Feature Factorization for top-n Recommendation: from item rating to features relevance

Vito Walter Anelli, Tommaso Di Noia,
Eugenio Di Sciascio
Polytechnic University of Bari
Via E. Orabona, 4
Bari
{vitowalter.anelli,tommaso.dinoia,disciascio}@poliba.it

Pasquale Lops
University of Bari "Aldo Moro"
Via E. Orabona, 4
Bari
pasquale.lops@uniba.it

ABSTRACT

In the last decade, collaborative filtering approaches have shown their effectiveness in computing accurate recommendations starting from the user-item matrix. Unfortunately, due to their inner nature, collaborative algorithms work very well with dense matrices but show their limits when they deal with sparse ones. In these cases, encoding user preferences only by means of past ratings may lead to unsatisfactory recommendations. In this paper we propose to exploit past user ratings to evaluate the relevance of every single feature within each profile thus moving from a user-item to a user-feature matrix. We then use matrix factorization techniques to compute recommendations. The evaluation has been performed on two datasets referring to different domains (music and books) and experimental results show that the proposed method outperforms the matrix factorization approach performed in the user-item space in terms of accuracy of results.

ACM Reference format:
Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio and Pasquale Lops. 2017. Feature Factorization for top-n Recommendation: from item rating to features relevance. In *Proceedings of RecSysKTL Workshop @ ACM RecSys '17, August 27, 2017, Como, Italy, ,* 6 pages.
DOI: N/A

1 INTRODUCTION

Recent years have seen the flourishing of many and diverse recommendation techniques based on the collaborative information encoded in the user-rating matrix. Factorization techniques working in such matrix have proven their effectiveness in improving the performance of recommendation engines and are implemented in many industrial and commercial systems [1, 14]. State-of-art algorithms can capture complex non-linear or latent factors-based relationships between users and items and this results more effective in all those scenarios where several users partially overlap their ratings or, in other words, the user-rating matrix is less sparse. In order to overcome the limits of pure collaborative approaches, hybrid ones [4] have been proposed that encode also side information about the items, typically content-based. Hybrid recommender systems have widely proved to improve performances in terms of accuracy and diversity of results[15, 18, 25, 29]. Whenever available, descriptions of the items can be used as a valuable source of information to augment the knowledge injected in and exploited by the system to compute the recommendation list of items. In this direction, an interesting class of recommender systems is the so called semantics-aware [8] where the information describing items goes beyond text and keywords and is represented by categorical/ontological data. SA approaches make use of ontologies or encyclopedic sources to encode and exploit domain-specific knowledge and in the last years many approaches have been proposed [2, 17, 19]. More recently, thanks to the Linking Open Data initiative, many structured data have become freely available to represent the content of items in different knowledge domains and then feed recommendation engines [9].

As a general remark, we can say that most of the recommendation algorithms available in the literature focus on computing the relevance of a set of items with reference to the user profile. Recommendation algorithms are designed around the computation of a relevance score to an item by evaluating its similarity with reference to other items. Features composing the description of an item, whatever the source, are not considered per se in the recommendation process but are usually exploited to evaluate the similarity between items or users. We believe that more attention should be paid to modeling the recommendation problem with a focus on recommending features rather then items. Expanding an item in its features brings with it some interesting side effects. On the one hand, all features may represent relations that, e.g., latent factor models are not able to look at. On the other hand, features give us a new set of explicit connections between items to be exploited with collaborative filtering algorithms. Finally, recommending items via feature recommendation may lead to an easier generation of explanation for the recommended list of items. Unfortunately, moving from items to features is not that straight as in a forest of many features, most of them may result not relevant to a user. Moreover, once we design an algorithm able to compute a recommendation list of features, we have to go back to the items space, as the ultimate goal of a recommender systems is to suggest items to a user.

