How to Make Latent Factors Interpretable by Feeding Factorization Machines with Knowledge Graphs

Vito Walter Anelli[1(✉)], Tommaso Di Noia[1], Eugenio Di Sciascio[1],
Azzurra Ragone[2], and Joseph Trotta[1]

[1] Polytechnic University of Bari, Bari, Italy
{vitowalter.anelli,tommaso.dinoia,eugenio.disciascio,
joseph.trotta}@poliba.it
[2] Milan, Italy
azzurraragone@gmail.com

Abstract. Model-based approaches to recommendation can recommend items with a very high level of accuracy. Unfortunately, even when the model embeds content-based information, if we move to a latent space we miss references to the actual semantics of recommended items. Consequently, this makes non-trivial the interpretation of a recommendation process. In this paper, we show how to initialize latent factors in Factorization Machines by using semantic features coming from a knowledge graph in order to train an interpretable model. With our model, semantic features are injected into the learning process to retain the original informativeness of the items available in the dataset. The accuracy and effectiveness of the trained model have been tested using two well-known recommender systems datasets. By relying on the information encoded in the original knowledge graph, we have also evaluated the semantic accuracy and robustness for the knowledge-aware interpretability of the final model.

1 Introduction

Transparency and interpretability of predictive models are gaining momentum since they been recognized as a key element in the next generation of recommendation algorithms. Interpretability may increase user awareness in the decision-making process and lead to fast (efficiency), conscious and right (effectiveness) decisions. When equipped with interpretability of recommendation results, a system ceases to be just a black-box [36,40,45] and users are more willing to extensively exploit the predictions [21,39]. Indeed, transparency increases their trust [17] (also exploiting specific semantic structures [16]), and satisfaction in using the system. Among interpretable models for Recommender Systems (RS), we may distinguish between those based on Content-based (CB) approaches

Authors are listed in alphabetical order.

and those based on Collaborative filtering (CF) ones. CB algorithms provide recommendations by exploiting the available content and matching it with a user profile [10,26]. The use of content features makes the model interpretable even though attention has to be paid since a CB approach "*lacks serendipity and requires extensive manual efforts to match the user interests to content profiles*" [46]. On the other hand, the interpretation of CF results will inevitably reflect the approach adopted by the algorithm. For instance, an item-based and a user-based recommendation could be interpreted, respectively, as "*other users who have experienced A have experienced B*" or "*similar users have experienced B*". Unfortunately, things change when we adopt more powerful and accurate Deep Learning [8] or model-based algorithms and techniques for the computation of a recommendation list. Such approaches project items and users in a new vector space of latent features [24] thus making the final result not directly interpretable. In the last years, many approaches have been proposed that take advantage of side information to enhance the performance of latent factor models. Side information can refer to items as well as users [43] and can be either structured [38] or semi-structured [6,9,47]. Interestingly, in [46] the authors argue about a new generation of knowledge-aware recommendation engines able to exploit information encoded in knowledge graphs (KG) to produce meaningful recommendations: "*For example, with knowledge graph about movies, actors, and directors, the system can explain to the user a movie is recommended because he has watched many movies starred by an actor*".

In this work, we propose a *knowledge-aware Hybrid Factorization Machine* (`kaHFM`) to train interpretable models in recommendation scenarios taking advantage of semantics-aware information (Sect. 2.1). `kaHFM` relies on Factorization Machines [29] and it extends them in different key aspects by making use of the semantic information encoded in a knowledge graph. With `kaHFM` we address the following research questions:

RQ1 Can we develop a model-based recommendation engine whose results are very accurate and, at the same time, interpretable with respect to an explicitly stated semantics coming from a knowledge graph?

RQ2 Can we evaluate that the original semantics of items features is preserved after the model has been trained?

RQ3 How to measure with an offline evaluation that the proposed model is really able to identify meaningful features by exploiting their explicit semantics?

We show how `kaHFM` may exploit data coming from knowledge graphs as side information to build a recommender system whose final results are accurate and, at the same time, semantically interpretable. With `kaHFM`, we build a model in which the meaning of each latent factor is bound to an explicit content-based feature extracted from a knowledge graph. Doing this, after the model has been trained, we still have an explicit reference to the original semantics of the features describing the items, thus making possible the interpretation of the final results. To answer RQ2, and RQ3 we introduce two metrics, Semantic Accuracy (SA@K) (Sect. 3.1) and Robustness (n-Rob@K) (Sect. 3.2), to measure the

interpretability of a knowledge-aware recommendation engine. The remainder of this paper is structured as follows: we evaluated `kaHFM` on two different publicly available datasets by getting content-based explicit features from data encoded in the `DBpedia` knowledge graph. We analyzed the performance of the approach in terms of accuracy of results (Sect. 4.1) by exploiting categorical, ontological and factual features (see Sect. 2.1). Finally, we tested the robustness of `kaHFM` with respect to its interpretability (Sects. 4.2 and 4.3) showing that it ranks meaningful features higher and is able to regenerate them in case they are removed from the original dataset.

2   Knowledge-Aware Hybrid Factorization Machines for Top-N Recommendation

In this section, we briefly recap the main technologies we adopted to develop `kaHFM`. We introduce Vector Space Models for recommender systems, and then we give a quick overview of Factorization Machines (FM).

Content-based recommender systems rely on the assumption that it is possible to predict the future behavior of users based on their personalized profile. Profiles for users can be built by exploiting the characteristics of the items they liked in the past or some other available side information. Several approaches have been proposed, that take advantage of side information in different ways: some of them consider tags [41], demographic data [49] or they extract information from collective knowledge bases [14] to mitigate the cold start problem [18]. Many of the most popular and adopted CB approaches make use of a Vector Space Model (VSM). In VSM users and items are represented by means of Boolean or weighted vectors. Their respective positions and the distance, or better the proximity, between them, provides a measure of how these two entities are related or similar. The choice of item features may substantially differ depending on their availability and application scenario: crowd-sourced tags, categorical, ontological, or textual knowledge are just some of the most exploited ones. All in all, in a CB approach we need (i) to get reliable items descriptions, (ii) a way to measure the strength of each feature for each item, (iii) to represent users and finally (iv) to measure similarities. Regarding the first point, nowadays we can easily get descriptions related to an item from the Web. In particular, thanks to the Linked Open Data initiative a lot of semantically structured knowledge is publicly available in the form of Linked Data datasets.

