PRIN: A Probabilistic Recommender with Item Priors and Neural Models

Alfonso Landin(✉), Daniel Valcarce, Javier Parapar, and Álvaro Barreiro

Information Retrieval Lab, Department of Computer Science, University of A Coruña, A Coruña, Spain
{alfonso.landin,daniel.valcarce,javierparapar,barreiro}@udc.es

Abstract. In this paper, we present PRIN, a probabilistic collaborative filtering approach for top-N recommendation. Our proposal relies on continuous bag-of-words (CBOW) neural model. This fully connected feedforward network takes as input the item profile and produces as output the conditional probabilities of the users given the item. With that information, our model produces item recommendations through Bayesian inversion. The inversion requires the estimation of item priors. We propose different estimates based on centrality measures on a graph that models user-item interactions. An exhaustive evaluation of this proposal shows that our technique outperforms popular state-of-the-art baselines regarding ranking accuracy while showing good values of diversity and novelty.

Keywords: Collaborative filtering · Neural models · Centrality measures

1 Introduction

In recent years, the way users interact with different services has shifted from a proactive approach, where users were actively looking for content, to one where users play a more passive role receiving content suggestions. This transformation has been possible thanks to the advances in the field of Recommender Systems (RS). These models produce personalized item recommendations based on user-item past interactions.

Approaches to item recommendation are usually classified in three families [2]. The first algorithms, content-based systems, use item metadata to produce tailored recommendations [13]. The second family, collaborative filtering (CF), exploits the past interactions of the users with the items to compute recommendations [21,26]. These interactions can be ratings, clicks, purchases, reproductions, etc. The third family, hybrid systems, combines techniques from the other two approaches to generate recommendations.

Collaborative filtering algorithms, which is the focus of this paper, can, in turn, be divided into two types. Model-based techniques, which build predictive models from the interaction data, and neighborhood-based techniques [26]

L. Azzopardi et al. (Eds.): ECIR 2019, LNCS 11437, pp. 133–147, 2019.
https://doi.org/10.1007/978-3-030-15712-8_9

(also called memory-based methods), which exploit past interactions directly. Neighborhood-based techniques rely on similar users or items, the neighborhoods, to compute the recommendations.

In this paper, we address the item recommendation task by proposing a model-based collaborative filtering technique. Our method is inspired by a word embedding model recently developed in the field of Natural Language Processing: the continuous bag-of-words (CBOW) model. Word embedding models are capable of learning word representations that take the form of dense vectors, called embeddings. Embeddings have much lower dimensionality than traditional sparse one-hot and bag-of-words representations and, moreover, are effective state-of-the-art methods in several tasks [22,24,25]. In particular, `word2vec` [24,25] has attracted great attention because of their efficiency and effectiveness. This tool provides two different models for generating word embeddings: the continuous bag-of-words model, which is designed to predict a word given its context, and the skip-gram model, which aims to predict the context of a word. When working on textual data, the context of a word in a document is composed of the surrounding words inside a fixed window.

In this paper, we propose PRIN, Probabilistic Recommender with Item Priors and Neural Models, a novel probabilistic model for the top-N recommendation task. PRIN uses the neural network topology of the CBOW model. However, instead of generating embeddings, we use the network to compute the conditional probabilities of the users given an item. Our probabilistic model requires the estimation of item prior probabilities to perform the Bayesian inversion of the conditional probabilities. To compute these item priors, we develop a graph-based interpretation of the user-item interactions. Over that graphs, we propose several estimates of item priors based on well-known graph centrality measures.

One additional advantage of the PRIN model is that it can be computed by leveraging current `word2vec` CBOW implementations. Moreover, our adaptation is even able to incorporate graded preference values (such as ratings) into the neural model.

Experiments are conducted on three datasets from different domains, with distinct sizes and sparsity figures. We show that our model can outperform several state-of-the-art collaborative baselines in ranking accuracy while maintaining good values of novelty and diversity. Moreover, PRIN inherits the efficiency and scalability of the CBOW model. For the sake of reproducibility, we also make our software publicly available[1].

2 Related Work

In this section, we present previous work on embeddings models, initially proposed in Natural Language Processing and nowadays commonly used in Recommender Systems. After that, we introduce graph centrality measures, whose aim is to reveal the importance of a node in a graph.

[1] https://gitlab.irlab.org/alfonso.landin/prin.

