Semantic Interpretation of Top-N Recommendations

Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzura Ragone, and Joseph Trotta

Abstract—Over the years, model-based approaches have shown their effectiveness in computing recommendation lists in different domains and settings. By relying on the computation of latent factors, they can recommend items with a very high level of accuracy. Unfortunately, when moving to the latent space, even if the model embeds content-based information, we miss references to the actual semantics of the recommended item. It makes the interpretation of the recommendation process non-trivial. In this paper, we show how to initialize latent factors in Factorization Machines by using semantic features coming from knowledge graphs to train an interpretable model, which is, in turn, able to provide recommendations with a high level of accuracy. In the presented approach, semantic features are injected into the learning process to retain the original informativeness of the items available in the dataset. By relying on the information encoded in the original knowledge graph, we also propose two metrics to evaluate the semantic accuracy and robustness of knowledge-aware interpretability. An extensive experimental evaluation on six different datasets shows the effectiveness of the interpretable model in terms of both accuracy and diversity of recommendation results and interpretability robustness.

Index Terms—Recommender Systems, Knowledge Graph, Interpretable models, Factorization Machines.

1 Introduction

Research on transparency and interpretability of predictive models is gaining momentum since the research community is recognizing them as decisive elements in the next generation of recommendation algorithms. When equipped with the interpretability of recommendation results, a system ceases to be just a black-box (transparency) [1], [2], [3], and users are more willing to exploit the predictions extensively [4]. Transparency increases users' trust [5] (also exploiting specific semantic structures [6]), and satisfaction in using the system. For a recommender system, the user's trust is a vital aspect since it also leads to better performance [7]. In a nutshell, we may say that interpretations for recommendation results can be item-based, user-based, or feature-based. Item-based interpretations make use of the shared set of items among users [8]; User-based interpretations rely on sets of most similar users, like in [4]; while Feature-based interpretations may exploit features of recommended items as director, genre, and cast. Among interpretable models, we may distinguish between those based on Content-based Filtering (CBF) approaches and those based on Collaborative filtering (CF) ones. CBF algorithms provide recommendations by exploiting the available content and matching it with a user profile [9]. The use of content features makes the model interpretable. It is worth noticing that these features can be dramatically different depending on the considered scenario: we may recommend a movie recommendation considering the director, actors, the producer, the genre, whereas we may interpret a book recommendation by the author, the book formats, or the saga. Sometimes, this prevents the straight adoption of a model in a specific knowledge domain. When content is missing, the recommender may rely only on the relationships between users (and the rates they provide to items), collaboratively [10]. Consequently, the interpretation of CF results inevitably reflects the approach adopted by the algorithm. For instance, an item-based and a user-based recommendation could be explained, respectively, as *"other users who have experienced A have experienced B"* or *"similar users have experienced B"*.

Unfortunately, things change when we adopt more accurate Deep Learning-based [11], Recurrent [12], [13], [14], or model-based algorithms and techniques. Such approaches project items and users in a new vector space of latent features [15], making the final result not directly interpretable. Indeed, it is possible to compute items and users' similarities via latent factor exploitation, but we entirely lose any reference to the original user-item interaction and then to an explicit justification for the recommendations. In the last years, the industry and the research community have proposed many approaches that take advantage of side information to enhance the performance of latent factor models. Side information can refer to items as well as users [16] and can be either structured [17] or semi-structured [18], [19]. Interestingly, in [20] the authors argue about a new generation of knowledge-aware recommendation engines able to exploit information encoded in knowledge graph ($\mathcal{KG}$) to produce meaningful recommendations.

In this work, we propose a `knowledge-aware Hybrid Factorization Machine` (`kaHFM`) to train interpretable models in recommendation scenarios. `kaHFM` relies on Factorization Machines [21], and it extends them in different key aspects making use of the semantic information encoded in a knowledge graph. Moreover, since it is unlikely that an inaccurate model will be adopted, it would be highly beneficial for users to develop accurate recommender

- *Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, and Joseph Trotta are with Polytechnic University of Bari.*
  E-mail: {vitowalter.anelli, tommaso.dinoia, eugenio.disciascio, jospeh.trotta} @poliba.it
- *Azzurra Ragone is an Independent Researcher.*
  E-mail: azzurra.ragone@gmail.com

systems, built upon an interpretable technique, so that the model can be, in case, exploited to generate explanations. In this sense, we show how `kaHFM` exploits 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 a semantic feature extracted from a knowledge graph. After the model training, 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. Furthermore, we show that the explicit mapping of latent features to content-based ones makes it possible to exploit the characteristics of these latter to implement a more effective initialization technique.

We have evaluated `kaHFM` on six different publicly available datasets. Initially, we have extracted content-based features from data encoded in the `DBpedia`[1] knowledge graph. Then, we have analyzed the performance of the approach in terms of accuracy, diversity, and novelty of results by exploiting categorical, ontological, and factual features (see Section 2.1). For each of them, we have exploited public mappings to `DBpedia`. Finally, we have tested the robustness of `kaHFM`, showing that it ranks important features higher and can regenerate them if we remove them from the original dataset. 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 regarding an explicitly stated semantics coming from a knowledge graph?
- `RQ2` Can we evaluate whether `kaHFM` preserves the original semantics of items features after training?
- `RQ3` How to measure with an offline evaluation that the proposed model can identify essential features by exploiting their explicit semantics?

We can summarize the main contributions of this paper as:

- presentation of `kaHFM`: a framework that exploits data available in knowledge graphs to build semantically interpretable models for recommendation tasks;
- an experimental evaluation based on six different datasets to assess the performance of `kaHFM` in terms of accuracy, diversity, and novelty of computed results;
- introduction of two metrics, Semantic Accuracy (SA@K) and robustness (n-Rob@K), to measure the interpretability of a knowledge-aware recommendation engine.

The remainder of the paper is structured as follows: in the next section, we introduce the background technologies behind `kaHFM`, and then we detail the overall approach in Section 3. The following section is devoted to the introduction of the two metrics we propose to assess the quality of `kaHFM` results in terms of interpretability. In Section 5, we describe the experimental setting while in Section 6, we report on related work. Conclusion and future work close the paper.

1. http://dbpedia.org

2 Background Technologies

In this section, we briefly recap the leading technologies we have adopted to develop `kaHFM`. First, we introduce Vector Space Models for recommender systems, and then we give a quick overview of knowledge graphs and their Linked (Open) Data implementation.

Content-based recommender systems rely on the assumption that it is possible to predict the future behavior of users based on their personalized profiles. We can build user profiles by exploiting the characteristics of the items they have liked in the past or some other available side information. Researchers have proposed several approaches that take advantage of side information in different ways: some of them consider tags [22], demographic data [23], or they extract information from collective knowledge bases [24]. Many of the most popular and adopted CBF approaches make use of a Vector Space Model (`VSM`).

In `VSM`, we represent users and items in the same vector space through Boolean or weighted vectors. Their respective positions and the distance (or better the proximity) between them provides a measure of how related (or similar) these two entities are. The vector space dimensions may refer to a user or item (or user-item) feature (e.g., the propensity to watch movies in the morning).

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. To sum up, in a CBF approach, we need (i) to get reliable items descriptions, (ii) to find 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 step, nowadays, we can smoothly get item descriptions from the Web. In particular, thanks to the Linked Open Data initiative, much semantically structured knowledge is publicly available in the form of Linked Data datasets.

2.1 Knowledge Graphs and Linked Data

In 2012, Google has announced its Knowledge Graph[2] ($\mathcal{KG}$) as a new tool to improve the identification and retrieval of entities in return to a search query. Most of the knowledge encoded in the Google Knowledge Graph comes from Freebase, which has been a crowd-sourced effort to create a base of facts in every possible knowledge domain.

The original idea of Semantic Web [25] has changed over the years, making possible the creation of a full stack of semantic technologies. More remarkably, these technologies have given birth to the Linking Open Data initiative[3], where a community of researchers and practitioners have devoted an enormous effort to build publicly available knowledge bases of machine-understandable data.

We can represent a knowledge base that exploits Semantic Web technologies through a graph (knowledge graph) in which entities are linked to each other by binary relations. The Semantic Web community has developed several technologies [26] and languages to manage this kind

2. https://googleblog.blogspot.it/2012/05/introducing-knowledge-graph-things-not.html
3. http://linkeddata.org

of knowledge model, e.g., the `Resource Description Framework` (`RDF`).

It provides a simple graph-based data model to encode knowledge in a structured way through triples in the form $\langle \sigma, \rho, \omega \rangle$. 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. In an `RDF` knowledge graph, we usually find different types of encoded information [27]:

- **Factual.** It refers to statements such as *The Matrix was directed by the Wachowskis* or *Melbourne is located in Australia* that describe attributes of an entity;
- **Categorical.** It states 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 hierarchically organized thus making it possible to define them in a more generic or specific way;
- **Ontological.** It is a more restrictive and formal way to classify entities via a hierarchical structure of classes. Differently from categories, between sub-classes and super-classes, an IS-A (transitive) relation subsists.

