrix P, the random walk with restart on heterogeneous network proposed by [18] is employed to generate possible route sequences for active nodes, denoted as $S = \{s_1, s_2, ..., s_{|\widetilde{\mathcal{V}}_t|}\}$. In the rest part of this section, we will illustrate the last two stages.

5.1 Community Integration

As the analogy between words in the text and vertices in walk sequences, we introduce the idea of processing streaming data in topic models to detect overlapping communities in heterogeneous dynamic networks. Before the introduction of community integration procedure, we make two assumptions on heterogeneous random walk sequences, graph vertices and communities as follows: (1) Each vertex in the HUI network can belong to multiple communities with different

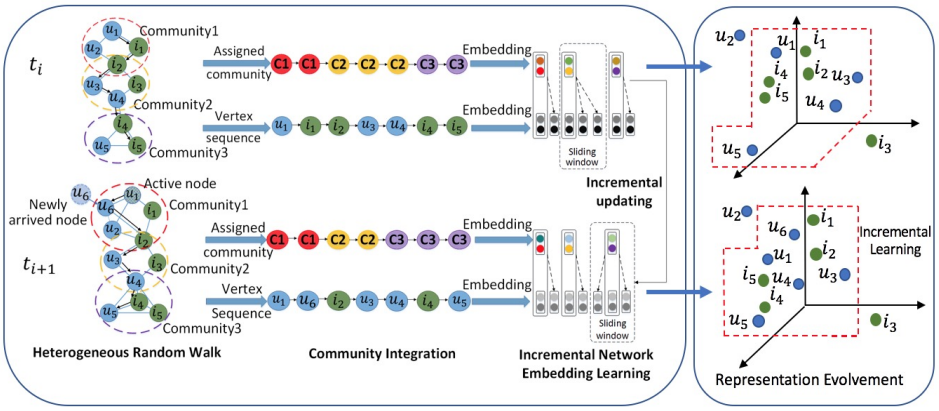


Fig. 2: The m-DNE Model.

preferences of $Pr(c|v)$, and each vertex sequence also owns its community distribution. (2) A vertex in a specific sequence belongs to a distinct community, and the community is determined by the community's distribution over sequences $Pr(c|s)$ and the vertex's distribution over communities $Pr(v|c)$.

With the above assumptions and heterogeneous random walk sequences, we can assign community labels to vertices in particular sequence. More specifically, for a vertex v in a sequence s, we compute the conditional probability of a community c with the following equation:

$$Pr(c|v,s) = \frac{Pr(c,v,s)}{Pr(v,s)} \propto Pr(v|c)Pr(c|s) \tag{7}$$

where $Pr(v|c)$ represents the role of v in community c, and $Pr(c|s)$ represents the community distribution in sequence s.

An ordinary way to estimate $Pr(v|c)$ and $Pr(c|s)$ is to use Gibbs Sampling. But it is not suitable for our updating progress. Thus, instead, we extend the Streaming Gibbs Sampling method proposed in [19] to achieve the conditional probability in our environment. According to the Bayesian Streaming Learning [19], if we fix the community distribution $\boldsymbol{C}^{1:t-1}$ of the previous arrived sequences, then $\boldsymbol{C}^{1:t}$ of the current timestamp can be achieved with $\boldsymbol{C}^{1:t-1}$, and normal Gibbs Sampling on $\boldsymbol{C}^t$. Therefore, the conditional distributions of $Pr(v|c)$ and $Pr(c|s)$ can be estimated as follows:

$$Pr(v|c) = \frac{N^t(v,c) + \beta_l}{\sum_{v' \in \mathcal{V}} N^t(v',c) + |\mathcal{V}|\beta_l}, \quad Pr(c|s) = \frac{N^t(c,s) + \alpha_l}{\sum_{c' \in \mathcal{C}} N^t(c',s) + \mathcal{K}\alpha_l} \tag{8}$$

where $N^t(v,c)$ is the number of times the vertex v assigned to community c at timestamp t and $N^t(c,s)$ is the number of vertices in sequence s are assigned to community c at t. Both $N^t(v,c)$ and $N^t(c,s)$ will be updated dynamically as community assignments change, and for different timestamps. β_l and α_l are smoothing factors in Latent Dirichlet Allocation[20]. With estimated $Pr(v|c)$ and $Pr(c|s)$, we assign a discrete community label c for each vertex v in sequence s.

