International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems
© World Scientific Publishing Company

Addressing the cold start with positive-only feedback through semantic-based recommendations

Paolo Tomeo
Politecnico di Bari, Via Orabona 4
Bari, 70125, Italy
paolo.tomeo@poliba.it

Ignacio Fernández-Tobías
Universidad Autónoma de Madrid, Calle Francisco Tomás y Valiente 11
Madrid, 28049, Spain
ignacio.fernandezt@uam.es

Iván Cantador
Universidad Autónoma de Madrid, Calle Francisco Tomás y Valiente 11
Madrid, 28049, Spain
ivan.cantador@uam.es

Tommaso Di Noia
Politecnico di Bari, Via Orabona 4
Bari, 70125, Italy
tommaso.dinoia@poliba.it

Recommender systems aim to provide users with accurate item suggestions in a personalized fashion, but struggle in the case of cold start users, for whom there is a scarcity of preference data. User preferences can be either explicitly stated by the users -often by means of ratings- , or implicitly acquired -for instance by mining text reviews, query searches or purchasing records-, but recommendation methods have been mostly designed to deal with numerical ratings. Real scenarios with user preferences expressed in the form of binary and unary (positive-only) feedbacks, such as the thumbs up/down in YouTube and the likes in Facebook, are increasingly popular, and make the user cold start problem even more challenging. To address this situation, i.e., cold start with positive-only feedback, in this paper we propose to exploit additional data to user preferences by means of specialized hybrid recommendation methods. In particular, we investigate a number of graph-based and matrix factorization recommendation models that jointly exploit user preferences and item semantic metadata automatically extracted from the well known DBpedia knowledge graph. Following a rigorous evaluation methodology for cold start, we empirically compare the above hybrid recommendation methods on a Facebook dataset composed of users likes for items in three domains, namely books, movies and music. The achieved experimental results show that the hybrid methods, exploiting item semantic metadata, outperform content-based and collaborative filtering baselines. In addition to recommendation accuracy, we also consider recommendation individual diversity and aggregate diversity as key quality factors for the users' satisfaction.

1. Introduction

Recommender Systems have become fundamental tools in helping users to find what is relevant for them in situations where information overload makes such task hard or even impossible. For such purpose, they capture, model and exploit user preferences, which can be obtained either explicitly by means of ratings, or implicitly e.g. by processing text reviews, and by mining item consuming and purchasing records. Two main types of recommendation approaches exist, namely content-based and collaborative filtering. The former commonly relies on content-based features to represent both user and item profiles; the latter, in contrast, works only with rating-based user/item similarities. The majority of the most effective collaborative filtering approaches have been designed to deal with numerical ratings, such as the 5-star ratings in Amazon[a] and Netflix[b], for both rating prediction and item ranking (a.k.a. top-N recommendation) tasks, and have been shown to generally outperform content-based approaches [1]. In many e-commerce and social network sites, however, user preferences are expressed in the form of binary and unary (positive-only) ratings, such as the thumbs up/down in YouTube[c] and the likes in Facebook[d], respectively. Moreover, in these cases, the well-known problem of cold-start in collaborative filtering [1], which refers to the scarcity of ratings at user level, is highly remarkable. In this context, the consideration of content-based features could benefit the understanding of the users' preferences, as well as the finding of similar users and items. For instance, in the movie recommendation domain, a user may be suggested with movies based on her and others' preferences for particular genres, directors and actors. Different hybrid recommender systems have been proposed to obtain the advantages of both the aforementioned approaches, also in order to tackle the cold-start problem. Hence, in this paper we address the cold-start in recommendations with positive-only feedback, exploring a number of graph-based and matrix factorization recommendation models that jointly exploit user ratings and item metadata, and evaluate them with Facebook *likes* as source of user preferences. The above mentioned sites do not provide content-based features that comprise items metadata. Hence, features can be (i) extracted from text descriptions about the items, e.g., movie plots, song lyrics, and book synopses; (ii) established by means of social tags manually assigned by users to items. Through the Facebook Graph API, items are identified by names which are plain texts freely set by users. Thus, to obtain metadata for available items, in this paper we propose to first link them with their corresponding entities in an external knowledge source. In particular, we present a method that automatically maps the items names to URIs of semantic entities in DBpedia[e], the Wikipedia ontology which is considered

[a] Amazon online shopping, www.amazon.com
[b] Netflix movie and TV series streaming, www.netflix.com
[c] YouTube online video sharing, www.youtube.com
[d] Facebook online social network, www.facebook.com
[e] wiki.dbpedia.org

as the core repository of the Linked Open Data (LOD) cloud. The use of LOD does not merely allow describing items by means of (isolated) content-based features, but also creating semantic networks that relate items, features and items with features, e.g., Kubrick's "The Full Metal Jacket" is a movie based on Hasford's "The Short-Timers" novel, and "Anti-War Films" is a subgenre of "Political Films." In this paper we propose to exploit semantic networks connecting users, liked items, and features for recommendation purposes in two ways: first, directly using the networks by graph-based models; second, extending content-based item profiles with related features, and incorporating the enriched profiles into matrix factorization models. Therefore, we evaluated the two types of approaches, along with several baselines, on a Facebook dataset comprising three distinct domains, namely books, movies and music. While new algorithms and approaches have been proposed over the years mainly devoted to maximizing recommendation accuracy, it has been recognized recently that predictive accuracy of recommendations is not enough to judge the effectiveness of a recommender system [?]. The most accurate recommendations for a user are often too similar to each other and attention has to be paid towards the goal of improving *diversity* in recommended items, known *individual diversity* [?]. Indeed, a recent user study pointed out a strong correlation between perceived accuracy and users satisfaction for algorithms able to better diversify the returned list of recommended items [2]. While this kind of dimension is list-wise, since it represents the diversity degree in the recommendations lists, the importance of a system-wise diversity has been revealed from other studies [3]. In particular, *aggregate diversity* has been proposed to assess the ability of a system to cover the items catalog and equally distribute the recommendations across all the items. Such quality factor results important for both business and user perspective: users may receive less obvious and more personalized recommendations, complying with the target to help users discover new content [4] and businesses may increase their sales [5]. This paper considerably extends our previous work [6] where we showed preliminary results only in terms of accuracy. In addition to recommendation accuracy, here we also consider individual diversity and aggregate diversity as key quality factors for the users' satisfaction. Moreover, we provide a comprehensive discussion of the results, considering the trade-off between the different evaluated quality dimensions. As a generic outcome, our experiments show that the proposed hybrid recommendation models, which exploit rating and semantic data, outperform content-based and collaborative filtering baselines, particularly in the most extreme cold-start scenarios.