If we take a look at the following `RDF` triples[4] from the `DBpedia` knowledge graph

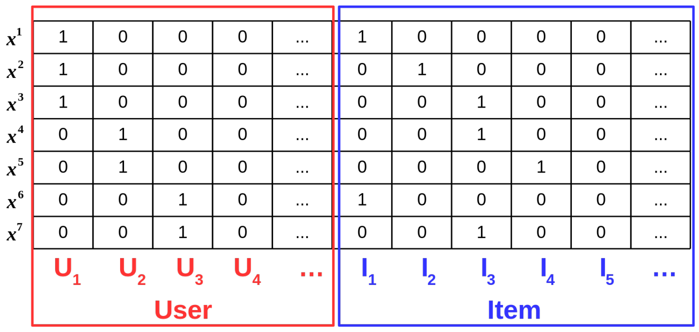
```
dbr:The_Matrix dbo:director
dbr:The_Wachowski_Brothers.
dbr:The_Matrix dct:subject dbc:Dystopian_films.
dbr:The_Matrix rdf:type dbo:Film.
```

we may see that each of them represents one of the types of data mentioned earlier. In the first triple, we state a fact about the movie *The Matrix* (represented by the corresponding URI `dbr:The_Matrix`) saying that it has been directed (`dbo:director`) by the *Wachowski Brothers* (`dbr:The_Wachowski_Brothers`). The second triple encodes categorical information through the predicate `dct:subject` about the same movie. In particular, here we say that it belongs to the category of dystopian films (`dbc:Dystopian_films`). Finally, with the last triple, we classify *The Matrix* as a Film (`dbo:Film`) thanks to the predicate `rdf:type`.

3 Knowledge-aware Hybrid Factorization Machines for Top-N Recommendation

3.1 Formal Model

Factorization models have proven to be among the best performing approaches as collaborative filtering methods to build a recommender system [21], [28]. Among all the different factorization models, factorization machines propose a unified general model to represent most of them. Here we report the definition and the formalization of a factorization model of order 2 for a recommendation problem involving only implicit ratings. Although, we could extend the model to a more expressive representation by taking into account, e.g., demographic and social information [29], multi-criteria [30], and even relations between contexts [31]. 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). Additionally, we indicate as m the overall number of interactions between users and items recorded on the platform (more precisely the fraction retained in the training set). Then, we denote with $\mathbf{X} \in \mathbb{R}^{m \times n}$ the matrix containing as rows all possible $\mathbf{x^{ui}}$ we can generate starting from the original user-item rating matrix as shown in Fig. 1.



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	...
	U_1	U_2	U_3	U_4	...	I_1	I_2	I_3	I_4	I_5	...
	User					Item					

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

We can define the factorization machine (FM) for each vector $\mathbf{x}$ as:

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

where the parameters are: w_0 representing the global bias; w_j giving the importance to every single x_j; the pair $v_{(j,f)}$ and $v_{(j',f)}$ in $\sum_{f=1}^{k} v_{(j,f)} \cdot v_{(j',f)}$ measuring the strength of the interaction between each pair of variables x_j and $x_{j'}$. We represent the number of latent factors by k. Researchers usually select the k value at design time.

To make the recommendation results computed by `kaHFM` semantically interpretable, we want to inject the knowledge encoded within a knowledge-graph in a Factorization Machine. Given a set of features retrieved from a $\mathcal{KG}$ [32], we first bind them to the latent factors Then, since we address a *Top-N* recommendation problem, we train the model by using the Bayesian Personalized Ranking (BPR) criterion.

In [33], the authors have proposed to encode a Linked Data knowledge graph in a vector space model to develop a CBF recommender system. Given a set of items $I = \{i_1, i_2, \dots, i_N\}$ 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 $F = \{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathcal{KG} \text{ with } i \in I\}$. We can represent each item as a vector of weights $\mathbf{i} = [v_{(i,1)}, \dots, v_{(i,\langle \rho,\omega \rangle)}, \dots, v_{(i,|F|)}]$ where $v_{(i,\langle \rho,\omega \rangle)}$ is the normalized TF-IDF value for $\langle \rho, \omega \rangle$:

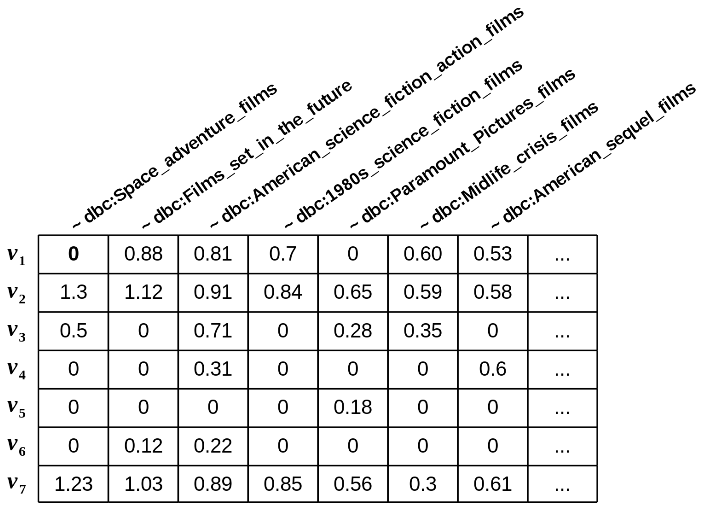
$$\begin{aligned} v_{(i,\langle \rho,\omega \rangle)} &= \underbrace{\frac{|\{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathcal{KG}\}|}{\sqrt{\sum_{\langle \rho,\omega \rangle \in F} |\{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathcal{KG}\}|^2}}}_{TF^{\mathcal{KG}}}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

4. For the sake of conciseness, we use the CURIE syntax that abbreviates URIs using their namespaces. In this paper, we refer to namespaces as available at http://prefix.cc.

Analogously, when we have a set U of users, we may represent users using the features describing the items they have 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, we denote with I^u the set of the items enjoyed by u and we have $\mathbf{u} = [v_{(u,1)}, \ldots, v_{(u,f)} \ldots, v_{(u,|F|)}]$ with

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

Given the vectors $\mathbf{u}_j$, with $j \in [1 \ldots |U|]$, and $\mathbf{i}_{j'}$, with $j' \in [1 \ldots |I|]$, 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 $\mathbf{i}_{j'}$. If we go back to Equation (1), we may see that, for each $\mathbf{x}$, the term $\sum_{j=1}^{n} \sum_{j'=j+1}^{n} x_j \cdot x_{j'} \cdot \sum_{f=1}^{k} v_{(j,f)} \cdot v_{(j',f)}$ is not zero only once, i.e., when both x_j and $x_{j'}$ 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



	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	
$\mathbf{v}_1$	**0**	0.88	0.81	0.7	0	0.60	0.53	...
$\mathbf{v}_2$	1.3	1.12	0.91	0.84	0.65	0.59	0.58	...
$\mathbf{v}_3$	0.5	0	0.71	0	0.28	0.35	0	...
$\mathbf{v}_4$	0	0	0.31	0	0	0	0.6	...
$\mathbf{v}_5$	0	0	0	0	0.18	0	0	...
$\mathbf{v}_6$	0	0.12	0.22	0	0	0	0	...
$\mathbf{v}_7$	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.

summation $\sum_{f=1}^{k} v_{(j,f)} \cdot v_{(j',f)}$ represents the dot product between two vectors $\mathbf{v}_j$ and $\mathbf{v}_{j'}$ with a size equal to k. Hence, $\mathbf{v}_j$ represents a latent representation of a user, $\mathbf{v}_{j'}$ that of an item in the same latent space, and we may evaluate the interaction through their dot product.

First, to inject the knowledge coming from $\mathcal{KG}$ into `kaHFM`, we keep Equation (1) and we set $k = |F|$. In other words, we impose the number of latent factors equal to the number of features describing all the items in our catalog. We want to stress that we aim not to represent each feature through a latent vector, but to associate each factor to a semantic feature. Finally, the latent vectors will be composed only of semantic features. Hence, we initialize the parameters $\mathbf{v}_j$ and $\mathbf{v}_{j'}$ with their corresponding rows from $\mathbf{V}$ which in turn represent respectively $\mathbf{u}_j$ and $\mathbf{i}_{j'}$. 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. However, it contains better values for $v_{(j,f)}$ and $v_{(j',f)}$ 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 $\mathbf{i}_{j'}$ (corresponding to $v_{(j,f)}$ and $v_{(j',f)}$ 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.

In Table 1 and Table 2 we show examples for values after the training (in the column `kaHFM`) together with the original TF-IDF ones computed for two movies from the `Yahoo!Movies`[5] dataset.

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

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

kaHFM	**TF-IDF**	**Predicate**	**Object**
1.2434	0.2858	dct:subject	dbc:Space_adventure_films
1.0355	0.3020	dct:subject	dbc:Films_set_in_the_future
0.8956	0.2605	dct:subject	dbc:American_science_fiction_action_films
0.8951	0.3451	dct:subject	dbc:Android_(robot)_films
0.7338	0.3105	dct:subject	dbc:Time_travel_films
0.6665	0.2701	dct:subject	dbc:Film_scores_by_Jerry_Goldsmith
0.6581	0.2205	dct:subject	dbc:1990s_action_films
0.6561	0.2279	dct:subject	dbc:1990s_science_fiction_films
0.6118	0.1988	dct:subject	dbc:American_sequel_films
0.5649	0.1713	dct:subject	dbc:Paramount_Pictures_films

TABLE 2
Top-10 features computed by `kaHFM` for the movie "`Star Trek - First Contact`".