2.1   From Factorization Machines to Knowledge-Aware Hybrid Factorization Machines

Factorization models have proven to be very effective in a recommendation scenario [31]. High prediction accuracy and the subtle modeling of user-item interactions let these models operate efficiently even in very sparse settings. Among all the different factorization models, factorization machines propose a unified general model to represent most of them. Here we report the definition related

to a factorization model of order 2 for a recommendation problem involving only implicit ratings. Nevertheless, the model can be easily extended to a more expressive representation by taking into account, e.g., demographic and social information [4], multi-criteria [3], and even relations between contexts [50].

For each user $u \in U$ and each item $i \in I$ we build a binary vector $\mathbf{x^{ui}} \in \mathbb{R}^{1 \times n}$, with $n = |U| + |I|$, representing the interaction between u and i in the original user-item rating matrix. In this modeling, $\mathbf{x^{ui}}$ contains only two 1 values corresponding to u and i while all the other values are set to 0 (see Fig. 1). We then denote with $\mathbf{X} \in \mathbb{R}^{n \times m}$ the matrix containing as rows all possible $\mathbf{x^{ui}}$ we can build starting from the original user-item rating matrix as shown in Fig. 1.

	U_1	U_2	U_3	U_4	...	I_1	I_2	I_3	I_4	I_5	...
x^1	1	0	0	0	...	1	0	0	0	0	...
x^2	1	0	0	0	...	0	1	0	0	0	...
x^3	1	0	0	0	...	0	0	1	0	0	...
x^4	0	1	0	0	...	0	0	1	0	0	...
x^5	0	1	0	0	...	0	0	0	1	0	...
x^6	0	0	1	0	...	1	0	0	0	0	...
x^7	0	0	1	0	...	0	0	1	0	0	...
	User					**Item**					

Fig. 1. A visual representation of $\mathbf{X}$ for sparse real valued vectors $\mathbf{x^{ui}}$.

The FM score for each vector $\mathbf{x}$ is defined as:

$$\hat{y}(\mathbf{x^{ui}}) = w_0 + \sum_{j=1}^{n} w_j \cdot x_j + \sum_{j=1}^{n} \sum_{p=j+1}^{n} x_j \cdot x_p \cdot \sum_{f=1}^{k} v_{(j,f)} \cdot v_{(p,f)} \qquad (1)$$

where the parameters to be learned are: w_0 representing the global bias; w_j giving the importance to every single x_j; the pair $v_{(j,f)}$ and $v_{(p,f)}$ in $\sum_{f=1}^{k} v_{(j,f)} \cdot v_{(p,f)}$ measuring the strength of the interaction between each pair of variables: x_j and x_p. The number of latent factors is represented by k. This value is usually selected at design time when implementing the FM.

In order to make the recommendation results computed by `kaHFM` as semantically interpretable, we inject the knowledge encoded within a knowledge graph in a Factorization Machine. In a knowledge graph, each triple represents the connection $\sigma \xrightarrow{\rho} \omega$ between two nodes, named *subject* (σ) and *object* (ω), through the *relation (predicate)* ρ. Given a set of features retrieved from a KG [13] we first bind them to the latent factors and then, since we address a Top-N recommendation problem, we train the model by using a Bayesian Personalized Ranking (BPR) criterion that takes into account entities within the original knowledge graph. In [15], the authors originally proposed to encode a Linked Data knowledge graph in a vector space model to develop a CB recommender system. Given a set of items $I = \{i_1, i_2, \ldots, i_N\}$ in a catalog and their associated triples $\langle i, \rho, \omega \rangle$ in a knowledge graph $\mathcal{KG}$, we may build the set of all possible

features as $F = \{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathcal{KG} \text{ with } i \in I\}$. Each item can be then represented as a vector of weights $\mathbf{i} = [v_{(i,1)}, \ldots, v_{(i,\langle \rho, \omega \rangle)}, \ldots, v_{(i,|F|)}]$, where $v_{(i,\langle \rho, \omega \rangle)}$ is computed as the normalized TF-IDF value for $\langle \rho, \omega \rangle$ as follows:

$$v_{(i,\langle \rho, \omega \rangle)} = \underbrace{\frac{|\{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathcal{KG}\}|}{\sqrt{\sum\limits_{\langle \rho, \omega \rangle \in F} |\{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathcal{KG}\}|^2}}}_{TF^{\mathcal{KG}}} \cdot \underbrace{\log \frac{|I|}{|\{j \mid \langle j, \rho, \omega \rangle \in \mathcal{KG} \text{ and } j \in I\}|}}_{IDF^{\mathcal{KG}}} \quad (2)$$

Since the numerator of $TF^{\mathcal{KG}}$ can only take values 0 or 1 and, each feature under the root in the denominator has value 0 or 1, $v_{(i,\langle \rho, \omega \rangle)}$ is zero if $\langle \rho, \omega \rangle \notin \mathcal{KG}$, and otherwise:

$$v_{(i,\langle \rho, \omega \rangle)} = \frac{\log |I| - \log |\langle j, \rho, \omega \rangle \cap \mathcal{KG} | j \in I|}{\sqrt{\sum\limits_{\langle \rho, \omega \rangle \in F} |\{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathcal{KG}\}|}} \quad (3)$$

Analogously, when we have a set U of users, we may represent them using the features describing the items they enjoyed in the past. In the following, when no confusion arises, we use f to denote a feature $\langle \rho, \omega \rangle \in F$. Given a user u, if we denote with I^u the set of the items enjoyed by u, we may introduce the vector $\mathbf{u} = [v_{(u,1)}, \ldots, v_{(u,f)} \ldots, v_{(u,|F|)}]$, where $v_{(u,f)}$ is:

$$v_{(u,f)} = \frac{\sum\limits_{i \in I^u} v_{(i,f)}}{|\{i \mid i \in I^u \text{ and } v_{(i,f)} \neq 0\}|}$$