2. Related Work

A user cold-start situation occurs when a RS does not have enough information about the interests or the past behaviour of a user to provide her good suggestions. Typically, users are in cold-start phase when they are new in the system. One of the solutions proposed so far to tackle such problem is the use of hybrid RSs, that

can take advantages of both content and collaborative information. Enrich et al. [7] proposed an extension of SVD++ that exploits social tags assigned to the items in order to improve the accuracy of the recommendations in a cold-start setting. Also focusing on the cold-start, A method that integrate information about the user's personality in the model's predictions has been proposed in [8]. Using cross-domain information has been demonstrated to be strongly effective for cold-start users [8], specially for domains strongly related like books and movies. Obviously, it requires information at least about the target users' interests in the source domain, namely the domain used for generating recommendation in the target domain. Another solution is the use of personality information, based on the assumption that users with similar personality traits have similar interests. Such information could be explicitly required to the users or inferred from their behaviour [9]. If cross-domain or personality information is not usable, preferences can be elicited by means of active learning methods, namely requiring directly cold-start users to provide some rating [9]. Finally, another solution, proposed in [10], regards the use of users social network content - e.g. Facebook friends and page likes.

3. Semantically Enriched Facebook Likes Dataset

The experimental evaluation presented in this paper is based on a Facebook dataset with user likes for book, movie and music items, which we extended with item metadata extracted from DBpedia. We comprehensively describe the dataset and the developed process to automatically extract DBpedia semantic networks relating items and features in [6]. Next a brief summary of the them is provided.

3.1. *Original Positive-only Feedback Data*

Our dataset initially consisted of a large set of *likes* assigned by users to items in Facebook. Using the Facebook Graph API, a user's like is retrieved in the form of a 4-tuple with the following information: the identifier, name and category of the liked item, and the timestamp of the like creation. The name of an item is given by the user who created the Facebook page of such item. As distinct names may exist for a particular item, users may express likes for different Facebook pages, which actually refer to the same item. Aiming at unifying and consolidating the items of the extracted Facebook likes, we developed a method, explained in [6], which automatically maps the items to the corresponding DBpedia entities, e.g., http://dbpedia.org/resource/The_Godfather for the identified names of The Godfather movie.

After filtering the items without a corresponding DBpedia resource, the final dataset contains 315870 likes provided by 1876 users on 4001 book pages, 1446017 likes provided by 26943 users on 3907 movie pages, and 1311974 likes provided by 49369 users on 5751 music pages. The data sparsity is more than 99% for all the three domains.

3.2. *Final Semantically Annotated Dataset*

For every linked entity, we finally accessed DBpedia to retrieve the entity metadata that afterwards would be used as input for the recommendation models. In this case, we launched a SPARQL query asking for all the properties and objects of the triples that have the target entity as subject. Since our ultimate goal is item recommendation, we only exploit metadata that may be relevant to relate common preferences of different users. Thus, we considered only certain properties in each domain. In particular, the complete list of DBpedia properties selected for each of the three domains of our dataset is showed in the our previous work [6].

3.3. *Semantically Enriched Item Profiles*

Fixing books, movies, and music artists and bands as the target items to be recommended, we can distinguish between three types of item metadata extracted from DBpedia. The first type of metadata is represented by the item attributes, e.g., the genre(s), director(s) and actors of a particular movie. Second, the item-item properties shown derive related items, e.g., the novel that a movie is based on (dbo:basedOn property), the prequel/sequel of a movie (dbo:previousWork / dbo:subsequentWork properties), and the musicians of a band (dbo:bandMember property). Finally, attribute-attribute properties generate **extended item attributes** that originally do not appear as metadata of the items, e.g., the subgenres of a particular music genre (dbo:musicSubgenre property). The above three types of item metadata constitute the semantically enriched item profiles that we propose to use in the recommendation models. In the conducted experiments, we note that the results achieved by exploiting the enriched profiles were better that those achieved by only using item attributes.

4. Evaluated recommendation models

In the following, we present the proposed graph-based and matrix factorization recommendation models that jointly exploit user rating and semantically enriched item profiles. The recommendation models used in this work jointly exploit user ratings, and item metadata automatically obtained from DBpedia semantic networks. The exploitation of these networks is done i) directly by means of graph-based models, and ii) indirectly by enriching item profiles that are incorporated into matrix factorization models. Hence, in the subsequent two sections we revise related work on graph-based and matrix factorization based recommender systems.

4.1. *Graph-based Models*

The importance of graph-based approaches to recommendation has emerged concurrently with the increasing availability of additional user and item information useful for the recommendation process itself. These approaches allow combining

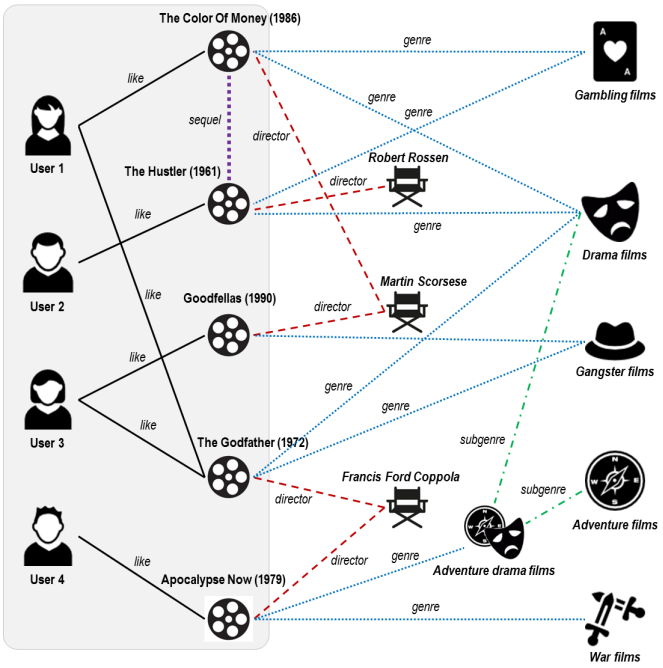

Fig. 1. Example of heterogeneous information network

the user-item rating matrix with side information into a graph, and then applying a graph mining technique. More specifically, as shown in Figure 1, the rating matrix is transformed into a bipartite graph component which consists of user and item nodes linked with rating/like edges extended to form a multipartite graph, including nodes representing additional entities, which are related to items. The graph also allows including other edges, representing e.g. contextual information for the ratings, social connections between users, and semantic relations between entities [11]. The result thus can be defined as a heterogeneous information network consisting of a multi-typed and multi-relational directed graph, with nodes and edges of different types [12].