3.2 Optimization

We can readily train Factorization Machines to reduce the prediction error via gradient descent methods, alternating least-squares (ALS), and MCMC. Since we have formulated our problem as a *Top-N* recommendation task, it is desirable to train `kaHFM` using a learning to rank approach like Bayesian Personalized Ranking Criterion (BPR) [34]. The BPR criterion optimizes 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 random sampling from a uniform distribution. We may formulate BPR optimization criterion as:

$$\begin{aligned} \texttt{BPR-OPT} &= \sum_{(u,i,j) \in D_s} ln\sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2 \\ &= \sum_{(u,i,j) \in D_s} ln\sigma(\hat{y}(\mathbf{x}^{\mathbf{ui}}) - \hat{y}(\mathbf{x}^{\mathbf{uj}})) - \lambda_\Theta \|\Theta\|^2 \end{aligned} \quad (3)$$

In this formulation, $\sigma(\cdot)$ is a sigmoid function, and we may define the update step as:

$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda \Theta \right) \quad (4)$$

where α is the chosen learning rate. Since we are in an implicit feedback setting, we may assume that there is only an instance for the pair user-item. Hence, in our model, we can derive $\hat{x}_{uij}$ as:

$$\begin{aligned} \hat{x}_{uij} &= \hat{y}(\mathbf{x}^{\mathbf{ui}}) - \hat{y}(\mathbf{x}^{\mathbf{uj}}) = w_i x_i - w_j x_j + \\ &+ \sum_{f \in F} x_u x_i v_{(u,f)} v_{(i,f)} - x_u x_j v_{(u,f)} v_{(j,f)} \end{aligned} \quad (5)$$

5. http://research.yahoo.com/Academic_Relations

For `kaHFM`, we can efficiently compute it by estimating the partial derivatives (to update the factorized parameters) for the only active entities involved in the transactions, w_i, w_i, v_u, v_i, and v_j:

$$\frac{\partial}{\partial \Theta}\hat{x}_{uij} = \begin{cases} 1, & \text{if } \theta = w_i, \\ -1, & \text{if } \theta = w_j, \\ v_{(u,f)}, & \text{if } \theta = v_{(i,f)}, \\ -v_{(u,f)}, & \text{if } \theta = v_{(j,f)}, \\ (v_{(i,f)} - v_{(j,f)}), & \text{if } \theta = v_{(u,f)}, \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Applying Equation (6) in Equation (4), we may iteratively update the model parameters to maximize the `BPR-OPT` criterion. The algorithm updates sequentially each sampled triple and continues the training until it reaches the provided number of iterations. Although we consider a high number of factors (features), the approach preserves the linear computational efficiency of BPR. Indeed, we perform each iteration in $\mathcal{O}(m)$ steps, where m is the overall number of transactions in the training set.

3.3 Personalized Recommendation

Once the training phase returns the optimal model parameters, the item recommendation step can take place. We extract the items vectors $\mathbf{v}_j$ from the matrix $\mathbf{V}$, with the associated optimal values. Afterward, we use them to implement an `Item-kNN` recommendation approach[6]. Hence, we measure similarities between each pair of items i and j by evaluating the cosine similarity of their corresponding vectors in $\mathbf{V}$:

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

Given a set of neighbors for the item i, denoted as N^i, such that $i \notin I^u$ and a user u we may predict the score assigned by u to i as:

$$score(u,i) = \frac{\sum\limits_{j \in N^i \cap I^u} cs(i,j)}{\sum\limits_{j \in N^i} cs(i,j)} \tag{7}$$

4 Semantic Accuracy and Generative Robustness of `kaHFM`

The proposed approach lets us keep the explicit meaning of the "latent" factors computed via a factorization machine, thus making possible an interpretation of the recommended results. In the following, we propose an automated offline procedure to assess that `kaHFM` preserves the semantics of the features represented in $\mathbf{V}$ after the training phase. We refer to the values computed by the procedure as *Semantic Accuracy*. A different but related aspect is that of evaluating whether `kaHFM` assigns a higher value to essential features. We refer to this behavior as *Robustness*. Interestingly, we may evaluate both *Semantic Accuracy* and *Robustness* in an offline setting.

6. Please note that this choice leaves the item bias out of the final score. To provide a fair comparison with kNN methods, we have chosen to exploit a standard similarity metric, renouncing to it. Going beyond standard cosine similarity, we may compute a custom similarity by considering, e.g., the product between w_i and w_j: $\frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \cdot \|\mathbf{v}_j\|} + \frac{w_i \cdot w_j}{\max(w_i, w_j)^2}$.

4.1 Semantic Accuracy

The main idea behind Semantic Accuracy is to evaluate, given an item i, how well `kaHFM` can return its original features in the top-K list $\mathbf{v}_i$. In other words, given the set of features of i represented by $F^i = \{f^i_1, \ldots, f^i_m, \ldots f^i_M\}$, with $F^i \subseteq F$, we check if the values in $\mathbf{v}_i$ corresponding to $f^i_m \in F^i$ 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 $\mathbf{v}_i$. Then, we normalize this number by the size of F^i and average on all the items within the catalog I.

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

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 item descriptions may consist of a different number of features, the size of the top list could be a function of the original size of the description. Thus, we have 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 $\mathbf{v}_i$.

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

4.2 Robustness

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

To measure the "meaningfulness" for a given feature, we suppose that a particular feature $\langle \rho, \omega \rangle$ may be useful to describe an item i. Nevertheless, 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.

To implement such a process, we may proceed by following these steps:

- we train `kaHFM` thus obtaining optimal values v_i for all the features in F^i;
- we identify the feature $f^i_{MAX} \in F^i$ with the highest value in v_i;
- we train the model again initializing $f^i_{MAX} = 0$, and we compute v'_i.

Hence, if $f^i_{MAX} \in top(v'_i, M)$, we may say that `kaHFM` shows 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 $f^i_{MAX} \in top(v'_i, M)$ divided by the size of I.

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

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

5 Experimental Evaluation

Datasets. To provide an answer to RQ1, we have evaluated the performance of our method on six datasets belonging to three different domains (Music, Books, and Movies). The Last.fm dataset corresponds to user-artist plays on Last.fm online music system released during *HETRec 2011*[7] *Workshop*. It contains social networking, tagging, artists, and music listening information from a set of $2,000$ users. LibraryThing represents books' ratings collected in the LibraryThing website[8] community. It contains social networking, tagging, and rating information on a [1..10] scale. Yahoo!Movies (*Yahoo! Webscope dataset ydata-ymovies-user-movie-ratings-content-v1_0*)[9] contains movie ratings generated on Yahoo! Movies up to November 2003. It provides content, rating, and demographic information on a [1..5] scale, and mappings to MovieLens and EachMovie datasets. Facebook Movies, Facebook Music, and Facebook Books datasets have been released for the Linked Open Data challenge colocated with *ESWC 2015*[10], and they refer to movies, music, and book domains, respectively. Only implicit feedback is available for these datasets, but for each item, they have provided a link to DBpedia. To map items in Last.fm and LibraryThing to DBpedia resources, we have exploited a freely available mapping[11]. We have extracted all the updated items-features mappings (Yahoo!Movies, LibraryThing,Last.fm, Facebook Movies, Facebook Music and Facebook Books), and we have made them publicly available.[12]. We have shown datasets statistics in Table 3.

Dataset	#Users	#Items	#Transactions	#Features	Sparsity
Yahoo!Movies	4,000	2,626	69,846	988,734	99.34%
LibraryThing	7,223	11,695	410,210	183,182	99.51%
Last.fm	1,375	8,289	60,701	434,817	99.47%
Facebook Music	52,068	5,749	1,374,994	345,452	99.54%
Facebook Movies	32,143	3,901	689,561	180,573	99.45%
Facebook Books	1,398	2,933	18,978	111,401	99.53%

TABLE 3
Datasets statistics.

Evaluation Protocol and Experimental Setting. We have chosen *"All Unrated Items"* [35] protocol to compare different algorithms. In All Unrated Items, we consider as candidate items all the items not yet rated by each user. We have split the dataset using Hold-Out 80-20 splitting strategy, retaining for every user the 80% of their ratings in the training set and the remaining 20% in the test set. Moreover, whenever timestamps associated with every transaction was available, we have performed a temporal split [36], [37]. We have tested our approach against the most related content-based and collaborative algorithms in terms of Accuracy, Diversity, and Novelty [38]. We have compared kaHFM[13] against a canonical 2-degree Factorization Machine (see Section 3.1) optimized via BPR (BPR-FM).

7. http://ir.ii.uam.es/hetrec2011/
8. https://www.librarything.com/
9. http://research.yahoo.com/Academic_Relations
10. https://2015.eswc-conferences.org/program/semwebeval.html
11. https://github.com/sisinflab/LODrecsys-datasets
12. https://github.com/sisinflab/LinkedDatasets/
13. Implementation available at https://github.com/sisinflab/HybridFactorizationMachines/

Moreover, since we exploit items similarity in the last step of our approach (see Equation (7)), we have compared kaHFM against an *Attribute Based Item-kNN* (ABItem-kNN) algorithm, that represents each item as a vector of weights, computed through a TF-IDF model. In ABItem-kNN, we have computed the attributes via Equation (2). For the sake of completeness we have compared kaHFM also against a pure Item-kNN, that is an item-based implementation of the k-nearest neighbors algorithm. Items in the neighborhood are then used to predict a score for each user-item pair. For all the methods that consider neighbors we have considered the neighbors in the range $[10, ...100]$. Regarding BPR parameters, we have considered the *learning rate* varying in $[0.005, ..., 0.5]$, the number of iterations in $[10, ..., 100]$, and the number of factors in $[20, ..., 100]$. The *bias regularization*, *user regularization*, *positive item regularization*, and *negative item regularization* have been set respectively to 0, $\frac{1}{20}$, $\frac{1}{20}$, and $\frac{1}{200}$ of the learning rate, while we have adopted a sampler *"without replacement"* to sample the triples, as suggested by authors [34]. We have compared kaHFM also against the corresponding User-based nearest neighbor scheme (User-kNN), and MostPopular, a simple baseline that shows high performance in specific scenarios [39]. In our context, we have considered mandatory to also compare against a pure knowledge-graph content-based baseline based on the Vector Space Model (VSM) [33]. Finally, to compare our method against state-of-art algorithms, we have chosen VAE-CF and BPR-SSLIM. In VAE-CF [40], the authors train a non-linear probabilistic model taking advantage of Bayesian inference to estimate the parameters. BPR-SSLIM is the BPR optimized variant of $SSLIM$ (specifically $cSLIM$ [41]), a well-known hybrid recommender system. It is based on a popular sparse linear model, $SLIM$, that efficiently learn a joint user-feature vector space exploiting users' implicit feedback.