Given the vectors $\mathbf{u}_j$, with $j \in [1 \ldots |U|]$, and $\mathbf{i}_p$, with $p \in [1 \ldots |I|]$, we build the matrix $\mathbf{V} \in \mathbb{R}^{n \times |F|}$ (see Fig. 2) where the first $|U|$ rows have a one to one mapping with $\mathbf{u}_j$ while the last ones correspond to $\mathbf{i}_p$. If we go back to Eq. (1) we may see that, for each $\mathbf{x}$, the term $\sum_{j=1}^{n} \sum_{p=j+1}^{n} x_j \cdot x_{j'} \cdot \sum_{f=1}^{k} v_{(j,f)} \cdot v_{(p,f)}$

	dbc:Space_adventure_films	dbc:Films_set_in_the_future	dbc:American_science_fiction_action_films	dbc:1980s_science_fiction_films	dbc:Paramount_Pictures_films	dbc:Midlife_crisis_films	dbc:American_sequel_films	
v_1	0	0.88	0.81	0.7	0	0.60	0.53	...
v_2	1.3	1.12	0.91	0.84	0.65	0.59	0.58	...
v_3	0.5	0	0.71	0	0.28	0.35	0	...
v_4	0	0	0.31	0	0	0	0.6	...
v_5	0	0	0	0	0.18	0	0	...
v_6	0	0.12	0.22	0	0	0	0	...
v_7	1.23	1.03	0.89	0.85	0.56	0.3	0.61	...

Fig. 2. Example of real valued feature vectors for different items v_j. For lack of space we omitted the predicate *dcterms:subject*

is not zero only once, i.e., when both x_j and x_p are equal to 1. In the matrix depicted in Fig. 1, this happens when there is an interaction between a user and an item. Moreover, the summation $\sum_{f=1}^{k} v_{(j,f)} \cdot v_{(p,f)}$ represents the dot product between two vectors: $\mathbf{v}_j$ and $\mathbf{v}_p$ with a size equal to k. Hence, $\mathbf{v}_j$ represents a latent representation of a user, $\mathbf{v}_p$ that of an item within the same latent space, and their interaction is evaluated through their dot product.

In order to inject the knowledge coming from $\mathcal{KG}$ into kaHFM, we keep Eq. (1) and we set $k = |F|$. In other words, we impose a number of latent factors equal to the number of features describing all the items in our catalog. We want to stress here that our aim is not representing each feature through a latent vector, but to associate each factor to an explicit feature, obtaining latent vectors that are composed by explicit semantic features. Hence, we initialize the parameters $\mathbf{v}_j$ and $\mathbf{v}_p$ with their corresponding rows from $\mathbf{V}$ which in turn represent respectively $\mathbf{u}_j$ and $\mathbf{i}_p$. In this way, we try to identify each latent factor with a corresponding explicit feature. The intuition is that after the training phase, the resulting matrix $\hat{\mathbf{V}}$ still refers to the original features but contains better values for $v_{(j,f)}$ and $v_{(p,f)}$ that take into account also the latent interactions between users, items and features. It is noteworthy that after the training phase $\mathbf{u}_j$ and $\mathbf{i}_p$ (corresponding to $v_{(j,f)}$ and $v_{(p,f)}$ in $\mathbf{V}$) contain non-zero values also for features that are not originally in the description of the user u or of the item i. We extract the items vectors $\mathbf{v}_j$ from the matrix $\hat{\mathbf{V}}$, with the associated optimal values and we use them to implement an Item-kNN recommendation approach. We measure similarities between each pair of items i and j by evaluating the cosine similarity of their corresponding vectors in $\hat{\mathbf{V}}$:

$$cs(i,j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\| \mathbf{v}_i \| \cdot \| \mathbf{v}_j \|}$$

Let us define N^i as the set of neighbors for the item i, composed by the items which are more similar to i according to the selected similarity measure. Denoted as N^i. It is possible to choose i such that $i \notin I^u$ and a user u, we predict the score assigned by u to i as

$$score(u,i) = \frac{\sum\limits_{j \in N^i \cap I^u} cs(i,j)}{\sum\limits_{j \in N^i} cs(i,j)} \tag{4}$$

Factorization machines can be easily trained to reduce the prediction error via gradient descent methods, alternating least-squares (ALS) and MCMC. Since we formulated our problem as a *top-N* recommendation task, kaHFM can be trained using a learning to rank approach like Bayesian Personalized Ranking Criterion (BPR) [32]. The BPR criterion is optimized using a stochastic gradient descent algorithm on a set D_S of triples (u, i, j), with $i \in I^u$ and $j \notin I^u$, selected through a random sampling from a uniform distribution. Once the training phase returns the optimal model parameters, the item recommendation step can take place.

In an RDF knowledge graph, we usually find different types of encoded information.

Table 1. Top-10 features computed by `kaHFM` for the movie `"Star Trek II - The Wrath of Khan"`.

kaHFM	TF-IDF	Predicate	Object
1.3669	0.2584	dct:subject	dbc:Space_adventure_films
1.1252	0.2730	dct:subject	dbc:Films_set_in_the_future
0.9133	0.2355	dct:subject	dbc:American_science_fiction_action_films
0.8485	0.3190	dct:subject	dbc:1980s_science_fiction_films
0.6529	0.1549	dct:subject	dbc:Paramount_Pictures_films
0.5989	0.3468	dct:subject	dbc:Midlife_crisis_films
0.5940	0.1797	dct:subject	dbc:American_sequel_films
0.5862	0.2661	dct:subject	dbc:Film_scores_by_James_Horner
0.5634	0.2502	dct:subject	dbc:Films_shot_in_San_Francisco
0.5583	0.1999	dct:subject	dbc:1980s_action_thriller_films

- **Factual.** This refers to statements such has *The Matrix was directed by the Wachowskis* or *Melbourne is located in Australia* when we describe attributes of an entity;
- **Categorical.** It is mainly used to state something about the subject of an entity. In this direction, the categories of Wikipedia pages are an excellent example. Categories can be used to cluster entities and are often organized hierarchically thus making possible to define them in a more generic or specific way;
- **Ontological.** This is a more restrictive and formal way to classify entities via a hierarchical structure of classes. Differently from categories, sub-classes and super-classes are connected through IS-A (transitive) relations.

