Exploiting Linked Open Data in Cold-start Recommendations with Positive-only Feedback

Paolo Tomeo[1], Ignacio Fernández-Tobías[2], Tommaso Di Noia[1], Iván Cantador[2]

[1]Politecnico di Bari
Via Orabona, 4
70125 Bari, Italy
{paolo.tomeo, tommaso.dinoia}@poliba.it

[2]Universidad Autónoma de Madrid
Calle Francisco Tomás y Valiente, 11
28049 Madrid, Spain
{ignacio.fernandezt, ivan.cantador}@uam.es

ABSTRACT

In recommender systems, user preferences can be acquired either explicitly by means of ratings, or implicitly –e.g., by processing text reviews, and by mining item browsing and purchasing records. Most existing collaborative filtering approaches have been designed to deal with numerical ratings, such as the *5-star ratings* in Amazon and Netflix, for both rating prediction and item ranking (a.k.a. top-N recommendation) tasks. In many e-commerce and social network sites, however, user preferences are usually expressed in the form of binary and unary (positive-only) ratings, such as the *thumbs up/down* in YouTube and the *likes* in Facebook, respectively. Moreover, in these cases, the well-known problem of cold-start –i.e., the scarcity of user preferences– is highly remarkable. To address this situation, we explore a number of graph-based and matrix factorization recommendation models that jointly exploit user ratings and item metadata. In this work, such metadata are automatically obtained from DBpedia –the queriable and structured version of Wikipedia which is considered as the core knowledge repository of the Linked Open Data initiative–, and the models are evaluated with a Facebook dataset covering three distinct domains, namely books, movies and music. The results achieved in our experiments show that the proposed hybrid recommendation models, which exploit rating and semantic data, outperform content-based and collaborative filtering baselines.

CCS Concepts

• Information systems→Collaborative filtering • Computing modelologies→Semantic networks • Mathematics of computing→Graph algorithms • Computing methodologies→ Non-negative matrix factorization.

Keywords

Recommender systems, hybrid recommendation, cold-start, positive-only feedback, Facebook, Linked Data, DBpedia.

1. INTRODUCTION

Recommender systems are information filtering systems that aim to identify and suggest the items –e.g., products, events, and contacts– a user may like or be interested in without the need of an explicit query, as commonly done in information retrieval systems. For such purpose, they capture, model and exploit user preferences. These latter 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. While content-based filtering methods suggest items that are similar to those the target user liked in the past, collaborative filtering methods suggest items liked by people with similar preferences to the target user. The former commonly use content-based features to represent both user and item profiles; the latter, in contrast, work with rating-based user/item similarities, and thus do not rely on machine analyzable content.

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[1] and Netflix[2], for both rating prediction and item ranking (a.k.a. top-N recommendation) tasks, and have been shown to generally outperform content-based approaches [21]. 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[3] and the *likes* in Facebook[4], respectively.

Moreover, in these cases, the well-known problem of cold-start in collaborative filtering [21], 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.

Hence, to address the presented situation –i.e., the cold-start in recommendations with positive-only feedback–, in this paper we explore 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 positive-only user feedback.

The above mentioned sites do not provide the content-based features that comprise the 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

CERI '16, June 14-16, 2016, Granada, Spain
© 2016 ACM. ISBN 978-1-4503-4141-7/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2934732.2934745.

[1] Amazon online shopping, http://www.amazon.com

[2] Netflix movie and TV series streaming, http://www.netflix.com

[3] YouTube online video sharing, http://www.youtube.com

[4] Facebook online social network, http://www.facebook.com

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 [2], the Wikipedia[5] ontology which is considered as the core repository of the Linked Open Data (LOD) [4] cloud.

In fact, the LOD initiative[6] aims at using the Web to connect related pieces of data, information, and knowledge about a variety of domains – such as geography, life sciences, government, and media, to name a few– using URIs (Uniform Resource Identifiers) via RDF[7] statements. 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.

We evaluate the two approaches on a Facebook dataset comprising three distinct domains, namely books, movies and music. The results achieved in our experiments show that the proposed hybrid recommendation models, which exploit rating and semantic data, outperform content-based and collaborative filtering baselines.

2. RELATED WORK

As mentioned before, the proposed recommendation models 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 (Section 2.1) and matrix factorization (Section 2.2) recommender systems.

2.1 Graph-based Recommender Systems

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 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 [19]. 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 [23].