2.1 Embeddings Models

Word and documents were traditionally represented using sparse high-dimensional vectors based on one-hot and bags-of-words (BOW [16]) models. However, nowadays, neural embedding methods provide more effective fixed-length dense vector representations [24,25,28]. In particular, the `word2vec` tool [24,25] implements efficient estimations of the continuous bag-of-words (CBOW) and the skip-gram (SG) word embedding models. While the SG model aims to predict the surrounding words within a fixed window, the CBOW model predicts the actual word given the surrounding words [24]. The neural network architecture of these models is the same: a fully connected feedforward network with a single hidden layer. The size of the input and output layers is the size of the vocabulary, and the size of the hidden layer size is given by the desired number of dimensions of the embeddings.

Our proposal approaches the recommendation task from a different perspective than previous efforts that used embedding models in collaborative filtering. In particular, the SG model has been previously adapted in [3,14] for the generation of item embeddings. In both cases, the methods discard the model once trained, and the embeddings are merely used with some memory-based techniques in the case of [14] and for category classification in [3]. In our proposal, we use the output of the neural model in combination with the item priors to produce the ranking of recommended items for a user in a model-based approach. Moreover, previous approaches do not tackle graded preference into the training process but use the data in a binarized form.

2.2 Centrality Measures

The importance of a node in a graph has been a subject of study for a long time. Researchers started exploring the dynamics in social groups from a mathematical perspective [9,34]. With this objective in mind, graphs were proposed to model the groups and the relations between their members. Finding the influence of a user has been reduced to the problem of measuring the importance of a node inside the social graph. Research from Bavelas [4] or Katz [19] in the early 50s showed the first attempts of defining centrality measures that can capture this property of the nodes of a graph. With the emergence of the World Wide Web, centrality measures were once again bought to the forefront as a way to analyze the graph formed by the pages contained within it. In this context, PageRank [27] and HITS [20] were defined.

Graph representations of collaborative filtering data has been previously used in tasks such as neighborhood selection for memory-based recommenders [7]. Centrality measures have also been used in the recommendation field, especially in social-based recommender systems since they exploit the social relationships between users [5,15]. In contrast, in our work, we use centrality measures to compute prior probabilities over items, taking advantage of their ability to capture the importance of the items in the whole graph.

3 Proposal

In this section we present our probabilistic recommender, PRIN, explaining how we train it and how the model computes the recommendations. A brief introduction to the notation used to present the model precedes this description. Lastly, we include some comments on the implementation details.

3.1 Notation

We denote the set of users of the systems as $\mathcal{U}$ and the set of items as $\mathcal{I}$. For a user $u \in \mathcal{U}$ and an item $i \in \mathcal{I}$, we use $r_{u,i}$ to indicate the rating given by u to i, having a value of zero in case the user did not rate the item. The set of items rated by a user u is represented by $\mathcal{I}_u$ and the set of users that have rated an item i is denoted by $\mathcal{U}_i$.

3.2 Probabilistic Recommender with Neural Model

The idea behind word embedding models is that both words occurring close to each other, inside a window of fixed length, or words that appear in different sentences surrounded by the same words are similar. We postulate that this also applies to collaborative filtering data.

We propose a probabilistic model based on the adaptation of the continuous bag-of-words (CBOW) model for the task of top-N recommendation. The CBOW model predicts a word given its context, defined by the surrounding words inside a fixed-length window [24]. In our scenario, users play the role of words and item profiles that of documents, defining an item profile as the set of users that have rated it. We choose the CBOW model for two main reasons. On the one hand, its efficiency is superior to the skip-gram model [24]. On the other hand, we think the task of finding if a user fits inside an item profile is more natural for the recommendation task than the skip-gram objective of finding the context that fits a user.

Figure 1 presents the architecture of the model, inspired by the CBOW neural model. The output of the network is the target user, and the input is its context, i.e. the item profile without the user. For a particular item i and target user u the input consists of the input context vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_{u-1}, \mathbf{x}_{u+1}, \ldots, \mathbf{x}_{|\mathcal{U}_i|}\}$, that are all the users that rated the item except user u, with $|\mathcal{U}_i|$ being the number of users that have rated item i. These vectors are encoded using a one-hot representation. For user v, $\mathbf{x}_v$ is a vector of the form $\{x_{v1}, \ldots, x_{v|\mathcal{U}|}\}$, where all components are zero except the v-th component which is one and $|\mathcal{U}|$ is the number of users in the dataset. This way the training examples are created from the item profiles, being able to generate $|\mathcal{U}_i|$ training examples for each item profile, one example for each user in the item profile.