In Table 1 we show an example for categorical values obtained after the training (in the column `kaHFM`) together with the original TF-IDF ones computed for a movie from the Yahoo! Movies[1] dataset.

3   Semantic Accuracy and Generative Robustness

The proposed approach let us keep the meaning of the "latent" factors computed via a factorization machine thus making possible an interpretation of the recommended results. To assess that `kaHFM` preserves the semantics of the features in $\mathbf{V}$ after the training phase, we propose an automated offline procedure to measure *Semantic Accuracy*. Moreover, we define as *Robustness* the ability to assign a higher value to important features after one or more feature removals.

3.1   Semantic Accuracy

The main idea behind Semantic Accuracy is to evaluate, given an item i, how well kaHFM is able to return its original features available in the computed top-K list $\mathbf{v}_i$. In other words, given the set of features of i represented by $F^i = \{f_1^i, \ldots, f_m^i, \ldots f_M^i\}$, with $F^i \subseteq F$, we check if the values in $\mathbf{v}_i$, corresponding to $f_{m,i} \in F^i$, are higher than those corresponding to $f \notin F^i$. For the set of M features initially describing i we see how many of them appear in the set $top(\mathbf{v}_i, M)$ representing the top-M features in $\mathbf{v}_i$. We then normalize this number by the size of F^i and average on all the items within the catalog I.

$$\texttt{Semantic Accuracy (SA@}M) = \frac{\sum\limits_{i \in I} \frac{|top(\mathbf{v}_i, M) \cap F^i|}{|F^i|}}{|I|}$$

In many practical scenarios we may have $|F| \gg M$. Hence, we might also be interested in measuring the accuracy for different sizes of the top list. Since items could be described with a different number of features, the size of the top list could be a function of the original size of the item description. Thus, we measured SA@nM with $n \in \{1, 2, 3, 4, 5, \ldots\}$ and evaluate the number of features in F^i available in the top-$n \cdot M$ elements of $\mathbf{v}_i$.

$$\texttt{SA@}nM = \frac{\sum\limits_{i \in I} \frac{|top(\mathbf{v}_i, n \cdot M) \cap F^i|}{|F^i|}}{|I|}$$

3.2   Robustness

Although SA@nM may result very useful to understand if kaHFM assigns weights according to the original description of item i, we still do not know if a high value in $\mathbf{v}_i$ really means that the corresponding feature is important to define i. In other words, are we sure that kaHFM promotes important features for i?

In order to provide a way to measure such "meaningfulness" for a given feature, we suppose, for a moment, that a particular feature $\langle \rho, \omega \rangle$ is useful to describe an item i but the corresponding triple $\langle i, \rho, \omega \rangle$ is not represented in the knowledge graph. In case kaHFM was robust in generating weights for unknown features, it should discover the importance of that feature and modify its value to make it enter the Top-K features in $\mathbf{v}_i$. Starting from this observation, the idea to measure robustness is then to "forget" a triple involving i and check if kaHFM can generate it. In order to implement such process we proceed by following these steps:

– we train kaHFM thus obtaining optimal values v_i for all the features in F^i;
– the feature $f_{MAX}^i \in F^i$ with the highest value in v_i is identified;
– we retrain the model again initializing $f_{MAX}^i = 0$ and we compute v_i'.

After the above steps, if $f_{MAX}^i \in top(v_i', M)$ then we can say that kaHFM shows a high robustness in identifying important features. Given a catalog I, we

may then define the *Robustness for 1 removed feature @M* (`1-Rob@M`) as the number of items for which $f_{MAX}^i \in top(v'_i, M)$ divided by the size of I.

$$\texttt{1-Rob@M} = \frac{\sum\limits_{i \in I} |\{i \mid f_{MAX}^i \in top(v'_i, M)\}|}{|I|}$$

Similarly to `SA@`nM, we may define `1-Rob@`n`M`.

4 Experimental Evaluation

In this section, we will detail three distinct experiments. We specifically designed them to answer the research questions posed in Sect. 1. In details, we want to assess if: (i) `kaHFM`'s recommendations are accurate; (ii) `kaHFM` generally preserves the semantics of original features; (iii) `kaHFM` promotes significant features.

Datasets. To provide an answer to our research questions, we evaluated the performance of our method on two well-known datasets for recommender systems belonging to movies domain. `Yahoo!Movies` (Yahoo! Webscope dataset ydata-ymovies-user-movie-ratings-content-v1_0)[2] contains movies ratings generated on Yahoo! Movies up to November 2003. It provides content, demographic and ratings information on a [1..5] scale, and mappings to `MovieLens` and `EachMovie` datasets. `Facebook Movies` dataset has been released for the Linked Open Data challenge co-located with ESWC 2015[3]. Only implicit feedback is available for this dataset, but for each item a link to DBpedia is provided. To map items in `Yahoo!Movies` and other well-known datasets, we extracted all the updated items-features mappings and we made them publicly available[4]. Datasets statistics are shown in Table 2.

Experimental Setting. "All Unrated Items" [37] protocol has been adopted to compare different algorithms. We have split the dataset using Hold-Out 80-20 retaining for every user the 80% of their ratings in the training set and the remaining 20% in the test set. Moreover, a temporal split has been performed [19] whenever timestamps associated to every transaction is available.

Extraction. Thanks to the publicly available mappings, all the items from the datasets represented in Table 2 come with a `DBpedia` link. Exploiting this reference, we retrieved all the $\langle \rho, \omega \rangle$ pairs. Some noisy features (based on the following predicates) have been excluded: `owl:sameAs`, `dbo:thumbnail`, `foaf:depiction`, `prov:wasDerivedFrom`, `foaf:isPrimaryTopicOf`.