5.1 Features extraction

The feature extraction is one of the most sensitive steps in our approach. A wrong feature selection may result in noisy data or the lack of some crucial features. We may logically split the preprocessing into three steps: (i) "**Extraction**", in which we retrieve data from the DBpedia knowledge graph, (ii) "**Selection**" where we select the features involved in the specific experiment, and (iii) "**Filtering**" in which we remove uninformative features [32].

Extraction. Thanks to the publicly available mappings, all the items from the datasets come with a DBpedia link. Exploiting this reference, we retrieve all the $\langle \rho, \omega \rangle$ pairs. We have excluded some noisy features (based on the following predicates) already in this early extraction. In particular we have removed: owl:sameAs, dbo:thumbnail, prov:wasDerivedFrom, foaf:depiction, foaf:isPrimaryTopicOf. The rationale is that they give us only a little information about the nature of the entity in the specific knowledge base (e.g., the links between DBpedia and WikiPedia pages) or they are links to multimedia data or even external datasets. After this cleaning step, we have indexed all the features, saved them separately, and associated them with the item *id*.

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

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

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. We have performed the preprocessing phase following [32] and [42] with a single threshold. We have reported the corresponding thresholds (*tm* [32], or, equivalently *p* [42] for missing values) for each dataset in Table 4.

We have discarded features with more than *tm* missing values. For a fair comparison, we have used the same features for Attribute-based Item-kNN (`ABItem-kNN`) and `kaHFM`. We have depicted the characteristics of each dataset (varying the setting) in Table 5.

Dataset	Threshold
`Yahoo!Movies`	99.62
`LibraryThing`	99.91
`Last.fm`	99.88
`Facebook Music`	99.83
`Facebook Movies`	99.74
`Facebook Books`	99.66

TABLE 4
Filtering thresholds.

	Categorical Setting		Ontological Setting		Factual Setting	
Datasets	Total	Selected	Total	Selected	Total	Selected
`Yahoo!Movies`	26155	747	38699	1240	950035	3186
`LibraryThing`	9443	1169	14585	1934	168597	5826
`Last.fm`	16422	1315	30734	3032	404083	9413
`Facebook Music`	15016	1057	27988	2531	317464	7881
`Facebook Movies`	8843	1103	13828	1848	166745	5427
`Facebook Books`	6231	263	9881	592	101520	1315

TABLE 5
Considered features in the different settings

5.2 Accuracy, Diversity and Novelty with `kaHFM`

To evaluate our approach, we have measured accuracy, novelty, and aggregate diversity metrics. The considered accuracy metrics are Precision@N ($Prec@N$) and Normalized Discounted Cumulative Gain ($nDCG@N$). $nDCG@N$ measures the usefulness of a suggested item, considering its relevance and its position in the recommendation list. Hence, we have computed it only for datasets that provide explicit ratings (i.e., `LibraryThing` and `Yahoo!Movies`). Since the recommended results may vary in length depending on the user, cumulative gain at each position is normalized across users. $EPC@N$ (Expected Popularity Complement) is used to measure novelty, or more precisely, the ability of the algorithm to select items that belong to the long tail. Finally, we have measured diversity through *Catalog Coverage* (aggregate diversity in *Top-N* list, $AD@N$), *Gini Index* ($Gini@N$), and *Shannon entropy* ($SE@N$). In particular, *Catalog Coverage* denotes the ability of a system in selecting as many elements as possible from the whole catalog while *Gini index* and *Shannon entropy* are used to measure the distributional inequality with different approaches. Both accuracy and novelty metrics have been computed by averaging their values per-user. To compute those metrics we have used the implementation provided by the RankSys[14] framework. We have performed the evaluation considering Top-10 ([39], [43], [44]) recommendations for all the datasets. When a rating score was available, we have adopted the *Threshold-based relevant items* condition [45] to take into account only relevant items. A relevance threshold of 4/5 and 8/10 has been set for `Yahoo!Movies` and `LibraryThing`, respectively. Finally, we have exploited $Prec@N$, or $nDCG@N$ (where available), to select the best models.

Tables 6, and 7 show the results of our experiments regarding accuracy and diversity. In the tables, we highlight in **bold** the best result while we <u>underline</u> the second one. We have denoted statistically significant results (adopting a Student's paired t-test with a 0.05 level) with a † mark.

`LibraryThing` experiments show that our approach outperforms the competing algorithms for all the considered metrics and settings. Instead, `MostPopular`, and then `User-kNN` show the worst results. If we observe `BPR-FM` aggregate diversity, we may notice relatively high values. This may suggest that `BPR-FM` is struggling to reach convergence.

`Yahoo!Movies` experiments show that in all three settings, our method is the most accurate, while it is the second one regarding diversity. Moreover, it shows similar values to the best performing one: `ABItem-kNN`. It is particularly noteworthy the `VSM` performance regarding accuracy. It behaves in three different ways depending on the considered setting. It is evidently due to the descriptions of items in the dataset, joint with the prediction formula. In this sense, `ABItem-kNN` performance confirms that applying an item-based scheme to features vectors leads to better results.

In `Last.fm`, `VAE-CF` is the best performing method regarding the accuracy, followed by `BPR-SSLIM`. However, both show low values concerning diversity. On the other hand, `kaHFM` shows a good trade-off between accuracy and diversity.

Even in `Facebook Movies`, `kaHFM` shows to be the most precise algorithm for all the considered settings. Regarding diversity, it is only the second-best after `ABItem-kNN`. However, `kaHFM` almost doubles the accuracy of `ABItem-kNN`, preserving high diversity values. On the other hand, `BPR-SSLIM` shows good accuracy results, but it provides less diversified and tailored recommendations.

Finally, `Facebook Music` and `Facebook Books` show a similar trend. `kaHFM` shows the best accuracy performance in all the experiments but the `Facebook Music` Factual Setting, in which it is still very close to `VSM`. Furthermore, for the two datasets, `ABItem-kNN` and `kaHFM` show the best diversity values, providing more personalized recommendation lists. More, we may notice that `VAE-CF` and `BPR-SSLIM` show excellent accuracy performance in `Facebook Books`, but even there, they provide less diversified recommendations.

Let us discuss the baselines more related to our approach. We have compared `kaHFM` against `ABItem-kNN` to check whether the collaborative trained features may lead to better similarity values. This hypothesis seems to be con-

14. http://ranksys.org/

	Facebook Movies					Facebook Music					Facebook Books				
Categorical Setting (CS)	Prec	EPC	AD	Gini	SE	Prec	EPC	AD	Gini	SE	Prec	EPC	AD	Gini	SE
ABItem-kNN	0.0197†	0.0224†	**3572**	**0.2411**	**9.8964**	0.0213†	0.0220†	5085	**0.2186**	**10.4950**	0.0102†	0.0111†	**1909**	**0.2808**	**9.9332**
BPR-FM	0.0126†	0.0123†	2468	0.0788	8.4310	0.0252†	0.0203†	1001	0.0052	4.9461	0.0066†	0.0065†	500	0.0402	7.1683
MostPopular	0.0118†	0.0099†	27	0.0029	3.8543	0.0146†	0.0089†	30	0.0020	3.8628	0.0032†	0.0030†	17	0.0034	3.6193
Item-kNN	0.0044†	0.0039†	731	0.0118	5.8832	0.0011†	0.0009†	894	0.0090	5.8271	0.0029†	0.0026†	332	0.0199	6.1754
User-kNN	0.0019†	0.0010†	260	0.0124	5.9282	0.0000†	0.0000†	233	0.0054	5.2449	0.0009†	0.0005†	56	0.0055	4.4969
VSM	0.0185†	0.0205†	3326	0.1769	9.5856	0.0289†	0.0325†	4581	0.1395	9.6625	0.0104†	0.0112†	1832	0.2631	9.8733
kaHFM	**0.0346**	**0.0371**	3409	0.1783	9.6429	**0.0360**	**0.0383**	**5146**	0.1912	9.9441	**0.0173**	**0.0196**	**1909**	0.2716	9.9037
VAE-CF	0.0177†	0.0169†	1717	0.0354	7.1551	0.0159†	0.0150†	2261	0.0552	8.5061	0.0119†	0.0115†	441	0.0401	7.2685
BPR-SSLIM	0.0240†	0.0233†	1432	0.0309	7.0290	0.0212†	0.0206†	2004	0.0350	7.8513	0.0118†	0.0122†	662	0.0381	7.0001
Ontological Setting (OS)	Prec	EPC	AD	Gini	SE	Prec	EPC	AD	Gini	SE	Prec	EPC	AD	Gini	SE
ABItem-kNN	0.0127†	0.0137†	**3553**	**0.2768**	**10.1921**	0.0222†	0.0232†	4798	**0.1961**	**10.1558**	0.0084†	0.0094†	2061	0.3003	10.0758
BPR-FM	0.0142†	0.0120†	39	0.0030	3.8999	0.0242†	0.0219†	1868	0.0397	7.9459	0.0102†	0.0081†	84	0.0048	4.3193
MostPopular	0.0123†	0.0102†	26	0.0029	3.8531	0.0114†	0.0070†	31	0.0020	3.8674	0.0033†	0.0031†	17	0.0034	3.6195
Item-kNN	0.0044†	0.0039†	711	0.0117	5.8784	0.0010†	0.0008†	911	0.0089	5.8077	0.0028†	0.0028†	326	0.0208	6.2392
User-kNN	0.0017†	0.0009†	245	0.0122	5.9147	0.0001†	0.0000†	235	0.0054	5.2318	0.0009†	0.0005†	79	0.0069	4.7921
VSM	0.0111†	0.0117†	2922	0.1194	8.8326	0.0117†	0.0123†	3475	0.0734	8.6511	0.0047†	0.0054†	1287	0.1254	8.7828
kaHFM	**0.0303**	**0.0333**	3537	0.2383	10.0461	**0.0315**	**0.0332**	**5072**	0.1892	9.9909	**0.0139**	**0.0158**	**2229**	**0.3359**	**10.2359**
VAE-CF	0.0179†	0.0171†	2080	0.0643	7.9327	0.0164†	0.0156†	2684	0.0629	8.6394	0.0135	0.0143	501	0.0484	7.5515
BPR-SSLIM	0.0256†	0.0248†	1297	0.0256	6.8169	0.0217†	0.0210†	2473	0.0466	8.2454	0.0124	0.0123†	782	0.0442	7.1501
Factual Setting (FS)	Prec	EPC	AD	Gini	SE	Prec	EPC	AD	Gini	SE	Prec	EPC	AD	Gini	SE
ABItem-kNN	0.0238†	0.0284†	**3642**	**0.2161**	**9.7051**	0.0331†	0.0348†	**5154**	**0.2006**	10.2277	0.0168†	0.0193†	**2261**	**0.3424**	**10.2636**
BPR-FM	0.0138†	0.0126†	2308	0.0624	7.8778	0.0134†	0.0127†	3200	0.0489	8.1593	0.0074†	0.0075†	477	0.0414	7.2526
MostPopular	0.0123†	0.0102†	26	0.0029	3.8531	0.0114†	0.0070†	31	0.0020	3.8674	0.0033†	0.0031†	17	0.0034	3.6195
Item-kNN	0.0044†	0.0039†	706	0.0117	5.8773	0.0010†	0.0008†	907	0.0089	5.8084	0.0028†	0.0028†	325	0.0207	6.2354
User-kNN	0.0017†	0.0009†	247	0.0122	5.9175	0.0001†	0.0000†	233	0.0052	5.1983	0.0009†	0.0005†	80	0.0069	4.7900
VSM	0.0228†	0.0259†	2980	0.1064	8.7811	**0.0362**	**0.0380**	3998	0.1018	9.2477	0.0129†	0.0144†	1874	0.2594	9.8843
kaHFM	**0.0331**	**0.0356**	3308	0.1512	9.5290	0.0351†	0.0367†	4886	0.1795	**10.2406**	**0.0212**	**0.0242**	2074	0.2630	9.8184
VAE-CF	0.0185†	0.0181†	873	0.0337	7.3088	0.0161†	0.0153†	2346	0.0565	8.5414	0.0134†	0.0132†	559	0.0459	7.4347
BPR-SSLIM	0.0278†	0.0258†	334	0.0143	6.0898	0.0233†	0.0224†	2513	0.0422	8.1154	0.0161†	0.0159†	839	0.0463	7.1440

