January 2020

Knowledge-aware Interpretable Recommender Systems

Vito Walter ANELLI [a], Vito BELLINI [a] , Tommaso DI NOIA [a]and Eugenio DI SCIASCIO [a]
[a] *Politecnico di Bari, Bari, Italy*

Abstract. Recommender systems are everywhere, from e-commerce to streaming platforms. They help users lost in the maze of available information, items and services to find their way. Among them, over the years, approaches based on machine learning techniques have shown particularly good performance for top-N recommendations engines. Unfortunately, they mostly behave as black-boxes and, even when they embed some form of description about the items to recommend, after the training phase they move such descriptions in a latent space thus loosing the actual explicit semantics of recommended items. As a consequence, the system designers struggle at providing satisfying explanations to the recommendation list provided to the end user. In this chapter, we describe two approaches to recommendation which make use of the semantics encoded in a knowledge graph to train interpretable models which keep the original semantics of the items description thus providing a powerful tool to automatically compute explainable results. The two methods relies on two completely different machine learning algorithms, namely, factorization machines and autoencoder neural networks. We also show how to measure the interpretability of the model through the introduction of two metrics: semantic accuracy and robustness.

Keywords. Recommender Systems, Knowledge Graphs, DBpedia, Interpretablity, Explanation

1. Introduction

Nowadays, it is well recognized that 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 and, consequently, this makes non-trivial the interpretation of a recommendation process.

On the other side, transparency and interpretability of predictive models are gaining momentum since they have been identified 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 [1,2,3] and users are more willing to extensively exploit the predictions [4,5]. Indeed, transparency increases their trust [6] (also exploiting specific semantic structures [7]), and satisfaction in using the system. Among interpretable models for Recommender Systems (RS), we may distinguish between those based on Content-based

(CB) approaches and those based on Collaborative filtering (CF) ones. CB algorithms provide recommendations by exploiting the available content and matching it with a user profile [8,9]. The use of content features makes easier to develop an interpretable model 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*" [10]. 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 [11] 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 [12] thus making the final result not directly interpretable. In the last years, many approaches have been proposed that take advantage of side information in recommendation algorithms to enhance the performance of latent factor models. Side information can refer to items as well as users [13] and can be either structured [14,15] or semi-structured [16,17,18]. Interestingly, in [10] 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 chapter we show how to properly inject semantics-aware data coming from an RDF knowledge graph[1] in model-based recommendation algorithms in order to go beyond a black-box approach and transform them in interpretable models. In the next section we will focus on factorization machines [19] while in Section 3 we will analyze a deep learning model based on autoencoder neural networks [20]. During the evaluation of the two approaches in terms of precision and interpretability of the final model, we refer to different dimensions of an RDF dataset. In particular, we know that an RDF knowledge graph encodes different types of information:

- **Factual.** This refers to statements that describe attributes of an entity such as *Star Wars: Episode IV – A New Hope was directed by the George Lucas* or *Jumanji is located in British Columbia*;
- **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.

We will show how the selection of these three classes of knowledge —and their combinations— may affect the final performance of a knowledge-aware recommender system.

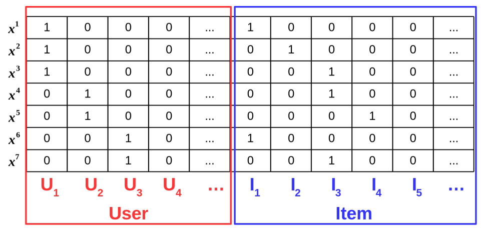
The chapter is structured as follows: in the next section we introduce and discuss a model based on knowledge-aware factorization machines for recommender systems

[1] We mainly refer to DBpedia.

while in Section 3 we describe a deep learnig model based on autoencoders that embeds a knowledge-graph in its structure. Section 4 is devoted to a brief description of related literature. Conclusion and future work close the chapter.

2. Knowledge-aware Hybrid Factorization Machines for Top-N Recommendation

As shown in [21], factorization models have proven their strength in recommendation scenarios. Main advantages of factorization models are their effectiveness in dealing with very sparse settings and their prediction accuracy thanks to the subtle modeling of user-item interactions. Several factorization models have been proposed in the literature and, among them, factorization machines generalize most of the factorization models unifying this class of algorithms. Here we report the definition related to a second order features-interaction factorization model 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 [22], multi-criteria [23], and even relations between contexts [24]. We build for each user $u \in U$ and each item $i \in I$ a binary vector $\mathbf{x}^{\mathbf{ui}} \in \mathbb{R}^{1 \times n}$, with $n = |U| + |I|$, representing the interaction between u and i in the original user-item rating matrix. In this modeling, $\mathbf{x}^{\mathbf{ui}}$ contains only two 1 values corresponding to u and i while all the other values are set to 0 (see Fig. 1). $\mathbf{X} \in \mathbb{R}^{n \times m}$ is a matrix that contains as rows all the possible $\mathbf{x}^{\mathbf{ui}}$ 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	...

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

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

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

where the parameters the model learns 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 latent factors number is denoted as k and its value is usually chosen at design time when implementing the FM.

If we want to make a factorization machine interpretable, we need a way to give an explicit semantics to latent factors. In this respect, knowledge graphs may result very useful since they provide information about several and different domains [25]. 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. Following [26], we bind the set of features retrieved from a knowledge graph to the latent factors of a Factorization Machine model. Since we are tackling a top-N recommendation problem, we use a Bayesian Personalized Ranking (BPR) criterion [27] to train our model. In [28], authors originally proposed to encode a Linked Data knowledge graph within a vector space model with the aim of developing a CB recommender system. Given $I = \{i_1, i_2, \dots, i_N\}$ as the set of items in a catalog and their associated triples $\langle i, \rho, \omega \rangle$ in a knowledge graph $\mathtt{KG}$, we build the set of all possible features as $F = \{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathtt{KG} \text{ with } i \in I\}$. We can then 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 calculated as the normalized TF-IDF value for $\langle \rho, \omega \rangle$ as follows:

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

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

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

Analogously, when we have a set U of users, we may represent them using the features that describe 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 $\mathbf{u} = [v_{(u,1)}, \dots, v_{(u,f)} \cdots, v_{(u,|F|)}]$, 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 $j \in [1 \dots |U|]$, and $\mathbf{i}_p$, with $p \in [1 \dots |I|]$, 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 Equation (1) we may see that, for each $\mathbf{x}$, the term $\sum_{j=1}^{n} \sum_{p=j+1}^{n} x_j \cdot x_{j'} \cdot \sum_{f=1}^{k} v_{(j,f)} \cdot v_{(p,f)}$ 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 $\mathtt{KG}$ into a factorization machine, we set $k = |F|$ in Equation (1). In other words, we impose the number of latent factors equal to the number of features that describe all the items in the catalog. We want to stress here that our aim is not to

	~ 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	
$\boldsymbol{v}_1$	**0**	0.88	0.81	0.7	0	0.60	0.53	...
$\boldsymbol{v}_2$	1.3	1.12	0.91	0.84	0.65	0.59	0.58	...
$\boldsymbol{v}_3$	0.5	0	0.71	0	0.28	0.35	0	...
$\boldsymbol{v}_4$	0	0	0.31	0	0	0	0.6	...
$\boldsymbol{v}_5$	0	0	0	0	0.18	0	0	...
$\boldsymbol{v}_6$	0	0.12	0.22	0	0	0	0	...
$\boldsymbol{v}_7$	1.23	1.03	0.89	0.85	0.56	0.3	0.61	...