Structuring all the available data in form of a graph leads to different advantages: (i) well-known graph-based algorithms can be used to develop hybrid recommender systems able to exploit the different types of information surfing the graph [13]; (ii) both content and collaborative aspects are represented in a uniform setting thus leveraging the multi-relational nature of the graph; (iii) the graph can be directly extended with information already available in the form of graphs, such as Linked Open Data ; (iv) exploring the graph jumping different hops could produce relevant but not obvious recommendations and also help on addressing the cold-start scenario, since exploring longer paths in the network could overcome the lack of connection information between users and items.

Given a graph G, our aim is to produce personalized recommendations leveraging the knowledge encoded in the graph. We describe our data by means of heterogeneous information network, which consists in a graph with different types of nodes and relations. Therefore, it is possible to find different paths among users and items composed by different types of relations. For example, an user may be connected to an item i by the relation (like, director, director^{-1})[f], which basically means that the user likes one or more items with same director of item i. More formally, these paths are called meta-paths and an actual sequence of nodes and relations, which generates the particular path, is called path instance [12].

HeteRec is a hybrid method based on matrix factorization that uses meta-path based latent features to represent the connectivity between users and items along different types of paths in a heterogeneous information network [13]. Briefly, HeteRec computes for each meta-path the relative diffused user preferences matrix extending the similarity measure PathSim [12] in order to include the user feedbacks. More formally, the user preference diffusion score between user u and item j, along a generic meta-path P, is defined as:

$$sim(u,j) = \sum_{i \in R(U)} \frac{2 \cdot r(u,i) \cdot |p_{i \to j} : p_{i \to j} :\in P|}{|p_{i \to i} : p_{i \to i} \in P| + |p_{j \to j} : p_{j \to j} \in P|} \tag{1}$$

where $p_{x \to y}$ is a path instance between the items x and y. Basically, this formula is a weighted sum of PathSim among the items in the user profiles and the target item j, where the numerator measures the connectivity defined by the number of path instances between them following P and the denominator represents the balance of their popularity in the graph, namely the number of path instances between themselves. Once the matrices are computed, HeteRec factorizes them with a low-rank matrix factorization technique. We found it infeasible in this case since the diffused user preferences matrices are usually dense. The truncation strategy can be used to keep the matrices sparse, reducing the amount of space and time consume [11], but could remove valuable information in the cold start scenario. Therefore, our model is directly based on the not factorized diffused user preferences matrices. Finally, the estimated user-item preference matrix is computed as the weighted sum of the different meta-path matrices: $R* = w_{P_1} R *_{P_1} + ... + w_{P_m} R*_{P_m}$, where m is the number of meta-paths, w_{P_1} and $R*_{P_1}$, respectively, the weight and the diffused user preferences matrix of i-th meta-path. HeteRec splits the users into clusters, and then computes the importance of each meta-paths with a learning-to-rank approach. As we face the user cold-start situation, clustering the users is impracticable with a few ratings and without additional information. Therefore, we compute the meta-paths weights globally for all the cold-start users.

PathRank is an extension of the Personalized PageRank algorithm able to exploit different paths on a heterogeneous graph during the random walk process [14]. At each iteration, the random walker has three options: *transition*, move to one

[f]Given a relation r going from x to y we denote with r^{-1} the relation going from y to x.

of adjacent nodes with probability w_trans ; *restart*, restart the random walk from one of the query nodes with probability $w_restart$; path following, considering one of the meta-paths with probability w_path. Therefore, PathRank vector $\vec{r}$ is computed as:

$$\vec{r} = w_{trans} M_G^T \vec{r} + w_{restart} \vec{t} + w_{path}(w_{P_1} M_{P_1}^T + ... + w_{P_m} M_{P_m}^T)\vec{r} \quad (2)$$

where M_G is the item-item transition matrix of the full graph G, M_{P_1} is the transition matrix of the i-th meta-path, $\vec{t}$ is is the teleport vector representing the recommendation query (user profile) initialized with $1/|R(u)|$ for each item in $R(u)$, 0 otherwise.

4.2. *Factorization-based Models*

Matrix factorization (MF) models are considered the state-of-the-art for collaborative filtering, and have been extensively studied in recent years [15]. These approaches gained most popularity in the context of the Netflix prize, and since then have been successfully used in many applications. Focusing on the rating prediction task, Funk [16] presented one of the first approaches that approximates the user-item rating matrix as the product of two low-rank matrices of user and item latent factors, respectively. Building on MF, Koren et al. [15] proposed the well-known SVD++ model. In this approach the user latent features are extended with additional parameters for each rated item. The motivation is that the information of whether a user chose or not to rate an item is also an indicator of her preferences, and should be taken into account in the rating prediction. Despite their success, the previous models were designed to deal with numeric, explicit ratings. However, typical feedbacks implicitly acquired by most real-world systems are positive-only and require different treatment. For such purpose, Hu et al. [17] presented a MF method that also models unobserved user-item interactions, as the lack of this information could indicate that the user dislikes the item or that she simply is unaware of it.

Basically, MF methods learn low-rank representations of the user-item matrix, deriving from the rating patterns some latent factors and then map both users and items to a joint latent factor space of dimensionality k, where k is usually much smaller then the size of the original user-item matrix. Therefore, the interactions between users and items are modeled as inner products in the latent factor space. More formally, each item i is associated with a vector $\boldsymbol{q_i} \in \mathbb{R}^k$, where each element in q_i indicate the extent to which the item possesses the corresponding factor. While each user u is associated with a vector $\boldsymbol{p_u} \in \mathbb{R}^k$, whose elements indicate the importance of the corresponding factor for the user. Finally, their scalar product captures the interaction between user u and item i and can be used for rating estimation as follows

$$r^*(u,i) = \boldsymbol{q_i^T p_u} \quad (3)$$

While the rating can be easily estimated once the model is computed, the major challenge remains to identify accurate mappings. Earliest implementations of matrix factorization methods relied on imputation techniques to remove the user-item matrix sparsity filling in the void cells, for instance with the average ratings for a user or for an item [18]. However, imputation increases the amount of data, making the computation more expansive, and moreover can lead to less accurate recommendations [15]. Recently, a number of MF methods that model directly the observed ratings only have been proposed. In the most common formulation of MF, to learn the model (i.e. q_i and p_u vectors) the systems aims at optimize a regularized squared error cost function, as follows

$$\min_{q^*,p^*} \sum_{(u,i)inK} (r(u,i) - r^*(u,i) + \lambda \left(\sum_u ||\boldsymbol{q_i}||^2 + \sum_i ||\boldsymbol{p_u}||^2 \right) \tag{4}$$

where K is the set of the (u,i) pairs representing all the known interactions in the user-item matrix, and the term λ controls the importance of the regularization term used to prevent overfitting. The choice of λ depends on the data and can be determined by cross validation. Usually, *Stochastic Gradient Descent* is used for minimizing that optimization problem and hence learning the factors, e.g. [16].