5.2 Incremental Network Embedding Learning

To initialize the learning process on HUI network $G_{mix} = (\mathcal{V}, \varepsilon)$, given a certain vertex sequence $s = \{v_1, v_2, ..., v_{|s|}\}$, for each vertex v_i and its assigned community c_i, we will learn the representations of both vertices and communities by maximizing the average log probability of predicting context vertices using both v_i and c_i as formalized below:

$$\mathcal{L}(s) = \frac{1}{|s|} \sum_{i=1}^{|s|} \sum_{i-|W| \leq j \leq i+|W|} logPr(v_j|v_i, c_i) \tag{9}$$

where v_j is the context node of the node v_i, and the probability $Pr(v_j|v_i, c_i)$ is defined using the softmax function:

$$Pr(v_j|v_i, c_i) = \frac{exp(\boldsymbol{v}'_j \cdot \overline{\boldsymbol{v}}_i)}{\sum_{v' \in \mathcal{V}} exp(\boldsymbol{v}' \cdot \overline{\boldsymbol{v}}_i)} \tag{10}$$

where $\boldsymbol{v}'_j$ is the context representation of its context node v_j. $\overline{\boldsymbol{v}}_i$ is the average vector representation of the center node v_i and community label c_i defined as $\overline{\boldsymbol{v_i}} = 1/2(\boldsymbol{v_i} + \boldsymbol{c_i})$. In such case, the local context and the global community structure can be incorporated to enhance vertex representation learning. Then subsequently, during incremental learning process at each timestamp $t > 1$, the heterogeneous random walk procedure will start with active node set $\widetilde{\mathcal{V}}_t$ to obtain possible route sequence set S.

To improve the computational efficiency of Eq. (10), in practical environment, we adopt hierarchical softmax[3], a computational efficient approximation of the full softmax in [21]. More precisely, given the average vector representation $\overline{\boldsymbol{v}}_i$ of v_i and c_i for target context v_j, let $L(v_j)$ be the length of its corresponding path, and let $b_n^{v_j} = 0$ when the path to v_j takes the left branch at the n-th layer and $b_n^{v_j} = 1$ otherwise. Then, the hierarchical softmax defines $Pr(v_j|v_i, c_i)$ as follows:

$$Pr(v_j|v_i, c_i) = \prod_{n=2}^{L(v_j)} ([\sigma(\overline{\boldsymbol{v}}_i^T \theta_{n-1}^{v_j})]^{1-b_n^{v_j}} \cdot [1 - \sigma(\overline{\boldsymbol{v}}_i^T \theta_{n-1}^{v_j})]^{b_n^{v_j}}) \tag{11}$$

where $\sigma(z) = \frac{1}{1+exp(-z)}$. All parameters are trained by using the Stochastic Gradient Descent method. To derive how θ is update at each time step, the gradient for $\theta_{n-1}^{v_j}$ is computed as follows:

$$\frac{\partial \mathcal{L}(v_j, n)}{\partial \theta_{n-1}^{v_j}} = [1 - b_n^{v_j} - \sigma(\overline{\boldsymbol{v}}_i^T \theta_{n-1}^{v_j})]\overline{\boldsymbol{v}}_i \tag{12}$$

To derive how the context embedding vectors are updated, the gradient for $\overline{\boldsymbol{v}}_i$ is computed as follows:

$$\frac{\partial \mathcal{L}(v_j, n)}{\partial \overline{\boldsymbol{v}}_i} = [1 - b_n^{v_j} - \sigma(\overline{\boldsymbol{v}}_i^T \theta_{n-1}^{v_j})]\theta_{n-1}^{v_j} \tag{13}$$

[3] The hierarchical softmax needs to evaluate only about $log(|\mathcal{V}|)$ nodes instead of all the $|\mathcal{V}|$ nodes to obtain the probability distribution.