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

represent each feature through a latent vector, but to map each factor with an explicit feature in order to obtain latent vectors that are made of explicit semantic features. To this aim, 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 rationale behind this idea is once the model has been trained, the resulting matrix $\hat{\mathbf{V}}$ still refers to the original features but it 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. At this point, if we extract the items vectors $\mathbf{v}_j$ from the matrix $\hat{\mathbf{V}}$, we may leverage optimal values of item vectors to implement an item-kNN recommendation approach. Similarities between each pair of items i and j are measured 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 N^i be the set of neighbors for the item i which contains the most similar items to i according to the selected similarity measure. We may choose i such that $i \notin I^u$ and a user u ad then predict the score assigned by u to i as:

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

From now on we will refer to this knowledge-aware factorization machine approach as `kaHFM`. Factorization machines can be easily trained to reduce the prediction error by using different optimization algorithms such as gradient descent methods, alternating least-squares (ALS) and MCMC. Since we formulated our problem as a *top-N* recommendation task, following [29], we trained `kaHFM` by using Bayesian Personalized Ranking Criterion (BPR) as a learning to rank approach. 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. At the end of the training phase, we can use the optimal model parameters for the item recommendation step.

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[2] 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"`.

2.1. Semantic Accuracy and Generative Robustness

`kaHFM` allows us to keep the meaning of the "latent" factors computed via a factorization machine, which turns out to be exploitable in order to interpret the recommended results. We propose an automated offline evaluation procedure to measure the *Semantic Accuracy* with the aim of verifying that `kaHFM` preserves the semantics of the features in $\mathbf{V}$ after the training phase. Furthermore, we define as *Robustness* the ability to assign a higher value to important features after one or more feature removal.

Semantic Accuracy. The rationale behind Semantic Accuracy is to evaluate, given an item i, how well `kaHFM` is able to correctly predict 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 $F^i = \{f_1^i, \dots, f_m^i, \dots f_M^i\}$, 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 $f \notin F^i$. Regarding the feature set M that initially describes i, we see how many features 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.

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

It not unusual to deal with scenarios for which we may have $|F| \gg M$. Therefore, we might also consider to measure the accuracy for different sizes of the top list. Since we

[2] `http://research.yahoo.com/Academic_Relations`

can describe items by using different number of features, the size of the top list might be a function of the original size of the features set that describes am item. In this direction, we measured $\texttt{SA@}nM$ with $n \in \{1,2,3,4,5,\dots\}$ and then we evaluated the number of features in F^i which are available in the top-$n \cdot M$ elements of $\mathbf{v}_i$.

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

Robustness. Although `SA@nM` may result very useful to check whether `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` provides a real mapping between latent factors and explicit features for i which in turn are the most important to describe the item? To provide a way of measuring "meaningfulness" for a given feature, we hypothesize that a particular feature $\langle \rho, \omega \rangle$ is useful for representing an item i, but the corresponding triple $\langle i, \rho, \omega \rangle$ is not described in the knowledge graph. In case `kaHFM` were effective in generating weights for unknown features, it should determine the importance of that feature and change its value to make it join the Top-K features in $\mathbf{v}_i$. Following this investigation, the rationale behind robustness is then to "forget" a triple involving i and check whether `kaHFM` is able to generate it back. We performed the following steps in order to implement this process:

- we train `kaHFM` to obtain 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 retrain the model initializing $f^i_{MAX} = 0$ and we compute v'_i.

After the above steps, if $f^i_{MAX} \in top(v'_i, M)$ then we can say that `kaHFM` shows a high robustness in identifying important features. In a catalog I, we want 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.

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

Similarly to $\texttt{SA@}nM$, we may define `1-Rob@nM`.

Experimental Evaluation. In this section, we provide details for all the three experiments we performed. Particularly, we want to test if:

- `kaHFM`'s recommendations are accurate;
- `kaHFM` preserves the semantics of original features;
- `kaHFM` promotes significant features.

Datasets. We evaluated `kaHFM`'s performance on two well-known datasets widely adopted in the recommender systems field on movie domain. `Yahoo!Movies` (Yahoo! Webscope dataset ydata-ymovies-user-movie-ratings-content-v1_0)[3] contains movies ratings on a [1..5] scale generated on Yahoo! Movies up to November 2003. It provides

[3]`http://research.yahoo.com/Academic_Relations`

content, demographic and mappings to `MovieLens` and `EachMovie` datasets. `Facebook Movies` dataset has been released for the Linked Open Data challenge co-located with ESWC 2015[4]. It contains implicit feedback only and it provides for each item a mapping to DBpedia. To map items in `Yahoo!Movies` and other datasets, we extracted all the updated items-features mappings and we made them publicly available[5]. Datasets statistics are shown in Table 2.

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%

Table 2. Datasets statistics.

Experimental Setting. In order to evaluate the proposed method with respect to other algorithms, we followed the ”All Unrated Items” [30] protocol. We split the datasets using the Hold-Out 80-20, so we retained for every user the 80% of their ratings in the training set and moved the remaining 20% in the test set. Furthermore, a temporal split has been performed [31,32] whenever timestamps associated to every transaction is available.
Extraction. Thanks to the publicly available mappings, we have the `DBpedia` link for each of the items in datasets listed in Table 2. Exploiting this mapping, we fetched all the $\langle \rho, \omega \rangle$ pairs associated to items. Some features had been excluded, in particular we excluded features based on the following predicates: `owl:sameAs`, `dbo:thumbnail`, `foaf:depiction`, `prov:wasDerivedFrom`, `foaf:isPrimaryTopicOf`.
Selection. We used three different settings to perform experiments because we want to analyze the impact of the different kind of features. Features have been chosen depending on their presence in all the different domains and because of their factual, categorical or ontological meaning.
Filtering. In this step we remove irrelevant features that bring trascurable value to the recommendation task, but, at the same time, pose scalability issues. We followed the approach presented in [26] for the pre-processing phase, and [33] with a unique threshold. Thresholds (corresponding to *tm* [26], and *p* [33] for missing values) and the considered features for each dataset are represented in Table 3.

		Categorical Setting		**Ontological Setting**		**Factual Setting**	
Datasets	Threshold	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

Table 3. Considered features in the different settings

2.2. Accuracy Evaluation

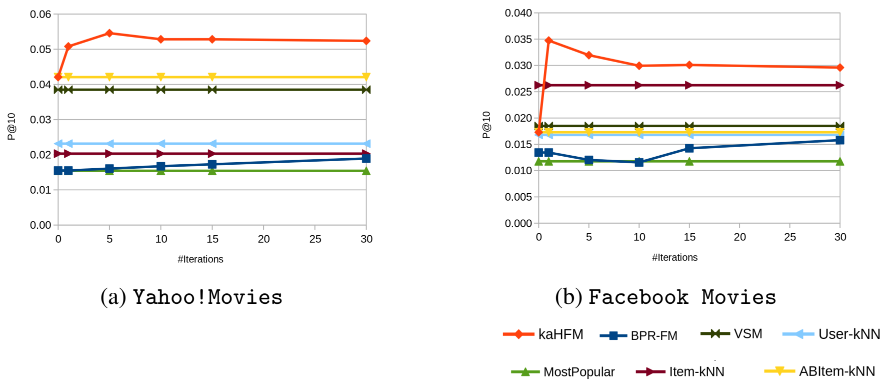
The objective of this evaluation is to verify whether the `Linked Data` injected in a controlled fashion can positively affect the training of Factorization Machines. To this aim, we do not compare `kaHFM` w.r.t. other state-of-art interpretable models but only with algorithms that are more related to our approach. We compared `kaHFM`[6] w.r.t. a canonical 2-