A specific matrix factorization method has been proposed in [19] for better handling implicit feedback, taking into account both observed and unobserved feedback in the training process. The motivation is that the model should not only be able to predict high scores for relevant items, but also whether an item was rated or not. However, non-observed user-item interactions can be due to the user not liking the item or not knowing about the item. Hence, the model includes a confidence hyperparameter $c_{(u,i)}$ in the loss function to penalize mistakes on observed and non-observed preference predictions differently

$$\min_{q^*,p^*} \sum_{(u,i)} c_{u,i}(p(u,i) - r^*(u,i))^2 + \lambda \left(\sum_u ||\boldsymbol{q_i}||^2 + \sum_i ||\boldsymbol{p_u}||^2 \right) \tag{5}$$

where $p(u,i)$ represents a binarized derivation of $r(u,i)$, that returns 1 if $r(u,i)$ is greater that 0, else returns 0.

The confidence parameter is set $c_{(u,i)} = 1 + \alpha r(u,i)$ with $\alpha > 0$, so that mistakes predicting observed feedback are more penalized. It is important to note that equation 5 considers all the possible (u,i) pairs, while the equation 4 uses only the known interactions in the user-item matrix. It is possible since $p(u,i)$ assumes 0 for all the unknown (u,i) pairs. The intuition behind this method is that the systems has minimal confidence in $p(u,i)$ for every user-item pair, but as it observes more evidence for positive preference, the confidence in $p(u,i)$ increases accordingly [19].

Collective Matrix Factorization (CMF) [20] is a representative matrix factorization method that originally showed significant improvements when item genres are taken into account for computing movie recommendations. The idea behind

CMF is to simultaneously factorize the user-item matrix and the item-item similarity matrix. Predictions are still computed using Equation 3, but CMF includes an additional set of item latent vectors $s_j \in \mathbb{R}^K$ feature to model the pairwise item interactions through the similarities. The loss function becomes:

$$\min_{q^*,p^*,r^*} \gamma \sum_{(u,i)} c_{u,i}(p(u,i) - r^*(u,i))^2 + (1-\gamma) \sum_i \sum_j (s_{i,j} - \boldsymbol{q_i^T s_j})^2 + \lambda \left(\sum_u ||\boldsymbol{q_i}||^2 + \sum_i ||\boldsymbol{p_u}||^2 + \sum_j ||\boldsymbol{s_j}||^2 \right) \tag{6}$$

where $s_{i,j}$ represents the content-based similarity between items i and j; γ (0,1] weights the importance of the item similarities in the factorization. If $\gamma = 1$ the result is the same of IMF, whereas γ close to 0 would ignore the preference predictions.

Factorization Machines (FMs) [21] are becoming increasingly popular, as they provide a principled and generic approach to integrate metadata into MF, showing promising results in the task of context-aware recommendation. Finally, a different set of approaches jointly factorize the user-item preference and item-metadata matrices, sharing the item latent factors between both decompositions. Therefore, Factorization Machines provide a generic way to extend the standard MF model with different kinds of side information. The idea is to (one-hot) encode the user/item metadata information in a single feature vector $x \in \mathbb{R}^{n=|U|+|I|+|F|}$ where $|U|, |I|, |F|$ are the number of users, items, and features, respectively. The model equation for a factorization machine of degree d = 2 is defined as:

$$r^*(u,i) = w_0 + \sum_{a=1}^{n} w_a x_a + \sum_{a=1}^{n} \sum_{b=a+1}^{n} \langle \boldsymbol{v_a}, \boldsymbol{v_b} \rangle \, x_a x_b \tag{7}$$

where the model parameters w_0, $\boldsymbol{w} \in \mathbb{R}^n$, $\boldsymbol{V} \in \mathbb{R}^{n \times k}$ have to be estimated, and $\langle \cdot, \cdot \rangle$ indicates the scalar product of two vectors. A row v_i in V represents the i-th variable with k factors. The w_a parameters model the contribution of each component in the feature vector, whereas the weights for the pairwise interactions are factorized as the product of two latent feature vectors v_a and v_b. FMs generalize all other MF models by taking in input any type of user/item metadata. Moreover, it has been demonstrated that the model equation 7 can be computed in linear time, hence parameters can be learned efficiently via stochastic gradient descent [21].

5. Experiments

In this section, we detail the setting and results of the experiments performed to evaluate the recommendation quality in the cold-start situation of graph-based and matrix factorization models (see Section 4) on the Facebook dataset (see Section 3), for three distinct domains (books, movies, and music). The goal is to evaluate the

effectiveness of considering jointly user likes and item metadata to produce accurate recommendations for cold-start users, and to compare the different approaches in this setting.

5.1. *Evaluation Methodology*

The evaluation of the proposed techniques was based on the TestItems evaluation methodology proposed in [22], using a modified user-based 5-fold cross-validation strategy proposed in [23] for the cold-start user scenario. First, we selected the users with at least 20 likes, shuffled and split them into five (roughly) equally sized subsets. In each cross-validation stage, we kept all the likes from four of the groups in the training set, whereas the likes from the users in the fifth group were randomly split into three subsets: training set (10 likes), validation set (5 likes), and testing (remaining likes, hence at least 5). In order to simulate different user profile sizes from one to ten likes, we repeat the training and the evaluation ten times, starting with the first like in the training set and incrementally increasing it one by one. This evaluation setting allows us to evaluate each profile size with the same test set, avoiding potential biases in the evaluation, since some accuracy metrics can be sensitive to the test set size [23]. To evaluate the ranking accuracy of the recommendations, we used Mean Reciprocal Rank (MRR), which computes the average reciprocal rank of the first relevant recommended item, and hence results particularly meaningful when users are provided with few but valuable recommendations (i.e., Top-1 or Top-3) [1]. As recommendation accuracy has been proved to be not enough to guarantee satisfying user experience, attention has been paid to other important quality factors such as individual diversity and aggregate diversity. The former is a list-wise property which indicates the ability of a system to provide diverse recommendations to each user. We used a intent-aware metric called BinomDiv [24] which measures the individual diversity also according to the user interests covered in the recommendation list. The latter is a systems-wise property that measures the coverage of the items catalog and the distribution of the items across the users. Hence we considered two metrics: catalog coverage (percentage of items recommended at least to once) and Entropy to analyse the items distribution [3].

5.2. *Baselines Models*

Besides the graph-based and factorization-based models presented in the previous section, we also evaluated a number of well-known content-based and collaborative filtering methods, and one hybrid method that integrates content similarity into user-based CF.

Popularity-based (POP). Non-personalized method that always recommends the most popular items not yet liked by the user.

Content-based (CB). Recommends the most similar items to those in the user profile. We compute the similarity between items as the cosine between their

TF-IDF feature vectors, obtained from the semantically-enriched item profiles.