In this paper we present `FF` (for `Features Factorization`), a *top-N* recommendation algorithm relying on user's feature preferences and collaborative filtering information in the features space. The main goal of `FF` is to compute an ordered list of features preferred by the user and, starting from such list, to reassemble the relevance values of each returned feature to produce a *top-N* list of items to recommend. All the side information adopted by `FF`

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
RecSysKTL Workshop @ ACM RecSys '17, August 27, 2017, Como, Italy
© 2017 Copyright held by the owner/author(s). .
DOI: N/A

is retrieved from DBpedia, the cornerstone dataset of the Linked Data cloud. For each item in the user profile we retrieve its features by querying DBpedia thus having them as a set of entities. This avoids all problems related to synonymy and polysemy which usually occur when dealing with keyword-based features. By combining the popularity of a feature in the user profile and the ratings assigned to items it is part of, for each user we compute a pair containing the relevance of the feature and its inferred rating. The resulting matrix in the user-feature space is then manipulated via factorization techniques to compute, for each user, a ranked list of features which is in turn post-processed to produce the final list of recommendations. Experimental evaluations of FF on two datasets related to the domains of books and music show its effectiveness in terms of accuracy of results in very sparse settings.

The remainder of the paper is structured as follows. In the next section we report some related work on LOD-based and feature-based approaches to recommendation. We continue in Section 3 by introducing and describing FF. Experimental evaluations are presented in Section 4 while in Section 5 we present and discuss the corresponding results. Conclusion and future works close the paper.

2 RELATED WORK

Several works have tried to build recommender systems by exploiting Linked Open Data (LOD) as side information for representing users or items, in addition to the user preferences usually collected through user ratings. Such approaches usually rely on DBpedia, the *nucleus* which acts as a hub for most of the knowledge in the so-called LOD cloud. In the following we review the recent literature on both LOD-based recommender systems and approaches which leverage the relevance of single features in the user profile.

LOD-based RS. A detailed review of recommender systems leveraging Linked Open Data is presented in [8]. Properties gathered from DBpedia may be used for different tasks, i.e. to produce cross-domain recommendations [10], to build a multirelational graph for a graph-based recommender [27], or to generate effective natural-language recommendation explanations [22]. On the other hand, DBpedia properties may be used in different ways: 1) to define semantic similarity measures for providing more accurate recommendations [18, 23, 30]; 2) to deal with problems as the *limited content analysis* or *cold-start*, e.g. by introducing new relevant features to improve item representations [3, 33], or to cope with the increasing data sparsity [21]; 3) to improve the overall accuracy of a recommender [20, 29], or to provide a good balance between different recommendation objectives, such as accuracy and diversity [15, 21, 28].

Feature-based RS. Several works attempt to analyze the user purchasing behavior based on item features. In [35], products are represented using vectors of features, and a customer profile module computes the level of interest of the customer in product features as the ratio of features among the products purchased, and the product quantity purchased by that customer. Similarly, in [12] a feature-based recommender system for domains without enough historical data to effectively measure user or item similarities is presented. The authors build the system based on the idea that *users who bought items with specific features also buy items with the same or similar features*. A similar approach is proposed in [26], in which effective strategies to incorporate item features for *top-N* recommender systems are developed. In graph-based recommender systems, an interesting work was proposed in [13], in which recommendations are produced inferring user preferences, evaluating item-preferences and attribute-preferences. The paper points out the importance of the feature evaluation and a method is proposed, which exploits explicit feature ratings, named attributes. Recently, an interesting approach called Feature Preferences Matrix Factorization (FPMF) has been proposed in [24]. FPMF incorporates user feature preferences in a matrix factorization to predict user likes. It is worth to note that none of the previous mentioned approaches rely on features coming from the Linked Open Data cloud.

3 PROPOSED APPROACH

3.1 Motivation

This work aims at investigating the role of feature **rating** and **relevance** in the item rating process. The main intuition behind FF is that items can be handled as a collection of features on which the recommendation process is then performed. Hence, when users rate an item, they are actually expressing their preference over the whole collection. The item rating action can be then summarized as the non trivial attempt to choose an overall rate for the entire set. If we want to discover the contribution of each single feature in the evaluation, first of all, we need to unpack each item in its composing features. Then, by combining the overall popularity of each feature in the user profile (feature relevance) and the rating assigned to items containing that feature we may estimate the implicit rating the user is giving to that specific feature. In the evaluation of a movie, the user implicitly evaluates the director, the actors, the producer, the country in which the movie is set. Each feature has its own rating and a relevance degree, hence a recommender system should consider these factors.

The second observation we based our work on, is that the relevance of an item in the user profile cannot be entirely encoded in its rating as the single rating represents a degree of liking about the specific item. The relevance of the item within a collection is not explicitly encoded anywhere with reference to the user's view. Our assumption is that such item-relevance naturally influences feature-relevances and vice-versa.