With this derivative, an embedding vector $\boldsymbol{v}_i$ and $\boldsymbol{c}_i$ in the context of node v_j can be updated as follows:

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \eta \sum_{n=2}^{L(v_j)} \frac{\partial \mathcal{L}(v_j, n)}{\partial \overline{\boldsymbol{v}}_i}, \quad \boldsymbol{c}_i \leftarrow \boldsymbol{c}_i + \eta \sum_{n=2}^{L(v_j)} \frac{\partial \mathcal{L}(v_j, n)}{\partial \overline{\boldsymbol{v}}_i} \tag{14}$$

In Algorithm 1, we summarize the learning process using hierarchical softmax for proposed m-DNE model. The algorithm iterates through all possible route sequences and updates the embedding vectors until the procedure converges. In each iteration, given a current node, the algorithm first obtains its embedding vectors and computes its context embedding vector. Based on the derivative above, the binary tree in hierarchical sampling is updated followed by the embedding vector (line 9-14). Given the vector size of d, the leaf nodes number $|V|$, the sequence length $|s|$ within one iteration and window length $|W|$, then the time complexity for an iteration is $\mathcal{O}(d \cdot |W| \cdot |s| \cdot log(|V|))$.

Parallelizability For real-world social networks, the frequency distribution of vertices in random walks follows a power law which results in a long tail of infrequent vertices [6]. Therefore, the updates of vertices' representation will be sparse in nature. Based on this, we adopt the lock-free solutions in the work [22] to parallelize asynchronous stochastic gradient descent (ASGD). Given that our updates are sparse and we do not acquire a lock to access the model shared parameters, ASGD will achieve an optimal rate of convergence.

Algorithm 1: Heterogeneous Softmax Algorithm for m-DNE

Input: Possible route sequence set S, window length $|W|$, embedding vector dimension d, sequence length $|s|$.
Output: The embedding representation $\boldsymbol{v}_i$ of v_i and representation $\boldsymbol{c_i}$ for c_i

1 Initialize the parameters randomly;
2 Shuffle the dataset;
3 **repeat**
4 Sample a route sequence $s = \{v_1, v_2, ..., v_{|s|}\}$ from S;
5 **for** $i = 1$ *to* $|s|$ **do**
6 Set $e \leftarrow 0$;
7 Compute the average representation $\overline{\boldsymbol{v_i}} = 1/2(\boldsymbol{v_i} + \boldsymbol{c_i})$;
8 **for** *each* $v_j \in s[i - |W|, i + |W|]$ **do**
9 **for** $n = 2$ *to* $L(v_j)$ **do**
10 $q \leftarrow \sigma(\overline{\boldsymbol{v}}_i \cdot \theta_{n-1}^{v_j})$;
11 $g \leftarrow \eta \cdot (b_n^{v_j} - 1 - q)$;
12 $e \leftarrow e + g \cdot \theta_{n-1}^{v_j}$;
13 Update $\theta_{n-1}^{v_j} \leftarrow \theta_{n-1}^{v_j} + g \cdot \overline{\boldsymbol{v}}_i$;
14 **end**
15 Update $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \eta \cdot e$;
16 Update $\boldsymbol{c}_i \leftarrow \boldsymbol{c}_i + \eta \cdot e$;
17 **end**
18 **end**
19 **until** *convergence*;

6 Recommendation Using m-DNE

Recommendation procedure can be performed after obtaining the embeddings for each vertex. To recommend top-K friends to a user $u_i \in \mathcal{U}$ with D dimensional

representation vector of $\overrightarrow{u_i} = (x_{i1}, x_{i2}, ..., x_{iD})$ and query time t, we compute the ranking score for user node u_j which does not have a direct link with u_i through the inner product of $\overrightarrow{u_i}$ and $\overrightarrow{u_j}$. Similar procedure can be found when recommending top-K items. Except that, to consider the freshness of the items such as tweets, we bring in the time decay function defined as $f(t_j, \lambda) = e^{-\lambda(t-t_j)}$, where t_j is the publication timestamp of item p_j and λ is employed to adjust the decay rate. Thus, the ranking score of item p_j can be obtained as follows: $S(u_i, p_j, t) = \overrightarrow{u_i} \cdot \overrightarrow{p_j} = f(t_j, \lambda) \sum_{n=1}^{D} x_{in} \cdot z_{jn}$. For computational efficiency, we adopt the Threshold-based Algorithm (TA) [23], which is capable of finding the top-k results by examining the minimum number of users/items.

7 Experiments

7.1 Experimental Setup

Dataset Description For experimental study, we evaluate the proposed m-DNE model on three real-world datasets: Twitter, Last.fm and Flickr. We collected Twitter dataset from January to March 2017 with Twitter API[4], which includes users and their posts, and the Last.fm dataset for 1 month through Last.fm API[5], which contains users and artists. We also adopted Flickr dataset[6] released online with friend relationships, images, and the activities of user comment image. In order to enrich the information about user and image, we extracted the timestamp of user comments, image uploading timestamp and image description with Flickr API[7]. For all datasets, the user-user links are constructed from bi-directional friendships between social network users, user-item links are constructed from the different activities of users (e.g., posting, listening or commenting items), and item-item links are constructed if the two artists/images share the same tag or the two posts have the same hashtag. The statistics of each dataset are summarized in Table 1.