Selection. We performed our experiments with three different settings to analyze the impact of the different kind of features. The features have been chosen as they are present in all the different domains and because of their factual, categorical or ontological meaning:

Table 2. Datasets statistics.

Dataset	#Users	#Items	#Transactions	#Features	Sparsity
Yahoo! Movies	4000	2,626	69,846	988,734	99.34%
Facebook Movies	32143	3,901	689,561	180,573	99.45%

- **Categorical Setting (CS):** We selected only the features containing the property `dcterms:subject`.
- **Ontological Setting (OS):** In this case the only feature we considered is `rdf:type`.
- **Factual Setting (FS):** We considered all the features but those involving the properties selected in OS, and CS.

Filtering. This last step corresponds to the removal of irrelevant features, that bring little value to the recommendation task, but, at the same time, pose scalability issues. The pre-processing phase has been done following [13], and [25] with a unique threshold. Thresholds (corresponding to *tm* [13], and *p* [25] for missing values) and the considered features for each dataset are represented in Table 3.

Table 3. Considered features in the different settings

Datasets	Threshold	Categorical setting		Ontological setting		Factual setting	
		Total	Selected	Total	Selected	Total	Selected
Yahoo!Movies	99.62	26155	747	38699	1240	950035	3186
Facebook Movies	99.74	8843	1103	13828	1848	166745	5427

4.1 Accuracy Evaluation

The goal of this evaluation is to assess if the controlled injection of `Linked Data` positively affects the training of Factorization Machines. For this reason, `kaHFM` is not compared with other state-of-art interpretable models but with only the algorithms that are more related to our approach. We compared `kaHFM`[5] w.r.t. a canonical 2 degree Factorization Machine (users and items are intended as features of the original formulation) by optimizing the recommendation list ranking via BPR (BPR-FM). In order to preserve the expressiveness of the model, we used the same number of hidden factors (see the "Selected" column in Table 3). Since we use items similarity in the last step of our approach (see Eq. (4)), we compared `kaHFM` against an *Attribute Based Item-kNN* (ABItem-kNN) algorithm, where each item is represented as a vector of weights,

---

[5] https://github.com/sisinflab/HybridFactorizationMachines.

computed through a TF-IDF model. In this model, the attributes are computed via Eq. (2). We also compared kaHFM also against a pure Item-kNN, that is an item-based implementation of the k-nearest neighbors algorithm. It finds the k-nearest item neighbors based on Pearson Correlation. https://github.com/sisinflab/HybridFactorizationMachines Regarding BPR parameters, *learning rate*, *bias regularization*, *user regularization*, *positive item regularization*, and *negative item regularization* have been set respectively to 0.05, 0, 0.0025, 0.0025 and 0.00025 while a sampler "without replacement" has been adopted in order to sample the triples as suggested by authors [32]. We compared kaHFM also against the corresponding User-based nearest neighbor scheme, and Most-Popular, a simple baseline that shows high performance in specific scenarios [11]. In our context, we considered mandatory to also compare against a pure knowledge-graph content-based baseline based on Vector Space Model (VSM) [15]. In order to evaluate our approach, we measured accuracy through Precision@N ($Prec@N$) and Normalized Discounted Cumulative Gain ($nDCG@N$). The evaluation has been performed considering Top-10 [11] recommendations for all the datasets. When a rating score was available (Yahoo!Movies), a *Threshold-based relevant items* condition [5,7] was adopted with a relevance threshold of 4 over 5 stars in order to take into account only relevant items. Figure 3 shows the results of our experiments regarding accuracy. In all the tables we highlight in **bold** the best result while we underline the second one. Statistically significant results are denoted with a * mark considering Student's paired t-test with a 0.05 level. Yahoo!Movies experiments show that in Categorical and Ontological settings our method is the most accurate. In the Yahoo!Movies mapping, a strong popularity bias is present and it is interesting to notice that this affects only the Factual setting leading our approach to be less precise than ABItem-kNN. In Facebook Movies we see very a good improvement in terms of accuracy as it almost doubles up the ABItem-kNN algorithm values. We compared kaHFM against ABItem-kNN to check if the collaborative trained features may lead to better similarity values. This hypothesis seems to be confirmed since in former experiments kaHFM beats ABItem-kNN in almost all settings. This suggests that collaborative trained features achieve better accuracy results. Moreover, we want to check if a knowledge-graph-based initialization of latent factors may improve the performance of Factorization Machines. kaHFM always beats BPR-FM, and in our opinion, this happens since the random initialization takes a while to drive the Factorization machine to reach good performance. Finally, we want to check if collaborative trained features lead to better accuracy results than a purely informativeness-based Vector Space Model even though it is in its knowledge-graph-aware version. This seems to be confirmed in our experiments, since kaHFM beats VSM in almost all cases. In order to strengthen the results we got, we computed recommendations with $0, 1, 5, 10, 15, 30$ iterations. For the sake of brevity we report here[6] only the plots related to Categorical setting (Fig. 3) It is worth to notice that in every case we considered, we show the best performance in

---

[6] Results of the full experiments: https://github.com/sisinflab/papers-results/tree/master/kahfm-results/.

	Facebook	Yahoo!	
Categorical Setting (CS)	P@10	P@10	nDCG@10
ABItem-kNN	0.0173*	<u>0.0421*</u>	<u>0.1174*</u>
BPR-FM	0.0158*	0.0189*	0.0344*
MostPopular	0.0118*	0.0154*	0.0271*
ItemKnn	<u>0.0262*</u>	0.0203*	0.0427*
UserKnn	0.0168*	0.0231*	0.0474*
VSM	0.0185*	0.0385*	0.1129*
kaHFM	**0.0296**	**0.0524**	**0.1399**
Ontological Setting (OS)	P@10	P@10	nDCG@10
ABItem-kNN	0.0172*	<u>0.0427*</u>	<u>0.1223*</u>
BPR-FM	0.0155*	0.0199*	0.0356*
MostPopular	0.0118*	0.0154*	0.0271*
ItemKnn	<u>0.0263*</u>	0.0203*	0.0427*
UserKnn	0.0168*	0.0232*	0.0474*
VSM	0.0181*	0.0349*	0.1083*
kaHFM	**0.0273**	**0.0521**	**0.1380**
Factual Setting (FS)	P@10	P@10	nDCG@10
ABItem-kNN	0.0234	<u>0.0619</u>	**0.1764**
BPR-FM	0.0157	0.0177	0.0305
MostPopular	0.0123	0.0154	0.0271
ItemKnn	**0.0273**	0.0203	0.0427
UserKnn	0.0176	0.0232	0.0474
VSM	0.0219	**0.0627**	<u>0.1725</u>
kaHFM	<u>0.0240</u>	0.0564	0.1434