In our model the user profile is not just a set of $\langle item, rating \rangle$ pairs but it contains information about the relevance of each feature composing the rated items and its estimated rating $\langle feature, relevance, rating \rangle$. In the following we will see principled methods to estimate both the user-feature rating and the user-feature relevance. Then, we focus the recommendation problem on the features composing the user profile. FF exploits a collaborative filtering step to get approximated information about the missing features in the users-features matrix and finally it combines the predicted ratings and relevance for each feature available in each item to compute a personalized ranked list of items.

3.2 Data Model

For a better understanding of the data we use to reshape the user profile as user-feature matrices, we first introduce the multidimensional graph we used to build them. As we can see from Figure 1

the user profile is built by considering information coming from both the user-item matrix and from DBpedia as external knowledge source. The graph-based nature of this latter one is exploited to identify features used to represent items. The knowledge encoded in Linked Data is represented as RDF labeled oriented graphs and the corresponding data model is based on the notion of triple $\langle subject, predicate, object \rangle$ where *predicate* represents the relation connecting the two entities *subject* and *object*. With reference to Figure 1, we have that each item in the catalog represents the subject of a triple $\langle i, p, e \rangle \in$ DBpedia. In order to catch the different knowledge encoded in the use of the same entity as object in triples with diverse predicates, in our model, we consider the chain $predicate - object$, (corresponding to $property - entity$, *pe* path in the knowledge graph) as a feature associated to the item i which in turn represents the subject of the corresponding triple.

Each item in the user profile is associated with a relevance function we denote with $\rho^{ui}(\cdot)$. Its value represents an estimation of how important is a particular item to the user u. Analogously, we have a value associated to each feature in the profile computed via the function $\rho^{uf}(\cdot)$ computing the relevance of the feature f (represented by the pair of property and entity *pe*) in the user profile. Actually, each feature is associated also with a rating $r^{uf}(\cdot)$ which is inferred by considering the rating of all the items containing f.

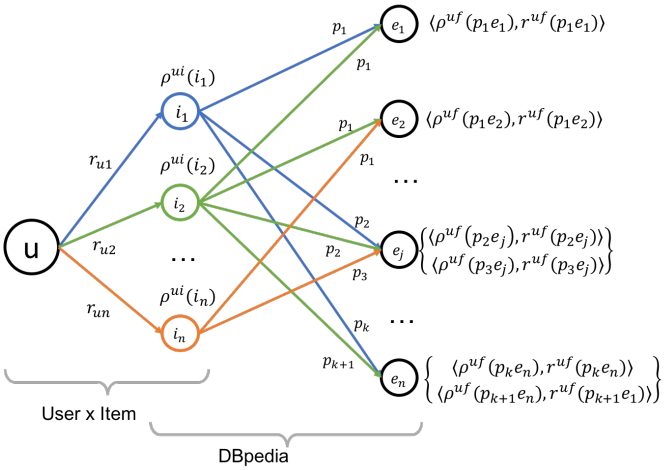

Figure 1: A graph-based representation of the data behind the computation of the user profile.

3.3 Problem Formulation

By considering the data associated to the user profile as described in the previous section we can move from a rating matrix connecting user and items to a user-feature matrix where each value is represented by the pair $\langle \rho^{uf}(\cdot), r^{uf}(\cdot) \rangle$. In other words, we may consider two user-feature matrices: the one $\mathcal{P}$ containing relevance values $\rho^{uf}(\cdot)$, the other $\mathcal{R}$ including the inferred ratings $r^{uf}(\cdot)$.

In FF, the relevance of a feature *pe* is computed as its probability of belonging to the set I_u representing the items already rated by a user u. More formally we have:

$$\rho^{uf}(pe) = \frac{\sum_{i \in I_u} |\{\langle i, p, e \rangle \mid \langle i, p, e \rangle \in \text{DBpedia}\}|}{|I_u|}$$

The idea behind this computation is quite straight: the more a feature is connected to the items in the user profile , the higher its relevance for the user.