[5] Wikipedia online encyclopedia, http://www.wikipedia.org

[6] Linked Open Data project, http://linkeddata.org

[7] Resource Description Framework, http://www.w3.org/RDF

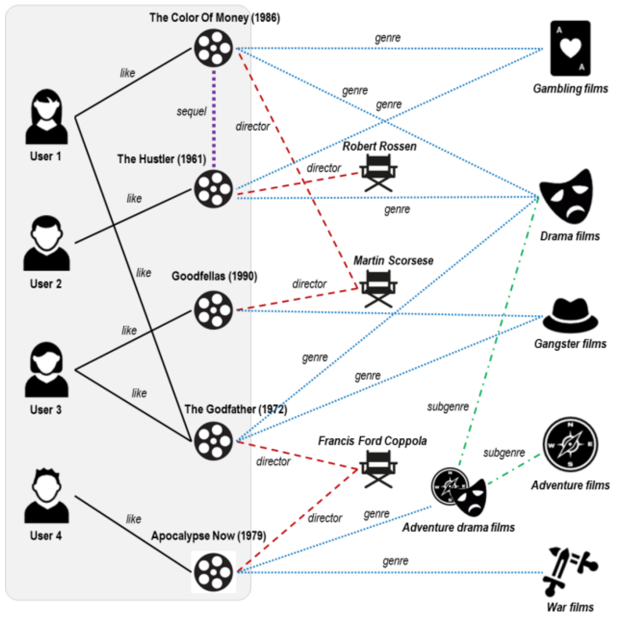

Figure 1. Example of heterogeneous information network

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 [24]; (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 [7]; (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.

PathRank [15] is an extension of the Personalized PageRank algorithm able to exploit different paths on a heterogeneous graph during the random walk process. At each iteration the random walker has three options: *transition*, move to one of adjacent nodes; *restart*, restart the random walk from one of the query nodes; *path following*, considering one of meta-paths that the authors call path-guides. A meta-path is a path consisting of a sequence of typed relations.

HeteRec [24] 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. HeteRec defines a user preference diffusion score extending the meta-path based similarity PathSim [23], including the user implicit feedback. This process propagates user preferences along the different meta-paths in the graph, producing a user-item matrix for each meta-path where each cell indicates the probability of certain user reaches a certain item under the relative meta-path. Then, it factorizes each matrix, and builds a recommendation model that estimates the rating for a user-item pair computing a weighted sum of the relative latent features in the matrices.

SPrank (Semantic Path-based ranking) [7] is a hybrid recommendation algorithm able to combine ontological knowledge belonging to the Web of Data with collaborative user preferences in a unified graph-based data model in a learning to rank setting.

Table 1. Considered item types and their DBpedia and YAGO classes for the three domains of the dataset. These classes were linked with the dataset items by the dbo:type and rdf:type properties in DBpedia, e.g., *dbr:The_Godfather* (movie) – *rdf:type* – *dbo:Film*

Books		Movies		Music	
Item type	***DBpedia class***	***Item type***	***DBpedia class***	***Item type***	***DBpedia class***
Book	dbo:Book	*Movie*	dbo:Film	*Composition*	dbo:Song
	yago:Book102870092		yago:Movie106613686		dbo:MusicalWork
	yago:Book102870526	*Genre*	dbo:MovieGenre		dbo:Single
Genre	yago:LiteraryGenres		yago:FilmGenres		dbo:ClassicalMusicComposition
Writer	dbo:Writer	*Director*	yago:FilmDirector110088200		dbo:Opera
	yago:Writer110794014		yago:Director110014939	*Genre*	dbo:MusicGenre
Fictional character	dbo:FictionalCharacter	*Actor*	dbo:Actor		yago:MusicGenres
	yago:FictionalCharacter109587565		yago:Actor109765278		yago:MusicGenre107071942
		Fictional character	dbo:FictionalCharacter	*Album*	dbo:Album
			yago:FictionalCharacter109587565		yago:Album106591815
				Musician	dbo:MusicalArtist
					yago:Musician110339966
					yago:Musician110340312
					yago:Composer109947232
				Band	dbo:Band
					yago:MusicalOrganization108246613

Namespaces

dbo:	http://dbpedia.org/ontology/
yago:	http://dbpedia.org/class/yago/
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
dbr:	http://dbpedia.org/resource/

2.2 Matrix Factorization Collaborative Filtering Systems

Matrix factorization (MF) models are considered the state-of-the-art for collaborative filtering, and have been extensively studied in recent years [14]. 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 [10] 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. The decomposition is obtained by minimizing the regularized squared loss of the actual observed ratings and the approximations computed with the latent feature matrices, e.g. using stochastic gradient descent. This method is efficient and scalable, as the rating matrix is usually very sparse and only the available ratings are taken into account.