TABLE 6

Accuracy, Diversity and Novelty results for Facebook Movies, Facebook Music and Facebook Books considering *Top*-10 recommendations

	LibraryThing						Yahoo!Movies						Last.fm				
Categorical Set.	Prec	nDCG	EPC	AD	Gini	SE	Prec	nDCG	EPC	AD	Gini	SE	Prec	EPC	AD	Gini	SE
ABItem-kNN	0.0474†	0.0588†	0.0514†	8009	0.2173	11.4276	0.0428†	0.1192†	0.0536†	**2456**	**0.3529**	**10.0210**	0.0249†	0.0240†	**4022**	**0.2181**	10.9180
BPR-FM	0.0287†	0.0431†	0.0304†	2455	0.0301	8.1288	0.0278†	0.0557†	0.0275†	490	0.0178	5.5298	0.0421†	0.0454†	779	0.0182	7.2498
MostPopular	0.0056†	0.0058†	0.0051†	34	0.0009	3.8301	0.0154†	0.0271†	0.0148†	48	0.0043	3.9038	0.0252†	0.0233†	35	0.0012	3.7052
Item-kNN	0.0639†	0.0961†	0.0782†	7944	0.2364	11.5397	0.0246†	0.0572†	0.0255†	1772	0.2106	9.4286	0.0312†	0.0318†	3235	0.1542	10.4596
User-kNN	0.0214†	0.0346†	0.0226†	1330	0.0103	6.7102	0.0234†	0.0484†	0.0239†	846	0.0451	7.0627	0.0402†	0.0448†	1075	0.0229	7.5653
VSM	0.0367†	0.0473†	0.0394†	7434	0.2106	11.4198	0.0385†	0.1129†	0.0496†	2320	0.2893	9.7604	0.0313†	0.0320†	3660	0.1881	10.7752
kaHFM	**0.0761**	**0.1115**	**0.0887**	**8251**	**0.2455**	**11.7059**	**0.0552**	**0.1518**	**0.0667**	2409	0.3150	9.8375	0.0420†	0.0436†	3847	0.2069	**10.9591**
VAE-CF	0.0282†	0.0478†	0.0307†	4773	0.0927	10.0765	0.0396†	0.0964†	0.0431†	1028	0.0926	8.1375	**0.0647**	**0.0700**	1509	0.0559	9.0385
BPR-SSLIM	0.0486†	0.0784†	0.0543†	4292	0.0650	9.3993	0.0450†	0.1167†	0.0483†	1487	0.1025	8.0746	0.0497	0.0529	2364	0.0821	9.4138
Ontological Set.	Prec	nDCG	EPC	AD	Gini	SE	Prec	nDCG	EPC	AD	Gini	SE	Prec	EPC	AD	Gini	SE
ABItem-kNN	0.0474†	0.0588†	0.0514†	8009	0.2173	11.4278	0.0183†	0.0477†	0.0211†	**2312**	**0.3321**	**10.0454**	0.0252†	0.0254†	3955	0.2262	11.0824
BPR-FM	0.0287†	0.0431†	0.0304†	2455	0.0301	8.1288	0.0278†	0.0557†	0.0275†	490	0.0178	5.5298	0.0435†	0.0471†	605	0.0112	6.4690
MostPopular	0.0056†	0.0058†	0.0051†	34	0.0009	3.8301	0.0154†	0.0271†	0.0148†	48	0.0043	3.9038	0.0252†	0.0233†	35	0.0012	3.7052
Item-kNN	0.0639†	0.0961†	0.0782†	7944	0.2364	11.5397	0.0246†	0.0572†	0.0255†	1772	0.2106	9.4286	0.0313†	0.0317†	3275	0.1572	10.4850
User-kNN	0.0214†	0.0346†	0.0226†	1330	0.0103	6.7102	0.0234†	0.0484†	0.0239†	846	0.0451	7.0627	0.0410†	0.0454†	1098	0.0233	7.5724
VSM	0.0367†	0.0473†	0.0394†	7434	0.2106	11.4198	0.0099†	0.0325†	0.0125†	1928	0.1883	9.0297	0.0092†	0.0094†	2533	0.0934	9.5057
kaHFM	**0.0761**	**0.1116**	**0.0887**	**8251**	**0.2455**	**11.7062**	**0.0475**	**0.1318**	**0.0573**	2310	0.2976	9.8485	0.0402†	0.0427†	**4193**	**0.2442**	**11.2105**
VAE-CF	0.0280†	0.0478†	0.0313†	4600	0.0861	9.9531	0.0371†	0.0951†	0.0411†	1292	0.1196	8.4051	**0.0648**	**0.0707**	1185	0.0372	8.4108
BPR-SSLIM	0.0486†	0.0784†	0.0543†	4292	0.0650	9.3993	0.0447†	0.1129†	0.0472†	1460	0.0949	7.9013	0.0490	0.0522†	2384	0.0794	9.2908
Factual Set.	Prec	nDCG	EPC	AD	Gini	SE	Prec	nDCG	EPC	AD	Gini	SE	Prec	EPC	AD	Gini	SE
ABItem-kNN	0.0534†	0.0667†	0.0571†	8303	0.2190	11.3562	0.0588†	0.1722†	0.0750†	2444	**0.3219**	**9.7896**	0.0396†	0.0382†	**4205**	**0.2385**	**11.1066**
BPR-FM	0.0094†	0.0117†	0.0094†	2975	0.0399	8.9305	0.0250†	0.0505†	0.0252†	878	0.0573	7.1329	0.0452†	0.0495†	640	0.0119	6.5636
MostPopular	0.0056†	0.0058†	0.0051†	34	0.0009	3.8301	0.0154†	0.0271†	0.0148†	48	0.0043	3.9038	0.0252†	0.0233†	35	0.0012	3.7052
Item-kNN	0.0662†	0.0993†	0.0803†	8220	0.2447	11.5963	0.0246†	0.0572†	0.0255†	1772	0.2106	9.4286	0.0319†	0.0321†	3286	0.1577	10.4899
User-kNN	0.0220†	0.0350†	0.0232†	1460	0.0117	6.8666	0.0234†	0.0484†	0.0239†	846	0.0451	7.0627	0.0412†	0.0457†	1112	0.0236	7.5784
VSM	0.0460†	0.0571†	0.0494†	6973	0.1968	11.3973	0.0603†	0.1665†	0.0730†	2233	0.2409	9.2662	0.0430†	0.0421†	3370	0.1639	10.5255
kaHFM	**0.0767**	**0.1135**	**0.0891**	**8442**	**0.2499**	**11.7381**	**0.0655**	**0.1816**	**0.0806**	**2388**	0.2806	9.6617	0.0473†	0.0468†	4034	0.2171	11.0217
VAE-CF	0.0251†	0.0466†	0.0282†	5592	0.1217	10.4968	0.0386†	0.0948†	0.0421†	1230	0.1122	8.3720	**0.0651**	**0.0698**	1264	0.0400	8.5197
BPR-SSLIM	0.0551†	0.0866†	0.0606†	4912	0.0723	9.3843	0.0529†	0.1371†	0.0577†	1683	0.1096	8.0641	0.0549	0.0611	2781	0.1058	9.6969

TABLE 7

Accuracy, Diversity and Novelty results for LibraryThing, Yahoo!Movies and Last.fm considering *Top*-10 recommendations firmed since kaHFM always beats ABItem-kNN regarding accuracy. Moreover, we want to check if a knowledge-graph-based initialization of latent factors may improve the performance of Factorization Machines. kaHFM beats BPR-FM 16 times over 18. This behavior may happen since the random initialization takes a while to drive the Factorization Machine to reach competitive performance. However, we have specifically investigated this aspect in section 5.3. Lastly, we are interested in checking whether collaborative trained features lead to better accuracy results than a purely informativeness-based Vector Space Model (in detail its knowledge-aware variant). It seems to be confirmed through experiments since kaHFM beats VSM 17 times over 18. Although we have measured very interesting results, it is worth mentioning that accuracy and interpretability come at a cost. Indeed, public DBpedia mappings are not always available. In this sense, the designer should ask either domain experts to describe the items semantically or provide a mapping. Moreover, the method could not recommend not-mapped items, thus forcing to work on dataset linking (toward DBpedia or other public datasets). Finally, the quality itself of the knowledge base may have an impact. Nevertheless, this aspect deserves a specific investigation that is outside the scope of this work.