Once we have computed the relevance of all the features in the user profile, we can move to the computation of the relevance for the items $i \in I_u$. This can be computed as the normalized summation of the relevance for all the features it is composed by. In formulas, we have

$$\rho^{ui}(i) = \frac{\sum_{\langle i, p, e \rangle \in \text{DBpedia}} \rho^{uf}(pe)}{|\{\langle i, p, e \rangle \mid \langle i, p, e \rangle \in \text{DBpedia}\}|}$$

Given a feature *pe*, the computation of the feature rating $r^{uf}(pe)$ exploits both the rating and the relevance of each item $i \in I_u$ containing *pe*.

$$r^{uf}(pe) = \frac{\sum_{\langle i, p, e \rangle \in \text{DBpedia}} r_{ui} \cdot \rho^{ui}(i)}{\sum_{\langle i, p, e \rangle \in \text{DBpedia}} \rho^{ui}(i)} \tag{1}$$

3.4 *top-N* Recommendation

The profiles we built contain only the features the user met before, but usually the number of those features is dramatically smaller than the overall number of features and this results in $\mathcal{P}$ and $\mathcal{R}$ being very sparse. In order to complete the information they contain, we compute, via Biased Matrix Factorization, the missing values $\hat{\rho}^{uf}(pe)$ for $\mathcal{P}$ and $\hat{r}^{uf}(pe)$ for $\mathcal{R}$. We run matrix factorization independently on $\mathcal{P}$ and $\mathcal{R}$. Biased Matrix Factorization is a matrix factorization model that minimizes RMSE using stochastic gradient descent [16]. It computes user's and item's biases to improve the estimation of the predicted value. Biased Matrix Factorization represents a state-of-the-art algorithm in rating prediction task. $\hat{\rho}^{uf}(pe)$ and $\hat{r}^{uf}(pe)$ represent the predicted relevance and the predicted rating for all those features not belonging to any of the items in I_u. As the resulting matrices contain both content-based and collaborative informations (due to the matrix factorization), we refer to them as *hybrid profile*.

With the *hybrid profile* we can estimate a ranked list for all the remaining items within the collection. In fact, the ranking of an item in the list is computed by considering the rating of the features belonging to the item and their relevance.

$$\begin{aligned} \hat{r}^{ui}(i) = & \sum_{(\langle i, p, e \rangle \in \text{DBpedia}) \wedge (i \in I_u)} \rho^{uf}(pe) \cdot r^{uf}(pe) + \\ & + \sum_{(\langle i, p, e \rangle \in \text{DBpedia}) \wedge (i \notin I_u)} \hat{\rho}^{uf}(pe) \cdot \hat{r}^{uf}(pe) \end{aligned} \tag{2}$$

It is important to point out that these estimations do not correspond to an actual rating but the correct item ranking is yet preserved.

3.4.1 Post-filtering. In order to improve the results of the final recommendation process, we propose a post-filtering step aimed at reducing the number of features considered while computing the final rank. The proposes filtering springs from the following observations:

- Not all the features items are relevant in the computation of the ranking for an item. All those features whose rating results low just introduce noise in the final values we compute.
- Feature ranking and relevance values evaluated via pure content-based approaches, i.e., before the matrix factorization, have a different influence if compared with the collaborative ones representing latent factors computed after the matrix factorization.

In order to lower the number of features involved in the computation and produce recommendations based only on the best ratings of the estimated features, we propose a filter that operates separately on directly estimated features (content-based) and estimated features coming from collaborative computation. We then introduce two thresholds α and β that act as filters on the feature rating values respectively in the content-based and in the collaborative cases. Hence, Equation (2) is slightly modified in

$$\hat{r}^{ui}(i) = \sum_{(\langle i,p,e\rangle \in \texttt{DBpedia}) \wedge (i \in I_u) \wedge r^{uf}(pe) > \alpha} \rho^{uf}(pe) \cdot r^{uf}(pe) + \\ + \sum_{(\langle i,p,e\rangle \in \texttt{DBpedia}) \wedge (i \notin I_u) \wedge \hat{r}^{uf}(pe) > \beta} \hat{\rho}^{uf}(pe) \cdot \hat{r}^{uf}(pe) \tag{3}$$

4 EXPERIMENTAL EVALUATION

In this section the experimental evaluation settings and the metrics used to evaluate the proposed algorithm are presented. We evaluated the algorithms in terms of ranking accuracy for *top-N* recommendations. The evaluation has been carried out on two datasets, `LibraryThing` and `Last.fm` belonging respectively to the domains of books and music. In order to remove the popularity bias from the evaluation results we removed the 1% most popular items [7]. Moreover we removed users with a number of ratings smaller than five as we want to evaluate the algorithms in a non cold start setting. The `LibraryThing` dataset contains 7,564 users, 39,515 items and 797,299 ratings. The minimum, mean and maximum number of ratings for user in the dataset are 20, 63, 3,018, respectively. `Last.fm` contains 1,892 users, 17,632 items and 92,834 ratings. In `LibraryThing`, ratings are distributed over a 1-10 scale. In `Last.fm` the rating is the number of times a song has been played, hence that number has been rescaled for each user in a 1-10 scale. Table 1 shows some statistics of the datasets subsets considering only the items mapped to `DBpedia` (using publicly available mappings [29]) after the pre-processing step. In case a mapping does not exist, a simple placeholder feature is used, that inherits the corresponding item values in terms of rating and relevance.