Building on MF, Koren et al. [13] 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. This approach was shown to significantly outperform the standard matrix factorization model, and was part of the winning solution of the Netflix prize.

Despite their success, the previous models were designed to deal with numeric, explicit ratings. However, the typical feedback implicitly acquired by most real-world systems is positive-only and requires different treatment. For such purpose, Hu et al. [11] 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. Hence, this approach works by factorizing the *full* rating matrix, which is computationally very expensive. The authors propose an Alternating Least Squares procedure to learn the model parameters in an efficient way, and show the superiority of this approach in the top-N recommendation task.

In parallel with these developments, some approaches have explored hybrid models that exploit user or item metadata within the MF framework. Enrich et al. [8] 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, Fernández-Tobías and Cantador [5] presented a method that extends Hu et al.'s approach to exploit information about the user's personality in the model's predictions. Alternatively, Factorization Machines [22] 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. Collective Matrix Factorization (CMF) [18] is a representative method of this approach that showed significant improvements when item genres are taken into account for computing movie recommendations.

In this paper, we evaluate MF methods for positive-only feedback that make use of semantic information extracted from Linked Open Data to improve the quality of the recommendations in the user cold-start. Moreover, we present an adaptation of CMF that exploits semantic-based item-item similarities in the task of top-N recommendation of books, movies, and music.

3. OBTAINING ITEM METADATA

The recommendation models evaluated in this paper jointly exploit user ratings and item metadata. Before presenting them, we first describe the Facebook *likes* (positive-only feedback) dataset utilized in the experiments, and how it was enriched with item metadata automatically acquired from DBpedia.

Hence, in Section 3.1 we depict the Facebook user preference raw data that constituted our original dataset. Then, in Section 3.2 we explain the method we implemented to automatically link the items of the original dataset with entities existing in DBpedia. Finally, in Sections 3.3 and 3.4 we describe the metadata extracted from DBpedia for the linked entities, and the item profiles enriched with such metadata.

Table 2. DBpedia properties considered as item metadata; "item" can be book, movie and composition, musician and band

Relation	***DBpedia property***	***Relation***	***DBpedia property***	***Relation***	***DBpedia property***
item – genre	dct:subject	*item – author*	dbo:author	*music item – album*	dbo:album
	dbo:genre		dbo:creator	*band – musician*	dbo:bandMember
book – genre	dbo:literaryGenre	*book – writer*	dbo:writer		dbo:formerBandMember
music genre – music genre	dbo:musicSubgenre	*movie – actor, character, director*	dbo:starring		dbo:musicalBand
	dbo:musicFusionGenre		dbo:cinematography		dbo:associatedBand
	dbo:movement		dbo:director	*item – item, character*	dbo:series
	dbo:derivative	*composition – musician*	dbo:artist	*item – character*	dbo:portrayer
	dbo:stylisticOrigin		dbo:composer	*item – item*	dbo:basedOn
			dbo:musicComposer		dbo:previousWork
			dbo:musicalArtist		dbo:subsequentWork
			dbo:associatedMusicalArtist		dbo:notableWork

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, e.g., `{id: "35481394342", name: "The Godfather", category: "Movie", created_time: "2015-05-14T12:35:08+0000"}`. The name of an item is given by the user who created the Facebook page of such item. In this context, distinct names may exist for a particular item, e.g., `"The Godfather"`, `"The Godfather: The Movie"`, `"The Godfather - Film series"`, and `"The Godfather (saga)"` for "The Godfather" movie. Users thus 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 Section 3.2, which automatically maps the items names with the unique URIs of the corresponding DBpedia entities, e.g., `http://dbpedia.org/resource/The_Godfather` for the identified names of "The Godfather" movie.

3.2 Linking Items to DBpedia Entities

Within the Semantic Web initiative, the Linked Data (LOD) project leads the extension of the Web with a global data space connecting diverse semantic entities, such as famous people, organizations, books, movies, music compositions, and reviews, to name a few. Moreover, the consolidation of specialized data storage and information retrieval technologies –e.g., the SPARQL[8] RDF query language and the Apache Fuseki[9] server– allows accessing LOD similarly to how a relational database is queried today.