The amount of units in the hidden layer, d, is a hyperparameter of the model that determines the dimension of the embeddings. These units have a linear activation function. We use the matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{U}| \times d}$ to denote the weights of the connections between the input layer and the hidden layer. Each row of the

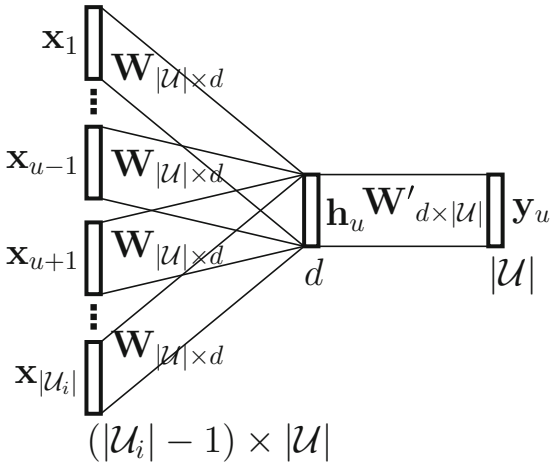

Fig. 1. Architecture of the model. The input layer consists of the aggregation of $|\mathcal{U}_i| - 1$ one-hot encoded vectors, each vector of dimension $|\mathcal{U}|$, the hidden layer has d units and the output layer has $|\mathcal{U}|$ units.

matrix, $\mathbf{v}_v$, corresponds to the input embedding vector of dimension d of the user v. The output of the hidden layer for the target user u, $\mathbf{h}_u$, is computed by averaging the embeddings of the input users corresponding to the context, weighted by the rating given by the users to the item i:

$$\mathbf{h}_u = \frac{\mathbf{W}}{\sum\limits_{v\in\mathcal{U}_i\setminus\{u\}} r_{v,i}} \sum_{v\in\mathcal{U}_i\setminus\{u\}} r_{v,i}\,\mathbf{x}_v = \frac{\sum\limits_{v\in\mathcal{U}_i\setminus\{u\}} r_{v,i}\,\mathbf{v}_v}{\sum\limits_{v\in\mathcal{U}_i\setminus\{u\}} r_{v,i}} \tag{1}$$

By weighting the average by the ratings given by the users, we can incorporate these values into the training process. Although we evaluated our proposal with explicit feedback dataset, one can incorporate information from implicit feedback, such as clicks or play counts, by substituting the ratings in Eq. 1.

The output layer is composed of $|\mathcal{U}|$ units with a softmax activation function. Similar to what we did before, we use the matrix $\mathbf{W}' \in \mathbb{R}^{d\times|\mathcal{U}|}$ to denote the weights of the connection between the hidden and the output layer. Each column of this matrix, $\mathbf{v}'_u$, is the d-dimensional output embedding vector of user u. This way the input of the output layer is given by $\mathbf{v}'^T_u\mathbf{h}_u$. The output of the network is the posterior probability distribution of users for the context, i.e. the item profile without the target user. These probabilities are calculated using the softmax function. The component u of the output vector for the target user, $\mathbf{y}_u$, is calculated as:

$$p\big(u \,|\, \mathcal{U}_i \setminus \{u\}\big) = (\mathbf{y}_u)_u = \frac{\exp\left(\mathbf{v}'^T_u\mathbf{h}_u\right)}{\sum\limits_{v\in\mathcal{U}} \exp\left(\mathbf{v}'^T_v\mathbf{h}_u\right)} \tag{2}$$

Each example of the training set consists of the profile of an item i and a target user u from the profile. Maximizing the likelihood of the data is equivalent to minimizing the negative log likelihood. Therefore, the objective function is:

$$\mathcal{L} = -\sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}_i} \log p\big(u \,|\, \mathcal{U}_i \setminus \{u\}\big) \tag{3}$$

The training of the model consists in learning the matrices $\mathbf{W}$ and $\mathbf{W}'$ by backpropagation. This model becomes impractical in large-scale scenarios because the cost of computing the gradient of $\log p\big(u \,|\, \mathcal{U}_i \setminus \{u\}\big)$ for each training example is proportional to the number of users, due to the softmax function (see Eq. 2). Mikolov et al. already noted this problem in [24,25], where they propose to solve it by using one of two approximations to the softmax function: hierarchical softmax and negative sampling. For this work, we choose negative sampling as it provides faster training than hierarchical softmax and similar effectiveness [24,25].

It can be observed that the objective function in Eq. 3 does not include any regularization term. Early experiments with the model showed that it was overfitting the data when training with too many iterations. To solve this problem we choose to use dropout regularization in the input layer [32]. We decided this over other forms of regularization, such as ℓ_2 regularization, because it provided better effectiveness and improved training time [29]. At the same time, we can leverage existing `word2vec` implementations when using dropout as we will see later on.