Table 1 also reports the sparsity values both for users-items and users-features matrices.

To evaluate `FF` we use the *all unrated items* [34] evaluation protocol, in which the ability to choose the correct set of items to propose to the users is favorite despite of the local ranking ability (*rated test-items* evaluation protocol). In *all unrated items* the recommendation list is produced using as candidate list the Cartesian product between users and item minus the items the user experimented in the training set. The evaluation has been conducted using a hold-out 80-20 splitting, in which 20% of the ratings are retained as test set.

`LibraryThing`	# users	# items	# ratings	sparsity (%)
user-item space	6,909	12,656	248,589	99.7157
	# users	# features	# ratings	sparsity (%)
user-feature space	6,909	141,531	8,680,619	99.11226
`Last.fm`	# users	# items	# ratings	sparsity (%)
user-item space	1,866	8,502	39,557	99.75066
	# users	# features	# ratings	sparsity (%)
user-feature space	1,866	274,523	4,989,281	99.02603

Table 1: Datasets Statistics.

We evaluated the accuracy of our approach by computing Precision ($P@N$), Recall ($R@N$) and nDCG ($nDCG@N$). Besides, as test-set could contain non-relevant documents, i.e. a low rated item we set a simple threshold in the 1 to 10 rating scale thus considering as relevant only the items that fall above it.

Baselines. In the experimental evaluation we compared `FF` with the popularity baseline (PopRank) and, as we rely on matrix factorization, the well known matrix factorization algorithm BPRMF [32] both in its pure collaborative version and in the hybrid one considering side information BPRMF+SI. We included also PopRank as it is acknowledged that popularity ranking can show good performance and it is an important baseline to compare against [7]. In order to produce recommendation lists from these well-known algorithms we used their *MyMediaLite*[1] implementation [11]. As for the selection of α and β parameters needed in Equation (3), in these experiments we kept a conservative approach and set respectively α to the mean μ of the rated items and β to the mean μ plus the standard deviation σ. Clearly, these values are not the optimal ones and the performances could be improved by a cross-validation setting of these parameters.

5 EXPERIMENTAL RESULTS

Tables 2 and 3 show the performance of `FF` compared with the competing algorithms described in Section 4. In **bold** we mark the best result for each metric. All the evaluations have been performed by using the same protocols as implemented in RankSys[2] library [6].

In Table 2 we show the evaluation results on `LibraryThing` dataset with a threshold set to 7/10 in a Top-10 recommendation list. The ranking accuracy performance, measured through nDCG, precision and recall shows that `Features Factorization` performs better than the competing algorithms. In details, `FF` performs 4 to 6 times better than BPRMF, the second best accurate algorithm, depending on the metrics.

As the rescaling operation in `Last.fm` affects the values of the items in the test set, we decided to perform evaluations considering all the items in test set as relevant (i.e. without any relevance threshold). Table 3 shows ranking accuracy evaluation results on `Last.fm` dataset with threshold of 0/10 for a Top-10 recommendation list. For precision metric the best performing algorithm is `FF` that performs 4 times better than BPRMF. For nDCG, `Features Factorization` performs at least 5 times better than the competing algorithms. The differences about accuracy metrics between `FF` and the other

[1] http://www.mymedialite.net/
[2] https://github.com/RankSys/RankSys

Alg	P@N	R@N	nDCG@N
FF	**0.03251**	**0.06576**	**0.06129**
BPRMF	0.00837	0.01280	0.01020
BPRMF+SI	0.00777	0.01325	0.01007
PopRank	0.00023	0.00095	0.00044

Table 2: Comparative results on LibraryThing dataset, Top-10 recommendation list and relevance threshold of 7/10.