Among the datasets existing in the Linked Data cloud, DBpedia plays the role of a knowledge hub thus connecting many other data repositories. It is the LOD version of Wikipedia[10] and, as of March 2016, its knowledge base describes 4.58M things, including 1,4M people, 735K places, 411K creative works, and 241K organizations. For each of these things, DBpedia gathers metadata obtained from structured data of the corresponding Wikipedia webpage. Such metadata are represented as 3-tuples (commonly referred as *triples*) of the form [subject → property → object], e.g., the [`dbr:The_Matrix`, `dbo:director`, `dbr:The_Wachowskis`] triple represents that "The Matrix" movie was directed by the Wachowskis brothers, where `dbr:` and `dbo:` are respectively the abbreviations of `http://dbpedia.org/resource/` and `http://dbpedia.org/ontology` namespaces.

As mentioned before, in this work we linked *liked* items in Facebook with their corresponding DBpedia entities, in order to obtain item metadata with which we can investigate semantic-based collaborative filtering approaches on positively-only feedback. This was done as follows.

Given a particular item, we first identified the DBpedia entities that are labelled with the name of the item. For such purpose, we launched a SPARQL query targeted on the subjects of triples that have `rdfs:label` as property (where `rdfs:` stands for the `http://www.w3.org/2000/01/rdf-schema#` namespace) and the item title as object. The next query is an example for "The Matrix 2" title.

```
SELECT DISTINCT ?item WHERE {
  {
    ?item rdf:type dbo:Film .
    ?item rdfs:label ?name .
    FILTER regex(?name,"the.*matrix.*2","i") .
  }
  UNION
  {
    ?item rdf:type dbo:Film .
    ?tmp dbo:wikiPageRedirects ?item .
    ?tmp rdfs:label ?name .
    FILTER regex(?name,"the.*matrix.*2","i") .
  }
}
```

To resolve ambiguities in those names that correspond to multiple items belonging to different domains, we specify the type of item we wanted to retrieve in each case. Specifically, the query includes a triple clause with `rdf:type` (or `dbo:type`) as property, being `rdf:` the `http://www.w3.org/1999/02/22-rdf-syntax-ns#` namespace. Hence, in the given example, the subject "The Matrix 2" refers to the "Movie" type, which is associated to the `dbo:Film` class in DBpedia. The item types were set from the item categories provided in Facebook (see Section 3.1), and their associated DBpedia and YAGO[11] classes were identified by manual inspection of the `rdf:type` values of several entities. Table 1 shows the list of item types and DBpedia/YAGO classes we considered for the three domains of our dataset.

[8] http://www.w3.org/TR/rdf-sparql-query

[9] http://jena.apache.org/documentation/fuseki2

[10] http://www.wikipedia.org

[11] http://www.mpi-inf.mpg.de/yago-naga/yago

Table 3. Dataset statistics. Underlined items are the ones considered as the target items to be recommended in each domain

books			*movies*			*music*			*Statistics*	**books**	**movies**	**music**
item type	*#items*	*#ratings*	*item type*	*#items*	*#ratings*	*item type*	*#items*	*#ratings*	#users	1876	26943	49369
books	4001	315870	movies	3907	1446017	artists	2903	1311974	#items	3557	3901	5748
genres	71	21285	actors	1293	309957	bands	2848	1288673	#likes	42869	876501	2084462
writers	395	14397	characters	120	40855	genres	202	143829	sparsity	0.994	0.992	0.993
characters	76	3560	genres	19	32746	albums	216	51061	avg #items per users	22.851	32.532	42.222
			directors	56	7577	compositions	205	39792	avg #users per items	12.052	224.686	362.641

Moreover, running the previous query template we observed that a number of items were not linked to DBpedia entities because the labels corresponded to Wikipedia redirection webpages. In these cases, to reach the appropriate entities the query makes use of the `dbo:wikiPageRedirects` property.

The result of the above query for "The Matrix 2" name is:

`http://dbpedia.org/resource/The_Matrix_Reloaded`

which actually is the DBpedia entity of the second movie in "The Matrix" saga. Here, it is important to note that thanks to the Wikipedia page redirect component we are able to link items whose names do not have a direct syntactic match with the label of its DBpedia entity, but with the label of a redirected entity, e.g., the "Matrix 2" title matches with "The Matrix Reloaded" entity.

3.3 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. Following the example given before, such a query would be:

```
SELECT ?p ?o WHERE {
  dbr:The_Matrix_Reloaded ?p ?o .
}
```

This query returns all the DBpedia property-value pairs of the `dbr:The_Matrix_Reloaded` entity. However, since our ultimate goal is item recommendation, we should only exploit metadata that may be relevant to relate common preferences of different users. Thus, we filtered the query results by considering certain properties in each domain. Specifically, Table 2 shows the list of DBpedia properties selected for each of the three domains of our dataset. Hence, for example, for the movie items, we would have as metadata the movies genres, directors, and actors, among others. The items and relations shown in the table thus represent a *semantic network* that is automatically obtained from DBpedia for each particular domain.Table 3 shows statistics of the dataset for the three domains of interest, namely books, movies, and music. The statistics are focused on the number of users, items and ratings from the positive-only feedback side, and the number of properties and triples from the item metadata side.