User-based Nearest Neighbors (UNN). Estimates the score of candidate item i for target user u by aggregating the preferences of other similar users: $r^*(u,i) = \sum_{v \in N(u) \cap U(i)} sim(u,v)$. Here $N(u)$ is the set of u's k most similar users, and $U(i)$ the set of users that liked item i. For the user-user similarity we considered Jaccard's coefficient between the sets $I(u)$ and $I(v)$ of items liked by users u and v, respectively.

Item-based Nearest Neighbors (INN). The INN method works similarly to CB, with the difference that the item similarity is computed in a collaborative filtering fashion by exploiting the users' interactions rather than the item content. Specifically, we compute the score of item i for user u as $r^*(u,i) = \sum_{j \in I(u)} sim(i,j)$, where the item similarity is computed as the Jaccard coefficient between sets $U(i)$ and $U(j)$.

Content-based Collaborative Filtering (HYB). This method integrates content information into the UNN algorithm by replacing the user similarity component. In particular, we generate the TF-IDF profile vector for each user aggregating the content-based profile vectors of her liked items, computed as in the CB method. We compute the user-user similarities as the cosine between their corresponding profile vectors, and use the same formula as in UNN to compute the recommendation scores. Although the method relies on content-based similarities, it still follows the CF paradigm by exploiting the information from other users in the neighborhood.

Sparse Linear Methods (SLIM). Implementation of the SLIM method [25] available in MyMediaLite[g]. SSLIM refers to SLIM with side information [26].

5.3. *Results*

Table 1 shows the performance of the evaluated algorithms in the three domains in terms of MRR and BinomDiv, while Table 2 shows the results in terms of Catalog Coverage and Entropy, which together assess the aggregate diversity dimension.

5.3.1. *Books*

Accuracy. PathRank obtains the best accuracy in the case of users with 1 and 2 likes; PathRank, UNN and HYB are the best methods with 3 likes; while HeteRec is best method from 4 to 10 likes. It is worth to note that POP baseline beats most of the methods except PathRank with 1 like, and also UNN and HYB with 2 likes. SLIM and SSLIM seem not able to face the cold-start problem, especially with users who provide less than 5 likes.

Individual Diversity. FMs provide the most diversified recommendations in all the configurations, except in the case of users with only 1 like where UNN is better.

[g]http://www.mymedialite.net

Table 1. MRR and BinomDiv values for different cold-start target profile sizes.

		MRR										BinonDiv									
	#likes	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
Books	CB	.086	.120	.144	.160	.168	.178	.199	.198	.205	.214	.491	.489	.487	.487	.482	.487	.492	.492	.493	.485
	POP	.244	.246	.248	.251	.252	.255	.255	.261	.263	.266	.674	.690	.702	.703	.710	.724	.732	.738	.740	.746
	INN	.145	.177	.216	.241	.262	.277	.303	.318	.331	.35	.655	.674	.654	.665	.672	.657	.673	.674	.672	.670
	UNN	.222	.265	**.286**	.289	.29	.306	.314	.323	.329	.337	**.733**	.706	.716	.709	.729	.722	.707	.716	.716	.716
	HYB	.247	.253	.283	.286	.292	.308	.322	.333	.339	.349	.640	.647	.661	.659	.656	.671	.657	.666	.669	.671
	SLIM	.13	.111	.194	.215	.242	.286	.323	.323	.335	**.377**	.724	.703	.692	.713	.692	.719	.701	.738	.743	.694
	SSLIM	.115	.119	.199	.212	.247	.271	.289	.303	.315	.326	.706	.722	.685	.615	.695	.703	.710	.665	.649	.653
	IMF	.171	.194	.235	.255	.271	.29	.285	.299	.307	.324	.583	.606	.630	.645	.657	.665	.661	.662	.659	.674
	CMF	.175	.186	.249	.258	.251	.285	.302	.317	.327	.358	.601	.640	.664	.680	.683	.703	.690	.708	.706	.710
	FMs	.213	.224	.223	.233	.236	.24	.245	.257	.253	.276	.716	**.726**	**.749**	**.741**	**.733**	**.740**	**.755**	**.757**	**.755**	**.764**
	HeteRec	.218	.244	.279	**.297**	**.316**	**.331**	**.345**	**.353**	**.358**	.366	.609	.623	.653	.672	.680	.692	.695	.694	.692	.692
	PathRank	**.251**	**.271**	.285	.292	.295	.302	.305	.309	.313	.317	.630	.640	.650	.650	.680	.680	.680	.695	.708	.706
Movies	CB	.082	.107	.119	.135	.144	.154	.162	.169	.175	.182	.298	.293	.289	.281	.274	.263	.256	.249	.241	.236
	POP	.287	.289	.292	.294	.297	.299	.302	.305	.308	.311	.304	.336	.354	.368	.378	.386	**.393**	**.400**	**.405**	**.410**
	INN	.233	.301	.336	.359	.377	.39	.405	.415	.423	.433	.289	.308	.315	.321	.323	.327	.329	.332	.333	.337
	UNN	**.332**	.320	.318	.330	.348	.378	.397	.405	.416	.426	.360	**.385**	**.404**	**.392**	**.396**	**.394**	**.393**	.393	.396	.395
	HYB	.3	.322	.343	**.366**	**.382**	**.398**	**.413**	**.426**	**.434**	**.446**	.348	.381	.389	.373	.368	.367	.362	.362	.362	.361
	SLIM	.157	.173	.236	.280	.306	.333	.349	.372	.390	.413	**.392**	.366	.341	.347	.366	.387	.366	.370	.395	.406
	SSLIM	.159	.192	.249	.290	.311	.338	.361	.382	.394	.417	**.392**	.353	.363	.372	.384	.384	.361	.371	.380	.377
	IMF	.256	.291	.314	.334	.348	.364	.376	.389	.400	.413	.299	.320	.335	.344	.347	.355	.358	.363	.366	.368
	CMF	.257	.297	.315	.337	.352	.371	.382	.391	.402	.420	.288	.314	.329	.337	.338	.348	.355	.357	.358	.363
	FMs	.29	.321	.334	.35	.358	.368	.375	.386	.391	.399	.357	.338	.343	.348	.352	.357	.353	.354	.361	.362
	HeteRec	.315	**.346**	**.357**	**.366**	.374	.382	.388	.395	.401	.408	.311	.328	.334	.337	.341	.343	.346	.348	.350	.354
	PathRank	**.333**	.336	.337	.340	.344	.345	.350	.354	.357	.361	.317	.327	.336	.342	.352	.353	.359	.361	.366	.368
Music	CB	.113	.135	.151	.167	.178	.187	.198	.207	.215	.223	.284	.286	.271	.256	.242	.233	.225	.217	.212	.205
	POP	.337	.340	.342	.345	.347	.349	.352	.354	.357	.359	.228	.262	.282	.295	.305	.313	.321	.327	.333	.338
	INN	.320	.391	.426	.455	.474	.489	.504	.518	.532	.542	.200	.213	.219	.223	.229	.231	.235	.236	.239	.240
	UNN	**.422**	.389	.389	.419	.448	.485	.503	.519	.533	.546	.296	.332	.348	.347	.330	.317	.309	.305	.301	.297
	HYB	.356	.383	.413	.443	.469	.491	.505	.522	**.536**	.548	.237	.260	.275	.266	.265	.263	.260	.260	.257	.257
	SLIM	.193	.184	.293	.346	.388	.418	.44	.468	.491	.505	**.355**	.347	.322	.309	.358	.290	.368	.361	.350	.351
	SSLIM	.207	.249	.334	.377	.413	.435	.458	.477	.502	.524	.294	.324	.307	.330	.338	.331	.334	.324	.297	.371
	IMF	.347	.396	.427	.451	.471	.488	.503	.516	.532	.544	.196	.217	.232	.241	.249	.253	.256	.260	.261	.265
	CMF	.357	.397	.432	.456	.476	**.493**	**.509**	**.525**	**.536**	**.550**	.187	.210	.225	.234	.243	.245	.250	.250	.255	.258
	FMs	.394	**.427**	**.450**	**.467**	**.480**	**.493**	.504	.514	.524	.533	.344	.331	.315	.306	.309	.305	.310	.300	.306	.303
	HeteRec	.358	.395	.421	.442	.463	.481	.496	.513	.524	.535	.227	.264	.280	.288	.296	.300	.302	.304	.306	.306
	PathRank	.410	.416	.420	.424	.428	.432	.436	.440	.443	.447	.350	**.380**	**.395**	**.404**	**.410**	**.413**	**.416**	**.419**	**.421**	**.422**