5.3 Evolution of **kaHFM** performance over iterations

In this experiment, we try to assess some crucial aspects. First, we want to know whether the knowledge-graph-based initialization is generally better than standard random initialization. Second, we desire to investigate if kaHFM can reach convergence sooner than BPR-FM. Third, we want to assess whether kaHFM training generally improves the original representation of features (for a *Top-N* recommendation task). For the experimental evaluation, we have analyzed kaHFM, BPR-FM, and ABItem-kNN performance. Since we need to compare them fairly, we have considered the same underlying model. In detail, we have used the same parameters adopted for kaHFM and the same number of hidden factors (see the *"Selected"* column in Table 5). Regarding BPR parameters, we have set *learning rate*, *bias regularization*, *user regularization*, *positive item regularization*, and *negative item regularization* to 0.05, 0, 0.0025, 0.0025, and 0.00025, respectively. For the sake of reproducibility, the BPR parameters correspond to mymedialite[15] implementation. Then, we have produced recommendations for the Categorical Setting of the six datasets considering 0,1,5,10,15,30 iterations. Results considering the other settings are available online[16]. In Figure 3, the plots show the evolution of Precision@10 for the considered iterations. First, we may focus on the difference between BPR-FM and ABItem-kNN. In almost all cases, ABItem-kNN Precision is higher than the BPR-FM starting value. It denotes that generally, the semantic features initialized with TF-IDF perform better than random. Of course, this cannot be a conclusive remark, because BPR associates an implicit meaning to the single factors during training. We may only affirm that a TF-IDF initialization puts features weights closer to a local minimum. Then, we may compare

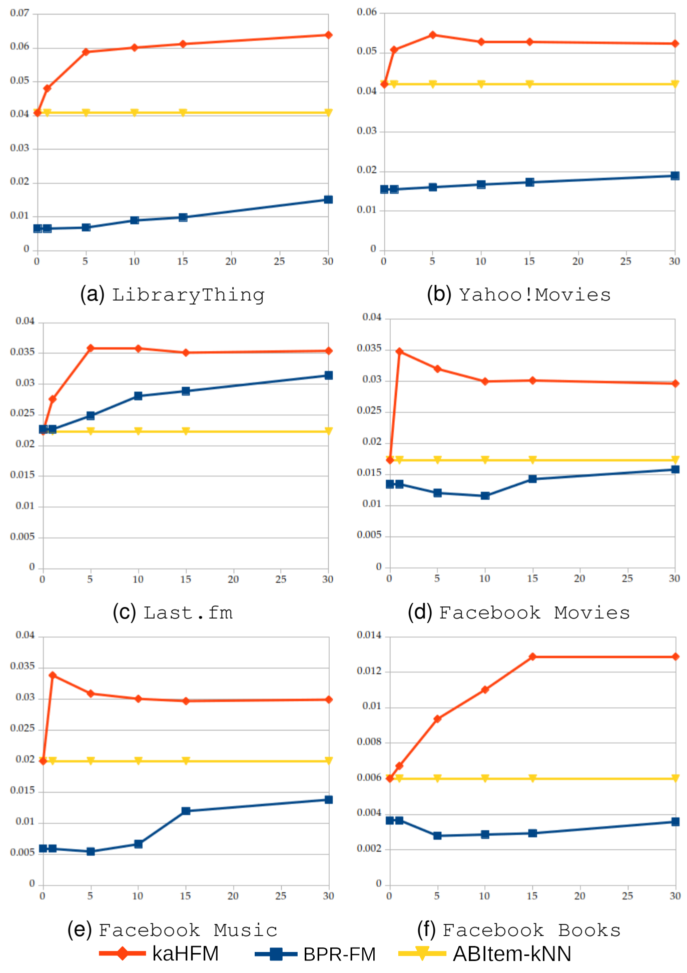


Fig. 3. Precision@10 varying # iterations 0, 1, 5 , 10 , 15, 30

the behavior of kaHFM and BPR-FM. The six pairs of curves show that kaHFM moves faster toward convergence. Finally, we may compare kaHFM and ABItem-kNN. All the plots seem to suggest that our approach improves the original item representation. However, it only implies that it leads to a Precision improvement, without any considerations about the preservation of the semantics. Given the obtained results we may say that the answer to RQ1 is positive when adopting kaHFM.

5.4 Semantic Accuracy

The previous experiments have shown the effectiveness of kaHFM in terms of accuracy, diversity, and novelty. In practical terms, we have shown that: (i) content initialization generally leads to better performance with kaHFM, (ii) the obtained items vectors are better fine-tuned than the original ones for a *Top-N* item recommendation task, (iii) results may depend on the kind of features we extract from the Knowledge Graph. However, we still do not know whether kaHFM preserves the original semantics of the features after the training (as we want to assess by posing RQ2). In Section 4.1, we have introduced Semantics Accuracy (SA@nM) as a metric to automatically check if item feature values reflect the actual meaning of that feature. Thus, we have measured SA@nM with $n \in \{1, 2, 3, 4, 5\}$ and $M = 10$, and we have evaluated the number of ground features available in the *Top-*nM elements of $\mathbf{v}_i$ for each of the six datasets.

15. http://www.mymedialite.net/
16. https://github.com/sisinflab/papers-results/tree/master/kahfm-results/

Semantics Accuracy	@M	@2M	@3M	@4M	@5M	F.A.
Yahoo!Movies	0.847	0.863	0.865	0.868	0.873	12.143
LibraryThing	0.960	0.996	0.998	0.999	0.999	3.820
Last.fm	0.960	0.987	0.991	0.994	0.995	6.615
Facebook Music	0.892	0.948	0.962	0.970	0.974	7.113
Facebook Movies	0.864	0.883	0.889	0.894	0.899	12.856
Facebook Books	0.995	1	1	1	1	3.133

TABLE 8
Semantics Accuracy results for different values of M. F.A. denotes the Feature Average number per item.

Table 8 shows the results for the Categorical setting of all the different datasets. In general, the results we obtain are noteworthy. As an example, we may examine the worst one to describe better the meaning of the values we obtain. In Yahoo!Movies Categorical setting, 747 different features compose each item vector (see Table 5). After the training phase, on average, more than ten (equal to $0.847 \cdot 12.143$) over twelve features (last column in Table 8) correspond to the original features. Consequently, kaHFM has computed almost the same best features starting from hundreds of them. Even then, the obtained results may provide an affirmative answer to RQ2.

5.5 Generative Robustness

In section 4.2, we have introduced a procedure to measure the capability of kaHFM to compute meaningful features. Here, we have computed 1-Rob@nM for the six adopted datasets, and we have depicted the results in Table 9.

1-Robustness	@M	@2M	@3M	@4M	@5M	F.A.
Yahoo!Movies	0.487	0.645	0.713	0.756	0.793	12.143
LibraryThing	0.275	0.481	0.554	0.597	0.632	3.820
Last.fm	0.125	0.281	0.346	0.394	0.430	6.615
Facebook Music	0.714	0.893	0.935	0.955	0.966	7.113
Facebook Movies	0.821	0.945	0.970	0.980	0.984	12.856
Facebook Books	0.315	0.516	0.605	0.682	0.745	3.133

TABLE 9
1-Robustness for different values of M. Column F.A. denotes the Feature Average number per item.

Even here, we focus on the CS setting. For a better understanding, we may start by focusing on Yahoo!Movies, which seemingly shows a bad behavior. As mentioned earlier, Table 8 shows that kaHFM has guessed ten on twelve different features for Yahoo!Movies. In this experiment, we eliminate a feature, thus making kaHFM to guess an average of nine over twelve features. What we assess is whether kaHFM can guess the removed feature in the remaining three slots. Results in Table 9 show that kaHFM puts that single removed feature in one of the three slots, the 48.7% of the times choosing among 747 overall features. We believe the example may help to appreciate even more the results on Facebook Music and Facebook Movies. For the remaining datasets, Table 8 shows that there are no free slots. Thus, after removing a feature, kaHFM has only one missing slot to fill with the right feature. Let us focus on Facebook Books. It contains 263 different features in the item vector (see Table 5) and a low average number of features per item in the $\mathcal{KG}$ (3.133). Results show that kaHFM fills correctly the missing slot 31% of the cases. Overall, results show that kaHFM promotes essential features, as asked by RQ3.