(a) Yahoo!Movies

(b) Facebook Movies

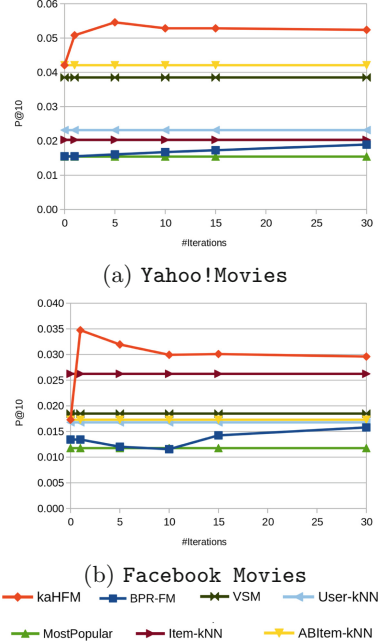kaHFM   BPR-FM   VSM   User-kNN   MostPopular   Item-kNN   ABItem-kNN

Fig. 3. Accuracy results for `Facebook Movies`, and `Yahoo!Movies`. In figures: Precision@10 varying # iterations 0, 1, 5, 10, 15, 30

one of these iterations. Moreover, the positive influence of the initialization of the feature vectors is particularly evident in all the datasets, with performances being very similar to the ones depicted in [32]. Given the obtained results we may say that the answer to RQ1 is positive when adopting `kaHFM`.

4.2 Semantic Accuracy

The previous experiments showed the effectiveness of our approach in terms of accuracy of recommendation. In practical terms, we proved that: (i) content initialization generally lead to better performance with our method, (ii) the obtained items vectors are fine-tuned better than the original ones for a *top-N* item recommendation task, (iii) results may depend on the features we extract from the Knowledge Graph. However, we still do not know if the original semantics of the features is preserved in the new space computed after the training of `kaHFM` (as we want to assess by posing RQ2). In Sect. 3.1 we introduced `Semantics Accuracy` ($SA@nM$) as a metric to automatically check if the importance computed by `kaHFM` and associated to each feature reflects the actual meaning of that feature. Thus, we measured `SA@`nM with $n \in \{1, 2, 3, 4, 5\}$ and $M = 10$, and evaluated the number of ground features available in the top-nM

elements of $\mathbf{v}_i$ for each dataset. Table 4 shows the results for all the different datasets computed in the Categorical setting. In general, the results we obtain are noteworthy. We now examine the worst one to better understand the actual meaning of the values we get. In `Yahoo!Movies`, 747 different features compose each item vector (see Table 3). After the training phase, on average, more than 10 (equal to 0.847×12.143) over 12 features (last column in Table 4) are equal to the original features list. This means that `kaHFM` was able to compute almost the same features starting from hundreds of them. Also in this case, given the obtained results we may provide a positive answer to RQ2.

Table 4. Semantics Accuracy results for different values of M. F.A. denotes the Feature Average number per item.

Semantics accuracy	SA@M	SA@2M	SA@3M	SA@4M	SA@5M	F.A.
`Yahoo!Movies`	0.847	0.863	0.865	0.868	0.873	12.143
`Facebook Movies`	0.864	0.883	0.889	0.894	0.899	12.856

4.3   Generative Robustness

The previous experiment showed that the features computed by `kaHFM` keep their original semantics if already present in the item description. In Sect. 3.2, we introduced a procedure to measure the capability of `kaHFM` to compute meaningful features. Here, we computed `1-Rob@nM` for the two adopted datasets. Results are represented in Table 5. In this case, we focus on the CS setting which provides the best results in terms of accuracy. For a better understanding of the obtained results, we start by focusing on `Yahoo!Movies` which apparently has bad behavior. Table 4 showed that `kaHFM` was able to guess 10 on 12 different features for `Yahoo!Movies`. In this experiment, we remove one of the ten features (thus, based on Table 4, `kaHFM` will guess an average of $10 - 1 = 9$ features). Since the number of features is 12 we have 3 remaining "slots". What we measure now is if `kaHFM` is able to guess the removed feature in these "slots". Results in Table 5 show that our method is able to put the removed feature in one of the three slots the 48.7% of the times starting from 747 overall features. This example should help the reader to appreciate even more `Facebook Movies` results. Hence, we

Table 5. 1-Robustness for different values of M. Column F.A. denotes the Feature Average number per item.

1-Robustness	`1-Rob@M`	`1-Rob@2M`	`1-Rob@3M`	`1-Rob@4M`	`1-Rob@5M`	F.A.
`Yahoo!Movies`	0.487	0.645	0.713	0.756	0.793	12.143
`Facebook Movies`	0.821	0.945	0.970	0.980	0.984	12.856

could confidently assess that `kaHFM` is able to propose meaningful features as we asked with RQ3.