Table 1: Some statistics of the datasets.

Datasets	#Users	#Items	#User-user links	#User-item links	#Item-item links
Twitter	69,830	6,284,665	429,836	69,131,820	30,795,807
Last.fm	41,258	10,361	235,417	11,486,510	1,820,649
Flickr	2,037,538	1,262,978	219,098,660	14,913,164	97,549,330

Baselines We compared our model with five state-of-the-art methods:

- **Weighted Regularized Matrix Factorization (WRMF).** A state-of-the-art offline matrix factorization model introduced by [24] is computed in batch mode, assuming the whole stream is stored and available for training.
- **Stream Ranking Matrix Factorization (RMFX).** It achieves partly online and much quicker updates of matrix factorization introduced in [11].

[4] https://dev.twitter.com/docs.
[5] http://www.last.fm/api/.
[6] http://arnetminer.org/lab-datasets/flickr/flickr.rar.
[7] https://www.flickr.com/services/api/.

- **Metapath2vec (M2V)** [15]. It uses metapath-based random walks on heterogeneous graphs to obtain node representations. Following [15], we employ 5 meaningful meta-paths whose lengths are not longer than 4, "UIU", "UUIU", "UIIU", "UIUIU" and "UUIIU", since long meta-paths are likely to introduce noisy semantics. Here, $'U' = User$ and $'I' = Item$.
- **PTE** [16]. We build three bipartite heterogeneous networks: user-user, user-item and item-item, and retrain it as an unsupervised embedding methods.
- **M-NMF** [9]. It jointly models node and community embedding using non-negative matrix factorization.

Parameter Settings WRMF setup is as follows: $\lambda_{WRMF} = 0.015$, $C = 1$, $epochs = 15$ for all datasets, which corresponds to a regularization parameter, a confidence weight that is put on positive observations, and the number of passes over observed data, respectively [24]. For RMFX, we set regularization constants λ_{RMFX}, learning rate η_0, and a learning rate schedule α equal to 0.1, 0.1, 1 for Twitter, 0.15, 0.05, 1.5 for Last.fm and 0.1, 0.15, 1 for Flickr using grid-search on stream data with cross validation [11]. Moreover, the number of iterations is set to the size of the reservoir. For all the embedding algorithms (metapath2vec, PTE, M-NMF and our model), the embedding dimensionality is set to 128, context window length is set to 8, walk length is set to 40, walks per vertex is set to 30, the neighborhood size is equal to 7 and the size of negative samples is equal to 5 for all datasets. For M-NMF, we followed the same tuning procedure in [9], and we found out that $\alpha = 0.1$ and $\beta = 5$ works at best for Twitter and Last.fm, while $\alpha = 10$ and $\beta = 5$ for Flickr. As for our m-DNE model for three datasets, we also set the dimension of community representation as 128. Following [20], the smoothing factors α_l and β_l are set to 2 and 0.5 respectively. We set decay rate $\lambda = 0.2$ for Twitter and 0.1 for Last.fm and Flickr. The number of communities $\mathcal{K}$ is set to 20 for m-DNE and M-NMF model [9]. We run experiments on Linux machines with eight 3.50GHz Intel Xeon(R) CPUs and 16GB memory.

Evaluation Criteria Given a dataset $\mathcal{D}$ ordered according to time, including user and item profiles, we use the first 50% of $\mathcal{D}$ as historical data pool to train the models, while the rest half data mimics the streaming input called "candidate set". For evaluation, we first randomly select a reference time as "current time" in candidate set. Then, we test our recommendations for the following week starting from reference time, while the data before reference time in candidate set are used to tune the hyper-parameters. However, WRMF and RMFX cannot explicitly handle new user/item introduction during the testing phase. For a fair comparison, all testing sets only cover users/items existing in training set. During evaluation phase, all experimental results are averaged over 10 different runs for reliability, and there is no temporal overlapping between any testing set.