[4]`https://2015.eswc-conferences.org/program/semwebeval.html`
[5]`https://github.com/sisinflab/LinkedDatasets/`
[6]`https://github.com/sisinflab/HybridFactorizationMachines`

degree Factorization Machine where users and items are intended as features of the original formulation. We optimized the recommendation list ranking via BPR (BPR-FM). In order to keep the expressiveness of the model, the same number of hidden factors have been used (see the "Selected" column in Table 3). Furthermore, we used items similarity [34] in the last step of our approach (see Equation (4)), for this reason, we compared `kaHFM` against an *Attribute Based Item-kNN* (ABItem-kNN) algorithm. In ABItem-kNN each item is represented by a vector of weights which are calculated through a TF-IDF model. In this model, the attributes are computed via Equation (2). Nevertheless, we also compared `kaHFM` against a pure Item-kNN, which is an item-based implementation of the k-nearest neighbors algorithm. It finds the k-nearest item neighbors based on Pearson Correlation. 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. Following [29], we adopted a sampler "without replacement" in order to sample the triples as suggested by authors. We also compared `kaHFM` against the corresponding User-based nearest neighbor scheme, and Most-Popular, a simple baseline that shows high performance in specific scenarios [35]. Since our method relies on knowledge-graphs, we considered mandatory a comparison against a pure content-based baselines such as a Vector Space Model (*VSM*) [28]. As evaluation metrics for our approach, we measured accuracy through Precision@N (*Prec@N*) and Normalized Discounted Cumulative Gain (*nDCG@N*) [36]. The evaluation has been performed considering Top-10 [35] recommendations for all the datasets. When a rating score was available (`Yahoo!Movies`), a *Threshold-based relevant items* condition [37,38] was adopted with a relevance threshold of 4 over 5 stars in order to take into account only relevant items. In Fig. 3 results of our experiments regarding accuracy are showed. 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. When Categorical and Ontological information are used, our method is the most accurate, as evidenced in `Yahoo!Movies` experiments. Very interestingly, even though `Yahoo!Movies` mapping is affected by a strong popularity bias, only the Factual setting leads our approach to be less effective then ABItem-kNN. In `Facebook Movies` we see a very consistent improvement of accuracy as it almost doubles up the ABItem-kNN algorithm values. We compared `kaHFM` against ABItem-kNN to verify whether the collaborative trained features might lead to better similarity values. We believe that this hypothesis is confirmed since in former experiments `kaHFM` beats ABItem-kNN in almost all settings. It turns out that collaborative trained features achieve better results in terms of accuracy. Moreover, we want to assess if the initialization of latent factors through a knowledge-graph based approach may improve the performance of Factorization Machines. `kaHFM` always beats BPR-FM, we suppose that this happens because the random initialization takes a while to converge towards an optimal solution. Finally, we want to check if collaborative trained features lead to better results in terms of accuracy w.r.t. a purely informativeness-based knowledge-graph-aware version of Vector Space Model. Our experiments confirm that `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. In the interest of brevity, we report here[7] only the plots for Categorical setting (Fig. 3) It is worth to mention that for all the cases we considered,

[7]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*	0.0421*	0.1174*
BPR-FM	0.0158*	0.0189*	0.0344*
MostPopular	0.0118*	0.0154*	0.0271*
ItemKnn	0.0262*	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	0.0427*	0.1223*
BPR-FM	0.0155*	0.0199*	0.0356*
MostPopular	0.0118*	0.0154*	0.0271*
ItemKnn	0.0263*	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	0.0619	**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**	0.1725
kaHFM	0.0240	0.0564	0.1434



(a) `Yahoo!Movies` (b) `Facebook Movies`

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

we show the best performance in one of these iterations. Nevertheless, in all the datasets we can notice a positive influence of the initialization of the feature vectors, with very similar performances to the ones depicted in [29].

2.3. Semantic Accuracy

Previous experiments showed that `kaHFM` is effective in terms of accuracy in recommendation scenarios. As a practical matter, we proved that:

- initializing latent factors with content-based weights leads to better performance with `kaHFM`;

- the obtained fine-tuned items vectors are better than the original ones in a *top-N* item recommendation scenario;
- results are dependant on the features we extract from the Knowledge Graph.

However, even though the proposed method outperforms the baselines, we still do not know if the original semantics of the features is preserved in the new space. In Section 2.1 we described `Semantics Accuracy` ($SA@nM$) as a metric to automatically assess if the importance calculated 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. Regarding the Categorical setting, results are showed in Table 4.

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

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

2.4. Generative Robustness

From previous experiments, it follows that features calculated through `kaHFM` keep their original semantics if already present in the item description. In section 2.1, we described a procedure to measure the capability of `kaHFM` to compute meaningful features. Here, we calculate `1-Rob@nM` for the two adopted datasets. Results are showed in Table 5. In this case, we concentrate on the CS setting which provides the best results in

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

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

terms of accuracy. To achieve a better understanding of the results, we start focusing on `Yahoo!Movies` for which apparently `kaHFM` has bad performance. In Table 4 we show 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". We want now assess if `kaHFM` is able to guess the removed feature in these "slots". Results of this experiment are showed in Table 5; as we can see, 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.

3. Knowledge-aware Autoencoder

In the last decade, Deep Learning (DL) has gained momentum as a disruptive technology. Several successes have largely proved it and, recently, researches have adopted DL for tackling the recommendation problem [39]. In this direction, DL recommendation techniques have shown to outperform state-of-the-art models regarding the accuracy of recommendations. Among the different DL techniques, Autoencoders are a Deep Neural Network (DNN) configuration often adopted for rating prediction task in a recommendation scenario. This configuration represents data in a low dimensional space preserving the important information for the recommendation process. Here, we describe SEMAUTO. An Autoencoder configuration that mimics the semantics-aware topology of a Knowledge Graph (KG). SEMAUTO leverages the knowledge coming from a KG and combines it with the performance of an Autoencoder.

Autoencoders. Autoencoders are a special configuration of Artificial Neural Nets (ANNs). It is an unsupervised learning algorithm that aims to replicate the input into the output layer. It takes advantage of a latent representation of data to reproduce the fed input. More formally, we can say that Autoencoder training corresponds to learning an approximate identity function, producing $\hat{x}$ similar to x. A typical Autoencoder is graphically represented in Figure 4.

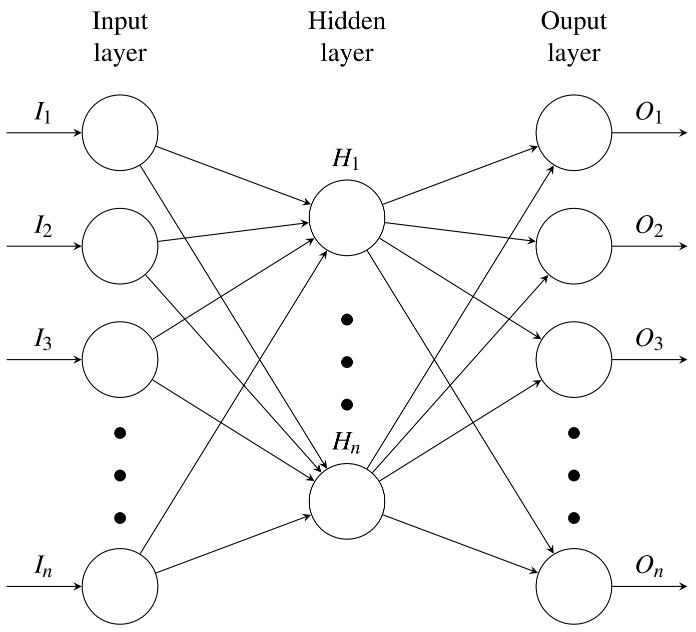


Figure 4. Architecture of an Autoencoder Neural Network.