5   Related Work

In recent years, several interpretable recommendation models that exploit matrix factorization have been proposed. It is well-known that one of the main issues of matrix factorization methods is that they are not easily interpretable (since latent factors meaning is basically unknown). One of the first attempts to overcome this problem was proposed in [47]. In this work, the authors propose Explicit Factor Model (EFM). Products' features and users' opinions are extracted with phrase-level sentiment analysis from users' reviews to feed a matrix factorization framework. After that, a few improvements to EFM have been proposed to deal with temporal dynamics [48] and to use tensor factorization [9]. In particular, in the latter the aim is to predict both user preferences on features (extracted from textual reviews) and items. This is achieved by exploiting the Bayesian Personalized Ranking (BPR) criterion [32]. Further advances in MF-based interpretable recommendation models have been proposed with Explainable Matrix Factorization (EMF) [1] in which the generated explanations are based on a neighborhood model. Similarly, in [2] an interpretable Restricted Boltzmann Machine model has been proposed. It learns a network model (with an additional visible layer) that takes into account a degree of explainability. Finally, an interesting work incorporates sentiments and ratings into a matrix factorization model, named Sentiment Utility Logistic Model (SULM) [6]. In [28] recommendations are computed by generating and ranking personalized explanations in the form of explanation chains. OCuLaR [42] provides interpretable recommendations from positive examples based on the detection of co-clusters between users (clients) and items (products). In [22] authors propose a Multi Level Attraction Model (MLAM) in which they build two attraction models, for cast and story. The interpretability of the model is then provided in terms of attractiveness of Sentence level, Word level, and Cast member. In [27] the authors train a matrix factorization model to compute a set of association rules that interprets the obtained recommendations. In [12] the authors prove that, given the conversion probabilities for all actions of customer features, it is possible to transform the original historical data to a new space in order to compute a set of interpretable recommendation rules. The core of our model is a general Factorization Machines (FM) model [30]. Nowadays FMs are the most widely used factorization models because they offer a number of advantages w.r.t. other latent factors models such as SVD++ [23], PITF [35], FPMC [33]. First of all, FMs are designed for a generic prediction task while the others can be exploited only for specific tasks. Moreover, it is a linear model and parameters can be estimated accurately even in high data sparsity scenarios. Nevertheless, several improvements have been proposed for FMs. For instance Neural Factorization

Machines [20] have been developed to fix the inability of classical FMs to capture non linear structure of real-world data. This goal is achieved by exploiting the non linearity of neural networks. Furthermore, Attentional Factorization Machines [44] have been proposed that use an attention network to learn the importance of feature interactions. Finally, FMs have been specialized to better work as Context-Aware recommender systems [34].

6 Conclusion and Future Work

In this work, we have proposed an interpretable method for recommendation scenario, kaHFM, in which we bind the meaning of latent factors for a Factorization machine to data coming from a knowledge graph. We evaluated kaHFM on two different publicly available datasets on different sets of semantics-aware features. In particular, in this paper we considered Ontological, Categorical and Factual information coming from DBpedia and we have shown that the generated recommendation lists are more precise and personalized. Summing up, performed experiments show that: (RQ1) the learned model shows very good performance in terms of accuracy and, at the same time, is effectively interpretable; (RQ2) the computed features are semantically meaningful; (RQ3) the model is robust regarding computed features. In the future we want to test the kaHFM performance in different scenarios, other than recommender systems. Moreover, the model can be improved in many different ways. Different relevance metrics could be beneficial in different scenarios, as the method itself is agnostic to the specific adopted measure. This work focused on the items' vector; however, an interesting key point would be analyzing the learned users' vectors to extract more accurate profiles. Furthermore, it would be useful to exploit kaHFM in order to provide suggestions to knowledge graphs maintainers while adding relevant missing features to the knowledge base. In this direction, we would like to evaluate our approach in knowledge graph completion task.

References

1. Abdollahi, B., Nasraoui, O.: Explainable matrix factorization for collaborative filtering. In: Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, 11–15 April 2016, Companion Volume, pp. 5–6 (2016)
2. Abdollahi, B., Nasraoui, O.: Explainable restricted Boltzmann machines for collaborative filtering. CoRR abs/1606.07129 (2016)
3. Adomavicius, G., Kwon, Y.O.: Multi-criteria recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 847–880. Springer, Boston, MA (2015). https://doi.org/10.1007/978-1-4899-7637-6_25
4. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 217–253. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_7

5. Anelli, V.W., Bellini, V., Di Noia, T., Bruna, W.L., Tomeo, P., Di Sciascio, E.: An analysis on time- and session-aware diversification in recommender systems. In: Bieliková, M., Herder, E., Cena, F., Desmarais, M.C. (eds.) Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, UMAP 2017, Bratislava, Slovakia, 09–12 July 2017, pp. 270–274. ACM (2017)

6. Bauman, K., Liu, B., Tuzhilin, A.: Aspect based recommendations: recommending items with the most valuable aspects based on user reviews. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017, pp. 717–725 (2017)

7. Campos, P.G., Díez, F., Cantador, I.: Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Model. User-Adapt. Interact. **24**(1–2), 67–119 (2014)

8. Chakraborty, S., et al.: Interpretability of deep learning models: a survey of results. In: 2017 IEEE SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI, pp. 1–6 (2017)

9. Chen, X., Qin, Z., Zhang, Y., Xu, T.: Learning to rank features for recommendation over multiple categories. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, 17–21 July 2016, pp. 305–314 (2016)

10. Cramer, H.S.M., et al.: The effects of transparency on trust in and acceptance of a content-based art recommender. User Model. User-Adapt. Interact. **18**(5), 455–496 (2008)

11. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, 26–30 September 2010, pp. 39–46 (2010)

12. Dhurandhar, A., Oh, S., Petrik, M.: Building an interpretable recommender via loss-preserving transformation. CoRR abs/1606.05819 (2016)

13. Di Noia, T., Magarelli, C., Maurino, A., Palmonari, M., Rula, A.: Using ontology-based data summarization to develop semantics-aware recommender systems. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 128–144. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_9

14. Di Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D.: Exploiting the web of data in model-based recommender systems. In: Sixth ACM Conference on Recommender Systems, RecSys 2012, Dublin, Ireland, 9–13 September 2012, pp. 253–256 (2012)

15. Di Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D., Zanker, M.: Linked open data to support content-based recommender systems. In: I-SEMANTICS 2012–8th International Conference on Semantic Systems, I-SEMANTICS 2012, Graz, Austria, 5–7 September 2012, pp. 1–8 (2012)

16. Drawel, N., Qu, H., Bentahar, J., Shakshuki, E.: Specification and automatic verification of trust-based multi-agent systems. Future Gener. Comput. Syst. (2018). https://doi.org/10.1016/j.future.2018.01.040