6 Related Work

The core of our model is a Factorization Machines (FM) model [21]. 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++ [46], PITF [47], FPMC [48]. First, FMs are designed for a generic prediction task while the others tackle specific tasks. Moreover, a FM with n features is equivalent to a linear model with $n + n * (n-1)/2$ features 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 [49] have been developed to fix the inability of classical FMs to capture non linear structure of real-world data. Furthermore, Attentional Factorization Machines [50] have been proposed that use an attention network to learn the importance of feature interactions. The factorization models have also been adopted to feed Active Learning-based Recommender Systems [51]. In detail, the authors propose the original idea of Dynamic Active Learning Budget to distribute the limited active learning resources. Usually only top recommended items are provided to the user. For this reason, ranking has become a much more important task than rating prediction [52]. This has led to *Learning to Rank* algorithms that can be further categorized in Point-wise [53], Pair-wise [34] and List-wise [44]. In particular, Pair-wise approaches are usually considered as a good trade-off between ordering performances and computational complexity. Among this class of algorithms, Bayesian Personalized Ranking (BPR) [34] is one of the most widely adopted. It exploits stochastic gradient descent algorithm to learn the relative order between positive and negative items (see Section 3.2). BPR can be applied to Matrix Factorization and Factorization Machines (as in our work and in [54]).

One of the first attempts to overcome the interpretability problem for matrix factorization is the Explicit Factor Model (EFM) [18], [55]. Products' features and users' opinions are extracted with phrase-level sentiment analysis from users' reviews to feed a matrix factorization framework. After that, we have observed a few improvements to EFM to deal with temporal dynamics [56] and to use tensor factorization [19]. They also adopt pair-wise learning to rank approach (in their case BPR on a three-fold user-item-feature space). A further advance in MF-based explainable recommendation models is Explainable Matrix Factorization (EMF) [57] in which they exploit a neighborhood model to generate explanations. Similarly, in [8], the authors propose an explainable Restricted Boltzmann Machine model. It learns a network model (with an additional visible layer) that takes into account a degree of explainability. In [58], the authors compute recommendations by generating and ranking personalized explanations in the form of explanation chains. OCuLaR [59] provides interpretable recommendations from positive examples based on the detection of co-clusters between users (clients) and items (products). In [60], the authors propose a Multi-Level Attraction Model (MLAM) in which they build two attraction models for cast and story. It provides the interpretability of the model in terms of attractiveness of the Sentence level, Word level, and Cast member. In [61], the authors train a matrix factorization model to complete the $U \times I$ matrix. They then use the complete (approximated) rating matrix to compute a set of association rules that explain the obtained recommendations. Among the works that exploit content information to produce explainable recommendations, Tagsplanations [62] is worth to mention. Community tags feed it, and it exploits a relevance measure to weight tags considering items, and user preferences. Furthermore, also demographic-based recommendation explanations have been inspected [23], to recommend items for specific types (age, location, gender) of users. Among model-based recommender systems, [63] proposes to reconstruct the user-item matrix using a non-convex function to approximate the rank in the *Top-N* recommendation setting. A particularly interesting work is [64], where authors have proposed to overpass the classic limitations of similarity-based methods, taking advantage of kernels. It is worth noticing that the techniques in the latter may drive an extension of `kaHFM`.

7 Conclusion and Future Work

In this work, we have proposed an interpretable method, `kaHFM`, in which we bind the meaning of latent factors for a Factorization machine to data coming from a knowledge graph. We have evaluated `kaHFM` on six different publicly available datasets, and we have compared it against state-of-the-art algorithms.

Results show that our approach generally outperforms the other approaches concerning accuracy, diversity, and novelty, considering different sets of semantics-aware features. We have shown that the generated recommendation lists are more precise and personalized, and they select more items from the long tail. In particular, we have considered Ontological, Categorical, and Factual information extracted from a freely available knowledge graph. Summing up, the experimental evaluation shows that: (`RQ1`) the learned model shows competitive performance regarding the accuracy, novelty, and diversity and, at the same time, is effectively interpretable; (`RQ2`) the estimated features show to be semantically meaningful; (`RQ3`) the model is robust to re-generate essential features after removing them.

In the future, we are interested in testing `kaHFM` in different scenarios, other than recommender systems. Moreover, we are already working on some model improvements. In detail, we are interested in considering other metrics for relevance that may precisely fit particular scenarios. In fact, this is possible since the method itself is agnostic to the specific adopted measure. Furthermore, it would be useful to exploit `kaHFM` to provide suggestions to knowledge graphs maintainers for adding relevant missing features to the knowledge base. In this sense, we would like to evaluate our approach in knowledge graph completion tasks.

References

[1] R. R. Sinha and K. Swearingen, "The role of transparency in recommender systems," in *Extended abstracts of the 2002 Conf. on Human Factors in Computing Systems, CHI 2002, Minneapolis, Minnesota, USA, April 20-25, 2002*, 2002, pp. 830–831.

[2] N. Tintarev and J. Masthoff, "Designing and evaluating explanations for recommender systems," in *Recommender Systems Handbook*, 2011, pp. 479–510.

[3] M. Zanker, "The influence of knowledgeable explanations on users' perception of a recommender system," in *Sixth ACM Conf. on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, 2012, pp. 269–272.

[4] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *CSCW 2000, Proceeding on the ACM 2000 Conf. on Computer Supported Cooperative Work, Philadelphia, PA, USA, December 2-6, 2000*, 2000, pp. 241–250.

[5] R. Falcone, A. Sapienza, and C. Castelfranchi, "The relevance of categories for trusting information sources," *ACM Trans. Internet Techn.*, vol. 15, no. 4, pp. 13:1–13:21, 2015.

[6] N. Drawel, H. Qu, J. Bentahar, and E. Shakshuki, "Specification and automatic verification of trust-based multi-agent systems," *Future Generation Computer Systems*, 2018.

[7] X. Li, H. Fang, Q. Yang, and J. Zhang, "Who is your best friend?: Ranking social network friends according to trust relationship," in *Proc. of the 26th Conf. on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018*, 2018, pp. 301–309.

[8] B. Abdollahi and O. Nasraoui, "Explainable restricted boltzmann machines for collaborative filtering," *CoRR*, vol. abs/1606.07129, 2016.

[9] H. S. M. Cramer, V. Evers, S. Ramlal, M. van Someren, L. Rutledge, N. Stash, L. Aroyo, and B. J. Wielinga, "The effects of transparency on trust in and acceptance of a content-based art recommender," *User Model. User-Adapt. Interact.*, vol. 18, no. 5, pp. 455–496, 2008.

[10] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ragone, and J. Trotta, "The importance of being dissimilar in recommendation," in *Proc. of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus*, C. Hung and G. A. Papadopoulos, Eds. ACM, 2019, pp. 816–821.

[11] S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. B. Srivastava, A. D. Preece, S. Julier, R. M. Rao, T. D. Kelley, D. Braines, M. Sensoy, C. J. Willis, and P. Gurram, "Interpretability of deep learning models: A survey of results," in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI 2017, San Francisco, CA, USA, August 4-8, 2017*, 2017, pp. 1–6.

[12] Y. Zhu, Y. Gong, Q. Liu, Y. Ma, W. Ou, J. Zhu, B. Wang, Z. Guan, and D. Cai, "Query-based interactive recommendation by meta-path and adapted attention-gru," in *Proc. of the 28th ACM Int. Conf. on Information and Knowledge Management, CIKM*. ACM, 2019, pp. 2585–2593.

[13] Y. Zhu, J. Zhu, J. Hou, Y. Li, B. Wang, Z. Guan, and D. Cai, "A brand-level ranking system with the customized attention-gru model," in *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence, IJCAI*. ijcai.org, 2018, pp. 3947–3953.

[14] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by time-lstm," in *Proc. of the 26th Int. Joint Conf. on Artificial Intelligence, IJCAI*. ijcai.org, 2017, pp. 3602–3608.

[15] Y. Koren, R. M. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[16] X. Wang, X. He, F. Feng, L. Nie, and T. Chua, "TEM: tree-enhanced embedding model for explainable recommendation," in *Proc. of the 2018 World Wide Web Conf. on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, 2018, pp. 1543–1552.

[17] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L. Huang, and C. Xu, "Recurrent knowledge graph embedding for effective recommendation," in *Proc. of the 12th ACM Conf. on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, 2018, pp. 297–305.

[18] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis," in *The 37th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, 2014, pp. 83–92.

[19] X. Chen, Z. Qin, Y. Zhang, and T. Xu, "Learning to rank features for recommendation over multiple categories," in *Proc. of the 39th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2016, pp. 305–314.

[20] Y. Zhang and X. Chen, "Explainable recommendation: A survey and new perspectives," *CoRR*, vol. abs/1804.11192, 2018.

[21] S. Rendle, "Factorization machines," in *Proc. of the 10th IEEE Int. Conf. on Data Mining ICDM*, 2010, pp. 995–1000.

[22] Y. Zhu, Z. Guan, S. Tan, H. Liu, D. Cai, and X. He, "Heterogeneous hypergraph embedding for document recommendation," *Neurocomputing*, vol. 216, pp. 150–162, 2016.

[23] W. X. Zhao, S. Li, Y. He, L. Wang, J. Wen, and X. Li, "Exploring demographic information in social media for product recommendation," *Knowl. Inf. Syst.*, vol. 49, no. 1, pp. 61–89, 2016.

[24] T. Di Noia, R. Mirizzi, V. C. Ostuni, and D. Romito, "Exploiting the web of data in model-based recommender systems," in *Sixth ACM Conf. on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, 2012, pp. 253–256.

[25] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, May 2001.

[26] V. W. Anelli, A. Calì, T. D. Noia, M. Palmonari, and A. Ragone, "Exposing open street map in the linked data cloud," in *Proc. of 29th Int. Conf. on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE*, ser. Lecture Notes in Computer Science, H. Fujita, M. Ali, A. Selamat, J. Sasaki, and M. Kurematsu, Eds., vol. 9799. Springer, 2016, pp. 344–355.