3.4 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. First, the reminder items appearing in the extracted DBpedia semantic networks, and shown in Table 2, can be considered as **item attributes**, e.g., the genre(s), director(s) and actors of a particular movie. Second, the item-item properties shown in Table 2 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. We note that they differ from the commonly used content-based item profiles composed of (plain) attributes. We also note that in the conducted experiments, the results achieved by exploiting the enriched profiles were better that those achieved by only using item attributes.

4. RECOMMENDATION MODELS

In the following, we present the proposed graph-based (Section 4.1) and matrix factorization (Section 4.2) recommendation models that jointly exploit user rating and semantically enriched item profiles. We also present a number of baseline recommendation models (Section 4.3) that were evaluated for comparison purposes.

4.1 Graph-based Models

Given a graph G, our aim is to produce personalized recommendations leveraging the knowledge encoded in the graph. As described in Section 2.1, 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})*[12], 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* [23].

HeteRec. We develop a graph-based recommender system based on an adaptation of HeteRec (Section 2.1). Briefly, HeteRec computes for each meta-path the relative diffused user preferences matrix extending the similarity measure PathSim [23] 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:

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

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.

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

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 [19], 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: $\hat{R} = w_{P_1}\hat{R}_{P_1} + \cdots + w_{P_m}\hat{R}_{P_m}$, where m is the number of meta-paths, w_{P_i} and $\hat{R}_{P_i}$, 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. We also implemented a graph-based algorithm presented in [15], which extends the Personalized PageRank considering the connectivity between users and items along different meta-paths. At each iteration, the random walker has three options: *transition*, move to one 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 the PathRank vector $\vec{r}$ is computed as:

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

where M_G is the item-item transition matrix of the full graph G, M_{P_i} is the transition matrix of the i-th meta-path, $\vec{t}$ 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 Matrix Factorization Models

Methods based on matrix factorization approximate the user-item preference matrix as the product of two user and item latent factor matrices. Specifically, for each user u, there is a corresponding latent feature vector $\mathbf{p}_u \in \mathbb{R}^K$, where K is the number of features to consider in the factorization. Likewise, each item i is associated to its corresponding feature vector $\mathbf{q}_i \in \mathbb{R}^K$. These feature vectors are assumed to capture the latent interests and properties of both users and items. All the three MF methods that we consider in this paper follow the above principle, but differ on the preferences that are estimated, and on the training procedure user to learn the model parameters.

Matrix Factorization for positive-only feedback (IMF). In Hu et al.'s method [11], the preference of user u to item i is computed as the dot product of their latent feature vectors:

$$\hat{r}_{u,i} = \langle \mathbf{p}_u, \mathbf{q}_i \rangle \tag{1}$$

The model parameters are automatically learned by minimizing the associated regularized squared loss over the full user-item matrix, i.e. both observed and unobserved feedback are taken into account 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 in the loss function to penalize mistakes on observed and non-observed preference predictions differently:

$$L = \sum_u \sum_i c_{u,i}\left(r_{u,i} - \hat{r}_{u,i}\right)^2 + \lambda\left(\sum_u \|\mathbf{p}_u\|^2 + \sum_i \|\mathbf{q}_i\|^2\right)$$

The confidence parameter is set $c_{u,i} = 1 + \alpha \cdot r_{u,i}$ with $\alpha > 0$, so that mistakes predicting observed feedback are more penalized. The hyperparameter λ controls the amount of regularization used to prevent overfitting. For efficiently minimizing L the authors propose an algorithm based on Alternating Least Squares (ALS). The key observation is that when all the $\mathbf{q}_i$ parameters are fixed (respectively $\mathbf{p}_u$), the minimization problem becomes a standard least-squares problem that can be solved analytically. We refer the reader to [11] for details on the specific learning algorithm.