17. Falcone, R., Sapienza, A., Castelfranchi, C.: The relevance of categories for trusting information sources. ACM Trans. Internet Technol. **15**(4), 13:1–13:21 (2015)

18. Fernández-Tobías, I., Cantador, I., Tomeo, P., Anelli, V.W., Noia, T.D.: Addressing the user cold start with cross-domain collaborative filtering: exploiting item metadata in matrix factorization. User Model. User-Adapt. Interact. **29**(2), 443–486 (2019)

19. Gunawardana, A., Shani, G.: Evaluating recommender systems. In: Recommender Systems Handbook, pp. 265–308 (2015)

20. He, X., Chua, T.: Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, 7–11 August 2017, pp. 355–364 (2017)

21. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: CSCW 2000, Proceeding on the ACM 2000 Conference on Computer Supported Cooperative Work, Philadelphia, PA, USA, 2–6 December 2000, pp. 241–250 (2000)

22. Hu, L., Jian, S., Cao, L., Chen, Q.: Interpretable recommendation via attraction modeling: learning multilevel attractiveness over multimodal movie contents. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, 13–19 July 2018, pp. 3400–3406 (2018)

23. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, 24–27 August 2008, pp. 426–434 (2008)

24. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Comput. **42**(8), 30–37 (2009)

25. Paulheim, H., Fürnkranz, J.: Unsupervised generation of data mining features from linked open data. In: 2nd International Conference on Web Intelligence, Mining and Semantics, WIMS 2012, Craiova, Romania, 6–8 June 2012, pp. 31:1–31:12 (2012)

26. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_10

27. Peake, G., Wang, J.: Explanation mining: post hoc interpretability of latent factor models for recommendation systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, 19–23 August 2018, pp. 2060–2069 (2018)

28. Rana, A., Bridge, D.: Explanation chains: recommendations by explanation. In: Proceedings of the Poster Track of the 11th ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, 28 August 2017 (2017)

29. Rendle, S.: Factorization machines. In: 2010 IEEE 10th International Conference on Data Mining (ICDM), pp. 995–1000. IEEE (2010)

30. Rendle, S.: Factorization machines. In: ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14–17 December 2010, pp. 995–1000 (2010)

31. Rendle, S.: Context-Aware Ranking with Factorization Models. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-16898-7

32. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009, pp. 452–461 (2009)

33. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, 26–30 April 2010, pp. 811–820 (2010)

34. Rendle, S., Gantner, Z., Freudenthaler, C., Schmidt-Thieme, L.: Fast context-aware recommendations with factorization machines. In: Proceeding of the 34th Inter-

national ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, 25–29 July 2011, pp. 635–644 (2011)

35. Rendle, S., Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation. In: Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, 4–6 February 2010, pp. 81–90 (2010)

36. Sinha, R.R., Swearingen, K.: The role of transparency in recommender systems. In: Extended abstracts of the 2002 Conference on Human Factors in Computing Systems, CHI 2002, Minneapolis, Minnesota, USA, 20–25 April 2002, pp. 830–831 (2002)

37. Steck, H.: Evaluation of recommendations: rating-prediction and ranking. In: Proceedings of the 7th ACM Conference on Recommender Systems, pp. 213–220. ACM (2013)

38. Sun, Z., Yang, J., Zhang, J., Bozzon, A., Huang, L., Xu, C.: Recurrent knowledge graph embedding for effective recommendation. In: Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, 2–7 October 2018, pp. 297–305 (2018)

39. Tintarev, N., Masthoff, J.: A survey of explanations in recommender systems. In: Proceedings of the 23rd International Conference on Data Engineering Workshops, ICDE 2007, 15–20 April 2007, Istanbul, Turkey, pp. 801–810 (2007)

40. Tintarev, N., Masthoff, J.: Designing and evaluating explanations for recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 479–510. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_15

41. Vig, J., Sen, S., Riedl, J.: Tagsplanations: explaining recommendations using tags. In: Proceedings of the 14th International Conference on Intelligent User Interfaces, IUI 2009, Sanibel Island, Florida, USA, 8–11 February 2009, pp. 47–56 (2009)

42. Vlachos, M., Duenner, C., Heckel, R., Vassiliadis, V.G., Parnell, T., Atasu, K.: Addressing interpretability and cold-start in matrix factorization for recommender systems. IEEE Trans. Knowl. Data Eng. **31**, 1253–1266 (2018)

43. Wang, X., He, X., Feng, F., Nie, L., Chua, T.: TEM: tree-enhanced embedding model for explainable recommendation. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France 23–27 April 2018, pp. 1543–1552 (2018)

44. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.: Attentional factorization machines: learning the weight of feature interactions via attention networks. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, 19–25 August 2017, pp. 3119–3125 (2017)

45. Zanker, M.: The influence of knowledgeable explanations on users' perception of a recommender system. In: Sixth ACM Conference on Recommender Systems, RecSys 2012, Dublin, Ireland, 9–13 September 2012, pp. 269–272 (2012)

46. Zhang, Y., Chen, X.: Explainable recommendation: a survey and new perspectives. CoRR abs/1804.11192 (2018)

47. Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., Ma, S.: Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In: The 37th International Conference on Research and Development in Information Retrieval, SIGIR 2014, Gold Coast, QLD, Australia, pp. 83–92 (2014)

48. Zhang, Y., et al.: Daily-aware personalized recommendation based on feature-level time series analysis. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, 18–22 May 2015, pp. 1373–1383 (2015)

49. Zhao, W.X., Li, S., He, Y., Wang, L., Wen, J., Li, X.: Exploring demographic information in social media for product recommendation. Knowl. Inf. Syst. **49**(1), 61–89 (2016)
50. Zheng, Y., Mobasher, B., Burke, R.D.: Incorporating context correlation into context-aware matrix factorization. In: Proceedings of the IJCAI 2015 Joint Workshop on Constraints and Preferences for Configuration and Recommendation and Intelligent Techniques for Web Personalization co-located with the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, 27 July 2015 (2015)