Once the parameters have been trained, it is possible to use the model to compute the posterior probability distribution of a user for each item. This calculation is done by applying Eqs. 1 and 2 to the whole item profile, without removing any user. It should be noted that after applying dropout regularization, during evaluation, the activations are reduced to account for the missing activations during training [32]. This process is not necessary in our case because the inputs to the hidden layer are averaged, as we can see in Eq. 1.

The output for each unit of the output layer is the probability of the corresponding user u given the item i, $p(u|i) = p(u|\mathcal{U}_i)$. It is not possible to use these probabilities to make a ranking of items for a user as $p(u|i)$ and $p(u|j)$ are not comparable $(i, j \in \mathcal{I}, i \neq j)$. It is possible to apply Bayes' rule to transform the probabilities and make them comparable:

$$p(i|u) = \frac{p(u|i)\,p(i)}{p(u)} \overset{\text{rank}}{=} p(u|i)\,p(i) \tag{4}$$

We describe in the next subsection the options we explored for the computation of the prior distribution of items, $p(i)$.

3.3 Item Priors with Centrality Measures

The objective of the centrality measures is to capture the importance of a node inside a graph. Several measures have been defined over the years, and their suitability for a task depends on the flow inside the graph [9]. For this reason, we

examined different types of measures [8]. The first category, geometric measures, is comprised of measures that assume that importance is a function of distances. In this group, we examined the indegree measure (the number of incoming edges of a node) and closeness [4]. The measures of the second category, spectral measures, compute the left dominant eigenvector of some matrix derived from the graph. We studied Katz's index [19], PageRank [27] and HITS [20] from this category. Lastly, we analyzed betweenness centrality, a path-based measure that takes into examination all the shortest path coming into a node [1,12].

To apply these measures to the computation of the prior distribution of items, we need to construct a graph-based model of the interactions between users and items in the system. We propose to construct a bipartite graph, where users and items play the role of the nodes, and the user-item interactions define the edges between users and the items. The weight of these edges is the rating assigned by the user to the item. We built two variants of the graph, one directed with the orientation of the edges going from users to the items and an undirected version. We do this because the direction of the edges can be meaningful when computing some centrality measures, but other measures are not useful when applied on a graph whose paths have a maximum length of one edge, as is the case of the directed graph.

3.4 Implementation Details

One of the advantages of the popularity of `word2vec` is that there are several publicly available implementations of the CBOW model. It is possible to leverage these implementations to build our model. We explain how to do that, also making possible to introduce the ratings of the items in the process and simulating the dropout in the input layer.

The original `word2vec` model is trained with a text corpus as the input. A corpus is composed of ordered sequences of words which we call documents, but can also be any other grouping of words such as sentences or paragraphs. The model has a hyperparameter for the window size, w, that controls how large is the context of a word, i.e. how many words before and after it are part of the context. For example, for $w = 1$, the context would be the preceding and the following words of the target word.

To train our model using collaborative filtering data, we build the analogous of a document in the format expected by the tool for each item profile. This pseudo-document contains all the identifiers of the users that have rated the item. To consider the whole item profile as the context, we set the window hyperparameter to the size of the larger item profile. The order of the items in the profile does not matter because the input of the hidden layer is the average of the input embeddings. To introduce the preference values into the model, we repeat each user identifier as many times as the rating given by the user to the item. Computing the average of the input constructed in this way is equivalent to the weighted average of Eq. 1.

Finally, we can introduce the dropout effect by modifying the hyperparameter w. If we set this parameter to a value smaller than the size of the profile, the

Table 1. Datasets statistics.

Dataset	Users	Items	Ratings	Density
MovieLens 20M	138,493	26,744	20,000,263	0.540%
R3-Yahoo	15,400	1,000	365,703	2.375%
LibraryThing	7,279	37,232	749,401	0.277%

context will be comprised of only some of the users of the item profile, *dropping out* the rest. We can add randomness to this procedure by shuffling the input each iteration. The combination of setting the w to a suitable value with the shuffling of the item profiles each iteration produces a similar effect to the dropout. This approach is a variant of the original technique that drops units randomly with a probability p [32]. Using this variant allows us to reuse existing `word2vec` CBOW implementations.

Training the PRIN model leads to a complexity for each training step of $\mathcal{O}(d \times (w+n))$, when using d dimensions, window size w and n negative samples. At each training step, there are w input embeddings, each corresponding to each input, of dimension d, that are averaged. It should be noted that w is bounded by the size of the larger item profile. The use of dropout in the form of a window produces notable improvements in the average computational cost of training the model. Moreover, using negative sampling allow approximating the softmax function with only n samples, instead of the whole user set. Finally, the number of training examples scales linearly with the number of user-item interactions in the collection. With all these facts, we can see that scalability of PRIN is well-suited for large-scale scenarios.