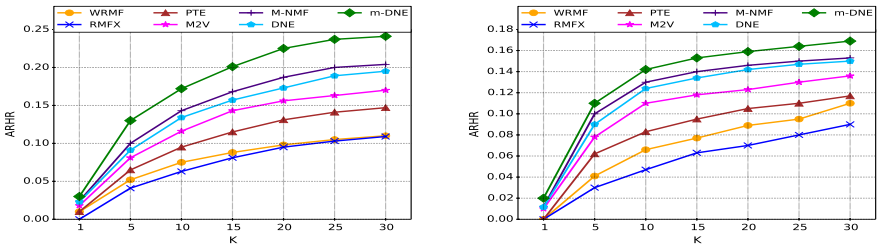
Since we are interested in measuring top-k recommendation instead of rating prediction, we measure the quality by looking at the $Recall@K$ [25] and Average Reciprocal Hit-Rank (ARHR) [26], which are widely used for evaluating top-k recommender systems. We show the performance when $k = \{1, 5, 10\}$, as a larger value of k is usually ignored for a typical top-k recommendation [25].

7.2 Results

Recommendation Effectiveness Table 2 summarizes the item and friend recommendation performance between our model and baselines. Besides, we also test our model without community attribute integration represented as DNE. From the results, we can observe that the *Recall*@K value grows gradually along with the increasing number of K, and the performance of item recommendation is better than friend recommendation. Besides, we can also observe on all datasets that: 1)Embedding-based algorithms (PTE, M2V, M-NMF, DNE and m-DNE) consistently perform better than non-embedding based benchmarks (WRMF, RMFX). It is because embedding-based algorithms can fully explore the network structure of the given information, which alleviates the issues of sparse and noisy signals. 2)The significant improvements show the promising benefit of the community integration and our incremental learning approach, which lead to the better performance of m-DNE than the other listed embedding methods.

Table 2: Top-k items and friends recommendation w.r.t. Recall@K (K=1,5,10).

Method	Twitter			Last.fm			Flickr		
	Recall@1	Recall@5	Recall@10	Recall@1	Recall@5	Recall@10	Recall@1	Recall@5	Recall@10
Top-k items recommendation									
WRMF	0.152	0.229	0.301	0.226	0.293	0.387	0.204	0.261	0.356
RMFX	0.115	0.194	0.273	0.197	0.276	0.358	0.171	0.252	0.334
PTE	0.219	0.292	0.379	0.276	0.352	0.433	0.246	0.327	0.394
M2V	0.236	0.307	0.392	0.291	0.374	0.467	0.263	0.341	0.435
M-NMF	0.264	0.328	0.426	0.342	0.407	0.506	0.311	0.386	0.479
DNE	0.251	0.324	0.417	0.331	0.403	0.498	0.306	0.374	0.470
m-DNE	**0.309**	**0.385**	**0.472**	**0.395**	**0.471**	**0.557**	**0.368**	**0.449**	**0.531**
Top-k friends recommendation									
WRMF	0.113	0.175	0.266	0.172	0.247	0.314	0.148	0.220	0.281
RMFX	0.097	0.146	0.204	0.136	0.225	0.290	0.118	0.196	0.267
PTE	0.152	0.226	0.313	0.224	0.276	0.347	0.198	0.255	0.329
M2V	0.176	0.235	0.327	0.234	0.292	0.348	0.203	0.267	0.334
M-NMF	0.226	0.267	0.339	0.262	0.321	0.378	0.237	0.304	0.351
DNE	0.213	0.256	0.332	0.254	0.317	0.363	0.228	0.296	0.344
m-DNE	**0.243**	**0.294**	**0.371**	**0.298**	**0.352**	**0.406**	**0.275**	**0.329**	**0.390**



(a) Top-k items recommendation (b) Top-k friends recommendation

Fig. 3: Recommendation performance w.r.t. ARHR.

Fig. 3 compares the performance of alternative approaches taking ARHR as metric. During experiments, we vary the number of recommendations K from 1 to 30. As expected, our m-DNE model performs better with ARHR as well, and M-NMF ranks the second place followed by DNE, which shows the same orders

in Table 2. In Fig. 3(a), as we recommend more items, since we have more chance to answer the true interested items correctly, ARHR grows gradually with increasing number K. The same trends appear in the friend recommendation task. To evaluate the efficiency of our model, we compare our m-DNE with other baselines on Twitter. As all baselines are not designed to handle dynamics except RMFX, we compare their cumulative running time over all time steps and plot it in a log scale. Each time step represents one day period. As can be seen in Fig. 4, m-DNE is much faster than the baselines which need to retrain and still show advantages compared with RMFX.