Collective Matrix Factorization (CMF). We instantiated the CMF method to inject item-item content similarities into the IMF method. The idea behind CMF [18] is to simultaneously factorize the user-item matrix and the item-item similarity matrix. Predictions are still computed using Equation (1), but we include an additional set of item latent feature vectors $\mathbf{r}_j \in \mathbb{R}^K$ to model the pairwise item interactions through the similarities. The loss function becomes:

$$L = \gamma \sum_u \sum_i c_{u,i}\left(r_{u,i} - \hat{r}_{u,i}\right)^2 + (1-\gamma)\sum_i \sum_j \left(s_{i,j} - \langle \mathbf{q}_i, \mathbf{r}_j \rangle\right)^2 + \lambda\left(\sum_u \|\mathbf{p}_u\|^2 + \sum_i \|\mathbf{q}_i\|^2 + \sum_j \|\mathbf{r}_j\|^2\right)$$

Here $s_{i,j}$ is the content-based similarity between items i and j. The tradeoff parameter $\gamma \in (0,1]$ controls the relative influence of the item similarities in the factorization. When $\gamma = 1$ we recover IMF, whereas $\gamma = 0$ would completely ignore preference predictions and thus we avoid this configuration. As previously, we learn the model parameters using ALS, which now involves an extra step fixing both all the $\mathbf{p_u}$'s and the $\mathbf{q_i}$'s to optimize the $\mathbf{r}_j$'s. Due to lack of space, we omit here the derivation of the parameter update rules and the specific derivation of the learning algorithm.

Factorization Machines (FMs). Factorization machines [22] 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 $\mathbf{x} \in \mathbb{R}^{n=|U|+|I|+|F|}$, where $|U|, |I|, |F|$ are the number of users, items, and features, respectively. The prediction formula depends on interactions of the components up to a certain degree d, which in the case of $d = 2$ becomes

$$\hat{r}_{u,i} = w_0 + \sum_{a=1}^{n} w_a x_a + \sum_{a=1}^{n} \sum_{b=a+1}^{n} \langle \mathbf{v}_a, \mathbf{v}_b \rangle x_a x_b$$

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 $\mathbf{v}_a$ and $\mathbf{v}_b$. Factorization machines generalize IMF by also taking into account user-feature and item-feature interactions, which are also factorized. In this work we use the implementation available in GraphLab (`http://graphlab.com`), and refer the reader to [22] for more details on the training algorithms.

4.3 Baseline Models

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.

Table 4. Mean Reciprocal Rank of the evaluated methods for different cold-start profile lengths in books, movies, and music.

	books										movies										music									
Profile length	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
CB	.086	.120	.144	.160	.168	.178	.199	.198	.205	.214	.082	.107	.119	.135	.144	.154	.162	.169	.175	.182	.113	.135	.151	.167	.178	.187	.198	.207	.215	.223
IMF	.171	.194	.235	.255	.271	.290	.285	.299	.307	.324	.256	.291	.314	.334	.348	.364	.376	.389	.400	.413	.347	.396	.427	.451	.471	.488	.503	.516	.532	.544
INN	.145	.177	.216	.241	.262	.277	.303	.318	.331	.350	.233	.300	.336	.359	.377	.390	.405	.415	.423	.433	.320	.391	.426	.455	.474	.489	.504	.518	.532	.542
POP	.244	.246	.248	.251	.252	.255	.255	.261	.263	.266	.287	.289	.292	.294	.297	.299	.302	.305	.308	.311	.337	.340	.342	.345	.347	.349	.352	.354	.357	.359
UNN	.222	.265	**.286**	.289	.290	.306	.314	.323	.329	.337	**.332**	.320	.318	.330	.348	.378	.397	.405	.416	.426	**.422**	.389	.389	.419	.448	.485	.503	.519	.533	.546
HYB	.247	.253	**.283**	.286	.292	.308	.322	.333	.339	.349	.300	.322	.343	**.366**	**.382**	**.398**	**.413**	**.426**	**.434**	**.446**	.356	.383	.413	.443	.469	.491	.505	.522	**.536**	**.548**
SLIM	.130	.111	.194	.215	.242	.286	.323	.323	.335	**.377**	.157	.173	.236	.280	.306	.333	.349	.372	.390	.413	.193	.184	.293	.346	.388	.418	.440	.468	.491	.505
SSLIM	.115	.119	.199	.212	.247	.271	.289	.303	.315	.326	.159	.192	.249	.290	.311	.338	.361	.382	.394	.417	.207	.249	.334	.377	.413	.435	.458	.477	.502	.524
FMs	.213	.224	.223	.233	.236	.240	.245	.257	.253	.276	.290	.321	.334	.350	.358	.368	.375	.386	.391	.400	.394	**.427**	**.450**	**.467**	**.480**	**.493**	.504	.514	.524	.533
CMF	.175	.186	.249	.258	.251	.285	.302	.317	.327	.358	.257	.297	.315	.337	.352	.371	.382	.391	.402	.420	.357	.397	.432	.456	.476	**.493**	**.509**	**.525**	**.536**	**.550**
HeteRec	.218	.244	.279	**.297**	**.316**	**.331**	**.345**	**.353**	**.358**	.366	.315	**.346**	**.357**	**.366**	.374	.382	.388	.395	.401	.408	.358	.395	.421	.442	.463	.481	.496	.513	.524	.535
PathRank	**.251**	**.271**	**.285**	.292	.295	.302	.305	.309	.313	.317	**.333**	.336	.337	.340	.344	.345	.350	.354	.357	.361	.410	.416	.420	.424	.428	.432	.436	.440	.443	.447

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 described in section 3.4.