Reproducing the input on the output layer, and hence learning an approximation of the identity function, is intuitive. However, the topology of the network we choose has many side-effects on the resulting function. As an example, if we limit the number of neurons in the hidden layer, the resulting function will seek correlations and combinations among input neurons to rebuild it. If we suppose to set 20 neurons in the hidden layer, while the input is composed of 100 neurons (i.e., 10×10 pixel image), the network can only use those 20 neurons to reconstruct the input. Thus, the network will learn a compressed representation of the input. A simple autoencoder designed to learn a low-dimensional representation is shown in Figure 4. Most Autoencoder configurations include only a single hidden layer. However, with the spread of DL, we are witnessing the proposal of many configurations that include several hidden layers. These Deep Autoencoders have more capacity and expressiveness, and hence they can perform a better approximation of the identity function [40].

Unfortunately, despite the excellent performance of Autoencoders, they behave like black boxes. Since they make use of latent representations, we cannot understand the reason why a compressed representation gives an output.

Semantics-Aware Autoencoder. The main intuition behind SEMAUTO is reflecting the KG connections between entities in a neural network. In detail, the same triples that describe items in the original KG activate the connections from layer `i` to layer `i+1`, as depicted in Figure 5.

In this pictorial representation, we have represented only categorical information into the autoencoder and we have omitted factual. These two sets of information in DBpedia have some peculiarities:

- the amount of categorical information is higher than the factual one. Regarding movies, the overall number of categories exceeds the number of entities reached by factual information;
- categorical information is more equally distributed over the items. In particular, categorical information can connect more items via the same category.

The idea is that a positive vote for *Cloud_Atlas* could be a signal of preference toward the corresponding category *Post-apocalyptic_films*.

We have leveraged the DBpedia categorical information to define the topology of the Neural Network. In particular, we have built a separate Autoencoder for each user on the platform, exploiting the items' descriptions available in DBpedia. This choice let us represent users with a different number of neurons, based on their interactions with the platform.

Let us define n as the number of items rated by user u. Let $C_i = \{c_{i1}, c_{i2}, \ldots, c_{im}\}$ be the set of m nodes (representing, e.g., categorical information) associated in the KG to the item i. Finally, let $F^u = \bigcup_{i=1}^{n} C_i$ be the set of features mapped into the hidden layer for the u, with $|F^u|$ being the overall number of units in the hidden layer.

It is worth to notice that the resulting network is not fully connected since it reflects the connections in the KG. Additionally, bias units are unnecessary, because they have no equivalent in the KG.

Moreover, the hidden layer units represent the categorical knowledge in KG. Once we start training the model, we take advantage of backpropagation algorithms to update the weights minimizing the prediction error for the user-item rating.

A possible interpretation for these weights is the importance given to them by the user for generating the rating score.

3.1. User profiles

At the end of the training phase, a new user-features latent representation is made available to the system.

Thanks to the categorical information encoded in the hidden layer, the method learns to predict user-item ratings employing the semantics of items.

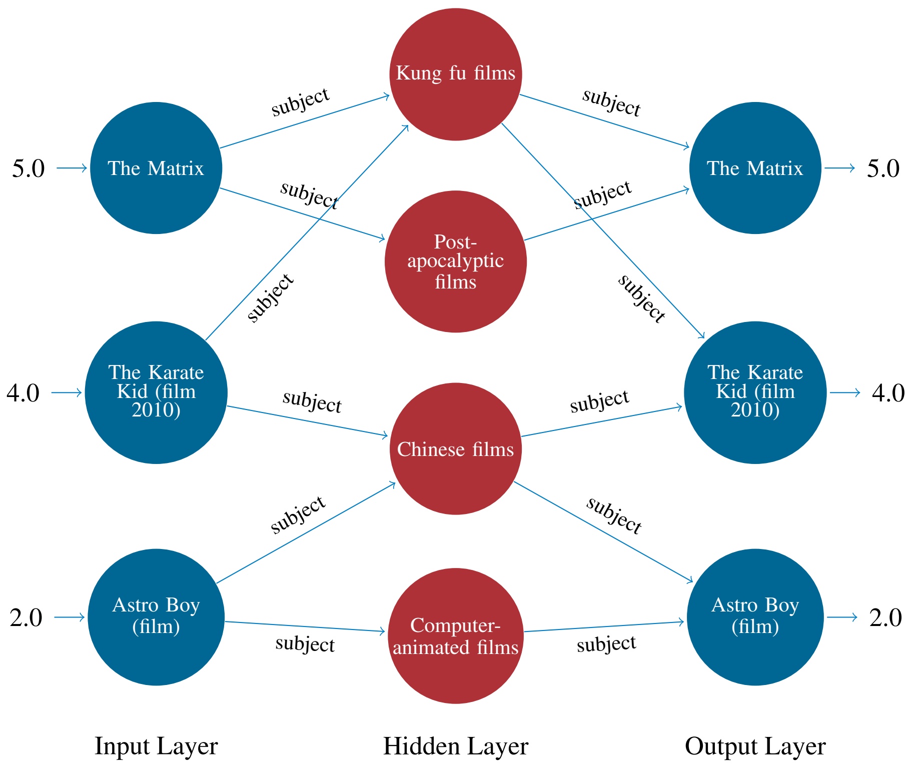


Figure 5. Architecture of a semantic autoencoder.

In the present work, we have adopted, as an activation function, the well known sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ since we have normalized the input to be in the range $[0,1]$. We have trained the autoencoders for 10,000 epochs using a learning rate of $r = 0.03$. The weights are initialized to random values as suggested by Xavier et al. in [41].

The resulting autoencoders contain much useful user information. In detail, we can build the user profile extracting the values of the weights. Since each node in the hidden layer corresponds to an explicit feature, we may assume that those values correspond to the user-feature importance in the user profile.

For a user u, the importance of a feature c is the summation of the weights $w_k^u(c)$ associated to the edges entering a node in the hidden layer. That unit represents the KG category c specialized on user u since we have trained the autoencoder with her ratings. More formally, we have:

$$\omega^u(c) = \sum_{k=1}^{|In(c)|} w_k^u(c)$$

where $In(c)$ is the set of the connections entering the unit that represents the feature c. We recall that, since the autoencoder is not fully connected, $|In(c)|$ is highly dependant on the connections between the items and the category c in KG.

Now we can leverage those weights to build a user profile. We model it as a weighted vector, in which each position is associated with a categorical feature.