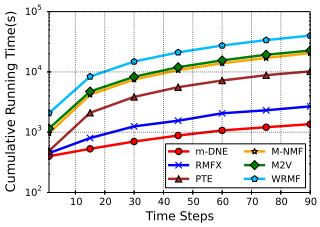
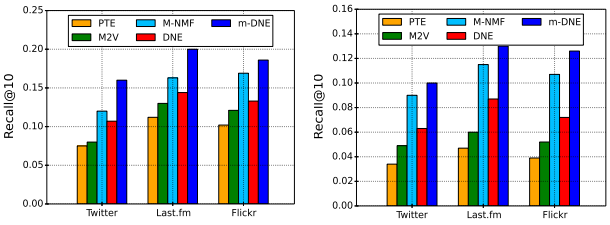


Fig. 4: Cumulative running time comparison.



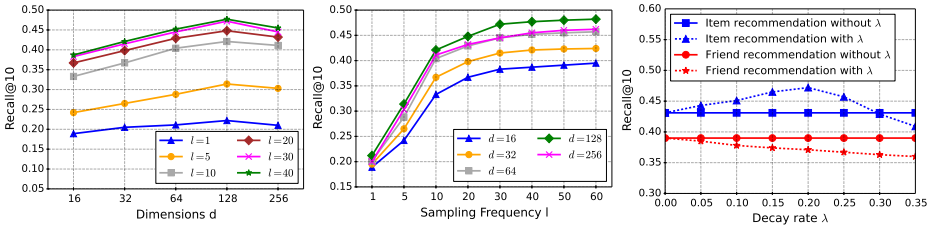
(a) Item recommendation (b) Friend recommendation

Fig. 5: Recommendations for cold-start cases.

Test for Cold Start Problem We also conduct experiments to study the effectiveness of different algorithms in addressing cold-start issues. As pre-processing, the target users who have less than 20 available items and social links in total are selected. As there are not many interaction records between users and items available for cold-start cases, WRMF and RMFX which are based on collaborative filtering, are not suitable for cold-start experiments. Thus, we compare m-DNE with the baselines which can leverage social information to recommend cold-start cases. The experimental results are shown in Fig. 5, from which we have the following observations: 1) m-DNE model still performs best consistently in recommending cold-start cases; 2) by comparing with Table 2, the *Recall* value of all algorithms decreases. For instance, the *Recall* value of M-NMF rapidly drops from 42.6% to 12% for twitter item recommendation but still better than DNE model, while m-DNE deteriorate slightly, which validates that community-aware high order proximity and the ability to capture the dynamic properties of the network are key factors affecting the recommendation performance.

Sensitivity to Parameters In this experiment, we study the influence of the embedding dimension d, the number of samples l and time decay rate λ by fixing the window size $|W| = 8$ and the random walk length $h = 40$. We vary one parameter each time to test the impact on recommendation performance with other parameters fixed. Because of the page limit, we only show the results on Twitter. But similar observations can be made on other datasets. Recommendation *Recall* value of m-DNE model is not highly sensitive to the dimension d, but still presents a tendency that its recommendation accuracy increases with the increasing number of d holistically, and it reaches peak when d is around 128.

However, m-DNE is sensitive to l with the *Recall* score varying a lot. First, the performance of m-DNE increases quickly with the increasing number of l, this is because the model has not achieved convergence. Then, it does not change significantly when the number of samples becomes large enough, since m-DNE has converged. Thus, to achieve a satisfying trade off between effectiveness and efficiency of model training, we set $l = 30$ and $d = 128$ on all datasets. In Fig. 6(c), λ shows different influence on item/user recommendation tasks. For item recommendation, the performance reaches the peak when $\lambda = 0.2$ but drops significantly afterwards. However, for user recommendation, the performance constantly decreases with the increasing value of λ. These phenomena show that in our case, items are more sensitive to time compared with users, and a suitable value of λ can help to improve the recommendation performance.



(a) dimensions, d (b) number of samples, l (c) time decay rate, λ

Fig. 6: Effect of different parameters on performance.