[27] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ragone, and J. Trotta, "How to make latent factors interpretable by feeding factorization machines with knowledge graphs," in *The Semantic Web - ISWC 2019 - Proc. of 18th Int. Semantic Web Conf., Auckland, New Zealand, Part I*, ser. Lecture Notes in Computer Science, vol. 11778. Springer, 2019, pp. 38–56.

[28] V. W. Anelli, T. D. Noia, E. D. Sciascio, C. Pomo, and A. Ragone, "On the discriminative power of hyper-parameters in cross-validation and how to choose them," in *Proc. of the 13th ACM Conf. on Recommender Systems, RecSys 2019, Copenhagen, Denmark*. ACM, 2019, pp. 447–451.

[29] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender Systems Handbook*, 2015, pp. 191–226.

[30] G. Adomavicius and Y. Kwon, "Multi-criteria recommender systems," in *Recommender Systems Handbook*, 2015, pp. 847–880.

[31] Y. Zheng, B. Mobasher, and R. D. Burke, "Incorporating context correlation into context-aware matrix factorization," in *Proc. of the IJCAI 2015 Joint Workshop CPCR+ITWP co-located with the 24th Int. Joint Conf. on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 27, 2015.*, 2015.

[32] T. Di Noia, C. Magarelli, A. Maurino, M. Palmonari, and A. Rula, "Using ontology-based data summarization to develop semantics-aware recommender systems," in *The Semantic Web, Proc. of 15th Int. Conf. ESWC*, 2018, pp. 128–144.

[33] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker, "Linked open data to support content-based recommender systems," in *I-SEMANTICS 2012 - 8th Int. Conf. on Semantic Systems*, 2012, pp. 1–8.

[34] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," in *UAI 2009, Proc. of the 25th Conf. on Uncertainty in Artificial Intelligence, Montreal, QC, Canada*, 2009, pp. 452–461.

[35] H. Steck, "Evaluation of recommendations: rating-prediction and ranking," in *Proc. of the 7th ACM Conf. on Recommender systems*. ACM, 2013, pp. 213–220.

[36] A. Gunawardana and G. Shani, "Evaluating recommender systems," in *Recommender Systems Handbook*, 2015, pp. 265–308.

[37] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ragone, and J. Trotta, "Local popularity and time in top-n recommendation," in *Proc. of 41st European Conf. on IR Research, ECIR 2019, Cologne, Germany, Part I*, vol. 11437. Springer, 2019, pp. 861–868.

[38] V. W. Anelli, V. Bellini, T. D. Noia, W. L. Bruna, P. Tomeo, and E. D. Sciascio, "An analysis on time- and session-aware diversification in recommender systems," in *Proc. of the 25th Conf. on User Modeling, Adaptation and Personalization, UMAP*. ACM, 2017, pp. 270–274.

[39] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proc. of the 4th ACM Conf. on Recommender Systems*, 2010, pp. 39–46.

[40] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. of the 2018 World Wide Web Conf. on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, P. Champin, F. L. Gandon, M. Lalmas, and P. G. Ipeirotis, Eds. ACM, 2018, pp. 689–698.

[41] X. Ning and G. Karypis, "Sparse linear methods with side information for top-n recommendations," in *6th ACM Conf. on Recommender Systems, RecSys*. ACM, 2012, pp. 155–162.

[42] H. Paulheim and J. Fürnkranz, "Unsupervised generation of data mining features from linked open data," in *2nd Int. Conf. on Web Intelligence, Mining and Semantics, WIMS*, 2012, pp. 31:1–31:12.

[43] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proc. of the 24th Int. Conf. on Machine Learning ICML*, 2007, pp. 129–136.

[44] Y. Shi, M. Larson, and A. Hanjalic, "List-wise learning to rank with matrix factorization for collaborative filtering," in *Proc. of the 4th ACM Conf. on Recommender Systems, RecSys*, 2010, pp. 269–272.

[45] P. G. Campos, F. Díez, and I. Cantador, "Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols," *User Model. User-Adapt. Interact.*, vol. 24, no. 1-2, pp. 67–119, 2014.

[46] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, 2008, pp. 426–434.

[47] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proc. of the Third Int. Conf. on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, 2010, pp. 81–90.

[48] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proc. of the 19th Int. Conf. on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, 2010, pp. 811–820.

[49] X. He and T. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. of the 40th Int. ACM SIGIR Conf. on Research and Development in IR, Shinjuku, Tokyo, Japan*.

[50] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," in *Proc. of the 26th Int. Joint Conf. on Artificial Intelligence, IJCAI 2017, Melbourne, Australia*.

[51] Y. Zhu, J. Lin, S. He, B. Wang, Z. Guan, H. Liu, and D. Cai, "Addressing the item cold-start problem by attribute-driven active learning," *IEEE Transactions on Knowledge and Data Engineering TKDE*, vol. 32, no. 4, pp. 631–644, 2020.

[52] S. M. McNee, J. Riedl, and J. A. Konstan, "Being accurate is not enough: how accuracy metrics have hurt recommender systems," in *Extended Abstracts Proc. of the 2006 Conf. on Human Factors in Computing Systems CHI*, 2006, pp. 1097–1101.

[53] Y. Koren and J. Sill, "Ordrec: an ordinal model for predicting personalized item rating distributions," in *Proc. of the 2011 ACM Conf. on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, 2011, pp. 117–124.

[54] I. Bayer, "fastfm: A library for factorization machines," *Journal of Machine Learning Research*, vol. 17, pp. 184:1–184:5, 2016.

[55] Y. Zhang, "Explainable recommendation: Theory and applications," *CoRR*, vol. abs/1708.06409, 2017.

[56] Y. Zhang, M. Zhang, Y. Zhang, G. Lai, Y. Liu, H. Zhang, and S. Ma, "Daily-aware personalized recommendation based on feature-level time series analysis," in *Proc. of the 24th Int. Conf. on World Wide Web, WWW 2015, Florence, Italy*.

[57] B. Abdollahi and O. Nasraoui, "Explainable matrix factorization for collaborative filtering," in *Proc. of the 25th Int. Conf. on World Wide Web, WWW, Companion Volume*, 2016, pp. 5–6.

[58] A. Rana and D. Bridge, "Explanation chains: Recommendations by explanation," in *Proc. of the Poster Track of the 11th ACM Conf. on Recommender Systems (RecSys)*, 2017.

[59] M. Vlachos, C. Duenner, R. Heckel, V. G. Vassiliadis, T. Parnell, and K. Atasu, "Addressing interpretability and cold-start in matrix factorization for recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

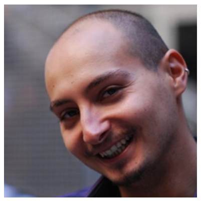
[60] L. Hu, S. Jian, L. Cao, and Q. Chen, "Interpretable recommendation via attraction modeling: Learning multilevel attractiveness over multimodal movie contents," in *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence, IJCAI*, 2018, pp. 3400–3406.

[61] G. Peake and J. Wang, "Explanation mining: Post hoc interpretability of latent factor models for recommendation systems," in *Proc. of the 24th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, 2018, pp. 2060–2069.

[62] J. Vig, S. Sen, and J. Riedl, "Tagsplanations: explaining recommendations using tags," in *Proc. of the 14th Int. Conf. on Intelligent User Interfaces, IUI*, 2009, pp. 47–56.

[63] Z. Kang, C. Peng, and Q. Cheng, "Top-n recommender system via matrix completion," in *Proc. of the 13th AAAI Conf. on Artificial Intelligence*. AAAI Press, 2016, pp. 179–185.

[64] ——, "Kernel-driven similarity learning," *Neurocomputing*, vol. 267, pp. 210–219, 2017.



Vito Walter Anelli is a PhD student in Electrical and Information Engineering at Polytechnic University of Bari, Italy. His current research interests fall in the areas of Recommender Systems, Knowledge representation, and User Modeling. He has published his works in national and international workshops and conferences as well as in international journals, and served as reviewer in venues of the aforementioned fields, including prestigious journals such as Knowledge-Based Systems, IEEE Transactions on Knowledge and Data Engineering.

Tommaso Di Noia is Full Professor of Information and Data Management with the Polytechnic University of Bari, Italy. Over the years, he has authored several papers in international journals and prominent conferences in his research areas. His current research interests include Recommender systems and machine learning, knowledge graphs and Semantic Web technologies, personalized information access, and preference representation and reasoning.

Azzurra Ragone is an independent researcher. Her current research interests fall in the areas of Fairness in Machine Learning, Recommender Systems, and Service Design. She worked as a researcher for several years at the Polytechnic of Bari, the University of Trento, the University of Michigan (USA) and the University of Milan Bicocca. She has published her research works in more than 80 national and international workshops and conferences as well as in international journals.

Eugenio Di Sciascio received the M.S. degree (Hons.) from the University of Bari, Bari, Italy, and the Ph.D. degree from the Polytechnic University of Bari, Bari. He has been an Assistant Professor with the University of Lecce and an Associate Professor with the Polytechnic University of Bari. He is currently a Full Professor of information technology engineering with the Polytechnic University of Bari, where he leads the research group of the SisInf Lab, the Information Systems Laboratory. His current research interests include multimedia information retrieval, knowledge representation and e-commerce. He is involved in several national and European research projects related to his research interests. Prof. Di Sciascio was a recipient of the Best Paper Award of the ICEC-2004, IEEE CEC-EEE-2006, ICEC-2007, SEMAPRO-2010, and ICWE-2010 conferences for his co-authored papers.

Joseph Trotta received the M.S. degree in Computer Engineering from Polytechnic University of Bari. He is currently a Research Assistant at Polytechnic University of Bari. His current research interests include Recommender Systems, and User Modeling. He has published his works in national and international workshops and conferences.