User-based Nearest Neighbors (UNN). Estimates the score of candidate item i for target user u by aggregating the preferences of other similar users: $\hat{r}_{u,i} = \sum_{v \in N(u) \cap U(i)} \mathrm{sim}(u, v)$. Here $N(u)$ is the set of u's k most similar users, and $U(i)$ the set of users that liked item i. We note that in this work we deal with binary feedback for item ranking and thus drop normalization constants and centering ratings to the mean. 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 $\hat{r}_{u,i} = \sum_{j \in I(u)} \mathrm{sim}(i, j)$, 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 [16] available in MyMediaLite[13]. SSLIM refers to SLIM with side information [17].

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.

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

5.1 Evaluation Methodology

The evaluation of the proposed techniques was based on the *TestItems* evaluation methodology proposed in [3], using a modified user-based 5-fold cross-validation strategy proposed in [12] 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 have been proven to be sensitive to the test set size [12]. To evaluate the ranking accuracy of the recommendations, we used Precision, Recall, and Mean Reciprocal Rank (MRR). The latter 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) [20]. However, we only show MRR results in this paper, since Precision and Recall ones have similar trends. We also used the Catalog Coverage to better understand the differences among the compared algorithms; it represents the number of items in the catalog that have been recommended at least once [1].

5.2 Results

Table 4 shows the performance of the evaluated algorithms in the three domains in terms of MRR.

Books. 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. Moreover, PathRank reaches the best catalog coverage (>20%) after HeteRec (>27%) and INN (>40%) with 1 and 2 likes; while HeteRec keeps the second position in almost all the cases. SLIM and SSLIM seem not able to face the cold-start problem, especially with 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). 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 $\gamma = 0.5$.

Movies. PathRank is again the strongest method for 1 like, closely followed by UNN and HeteRec. We note nonetheless that the coverage of the latter is much higher (>45% compared to about 10%). Until 4 likes are available, HeteRec yields the best performance, still maintaining good coverage. As more likes are observed, the HYB method consistently outperforms the rest, with coverage in the range of 11.7% to 14.6%. We see that the performance of INN in the same setting is close to HYB, however providing much better catalog coverage (20.7% to 33.2%). On a side note, as FMs beat IMF with few likes, item metadata results also valuable in MF-based models. Regarding CMF and SLIM, we observe a similar behavior to the books domain. In summary, once again we conclude that content information is especially beneficial in the most extreme cold-start. As more likes are available, content-based collaborative filtering (HYB) also provides the best accuracy. We also observe that graph-based methods are better than MF-based approaches when very few likes are observed, even though the latter also benefit from content information.

Music. With only a like in the user profile, UNN is the best method; PathRank results quite close but with worse coverage (9.1% against 22.3%). From 2 to 6 likes, FMs emerges the best option; again PathRank obtains a high MRR with 2 likes but with lowest coverage (4% against 7.7%). From 7 t o10, CMF obtains the best accuracy values. Again, SLIM seems a weak method for cold start users, also using side information. In terms of coverage, HeteRec and INN have the best values (from ~18% to ~84%), whereas CMF has a coverage value around 15%. We can conclude that matrix factorization models perform better in this domain and are also able to exploit item metadata, since CMF and FMs beats 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 item metadata extracted from Linked Data. 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. The results demonstrated that by exploiting item metadata, the proposed methods are able to provide relevant recommendations even for users with very few likes, hence addressing the cold-start problem. Moreover, graph-based methods were more effective than matrix factorization approaches in books and movie recommendations, whereas the latter provided better performance in the music domain. We conjecture that this is due to the importance of collaborative information in this domain, and to the ability of matrix factorization models to better balance the impact of item metadata. In contrast, we argue that graph-based methods more effectively exploit content information. More extensive experiments are needed in order to confirm our hypothesis. In the future we will extend the analysis to the cross-domain recommendation task [5], leveraging item metadata to support the transfer of knowledge between the domains. Additionally, we plan to analyze other important quality factors, such as recommendation diversity [6].