Let F^u be the overall set of categories connected to the items rated by u and let $F = \bigcup_{u \in U} F^u$ be the overall set of features considering the entire population. For each user $u \in U$ and for each feature $c \in F$ we have the user profile defined as:

$$P(u) = \{\langle c, \omega \rangle \mid \omega = \omega^u(c) \text{ if } c \in F^u\}$$

However, since users have rated a different number of items, we will witness different user profiles' dimensions. Also, as users rate only a small fraction of the catalog, the majority of the features for each user remains without a value. To compute those values we have taken advantage of unsupervised learning motivated by *word2vec* [42]. *word2vec* is designed to compute word embeddings exploiting the distribution of words in sentences of a corpus. It projects semantically similar words to close points in a new atent space. Even though two words never co-occur in the same sentance, we are still able to compute their similarity. More formally, let $[x_1, \ldots, x_n]$ be a word sequence in a window. *word2vec* will estimate the probability that a word is the element of the sequence $p(x' \mid [x_1, \ldots, x_n])$. In the current setting, instead of sequences of words, we are dealing with sequences of categories in the user profile. *word2vec* will, therefore, estimate the weight for the missing features $c' \notin F^u$ to be part of the sequence. To estimate the scores, we first need to generate the corpus with all the user profiles $P(u)$. Exploiting ω we can order the features within the profiles. Second, we need to associate a consistent value for the ordered elements $c \in F^u$ on all $u \in U$ (with different profiles' sizes). Formally, we transform the original set $P(u)$ to a sequence of categories $s(u)$. In detail, we map each $\langle c, \omega \rangle \in P(u)$ to $\langle c, norm(\omega) \rangle$ using *norm* as a mapping function:

$$norm : [0,1] \mapsto \{0.1, 0.2, 0.3, \ldots, 1\}$$

that linearly maps[8] a value in the interval $[0,1]$ to a real value in the set $\{0.1, 0.2, 0.3, \ldots, 1\}$. The new pairs form the set

$$P^{norm}(u) = \{\langle c, norm(\omega) \rangle \mid \langle c, \omega \rangle \in P(u)\}$$

For each normalized user profile set $P^{norm}(u)$ we build the corresponding sequence

$$s(u) = [\ldots, \langle c_i, norm(\omega_i^u) \rangle, \ldots \langle c_j, norm(\omega_j^u) \rangle, \ldots]$$

with $\omega_i^u \geq \omega_j^u$.

The set $S = \{s(u) \mid u \in U\}$ is the corpus we can feed the *word2vec* algorithm with. Then, *word2vec* will be able to discover features patterns following to categories distribution over all users. We take advantage of user's sequence of features $s(u)$ to estimate the probability of $\langle c', norm(\omega') \rangle \in \bigcup_{v \in U} P^{norm}(v) - P^{norm}(u)$ to belong to the context. This probability is a measure of importance of the category for u. Consequently, we estimate $p(\langle c', norm(\omega') \rangle \mid s(u))$.

It is important to highlight that $c' \in F^u$ can appear in several pairs having c' as first category in $\bigcup_{v \in U} P^{norm}(v) - P^{norm}(u)$. As an example, the category `dbc:Kung_fu_films` may appear in $\langle$`dbc : Kung_fu_films`$, 0.2\rangle$ and $\langle$`dbc : Kung_fu_films`$, 0.5\rangle$, along with

[8]In this work we adopt a standard `minmax` normalization.

the related probabilities $p(\langle \texttt{dbc:Kung\_fu\_films}, 0.2\rangle \mid s(u))$, $p(\langle \texttt{dbc:Kung\_fu\_films}, 0.5\rangle \mid s(u))$. In these situations, we choose the category pair with the highest probability since we require to add the category `dbc:Kung_fu_films` and the associated weight only once in the user profile. After this processing, we obtain a new user profile:

$$\hat{P}(u) = P(u) \cup \{\langle c, \omega\rangle \mid \underset{\omega\in\{0.1,\dots,1\}}{\operatorname{argmax}} \; p(\langle c, \omega\rangle \mid s(u)) \text{ and } \langle c, \omega\rangle \notin P^{norm}(u)\}$$

It is worth noticing that this new user profile $\hat{P}(u)$ acknowledges collaborative information, while the initial $P(u)$ makes use of only content knowledge. This is since $\hat{P}(u)$ is built using the set S that represents information coming from all the users.

3.2. Computing Recommendations

The choice of representing user profiles as weighted vectors is particularly useful to compute recommendations adopting a user-based k Nearest Neighbors (kNN) approach. As in Vector Space Model, the user profile represents the projection of the user in a new multi-dimensional space, where the similarity between the users u and v can be easily computed exploiting Cosine Vector Similarity. Then, the k most similar users (for user u) are used to estimate the rating r for the item i as a weighted average of the ratings on i among the neighbors:

$$r(u,i) = \frac{\sum_{j=1}^{k} sim(u, v_j) \cdot r(v_j, i)}{\sum_{j=1}^{k} sim(u, v_j)} \tag{5}$$

where $r(v_j, i)$ is the rating assigned to i by the user v_j. We are now able to provide top-N recommendations for each user ordering the ratings computed with Equation (5).

3.3. Experiments

We have tested SEMAUTO using three well-known datasets. In detail, the first part of this section is devoted to introducing these datasets. Later, we introduce the adopted evaluation protocol along with the evaluation metrics chosen. A detailed discussion of the experimental results closes the section. For the sake of reproducibility, we have made available a public implementation of the method[9].

Dataset. We have evaluated the performance of the competing methods considering three well-known datasets belonging to three different domains. The statistics of the datasets are depicted in Table 6.

	#users	#items	#ratings	sparsity
MovieLens 20M	138,493	26,744	20,000,263	99.46%
Amazon Digital Music	478,235	266,414	836,006	99.99%
LibraryThing	7,279	37,232	626,000	99.77%

Table 6. Datasets

[9] `https://github.com/sisinflab/SEMAUTO-2.0`

For MovieLens 20M[10] we have considered a mapping containing 22,959 items, for Amazon Digital Music[11] we have mapped 4,077 items, and for LibraryThing[12] we have considered 9,926 items. For these experiments, only the items with a mapping to DBpedia were considered.

Evaluation protocol. As evaluation protocol, we have considered the "all unrated items" protocol [43]. In this protocol, all the items but those already rated by user *u* are considered candidate items. We have split the original data adopting a Hold-Out 80-20 strategy, in which the 80% of user ratings are retained as a training set. The remaining 20% is considered as the test set. Here, we show how we evaluated the performances of our methods in recommending items.

As evaluation metrics, we adopted some well-known accuracy and diversity metrics. As for accuracy, we have measured Precision, Recall, F-1 score, nDCG [44], while for diversity we have computed Aggregate Diversity, and Gini index as a measure of sales diversity [45].

Results Discussion. We have compared SEMAUTO against some well-known state-of-art recommendation algorithms: BPRMF, WRMF and a single-layer autoencoder for rating prediction. BPRMF [29] is a simple Matrix Factorization algorithm optimized adopting the Bayesian Personalized Ranking criterion. WRMF [46,47] is a Weighted Regularized Matrix Factorization method that makes use of regularized Least-Squares (LS), and uses a weighting matrix to differentiate the observed positive feedback from the others. BPRMF and WRMF can be enhanced to take advantage of side information, i.e., the description of items content. In detail, in this experimental evaluation, we have extracted the categorical information from the DBpedia KG and used it as side information. We have computed BPRMF and WRMF recommendation lists adopting the publicly available MyMediaLite[13] implementation. On the other side, the Autoencoder was implemented by scratch using Keras[14]. Also, we have performed the Wilcoxon Signed Rank test to assess the statistical significance of the results. The resulting *p-values* are constantly lower than 0.05.
Table 8 shows the results of the competing methods on the three aforementioned datasets. As you may notice, regarding SEMAUTO, we have considered different values for the size of the neighborhood *k*.
Concerning accuracy, it is worth noticing that SEMAUTO outperforms the considered baselines on MovieLens 20M and Amazon Digital Music. However, on the LibraryThing dataset, the results show a very similar behavior between the methods, with a slightly better performance shown by fully connected Autoencoder.
As for diversity, SEMAUTO behaves better than the competing algorithms. In detail, the results show that the recommendation lists are much more tailored to users, preserving the accuracy of recommendations. Finally, in the cases in which the accuracy results are very close to another method, our method provides differentiated lists and a much higher overall number of items.