Table 2. Catalog Coverage and Entropy values for different cold-start target profile sizes.

		Catalog Coverage										Entropy									
	#likes	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
Books	CB	.346	.359	**.345**	**.328**	**.315**	**.307**	**.295**	**.285**	**.277**	**.273**	9.78	10.05	9.93	9.80	9.73	9.63	9.55	9.48	9.42	9.38
	POP	.003	.003	.003	.003	.003	.003	.004	.004	.004	.004	3.37	3.40	3.43	3.46	3.48	3.51	3.54	3.56	3.58	3.60
	INN	.362	**.374**	.341	.314	.291	.272	.257	.237	.223	.211	9.59	9.56	9.17	8.90	8.70	8.51	8.35	8.14	8.00	7.87
	UNN	.117	.081	.073	.071	.068	.065	.062	.060	.058	.059	6.68	6.08	6.07	6.09	6.10	6.01	5.98	5.98	5.98	6.01
	HYB	.102	.091	.085	.080	.078	.076	.073	.072	.067	.069	6.76	6.59	6.56	6.51	6.51	6.51	6.45	6.44	6.39	6.41
	SLIM	.398	.310	.243	.186	.171	.148	.144	.133	.126	.115	11.24	10.89	10.57	10.20	10.05	9.79	9.69	9.54	9.42	9.30
	SSLIM	**.426**	.357	.284	.236	.207	.171	.159	.142	.134	.116	**11.44**	**11.17**	**10.78**	**10.46**	**10.36**	**10.07**	**9.92**	**9.75**	**9.65**	**9.46**
	IMF	.057	.054	.051	.049	.048	.048	.047	.045	.046	.045	7.18	6.97	6.82	6.74	6.68	6.64	6.60	6.58	6.58	6.59
	CMF	.098	.087	.078	.075	.071	.072	.070	.065	.069	.065	7.81	7.51	7.28	7.18	7.13	7.05	7.06	7.02	7.07	7.02
	FMs	.077	.086	.111	.109	.109	.111	.127	.118	.120	.120	6.46	6.82	7.29	7.35	7.40	7.39	7.66	7.55	7.57	7.60
	HeteRec	.245	.241	.223	.199	.175	.162	.141	.120	.102	.092	7.99	7.81	7.50	7.17	6.86	6.67	6.44	6.21	6.03	5.92
	PathRank	.182	.137	.113	.093	.082	.077	.069	.065	.058	.055	6.98	6.32	5.94	5.63	5.46	5.37	5.25	5.19	5.11	5.06
Movies	CB	.861	**.822**	**.771**	**.734**	**.698**	**.673**	**.651**	**.631**	**.612**	**.598**	5.48	5.38	5.26	5.17	5.09	5.03	4.97	4.93	4.89	4.85
	POP	.003	.003	.003	.004	.004	.004	.004	.004	.005	.005	1.83	1.85	1.86	1.88	1.89	1.91	1.92	1.93	1.95	1.96
	INN	.856	.650	.495	.394	.332	.292	.260	.239	.221	.206	4.84	4.30	4.00	3.81	3.69	3.59	3.52	3.47	3.42	3.38
	UNN	.185	.173	.180	.138	.122	.107	.101	.097	.095	.092	3.09	3.28	3.27	3.16	3.08	3.02	3.02	3.02	3.03	3.04
	HYB	.207	.196	.173	.158	.146	.135	.129	.123	.120	.117	3.53	3.53	3.45	3.40	3.37	3.35	3.34	3.33	3.33	3.33
	SLIM	.855	.680	.591	.526	.490	.462	.439	.427	.419	.398	**11.35**	**10.93**	**10.70**	**10.51**	**10.38**	**10.26**	**10.18**	**10.09**	**10.03**	**9.99**
	SSLIM	**.891**	.708	.572	.504	.460	.430	.410	.397	.381	.369	**11.35**	**10.93**	10.62	10.41	10.27	10.12	10.04	9.96	9.90	9.84
	IMF	.162	.156	.149	.141	.137	.134	.131	.130	.129	.128	4.29	4.15	4.05	3.99	3.95	3.93	3.92	3.91	3.91	3.91
	CMF	.142	.137	.132	.126	.122	.120	.114	.116	.111	.111	4.22	4.06	3.97	3.93	3.87	3.85	3.84	3.83	3.82	3.83
	FMs	.036	.195	.216	.224	.227	.235	.237	.247	.240	.244	2.28	3.52	3.65	3.74	3.78	3.82	3.86	3.89	3.91	3.93
	HeteRec	.450	.292	.216	.166	.134	.115	.098	.087	.077	.070	3.62	3.23	3.03	2.90	2.81	2.75	2.71	2.67	2.65	2.63
	PathRank	.098	.044	.025	.020	.017	.015	.014	.013	.013	.012	2.47	2.30	2.24	2.22	2.21	2.20	2.20	2.20	2.20	2.21
Music	CB	.867	**.835**	**.799**	**.762**	**.732**	**.698**	**.673**	**.649**	**.628**	**.608**	11.19	10.98	10.76	10.55	10.37	10.20	10.04	9.91	9.78	9.67
	POP	.002	.002	.002	.002	.003	.003	.003	.003	.003	.003	3.36	3.39	3.42	3.45	3.47	3.50	3.52	3.54	3.56	3.59
	INN	.841	.604	.469	.398	.351	.316	.294	.280	.266	.256	9.84	8.79	8.28	7.97	7.77	7.62	7.51	7.43	7.36	7.31
	UNN	.223	.207	.222	.171	.159	.146	.138	.137	.136	.135	7.03	7.30	7.09	6.75	6.61	6.56	6.59	6.61	6.62	6.65
	HYB	.290	.244	.209	.191	.176	.166	.158	.152	.147	.145	8.45	8.12	7.77	7.55	7.40	7.31	7.24	7.19	7.16	7.13
	SLIM	**.929**	.746	.658	.605	.550	.526	.520	.504	.495	.484	11.88	11.35	**11.14**	**10.94**	**10.78**	**10.70**	**10.62**	**10.57**	**10.51**	**10.46**
	SSLIM	.702	.645	.481	.428	.396	.376	.352	.340	.336	.322	**12.23**	**11.48**	10.99	10.73	10.56	10.45	10.36	10.29	10.24	10.19
	IMF	.146	.139	.131	.129	.125	.123	.121	.119	.119	.119	8.67	8.23	7.99	7.84	7.77	7.71	7.67	7.65	7.64	7.63
	CMF	.161	.142	.135	.128	.124	.124	.120	.120	.119	.119	8.72	8.24	7.96	7.84	7.77	7.72	7.68	7.66	7.66	7.65
	FMs	.051	.077	.093	.103	.112	.116	.122	.124	.130	.133	5.32	5.94	6.30	6.48	6.65	6.74	6.85	6.92	6.99	7.05
	HeteRec	.519	.426	.362	.316	.279	.253	.231	.214	.201	.189	8.33	7.69	7.25	6.95	6.74	6.58	6.45	6.36	6.28	6.21
	PathRank	.092	.041	.025	.021	.018	.016	.015	.014	.014	.014	4.81	4.41	4.29	4.25	4.22	4.22	4.22	4.22	4.22	4.23