4 Experiments

In this section, we introduce the datasets, the evaluation protocol and the metrics used in our experiments. We finish the section by presenting the results of the experiments, confronting them with representative baselines.

4.1 Datasets

To evaluate the effectiveness of our proposal we conducted experiments on several collections, from different domains: the MovieLens 20M movie dataset[2], the R3-Yahoo! music dataset[3] and the LibraryThing book dataset. Details from each collection can be seen in Table 1. The datasets where partitioned randomly in two sets, one containing 80% of the ratings of each user, used for training, and a second split, with the remaining 20%, used for evaluation purposes.

[2] http://grouplens.org/datasets/movielens.
[3] http://webscope.sandbox.yahoo.com.

4.2 Evaluation Protocol

To evaluate the algorithms for the top-N recommendation task, we use the TestItems evaluation approach as described in [6]. For each user u, we rank all the items that have a rating by any user in the test set and were not rated by user u in the training set. This protocol provides a reliable assessment of the quality of the recommendation because it measures how well a recommender discerns relevant items in the collection [6].

To assess the accuracy of the recommendation rankings we use the Normalized Discounted Cumulative Gain (nDCG), using the *standard formulation* as described in [33], with the ratings in the test set as graded relevance judgments. We also measured diversity using the complement of the Gini index [11]. Last, we assess the novelty of the recommendations using the mean self-information (MSI) [35]. All the metrics are evaluated at a cut-off of 10 because we want to study the quality of the top recommendations, the ones the user usually consumes. To penalize a recommender not being able to provide recommendations to every user, the score in all metrics for those users is assigned a value of zero.

We study the statistical significance of the improvements regarding nDCG@10 and MSI@10 using a permutation test ($p < 0.01$) [31]. We cannot apply this procedure to the Gini index because we are using a paired test and Gini is a global metric. The statistical significance of the results is annotated in Table 3.

4.3 Baselines

We compare our proposed model to a representative set of state-of-the-art baselines. First, from the memory-based category of recommenders, we use NNCosNgbr [10], an item-based neighborhood approach. We also employ several techniques based in matrix factorization: PureSVD [10], BPRMF [30] and WRMF [18]. We compared with CoFactor [23], a variant of WRMF that jointly factorizes the user-item matrix and an embedding model, and NeuMF, a novel neural collaborative filtering approach [17]. Finally, we also include the results of the item-based counterpart of our model. We called this probabilistic recommender with neural models PRN.

The networks architecture of PRN is analogous to the architecture of PRIN (shown in Fig. 1), with an input of $|\mathcal{I}_u| - 1$ one-hot encoded vectors of dimension $|\mathcal{I}|$, where $\mathcal{I}$ is the set of items and $\mathcal{I}_u$ is the set of items rated by user u. The output is calculated with the dual equations of Eqs. 1 and 2. PRN takes as input a user profile and the output is the posterior probability distribution of the items for that user. This probability is usable as the basis of ranking, obviating the need for a prior distribution as in the case of PRIN.

We performed a grid search to tune all the hyperparameters of the baselines to maximize nDCG@10. Table 2 reports the optimal values of the hyperparameters (using the notation from the original papers) for all the techniques to favour reproducibility.

Table 2. Optimal values of the hyperparameters for nDCG@10 for NNCosNgbr, PureSVD, BPRMF, WRMF, CoFactor, NeuMF and our proposal PRIN and its item-based counterpart PRN.