[10]`https://grouplens.org/datasets/movielens/20m/`
[11]`http://jmcauley.ucsd.edu/data/amazon/`
[12]`https://www.librarything.com`
[13]`http://mymedialite.net`
[14]`https://keras.io`

	avg #features	std	avg #features/avg #items
Movielens 20M	1015.87	823.26	8.82
Amazon Digital Music	7.22	9.77	5.17
LibraryThing	206.88	196.64	1.96

Table 7. Summary of hidden units for mapped items only.

Table 7 shows that SEMAUTO performs better in domains with highly described items. This finding is clear if we observe the number of hidden units. This is likely due to the high expressiveness of the model, as suggested by the Universal Approximation Theorem. Consequently, SEMAUTO shows better performance on the MovieLens 20M dataset than the others. On the other side, on the LibraryThing dataset, SEMAUTO shows worse performance since we have not sufficient categories to make an expressive enough model.

4. Related Work

In the past few years, many interpretable recommendation model based on matrix factorization have been proposed. As is well known, matrix factorization approaches are not easy interprable because the meaning of their latent factors is unknown. One of the first attempts to address this problem was proposed in [16] in which the authors propose Explicit Factor Model (EFM). In their work, a matrix factorization framework takes as input both products' features and users' opinions extracted with phrase-level sentiment analysis from users' reviews. Thereafter, few improvements to EFM have been proposed to deal with temporal dynamics [48] and to use tensor factorization [18]. Specifically, in the latter the aim is to predict both user preferences on features (extracted from textual reviews) and items; in their work, the authors adopted the Bayesian Personalized Ranking (BPR) criterion [29]. Advances in MF-based recommendation models which are interpretable have been proposed lately with Explainable Matrix Factorization (EMF) [49] in which models rely on neighbors to compute explanations. In the same way, an interpretable Restricted Boltzmann Machine model has been proposed in [50]. 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) [17]. In another work, recommendations are computed by generating and ranking personalized explanations in the form of explanation chains [51]. Other methods exploits clustering techniques to provide an explanation, for instance in [52] the proposed method provides interpretable recommendations from positive examples based on the detection of co-clusters between users (clients) and items (products). In [53] 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. Differently, in [54], authors train a matrix factorization model in order to compute a set of association rules that interprets the obtained recommendations. In [55], authors prove that given the conversion probabilities for all actions of customer features, the original historical data is transformable into a new space

	k	**F1**	**Prec.**	**Recall**	**nDCG**	**Gini**	**aggrdiv**
				MOVIELENS 20M			
AUTOENCODER	–	0.21306	0.21764	0.20868	0.24950	0.01443	1587
BPRMF	–	0.14864	0.15315	0.14438	0.17106	**0.00375**	3263
BPRMF + SI	–	0.16838	0.17112	0.16572	0.19500	0.00635	3552
WRMF	–	0.19514	0.19806	0.19231	0.22768	0.00454	766
WRMF + SI	–	0.19494	0.19782	0.19214	0.22773	0.00450	759
SEMAUTO	5	0.18857	0.18551	0.19173	0.21941	0.01835	**5214**
	10	0.21268	0.21009	0.21533	0.24945	0.01305	3350
	20	0.22886	0.22684	0.23092	0.27147	0.01015	2417
	40	0.23675	0.23534	0.23818	0.28363	0.00827	1800
	50	0.23827	0.23686	0.23970	0.28605	0.00780	1653
	100	**0.23961**	**0.23832**	**0.24090**	**0.28924**	0.00662	1310
				AMAZON DIGITAL MUSIC			
AUTOENCODER	–	0.00060	0.00035	0.00200	0.00102	0.33867	**3559**
BPRMF	–	0.01010	0.00565	0.04765	0.02073	**0.00346**	539
BPRMF + SI	–	0.00738	0.00413	0.03480	0.01624	0.06414	2374
WRMF	–	0.02189	0.01236	0.09567	0.05511	0.01061	103
WRMF + SI	–	0.02151	0.01216	0.09325	0.05220	0.01168	111
SEMAUTO	5	0.01514	0.00862	0.06233	0.04365	0.03407	3378
	10	0.01920	0.01091	0.07994	0.05421	0.05353	3449
	20	0.02233	0.01267	0.09385	0.06296	0.08562	3523
	40	0.02572	0.01460	0.10805	0.06980	0.14514	3549
	50	0.02618	0.01486	0.10974	0.07032	0.17192	3549
	100	**0.02835**	**0.01608**	**0.11964**	**0.07471**	0.24859	3448
				LIBRARYTHING			
AUTOENCODER	–	**0.01562**	**0.01375**	**0.01808**	**0.01758**	0.07628	2328
BPRMF	–	0.01036	0.00954	0.01134	0.01001	0.06764	3140
BPRMF + SI	–	0.01065	0.00994	0.01148	0.01041	0.10753	**4946**
WRMF	–	0.01142	0.01071	0.01223	0.01247	**0.00864**	439
WRMF + SI	–	0.01116	0.01030	0.01217	0.01258	0.00868	442
SEMAUTO	5	0.00840	0.00764	0.00931	0.00930	0.13836	4895
	10	0.01034	0.00930	0.01163	0.01139	0.07888	3558
	20	0.01152	0.01029	0.01310	0.01248	0.04586	2245
	40	0.01195	0.01073	0.01347	0.01339	0.02800	1498
	50	0.01229	0.01110	0.01378	0.01374	0.02403	1312
	100	0.01278	0.01136	0.01461	0.01503	0.01521	873

Table 8. Experimental Results

so that the computation of a set of interpretable recommendation rules allows the model to provide an explanation. The core of our model is a general Factorization Machines (FM) model [56]. 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++ [57], PITF [58], FPMC [59]. First and foremost, FMs are designed for a generic prediction task, on the contrary, others factorization models are usually deployed for specific tasks

[60]. In addition, it is a linear model and parameters can be estimated accurately even in presence of high sparsity data. At the same time, several improvements have been proposed for FMs, such as Neural Factorization Machines [61] that levarage neural networks to capture non linear structure of real-world data. Moreover, Attentional Factorization Machines use an attention network to learn the importance of feature interactions [62]. Finally, FMs have been specialized to better work as Context-Aware recommender systems [63].

5. Conclusion and Future Work

Recommender systems are, with no doubt, part of our daily life and as such they have the power to drive our decisions. Based on machine learning techniques, their are getting more and more complicated from an algorithmic point of view and they act as black-boxes while computing recommendation list. This calls for a new generation of machine learning algorithms that can be interpretable by a human user and, whenever possible, provide human-readable explanations. In this respect, knowledge graphs play a crucial role in the development of this new breed of tools thanks to the explicit semantics encoded in their well curated data. Indeed, they can be injected within a recommender system algorithm thus allowing them to provide a semantically-enriched interpretation of the results they compute. In this chapter we have presented two algorithms fed by DBpedia, `kaHFM` and SEMAUTO, based on machine learning techniques, namely factorization machines and autoencoder neural networks, which are able to compute interpretable recommendation lists. We have shown that, not only they train interpretable models but their performances are comparable with, and can also beat, state-of-the-art algorithms specifically designed to implement recommender systems. Moreover, experimental results have shown an interesting dependency of the performance of an algorithm from the kind of knowledge injected in the training process. Specifically, we have shown that results may change in the presence of categorical, factual or ontological knowledge extracted from a source knowledge graph. The concepts, ideas and motivation behind the development of `kaHFM` and SEMAUTO pave the way to the design of new interpretable algorithms for recommender systems able to automatically generate human-understandable explanations.

References

[1] R.R. Sinha and K. Swearingen, 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, April 20-25, 2002*, L.G. Terveen and D.R. Wixon, eds, ACM, 2002, pp. 830–831. doi:10.1145/506443.506619.