8 Conclusion

In this paper, we propose m-DNE, an efficient model which learns the embedding of heterogeneous social network by jointly modelling the temporal semantic effects, social relationships and user behavior sequential patterns in a unified way. Community-aware high-order proximity is applied to optimize the node representations. Besides, a parallel incremental learning algorithm and an efficient query processing technique are employed for recommendation efficiency. The experimental results show the effectiveness of our m-DNE on social recommendations. In the future, we will consider to integrate attributes from multiple social sites. Additionally, short-term user interest changes also need to be considered with the use of advanced deep learning models such as Recurrent Neural Network.

Acknowledgments. This work was supported by the Research Council of Norway (grant number 245469).

References

1. Sedhain, S., Sanner, S., Braziunas, D., Xie, L. and Christensen, J.: Social collaborative filtering for cold-start recommendations. In: RecSys. pp. 345-348 (2014)
2. Kouki, P., Fakhraei, S., Foulds, J., Eirinaki, M. and Getoor, L.: Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In: RecSys. pp. 99-106. ACM (2015)

3. Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V. and Sharp, D.: E-commerce in your inbox: Product recommendations at scale. In: SIGKDD. pp. 1809-1818. ACM (2015)
4. Covington, P., Adams, J. and Sargin, E.: Deep neural networks for youtube recommendations. In: RecSys. pp. 191-198. ACM (2016)
5. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J. and Mei, Q.: Line: Large-scale information network embedding. In: WWW. pp. 1067-1077 (2015)
6. Perozzi, B., Al-Rfou, R. and Skiena, S.: Deepwalk: Online learning of social representations. In: SIGKDD. pp. 701-710. ACM (2014)
7. Cavallari, S., Zheng, V.W., Cai, H., Chang, K.C.C. and Cambria, E.: Learning community embedding with community detection and node embedding on graphs. In: CIKM. pp. 377-386. ACM (2017)
8. Li, J., Dani, H., Hu, X., Tang, J., Chang, Y. and Liu, H.: Attributed network embedding for learning in a dynamic environment. In: CIKM. pp. 387-396 (2017)
9. Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W. and Yang, S.: Community Preserving Network Embedding. In: AAAI. pp. 203-209 (2017)
10. Agarwal, D., Chen, B.C. and Elango, P.: Fast online learning through offline initialization for time-sensitive recommendation. In: SIGKDD. pp. 703-712 (2010)
11. Diaz-Aviles, E., Drumond, L., Schmidt-Thieme, L. and Nejdl, W.: Real-time top-n recommendation in social streams. In: RecSys. pp. 59-66. ACM (2012)
12. Huang, Y., Cui, B., Zhang, W., Jiang, J. and Xu, Y.: Tencentrec: Real-time stream recommendation in practice. In: SIGMOD. pp. 227-238. ACM (2015)
13. Subbian, K., Aggarwal, C. and Hegde, K.: Recommendations for streaming data. In: CIKM. pp. 2185-2190. ACM (2016)
14. Grover, A. and Leskovec, J.: node2vec: Scalable feature learning for networks. In: SIGKDD. pp. 855-864. ACM (2016)
15. Dong, Y., Chawla, N.V. and Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: SIGKDD. pp. 135-144. ACM (2017)
16. Tang, J., Qu, M. and Mei, Q.: Pte: Predictive text embedding through large-scale heterogeneous text networks. In: SIGKDD. pp.1165-1174. ACM (2015)
17. Le, Q. and Mikolov, T.: Distributed representations of sentences and documents. In: ICML. pp. 1188-1196 (2014)
18. Li, Y. and Patra, J.C.: Genome-wide inferring genephenotype relationship by walking on the heterogeneous network. Bioinformatics, 26(9), 1219-1224 (2010)
19. Gao, Y., Chen, J. and Zhu, J.: Streaming gibbs sampling for lda model. arXiv preprint arXiv:1601.01142 (2016)
20. Blei, D.M., Ng, A.Y. and Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research, 3(1), 993-1022 (2003)
21. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
22. Recht, B., Re, C., Wright, S. and Niu, F.: Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In: NIPS (2011)
23. Fagin, R., Lotem, A. and Naor, M.: Optimal aggregation algorithms for middleware. Journal of computer and system sciences, 66(4), 614-656 (2003)
24. Hu, Y., Koren, Y. and Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: ICDM. pp. 263-272. IEEE (2008)
25. Cremonesi, P., Koren, Y. and Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: RecSys. pp. 39-46. ACM (2010)
26. Deshpande, M. and Karypis, G.: Item-based top-n recommendation algorithms. In: TOIS, 22(1), pp.143-177 (2004)