Model	MovieLens 20M	R3-Yahoo!	LibraryThing
NNCosNgbr	$k = 50$	$k = 25$	$k = 25$
PureSVD	$d = 30, \kappa = 10^{-6}$	$d = 15, \kappa = 10^{-6}$	$d = 700, \kappa = 10^{-6}$
BPRMF	$d = 50$, $\lambda = 0.01$, $\alpha = 0.01$, $i = 10^5$	$d = 175$, $\lambda = 0.01$, $\alpha = 0.01$, $i = 10^5$	$d = 600$, $\lambda = 0.001$, $\alpha = 0.01$, $i = 10^6$
WRMF	$d = 50$, $\lambda = 0.01$, $\alpha = 1$, $i = 50$	$d = 50$, $\lambda = 1$, $\alpha = 2$, $i = 50$	$d = 400$, $\lambda = 0.1$, $\alpha = 1$, $i = 50$
CoFactor	$d = 100$, $c_0 = 0.3$, $c_1 = 3$, $\lambda_\theta = \lambda_\beta = \lambda_\gamma = 10^{-5}$, $k = 1$	$d = 30$, $c_0 = 1$, $c_1 = 10$, $\lambda_\theta = \lambda_\beta = \lambda_\gamma = 10^{-5}$, $k = 1$	$d = 500$, $c_0 = 1$, $c_1 = 10$, $\lambda_\theta = \lambda_\beta = \lambda_\gamma = 10^{-5}$, $k = 1$
NeuMF	$d = 64$, $i = 20$, $n = 5$	$d = 12$, $i = 5$, $n = 5$	$d = 1024$, $i = 20$, $n = 5$
PRIN	$d = 1000$, $w = 50$, $it = 1000$, indegree	$d = 50$, $w = 10$, $it = 200$, Katz	$d = 200$, $w = 10$, $it = 200$, PageRank
PRN	$d = 500$, $w = 100$, $it = 300$	$d = 200$, $w = 2$, $it = 100$	$d = 500$, $w = 1$, $it = 1000$

4.4 Results and Discussion

To tune our model, we perform a grid search over the hyperparameters, the same way we did with the baselines, to maximize nDCG@10. Although our implementation is based on the CBOW model, to keep things simple we only tune the parameters relevant to our model: the dimension of the hidden layer d, the window size w for the regularization effect and the number of training iterations it. The parameter for the negative sampling training is fixed, with a value of 10 negative samples. We also report the centrality measure that yields the best results. Table 2 reports the optimal values for each collection with the values of the hyperparameters of the baselines.

Table 3 shows the values for nDCG@10, Gini@10 and MSI@10 for all the recommenders. The results show that PRIN outperforms all the baselines concerning nDCG@10. In the MovieLens dataset, it surpasses the best baseline, WRMF, while also obtaining a better result in diversity but a lower score in novelty. When comparing to the next best result in R3-Yahoo!, BPRMF, our model is also able to perform better in novelty and diversity. In the case of the LibraryThing dataset, the improvement in nDCG@10 is statistically significant over all the baselines except CoFactor. When it comes to novelty and diversity in this dataset, the results are not as good as other baselines. This fact is not unexpected, diversity and accuracy are frequently considered as two irreconcilable goals in the field of Recommender Systems. Usually, systems with good figures

Table 3. Values of nDGC@10, Gini@10, MSI@10 on MovieLens 20M, R3-Yahoo! and LibraryThing datasets. Statistical significant improvements (according to permutation test with $p < 0.01$) in nDCG@10 and MSI@10 with respect to NNCosNgbr, PureSVD, BPRMF, WRMF, CoFactor, NeuMF and our proposal PRIN and its dual model PRN are superscripted with a, b, c, d, e, f, g and h, respectively.

Model	Metric	ML 20M	R3-Yahoo!	LibraryThing
NNCosNgbr	nDCG@10	0.1037	0.0172	0.1438
	Gini@10	0.0209	0.1356	0.1067
	MSI@10	29.3332^{bcdefg}	$\mathbf{36.8264}^{bcdefgh}$	$\mathbf{47.0790}^{bcdefgh}$
PureSVD	nDCG@10	0.3477^{acfh}	0.0233^{a}	0.2283^{af}
	Gini@10	0.0079	0.0587	0.0535
	MSI@10	15.4201	21.9703^{c}	40.7276^{cdefg}
BPRMF	nDCG@10	0.2671^{ah}	0.0278^{abdf}	0.2479^{abf}
	Gini@10	0.0103	0.1071	0.0474
	MSI@10	15.9674^{b}	21.4253	34.5252
WRMF	nDCG@10	0.3682^{abcefh}	0.0266^{a}	0.2532^{abcfh}
	Gini@10	0.0138	0.1191	0.0512
	MSI@10	17.3695^{bcg}	24.7479^{bcefg}	38.2290^{cfg}
CoFactor	nDCG@10	0.3555^{abcfh}	0.0258^{ab}	0.2568^{abcdfh}
	Gini@10	0.0215	0.1407	0.0690
	MSI@10	19.5491^{bcdg}	25.7688^{bcfg}	39.7497^{cdfg}
NeuMF	nDCG@10	0.3185^{ach}	0.0258^{ab}	0.1835^{a}
	Gini@10	0.0328	0.0993	0.0613
	MSI@10	21.2605^{bcdeg}	22.2208^{bc}	36.5621
PRIN	nDCG@10	$\mathbf{0.3751}^{abcdefh}$	$\mathbf{0.0299}^{abcdefh}$	$\mathbf{0.2578}^{abcdfh}$
	Gini@10	0.0155	0.1966	0.0482
	MSI@10	16.5353^{bc}	24.0921^{bcf}	34.4458^{cg}
PRN	nDCG@10	0.1909^{a}	0.0276^{abd}	0.2423^{abf}
	Gini@10	**0.2175**	**0.3221**	**0.1208**
	MSI@10	$\mathbf{49.4532}^{abcdefg}$	29.4596^{bcdefg}	42.1890^{bcdefg}