[2] N. Tintarev and J. Masthoff, Designing and Evaluating Explanations for Recommender Systems, in: *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira and P.B. Kantor, eds, Springer, 2011, pp. 479–510. doi:10.1007/978-0-387-85820-3_15.

[3] M. Zanker, The influence of knowledgeable explanations on users' perception of a recommender system, in: *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, P. Cunningham, N.J. Hurley, I. Guy and S.S. Anand, eds, ACM, 2012, pp. 269–272. doi:10.1145/2365952.2366011.

[4] N. Tintarev and J. Masthoff, 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*, IEEE Computer Society, 2007, pp. 801–810. doi:10.1109/ICDEW.2007.4401070.

[5] J.L. Herlocker, J.A. Konstan and J. Riedl, Explaining collaborative filtering recommendations, in: *CSCW 2000, Proceeding on the ACM 2000 Conference on Computer Supported Cooperative Work, Philadelphia, PA, USA, December 2-6, 2000*, W.A. Kellogg and S. Whittaker, eds, ACM, 2000, pp. 241–250. doi:10.1145/358916.358995.

[6] R. Falcone, A. Sapienza and C. Castelfranchi, The Relevance of Categories for Trusting Information Sources, *ACM Trans. Internet Techn.* **15**(4) (2015), 13:1–13:21. doi:10.1145/2803175.

[7] N. Drawel, H. Qu, J. Bentahar and E. Shakshuki, Specification and automatic verification of trust-based multi-agent systems, *Future Generation Computer Systems* (2018). doi:10.1016/j.future.2018.01.040.

[8] M.J. Pazzani and D. Billsus, Content-Based Recommendation Systems, in: *The Adaptive Web, Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa and W. Nejdl, eds, Lecture Notes in Computer Science, Vol. 4321, Springer, 2007, pp. 325–341. doi:10.1007/978-3-540-72079-9_10.

[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.* **18**(5) (2008), 455–496. doi:10.1007/s11257-008-9051-3.

[10] Y. Zhang and X. Chen, Explainable Recommendation: A Survey and New Perspectives, *CoRR* **abs/1804.11192** (2018). `http://arxiv.org/abs/1804.11192`.

[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*, IEEE, 2017, pp. 1–6. doi:10.1109/UIC-ATC.2017.8397411.

[12] Y. Koren, R.M. Bell and C. Volinsky, Matrix Factorization Techniques for Recommender Systems, *IEEE Computer* **42**(8) (2009), 30–37. doi:10.1109/MC.2009.263.

[13] X. Wang, X. He, F. Feng, L. Nie and T. Chua, 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, April 23-27, 2018*, P. Champin, F.L. Gandon, M. Lalmas and P.G. Ipeirotis, eds, ACM, 2018, pp. 1543–1552. doi:10.1145/3178876.3186066.

[14] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L. Huang and C. Xu, Recurrent knowledge graph embedding for effective recommendation, in: *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, S. Pera, M.D. Ekstrand, X. Amatriain and J. O'Donovan, eds, ACM, 2018, pp. 297–305. doi:10.1145/3240323.3240361.

[15] V.W. Anelli and T.D. Noia, 2nd Workshop on Knowledge-aware and Conversational Recommender Systems - KaRS, in: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, W. Zhu, D. Tao, X. Cheng, P. Cui, E.A. Rundensteiner, D. Carmel, Q. He and J.X. Yu, eds, ACM, 2019, pp. 3001–3002. doi:10.1145/3357384.3358805.

[16] 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. Conf. on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia*, S. Geva, A. Trotman, P. Bruza, C.L.A. Clarke and K. Järvelin, eds, ACM, 2014, pp. 83–92. ISBN ISBN 978-1-4503-2257-7. doi:10.1145/2600428.2609579.

[17] K. Bauman, B. Liu and A. Tuzhilin, 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, August 13 - 17, 2017*, ACM, 2017, pp. 717–725. doi:10.1145/3097983.3098170.

[18] X. Chen, Z. Qin, Y. Zhang and T. Xu, 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, July 17-21, 2016*, R. Perego, F. Sebastiani, J.A. Aslam, I. Ruthven and J. Zobel, eds, ACM, 2016, pp. 305–314. doi:10.1145/2911451.2911549.

[19] 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, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I.F. Cruz, A. Hogan, J. Song, M. Lefrançois and F. Gandon, eds, Lecture Notes in Computer Science, Vol. 11778, Springer, 2019, pp. 38–56. doi:10.1007/978-3-030-30793-6_3.

[20] V. Bellini, T.D. Noia, E.D. Sciascio and A. Schiavone, Semantics-Aware Autoencoder, *IEEE Access* **7** (2019), 166122–166137. doi:10.1109/ACCESS.2019.2953308.

[21] S. Rendle, *Context-aware ranking with factorization models*, Springer, 2011.

[22] G. Adomavicius and A. Tuzhilin, Context-Aware Recommender Systems, in: *Recommender Systems Handbook*, F. Ricci, L. Rokach and B. Shapira, eds, Springer, 2015, pp. 191–226. doi:10.1007/978-1-4899-7637-6_6.

[23] G. Adomavicius and Y. Kwon, Multi-Criteria Recommender Systems, in: *Recommender Systems Handbook*, F. Ricci, L. Rokach and B. Shapira, eds, Springer, 2015, pp. 847–880. doi:10.1007/978-1-4899-7637-6_25.

[24] Y. Zheng, B. Mobasher and R.D. Burke, Incorporating Context Correlation into Context-aware Matrix Factorization, in: *Proc. 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 Int. Joint Conf. on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 27, 2015.*, D. Jannach, J. Mengin, B. Mobasher, A. Passerini and P. Viappiani, eds, CEUR Workshop Proceedings, Vol. 1440, CEUR-WS.org, 2015. `http://ceur-ws.org/Vol-1440/Paper5.pdf`.

[25] V.W. Anelli, A. Calì, T.D. Noia, M. Palmonari and A. Ragone, Exposing Open Street Map in the Linked Data Cloud, in: *Trends in Applied Knowledge-Based Systems and Data Science - Proc. of 29th Int. Conf. on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2016, Morioka, Japan, August*, H. Fujita, M. Ali, A. Selamat, J. Sasaki and M. Kurematsu, eds, Lecture Notes in Computer Science, Vol. 9799, Springer, 2016, pp. 344–355. doi:10.1007/978-3-319-42007-3_29.

[26] T.D. 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 - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, A. Gangemi, R. Navigli, M. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai and M. Alam, eds, Lecture Notes in Computer Science, Vol. 10843, Springer, 2018, pp. 128–144. doi:10.1007/978-3-319-93417-4_9.

[27] S. Rendle, C. Freudenthaler, Z. Gantner and L. Schmidt-Thieme, BPR: Bayesian Personalized Ranking from Implicit Feedback, in: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, AUAI Press, Arlington, Virginia, United States, 2009, pp. 452–461. ISBN ISBN 978-0-9749039-5-8.

[28] T.D. 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 International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012*, V. Presutti and H.S. Pinto, eds, ACM, 2012, pp. 1–8. doi:10.1145/2362499.2362501.

[29] S. Rendle, C. Freudenthaler, Z. Gantner and L. Schmidt-Thieme, BPR: Bayesian Personalized Ranking from Implicit Feedback, in: *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, J.A. Bilmes and A.Y. Ng, eds, AUAI Press, 2009, pp. 452–461. `https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1630&proceeding_id=25`.

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

[31] A. Gunawardana and G. Shani, Evaluating Recommender Systems, in: *Recommender Systems Handbook*, F. Ricci, L. Rokach and B. Shapira, eds, Springer, 2015, pp. 265–308. doi:10.1007/978-1-4899-7637-6_8.