Generally, POP, UNN and SLIM result close to FMs, while CB provides always the worst diversity.

Aggregate Diversity. In terms of catalog coverage, SSLIM is the best method with 1 like, INN with 2 likes, and CB for all the other sizes; while considering the entropy, SSLIM is always the best method, closely followed by SLIM. Generally, CB, INN, SLIM, and SSLIM provide the best values in term of both the two metrics, respect to all the other evaluated methods. POP is obviously the worst method, as it recommends only the most popular items.

Overall Analysis. Since accuracy, individual and aggregate diversity need to be considered together, here we discuss the results with a complete overview. None of the analysed methods is able to overcome all the others in terms of all the metrics. Analysing MRR and BinonDiv together, INN seems to achieve the best trade-off with almost all the profile size, followed by HeteRec from 5 to 10 likes. Analysing MRR and catalog coverage, INN and HeteRec provide the best trade-offs for all the profile sizes, followed by SLIM and SSLIM. While, considering the trade-off between MRR and entropy, SLIM and SSLIM provide the best results, followed by INN and HeteRec. Even though SLIM and SSLIM provide good catalog coverage and the best entropy values, it is important to note that their recommendation accuracy is very low for the most cold users (less than 5 likes). Summing up, graph-based methods obtain the best results in books domain with best accuracy and good coverage, but in different situations: PathRank is better with less likes (strong cold-start) but HeteRec overcomes it where more likes are available (weak cold-start). However, they provide less diversified list of recommendations (individual diversity) respect to most of the other methods. The three factorization-based models (IMF, CMF, FMs) are not particularly effective in this domain. We also notice that using metadata information leads to better recommendations. In particular, we can see that HYB beats UNN in seven out of ten cases. Moreover, CMF gives the same importance to user preferences and item metadata, obtaining the better accuracy with the trade-off parameter $\gamma = 0.5$.

5.3.2. *Movies*

Accuracy. PathRank and UNN provide the more accurate recommendations to users with 1 like, closely followed by HeteRec. Until 4 likes are available, HeteRec yields the best performance. As more likes are observed, the HYB method consistently outperforms the rest, closely followed by INN and UNN. Regarding CMF and SLIM, we observe a similar behavior to the books domain. CB is always the worst method, while SLIM and SSLIM are able to overcome POP only in the case of users with more than 4 likes. On a side note, as FMs beat IMF with few likes, item metadata results also valuable in MF-based models, especially in the most cold-start situations (likes less than 5).

Individual Diversity. UNN provides the most diversified list of recommendations in almost all the configurations, followed by HYB, SLIM and SSLIM. CB results

always the worst diversity, except for users with 1 like.
Aggregate Diversity. In terms of catalog coverage, SSLIM is again the best method with 1 like, while in the other configurations CB overcomes all the other methods, followed by SLIM, SLIM, INN. Recalling that POP should be always the worst method in terms of coverage, we note that PathRank provides results not significantly greater than POP and is hence unable to cover the catalog.
Overall Analysis. Again, none of the analysed methods results the best solution in all the quality dimensions. Analysing MRR and BinonDiv together, UNN seems to obtain the best trade-off, followed by HYB; while CB is always the worst choice. Among the factorization-based methods, IMF and FMs show very similar trends, while CMF provide slightly less diversified recommendation lists. While the two graph-based methods are the most accurate methods for users with less than 5 likes, providing individual diversity results on the average. Analysing MRR and catalog coverage, INN provides one of the best trade-offs between the two dimension considering almost all the profile sizes. However, SLIM and SSLIM result the best choice for users with more that 7 likes. Again, SLIM and SSLIM are the best methods when MRR and entropy are considered together. Once again we note that content information is especially beneficial in terms of accuracy, while the other quality dimension do not receive a particular improvement. We also observe that graph-based methods are better than MF-based approaches in terms of accuracy and individual diversity when very few likes are observed, but they strongly penalize the aggregate diversity.