of accuracy tend to degrade de diversity of the recommendation, and systems with bad performance in accuracy show better diversity, in the extreme case a random recommender would produce very diverse recommendations.

Another significant result is that PRIN is consistently the best method regarding accuracy across collections. This property is essential in order to select an algorithm for use in a commercial solution. This property does not appear with the other methods. For instance, when observing the other neural/embedding-based models we can observe that CoFactor ranks third in the ML dataset, fifth in the R3 collection and sixth with the LibraryThing data, in turn, NeuMF ranks fifth, sixth and seventh respectively.

The optimal values for the hyperparameters vary for each collection. This fact indicates the need, shared with all the baselines, to tune these hyperparameters to the particular data. In the case of the size of the hidden layer, there is a trend for the need of larger hidden layers the larger the dataset. This fact supports the intuition that with more data there is a need for more features to be able to capture the properties of the data.

Regarding the centrality measures, each dataset performs better with a different one. The best results with MovieLens are obtained using indegree, whose value for the items is independent of whether the directed or the undirected graph is used. For R3-Yahoo!, using Katz's index on the directed graph yields the best results. In this dataset, using the indegree measure leads to similar results in nDCG@10 but worse on novelty and diversity. When it comes to LibraryThing, it is PageRank, computed on the undirected graph, that gives the best performance. Therefore, we can conclude that the centrality measures have to be adapted to the nature of the graph. For instance, dataset sparsity affects the connectivity of the graph, and the existence of the connected components reflects user communities. Therefore before selecting an item prior, we have to analyse the connectivity, edge meaning and size of the graph.

5 Conclusions

In this paper, we presented PRIN, a novel probabilistic collaborative filtering technique. Our probabilistic model exploits the output of a neural user embedding model. This embedding model can be computed by leveraging existing `word2vec` CBOW implementations. The probabilistic formulation of PRIN also requires an item prior estimate. We evaluated several centrality measures of two graph-based interpretations of the user-item interactions as item prior probability estimates.

Our experiments showed that PRIN outperforms all the baselines on three datasets regarding ranking accuracy. PRIN is also able to provide good figures of diversity and novelty.

As future work, we envision to study other no graph-based prior estimates to further improve PRIN. Additionally, we think that it would be interesting to analyze the adaptation of the skip-gram model and also explore deeper or more complex network topologies for the neural model. Another prospect is the evaluation of the model when using an implicit feedback dataset.

Acknowledgments. This work has received financial support from project TIN2015-64282-R (MINECO/ERDF) and accreditation ED431G/01 (Xunta de Galicia/ERDF). The first author acknowledges the support of grant FPU17/03210 (MICIU) and the second author acknowledges the support of grant FPU014/01724 (MICIU).