[32] V.W. Anelli, T.D. Noia, E.D. Sciascio, A. Ragone and J. Trotta, Local Popularity and Time in top-N Recommendation, in: *Advances in Information Retrieval - Proc. of 41st European Conf. on IR Research, ECIR 2019, Cologne, Germany, Part I*, Vol. 11437, L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff and D. Hiemstra, eds, Springer, 2019, pp. 861–868. doi:10.1007/978-3-030-15712-8_63.

[33] H. Paulheim and J. Fürnkranz, Unsupervised generation of data mining features from linked open data, in: *2nd International Conference on Web Intelligence, Mining and Semantics, WIMS '12, Craiova, Romania, June 6-8, 2012*, D.D. Burdescu, R. Akerkar and C. Badica, eds, ACM, 2012, pp. 31:1–31:12. doi:10.1145/2254129.2254168.

[34] 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 Comput-*

ing, SAC 2019, Limassol, Cyprus, C. Hung and G.A. Papadopoulos, eds, ACM, 2019, pp. 816–821. doi:10.1145/3297280.3297360.

[35] P. Cremonesi, Y. Koren and R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, X. Amatriain, M. Torrens, P. Resnick and M. Zanker, eds, ACM, 2010, pp. 39–46. doi:10.1145/1864708.1864721.

[36] 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*, T. Bogers, A. Said, P. Brusilovsky and D. Tikk, eds, ACM, 2019, pp. 447–451. doi:10.1145/3298689.3347010.

[37] 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.* **24**(1–2) (2014), 67–119. doi:10.1007/s11257-012-9136-x.

[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 2017, Bratislava, Slovakia 2017*, M. Bieliková, E. Herder, F. Cena and M.C. Desmarais, eds, ACM, 2017, pp. 270–274. doi:10.1145/3079628.3079703.

[39] P. Covington, J. Adams and E. Sargin, Deep Neural Networks for YouTube Recommendations, in: *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, S. Sen, W. Geyer, J. Freyne and P. Castells, eds, ACM, 2016, pp. 191–198. doi:10.1145/2959100.2959190.

[40] G.E. Hinton and R.R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, *Science (New York, N.Y.)* **313** (2006), 504–7. doi:10.1126/science.1127647.

[41] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, Y.W. Teh and D.M. Titterington, eds, JMLR Proceedings, Vol. 9, JMLR.org, 2010, pp. 249–256. `http://proceedings.mlr.press/v9/glorot10a.html`.

[42] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado and J. Dean, Distributed Representations of Words and Phrases and their Compositionality, in: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C.J.C. Burges, L. Bottou, Z. Ghahramani and K.Q. Weinberger, eds, 2013, pp. 3111–3119. `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality`.

[43] H. Steck, Evaluation of Recommendations: Rating-prediction and Ranking, in: *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, ACM, New York, NY, USA, 2013, pp. 213–220. ISBN ISBN 978-1-4503-2409-0.

[44] K. Järvelin and J. Kekäläinen, IR evaluation methods for retrieving highly relevant documents, *SIGIR Forum* **51**(2) (2017), 243–250. doi:10.1145/3130348.3130374.

[45] D.M. Fleder and K. Hosanagar, Recommender systems and their impact on sales diversity, in: *Proceedings 8th ACM Conference on Electronic Commerce (EC-2007), San Diego, California, USA, June 11-15, 2007*, J.K. MacKie-Mason, D.C. Parkes and P. Resnick, eds, ACM, 2007, pp. 192–199. doi:10.1145/1250910.1250939.

[46] R. Pan, Y. Zhou, B. Cao, N.N. Liu, R.M. Lukose, M. Scholz and Q. Yang, One-Class Collaborative Filtering, in: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, IEEE Computer Society, 2008, pp. 502–511. doi:10.1109/ICDM.2008.16.

[47] Y. Hu, Y. Koren and C. Volinsky, Collaborative Filtering for Implicit Feedback Datasets, in: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, IEEE Computer Society, 2008, pp. 263–272. doi:10.1109/ICDM.2008.22.

[48] 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: *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, A. Gangemi, S. Leonardi and A. Panconesi, eds, ACM, 2015, pp. 1373–1383. doi:10.1145/2736277.2741087.

[49] B. Abdollahi and O. Nasraoui, Explainable Matrix Factorization for Collaborative Filtering, in: *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April*

11-15, 2016, Companion Volume, J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks and B.Y. Zhao, eds, ACM, 2016, pp. 5–6. doi:10.1145/2872518.2889405.

[50] B. Abdollahi and O. Nasraoui, Explainable Restricted Boltzmann Machines for Collaborative Filtering, *CoRR* **abs/1606.07129** (2016). `http://arxiv.org/abs/1606.07129`.

[51] A. Rana and D. Bridge, Explanation Chains: Recommendations by Explanation, in: *Proceedings of the Poster Track of the 11th ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 28, 2017.*, D. Tikk and P. Pu, eds, CEUR Workshop Proceedings, Vol. 1905, CEUR-WS.org, 2017. `http://ceur-ws.org/Vol-1905/recsys2017_poster4.pdf`.

[52] 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).

[53] L. Hu, S. Jian, L. Cao and Q. Chen, 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, July 13-19, 2018, Stockholm, Sweden.*, J. Lang, ed., ijcai.org, 2018, pp. 3400–3406. doi:10.24963/ijcai.2018/472.

[54] G. Peake and J. Wang, 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, August 19-23, 2018*, Y. Guo and F. Farooq, eds, ACM, 2018, pp. 2060–2069. doi:10.1145/3219819.3220072.

[55] A. Dhurandhar, S. Oh and M. Petrik, Building an Interpretable Recommender via Loss-Preserving Transformation, *CoRR* **abs/1606.05819** (2016). `http://arxiv.org/abs/1606.05819`.

[56] S. Rendle, Factorization Machines, in: *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, G.I. Webb, B. Liu, C. Zhang, D. Gunopulos and X. Wu, eds, IEEE Computer Society, 2010, pp. 995–1000. doi:10.1109/ICDM.2010.127.

[57] Y. Koren, 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, August 24-27, 2008*, Y. Li, B. Liu and S. Sarawagi, eds, ACM, 2008, pp. 426–434. doi:10.1145/1401890.1401944.

[58] S. Rendle and L. Schmidt-Thieme, Pairwise interaction tensor factorization for personalized tag recommendation, in: *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, B.D. Davison, T. Suel, N. Craswell and B. Liu, eds, ACM, 2010, pp. 81–90. doi:10.1145/1718487.1718498.

[59] S. Rendle, C. Freudenthaler and L. Schmidt-Thieme, 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, April 26-30, 2010*, M. Rappa, P. Jones, J. Freire and S. Chakrabarti, eds, ACM, 2010, pp. 811–820. doi:10.1145/1772690.1772773.

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

[61] X. He and T. Chua, 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, Japan, August 7-11, 2017*, N. Kando, T. Sakai, H. Joho, H. Li, A.P. de Vries and R.W. White, eds, ACM, 2017, pp. 355–364. doi:10.1145/3077136.3080777.

[62] 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: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, C. Sierra, ed., ijcai.org, 2017, pp. 3119–3125. doi:10.24963/ijcai.2017/435.

[63] S. Rendle, Z. Gantner, C. Freudenthaler and L. Schmidt-Thieme, Fast context-aware recommendations with factorization machines, in: *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, W. Ma, J. Nie, R.A. Baeza-Yates, T. Chua and W.B. Croft, eds, ACM, 2011, pp. 635–644. doi:10.1145/2009916.2010002.