Acknowledgements. This work was supported by the Spanish Ministry of Science and Innovation (TIN2013-47090-C3-2).

7. REFERENCES

[1] Adomavicius, G., Kwon, Y. 2012. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5), 896–911.

[2] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z. 2007. DBpedia: A Nucleus for a Web of Open Data. *6th International Semantic Web Conference*, 722-735.

[3] Bellogin, A., Castells, P., and Cantador, I. 2011, Precision-oriented evaluation of recommender systems: An algorithmic comparison. *5th ACM Conference on Recommender Systems*, 333–336.

[4] Bizer, C., Heath, T., Berners-Lee, T. 2009. Linked Data - The Story So Far. *Journal on Semantic Web and Information Systems* 5(3), 1-22.

[5] Cantador, I., Fernández-Tobías, I., Berkovsky, S., Cremonesi, P. 2015. Cross-Domain Recommender Systems. *Recommender Systems Handbook (2nd edition)*, 919-959.

[6] Di Noia, T., Ostuni, V.C., Rosati, J., Tomeo, P., Di Sciascio, E. 2014. An Analysis of Users' Propensity toward Diversity in Recommendations. *8th ACM Conference on Recommender Systems*. 285-288.

[7] Di Noia, T., Ostuni, V.C., Tomeo, P., Di Sciascio, E. 2016. SPRank: Semantic Path-based Ranking for Top-N Recommendations using Linked Open Data. *ACM TIST* (to appear), 30 pages.

[8] Enrich, M., Braunhofer, M., Ricci, F. 2013. Cold-Start Management with Cross-Domain Collaborative Filtering and Tags. *14th Intl. Conference on E-Commerce and Web Technologies*, 101-112.

[9] Fernández-Tobías, I., Braunhofer, M., Elahi, M., Ricci, F., Cantador, I. 2016. Alleviating the New User Problem in Collaborative Filtering by Exploiting Personality Information. In *UMUAI* (to appear), 35 pages.

[10] Funk, S. 2006. Netflix Update: Try This At Home. `http://sifter.org/~simon/journal/20061211.html`

[11] Hu, Y., Koren, Y., Volinsky, C. 2008. Collaborative Filtering for Implicit Feed-back Datasets. *8th IEEE Conference on Data Mining*, 263–272.

[12] Kluver, D., Konstan, J.A. 2014. Evaluating Recommender Behavior for New Users. *8th ACM Conference on Recommender Systems*, 121–128.

[13] Koren, Y. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. *14th ACM Conference on Knowledge Discovery and Data Mining*, 426-434.

[14] Koren, Y., Bell, R. 2011. Advances in Collaborative Filtering. *Recommender Systems Handbook*, 145-186.

[15] Lee, S., Park, S., Kahng, M., Goo Lee, S. 2012. PathRank: A Novel Node Ranking Measure on a Heterogeneous Graph for Recommender Systems. *21st ACM Conf. on Information and Knowledge Management*, 1637-1641.

[16] Ning, X., Karypis, G., 2011. Slim: Sparse Linear methods for top-n recommender systems. ICDM, 497-506.

[17] Ning, X., Karypis, G., 2012. Sparse Linear methods with Side Information for Top-N recommender systems. *ACM RecSys 2012*, 155-162.

[18] Singh, A. P., Gordon, G. J. 2008. Relational Learning Via Collective Matrix Factorization. *14th ACM Conference on Knowledge Discovery and Data Mining*, 650-658.

[19] Shi, C., Kong, X., Huang, Y., Philip, S.Y., Wu, B. 2013. HeteSim: A General Framework for Relevance Measure in Heterogeneous Networks. *IEEE Trans. on Knowledge and Data Engineering* 26(10), 2479-2492.

[20] Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., Hanjalic, A. 2012. CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-more Filtering. *ACM RecSys 2012*, 139-146.

[21] Shi, Y., Larson, M., Hanjalic, A. 2014. Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges. *Journal ACM Computing Surveys* 47(1), No. 3.

[22] Rendle, S. 2010. Factorization Machines. *10th IEEE International Conference on Data Mining*, 995-1000.

[23] Sun, Y., Han, J., Yan, X., Yu, P. S., Wu, T. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *2011 Conference on Very Large Database Endowment*, 992-1003.

[24] Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., Khandewal, U., Norick, B., Han, J. 2014. Personalized Entity Recommendation: A Heterogeneous Information Network Approach. *7th ACM Conference on Web Search and Data Mining*, 283–292.