Alg	P@N	R@N	nDCG@N
FF	**0.01543**	**0.02701**	**0.02330**
BPRMF	0.00348	0.00902	0.00495
BPRMF+SI	0.00032	0.00073	0.00028
PopRank	0.00027	0.00089	0.00021

Table 3: Comparative results on Last.fm dataset, Top-10 recommendation list and no relevance threshold.

algorithms are statistically significant according to the Student's paired *t-test* with $p < 0.001$ for every cases.

The differences in the behavior for the two datasets can be explained by looking at different dimensions of the Last.fm dataset (both the original and the feature-augmented one we used in our experiments). As for the original dataset, while for LibraryThing we used the original ratings of the user, in Last.fm we rescaled the users feedback represented as the number of times they played a song and normalized it in a 1-10 scale. This could have affected the final results especially in terms of accuracy. Indeed, the pure content-based feature ratings we predict highly depend on the original rating value (see Equation 1).

If we consider the feature-augmented dataset, by looking at the data represented in Table 1 the first observation we make is that the number of features in Last.fm is two order of magnitude higher than the number of items while in LibraryThing it is just one. Then, the decrease in performance of FF may be attributed also to the curse of dimensionality problem. Moreover, a deeper investigation on the quality of the adopted LOD dataset is needed. Recently, a few papers have been published on this topic [5, 36] but there is not yet a common view on the metrics to be adopted to evaluate the quality of the knowledge encoded in a Linked Data dataset and, more generally, in a knowledge graph.

6 CONCLUSION

In this paper we presented FF, a novel algorithm that bases on feature recommendation as an intermediate step for computing *top-N* items recommendation lists. The main idea behind FF is that feature relevance in a user profile plays a key role in the selection and rating of an item in a collection. Based on this observation we developed an algorithm that shifts the recommendation problem from a user-item space to a user-feature one. In this new space we introduced the explicit notion of feature relevance and feature rating and combined them with well known factorization techniques to perform a Features Factorization aimed at predicting a rating and a relevance for each feature unknown to the user. We compared FF with well known factorization techniques (both pure collaborative and hybrid with side information) on two datasets in the domains of books and music. In both datasets FF results the best algorithm in terms of recommending accurate items. This can be considered as a strong clue to confirm our intuition that recommending items via feature ranking is a feasible way to develop content-aware recommendation engines. As future work, we are investigating the behavior of FF with respect to novelty and diversity of results. We are also interested in exploring the behavior of FF approach with different collaborative filtering algorithms, other than factorization techniques in the item-feature space and in particular with Factorization Machines [31]. Moreover, since we collected content-based data from Linked Open Data datasets, an analysis on the influence of such datasets on the recommendation results is also in progress. Another aspect we are willing to deepen is related to results explanation. Indeed, very interestingly, item recommendation via feature ranking paves the way to new proposals for explanation services.

REFERENCES

[1] Robert M Bell and Yehuda Koren. 2007. Lessons from the Netflix prize challenge. *Acm Sigkdd Explorations Newsletter* 9, 2 (2007), 75–79.

[2] Yolanda Blanco-Fernandez, Jose J Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, and Martin Lopez-Nores. 2008. Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *IEEE Transactions on Consumer Electronics* 54, 2 (2008).

[3] Svetlin Bostandjiev, John O'Donovan, and Tobias Höllerer. 2012. TasteWeights: a visual interactive hybrid recommender system. In *Proceedings of the sixth ACM conference on Recommender Systems*. 35–42.

[4] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.

[5] Cinzia Cappiello, Tommaso Di Noia, Bogdan Alexandru Marcu, and Maristella Matera. 2016. A Quality Model for Linked Data Exploration. In *Web Engineering - 16th International Conference, ICWE*. 397–404.

[6] Pablo Castells, Neil J. Hurley, and Saul Vargas. 2015. *Novelty and Diversity in Recommender Systems*. Springer US, 881–918.

[7] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems (RecSys '10)*. ACM, New York, NY, USA, 39–46. DOI: https://doi.org/10.1145/1864708.1864721

[8] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. 2015. Semantics-Aware Content-Based Recommender Systems. In *Recommender Systems Handbook*. 119–159.

[9] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. 2012. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems*. ACM, 1–8.

[10] Ignacio Fernández-Tobías, Paolo Tomeo, Iván Cantador, Tommaso Di Noia, and Eugenio Di Sciascio. 2016. Accuracy and diversity in cross-domain recommendations for cold-start users with positive-only feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 119–122.