References

1. Anthonisse, J.: The rush in a directed graph. Stichting Mathematisch Centrum. Mathematische Besliskunde (BN 9/71), January 1971
2. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997). https://doi.org/10.1145/245108.245124
3. Barkan, O., Koenigstein, N.: Item2vec: neural item embedding for collaborative filtering. In: 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6, September 2016. https://doi.org/10.1109/MLSP.2016.7738886
4. Bavelas, A., Barrett, D.: An Experimental Approach to Organizational Communication. American Management Association (1951)
5. Bellogín, A., Cantador, I., Díez, F., Castells, P., Chavarriaga, E.: An empirical comparison of social, collaborative filtering, and hybrid recommenders. ACM Trans. Intell. Syst. Technol. **4**(1), 1–29 (2013). https://doi.org/10.1145/2414425.2414439
6. Bellogín, A., Castells, P., Cantador, I.: Precision-oriented evaluation of recommender systems. In: Proceedings of the 5th ACM Conference on Recommender systems, RecSys 2011, pp. 333–336. ACM, New York (2011). https://doi.org/10.1145/2043932.2043996
7. Bellogin, A., Parapar, J.: Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In: Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys 2012, pp. 213–216. ACM, New York (2012). https://doi.org/10.1145/2365952.2365997
8. Boldi, P., Vigna, S.: Axioms for centrality. Internet Math. **10**(3–4), 222–262 (2014). https://doi.org/10.1080/15427951.2013.865686
9. Borgatti, S.P.: Centrality and network flow. Soc. Netw. **27**(1), 55–71 (2005). https://doi.org/10.1016/j.socnet.2004.11.008
10. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on Top-N recommendation tasks. In: Proceedings of the 4th ACM Conference on Recommender Systems, RecSys 2010, pp. 39–46. ACM, New York (2010). https://doi.org/10.1145/1864708.1864721
11. Fleder, D., Hosanagar, K.: Blockbuster culture's next rise or fall: the impact of recommender systems on sales diversity. Manage. Sci. **55**(5), 697–712 (2009). https://doi.org/10.1287/mnsc.1080.0974
12. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry **40**(1), 35–41 (1977). https://doi.org/10.2307/3033543
13. de Gemmis, M., Lops, P., Musto, C., Narducci, F., Semeraro, G.: Semantics-aware content-based recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 119–159. Springer, Boston (2015). https://doi.org/10.1007/978-1-4899-7637-6_4
14. Grbovic, M., et al.: E-commerce in your inbox: Product recommendations at scale. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2015, pp. 1809–1818. ACM, New York (2015). https://doi.org/10.1145/2783258.2788627
15. Guy, I.: Social recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 511–543. Springer, Boston, MA (2015). https://doi.org/10.1007/978-1-4899-7637-6_15
16. Harris, Z.S.: Distributional structure. WORD **10**(2–3), 146–162 (1954). https://doi.org/10.1080/00437956.1954.11659520

17. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, WWW 2017, pp. 173–182. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2017). https://doi.org/10.1145/3038912.3052569
18. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, ICDM 2008, pp. 263–272. IEEE, Washington (2008). https://doi.org/10.1109/ICDM.2008.22
19. Katz, L.: A new status index derived from sociometric analysis. Psychometrika **18**(1), 39–43 (1953). https://doi.org/10.1007/BF02289026
20. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999). https://doi.org/10.1145/324133.324140
21. Koren, Y., Bell, R.: Advances in collaborative filtering. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 77–118. Springer, Boston (2015). https://doi.org/10.1007/978-1-4899-7637-6_3
22. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning, pp. 1188–1196. Proceedings of Machine Learning Research, PMLR, Bejing, China, 22–24 June 2014
23. Liang, D., Altosaar, J., Charlin, L., Blei, D.M.: Factorization meets the item embedding: regularizing matrix factorization with item co-occurrence. In: Proceedings of the 10th ACM Conference on Recommender Systems, RecSys 2016, pp. 59–66. ACM, New York (2016). https://doi.org/10.1145/2959100.2959182
24. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. CoRR abs/1301.3, January 2013
25. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26, NIPS 2013, pp. 3111–3119. Curran Associates, Inc. (2013)
26. Ning, X., Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 37–76. Springer, Boston (2015). https://doi.org/10.1007/978-1-4899-7637-6_2
27. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, November 1999
28. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, pp. 1532–1543. ACL, Stroudsburg (2014). https://doi.org/10.3115/v1/D14-1162
29. Phaisangittisagul, E.: An Analysis of the regularization between L2 and dropout in single hidden layer neural network. In: Proceedings of the 7th International Conference on Intelligent Systems, Modelling and Simulation, ISMS 2016, pp. 174–179. IEEE (2016). https://doi.org/10.1109/ISMS.2016.14
30. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2009, pp. 452–461. AUAI Press, Arlington (2009)

31. Smucker, M.D., Allan, J., Carterette, B.: A comparison of statistical significance tests for information retrieval evaluation. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM 2007, p. 623. ACM, New York (2007). https://doi.org/10.1145/1321440.1321528
32. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014). https://doi.org/10.1214/12-AOS1000
33. Wang, Y., Wang, L., Li, Y., He, D., Chen, W., Liu, T.Y.: A theoretical analysis of NDCG ranking measures. In: Proceedings of the 26th Annual Conference on Learning Theory, COLT 2013, pp. 1–30. JMLR.org (2013)
34. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Structural Analysis in the Social Sciences. Cambridge University Press, Cambridge (1994)
35. Zhou, T., Kuscsik, Z., Liu, J.G., Medo, M., Wakeling, J.R., Zhang, Y.C.: Solving the apparent diversity-accuracy dilemma of recommender systems. Proc. Nat. Acad. Sci. **107**(10), 4511–4515 (2010). https://doi.org/10.1073/pnas.1000488107