5.3.3. *Music*

Accuracy. As for movies domain, UNN provides the most accurate suggestions for users with only 1 like. Then, FMs and CMF overcome almost all the other methods in the other profile size configurations. In particular, FMs result the best method from 2 to 6 likes, and CMF from 6 to 10. PathRank seems to be still competitive in the extreme cold-start scenario (1 and 2 likes). Therefore, we can confirm again that hybrid recommender systems are effective for generating accurate recommendations. In particular we see that CMF and FMs always overcome IMF, which use only collaborative information, and SLIM with side information (SSLIM) is better that SLIM.
Individual Diversity. In contrast with the other two domains, PathRank is able to overcome all the other methods in terms of BinomDiv. FMs is the best choice among the factorization-based methods, while the other two methods (IMF and CMF) provide the less diversified list of recommendations compared to all the other methods.
Aggregate Diversity. Again CB overcomes all the other methods in terms of of catalog coverage, except in the case of 1 like where SLIM is still the best method. SLIM follows CB from 2 to 10, while SSLIM and INN place themselves down. Conversely, all the three factorization-based methods and PathRank provide the

lowest values of catalog coverage.

Overall Analysis. PathRank, UNN and FMs show the best trade-off between MRR and BinonDiv from 1 to 5 likes; while SSLIM, UNN, and FMs seem generally to be the best methods from 5 to 10 likes, except from 8 to 10 likes, where SLIM provides the best trade-off. Analysing MRR and catalog coverage together, INN is again one of the best methods; when users have from 7 to 10 likes, SLIM results a good alternative to INN, with less accuracy but better catalog coverage. Considering the trade-off between MRR and entropy, INN, CMF, and IMF generally provide a good balance, in particular in the most extreme cold-start scenarios. With more than 7 likes, SLIM and SSLIM deserve to be taken into account, since they obtain very high entropy with slight loss of accuracy. We can conclude that matrix factorization models perform better in this domain and are also able to exploit item metadata, since CMF and FMs beat IMF in each configuration. In particular, CMF is able to adequately combine likes and item metadata. As the optimal trade-off parameter for CMF is 0.1 in this domain, this method gives more importance to the content information as opposed to user preferences, demonstrating that metadata is valuable in the cold-start scenario.

6. CONCLUSIONS AND FUTURE WORK

Providing relevant suggestions of items for cold-start users is a well-known problem in recommender systems. In this paper, we carried out a comparison of different hybrid recommendation methods that jointly exploit user ratings and semantic data descriptive of the items extracted from DBpedia. We evaluated graph-based and matrix factorization algorithms in the top-N recommendation task with positive-only feedback, using a Facebook likes dataset covering three distinct domains, in terms of not just accuracy, but also individual and aggregate diversity. The results demonstrated that by exploiting semantic information about the item, the proposed methods are able to provide relevant recommendations even for users with very few likes, hence addressing the cold-start problem. While individual and aggregate diversity do not always benefit of the integration of semantic information into hybrid approaches. In the future we will extend the analysis to other domains and other recommendation algorithms.

References

1. Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "CLiMF: Learning to maximize reciprocal rank with collaborative less-is-more filtering," in *Proceedings of the Sixth ACM Conference on RecSys*, pp. 139–146, ACM, 2012.
2. M. D. Ekstrand, F. M. Harper, M. C. Willemsen, and J. A. Konstan, "User perception of differences in recommender algorithms," in *Proceedings of the 8th ACM Conference on RecSys*, pp. 161–168, ACM, 2014.
3. G. Adomavicius and Y. Kwon, "Improving aggregate recommendation diversity using ranking-based techniques," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, pp. 896–911, 2012.

4. S. Vargas and P. Castells, "Improving sales diversity by recommending users to items," in *Eighth ACM Conference on RecSys*, pp. 145–152, 2014.
5. D. Fleder and K. Hosanagar, "Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity," *Management science*, vol. 55, no. 5, pp. 697–712, 2009.
6. P. Tomeo, I. Fernández-Tobías, T. Di Noia, and I. Cantador, "Exploiting linked open data in cold-start recommendations with positive-only feedback," in *CERI '16*, 2016.
7. M. Enrich, M. Braunhofer, and F. Ricci, "Cold-start management with cross-domain collaborative filtering and tags," in *EC-Web '13*, pp. 101–112, 2013.
8. I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi, "Cross-domain recommender systems," in *Recommender Systems Handbook*, pp. 919–959, Springer, 2015.
9. I. Fernández-Tobías, M. Braunhofer, M. Elahi, F. Ricci, and I. Cantador, "Alleviating the new user problem in collaborative filtering by exploiting personality information," *User Model. User-Adapt. Interact.*, vol. 26, no. 2-3, pp. 221–255, 2016.
10. S. Sedhain, S. Sanner, D. Braziunas, L. Xie, and J. Christensen, "Social collaborative filtering for cold-start recommendations," in *Proceedings of the 8th ACM Conference on RecSys*, pp. 345–348, ACM, 2014.
11. C. Shi, X. Kong, Y. Huang, P. S. Yu, and B. Wu, "Hetesim: A general framework for relevance measure in heterogeneous networks.," *CoRR*, vol. abs/1309.7393, 2013.
12. Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," in *In VLDB 11*, 2011.
13. X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in *Proceedings of the 7th ACM WSDM*, pp. 283–292, ACM, 2014.
14. S. Lee, S. Park, M. Kahng, and S.-g. Lee, "Pathrank: A novel node ranking measure on a heterogeneous graph for recommender systems," in *Proceedings of the 21st ACM CIKM*, CIKM '12, pp. 1637–1641, ACM, 2012.
15. Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, pp. 30–37, Aug. 2009.
16. S. Funk, "Netflix update: Try this at home," in *http://sifter.org/ simon/journal/20061211.html*, 2006.
17. Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *ICDM '08*, pp. 263–272, 2008.
18. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, "Application of dimensionality reduction in recommender system - a case study," in *In ACM WebKDD Workshop*, 2000.
19. Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pp. 263–272, IEEE Computer Society, 2008.
20. A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proceedings of the 14th ACM SIGK KDD*, pp. 650–658, ACM, 2008.
21. S. Rendle, "Factorization machines," in *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pp. 995–1000, IEEE Computer Society, 2010.
22. A. Bellogin, P. Castells, and I. Cantador, "Precision-oriented evaluation of recommender systems: An algorithmic comparison," in *ACM RecSys '11*, pp. 333–336, 2011.
23. D. Kluver and J. A. Konstan, "Evaluating recommender behavior for new users," in *RecSys '14*, pp. 121–128, 2014.
24. S. Vargas, L. Baltrunas, A. Karatzoglou, and P. Castells, "Coverage, redundancy and size-awareness in genre diversity for recommender systems," in *RecSys '14*, pp. 209–

216, 2014.
25. X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pp. 497–506, IEEE Computer Society, 2011.
26. X. Ning and G. Karypis, "Sparse linear methods with side information for top-n recommendations," in *Proceedings of the Sixth ACM Conference on RecSys*, pp. 155–162, ACM, 2012.