[11] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: a free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems (RecSys '11)*. ACM, New York, NY, USA, 305–308.

[12] Eui-Hong Han and George Karypis. 2005. Feature-based recommendation system. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management*. 446–452.

[13] Luheng He, Nathan Nan Liu, and Qiang Yang. 2011. Active Dual Collaborative Filtering with Both Item and Attribute Feedback.. In *AAAI*.

[14] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 43–50.

[15] Houda Khrouf and Raphaël Troncy. 2013. Hybrid Event Recommendation Using Linked Data and User Diversity. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, New York, NY, USA, 185–192. DOI: https://doi.org/10.1145/2507157.2507171

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[17] John Lees-Miller, Fraser Anderson, Bret Hoehn, and Russell Greiner. 2008. Does wikipedia information help netflix predictions?. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*. IEEE, 337–343.

[18] Rouzbeh Meymandpour and Joseph G Davis. 2015. Enhancing Recommender Systems Using Linked Open Data-Based Semantic Analysis of Items. In *Proceedings*

of the 3rd Australasian Web Conference (AWC 2015), Vol. 27. 11–17.

[19] Stuart E Middleton, Nigel R Shadbolt, and David C De Roure. 2004. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 54–88.

[20] Cataldo Musto, Pierpaolo Basile, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2014. Linked Open Data-enabled Strategies for Top-N Recommendations. In *Proceedings of the 1st Workshop on New Trends in Content-based Recommender Systems co-located with the 8th ACM Conference on Recommender Systems, CBRecSys@RecSys*.

[21] Cataldo Musto, Pierpaolo Basile, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2017. Introducing linked open data in graph-based recommender systems. *Information Processing & Management* 53, 2 (2017), 405–435.

[22] Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2016. ExpLOD: A Framework for Explaining Recommendations based on the Linked Open Data Cloud. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 151–154.

[23] C. Musto, G. Semeraro, P. Lops, M. de Gemmis, and F. Narducci. 2012. Leveraging Social Media Sources to Generate Personalized Music Playlists. In *E-Commerce and Web Technologies - 13th International Conference, EC-Web 2012*. 112–123.

[24] Mona Nasery, Matthias Braunhofer, and Francesco Ricci. 2016. Recommendations with Optimal Combination of Feature-Based and Item-Based Preferences. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, UMAP*. 269–273.

[25] Xia Ning and George Karypis. 2012. Sparse linear methods with side information for top-n recommendations. In *Proceedings of the sixth ACM conference on Recommender systems (RecSys '12)*. ACM, New York, NY, USA, 155–162.

[26] Xia Ning and George Karypis. 2012. Sparse linear methods with side information for top-n recommendations. In *Sixth ACM Conference on Recommender Systems, RecSys*. 155–162.

[27] Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. 2016. Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 9.

[28] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. 2017. Sound and Music Recommendation with Knowledge Graphs. *ACM TIST* 8, 2 (2017), 21:1–21:21.

[29] Vito Claudio Ostuni, Tommaso Di Noia, Eugenio Di Sciascio, and Roberto Mirizzi. 2013. Top-N Recommendations from Implicit Feedback Leveraging Linked Open Data. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, New York, NY, USA, 85–92. DOI: https://doi.org/10.1145/2507157.2507172

[30] Guangyuan Piao and John G Breslin. 2016. Measuring semantic distance for linked open data-enabled recommender systems. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. 315–320.

[31] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*. IEEE Computer Society, Washington, DC, USA, 995–1000. DOI: https://doi.org/10.1109/ICDM.2010.127

[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09)*. AUAI Press, Arlington, Virginia, United States, 452–461.

[33] Max Schmachtenberg, Thorsten Strufe, and Heiko Paulheim. 2014. Enhancing a Location-based Recommendation System by Enrichment with Structured Data from the Web. In *4th International Conference on Web Intelligence, Mining and Semantics WIMS*. 17:1–17:12.

[34] Harald Steck. 2013. Evaluation of recommendations: rating-prediction and ranking. In *RecSys*. 213–220.

[35] Sung-Shun Weng and Mei-Ju Liu. 2004. Feature-based recommendations for one-to-one marketing. *Expert Syst. Appl.* 26, 4 (2004), 493–508.

[36] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. 2016. Quality assessment for linked data: A survey. *Semantic Web* 7, 1 (2016), 63–93.