This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Semantics-Aware Autoencoder

VITO BELLINI[1], TOMMASO DI NOIA[2], EUGENIO DI SCIASCIO[3] ANGELO SCHIAVONE.[4]
[1]Polytechnic University of Bari, Italy (e-mail: vito.bellini@poliba.it)
[2]Polytechnic University of Bari, Italy (e-mail: tommaso.dinoia@poliba.it)
[3]Polytechnic University of Bari, Italy (e-mail: eugenio.disciascio@poliba.it)
[4]Polytechnic University of Bari, Italy (e-mail: angelo.schiavone@poliba.it)

ABSTRACT Recommender Systems are widely adopted in nowadays services such as e-commerce websites, multimedia streaming platforms, and many others. They help users to find what they are looking for by suggesting relevant items leveraging their past preferences. Deep Learning models are very effective in solving the recommendation problem; as a matter of fact, many deep learning architectures have been proposed over the years. Even if deep learning models outperform many state-of-the-art algorithms, the worst disadvantage is about their interpretability: explaining the reason a specific item has been recommended to a user is quite a difficult task since the model is not interpretable. Accuracy in the recommendation is no more enough since users are also expecting a useful explanation for the suggested items. Users, on the other hand, want to know why. In this paper, we present `SemAuto`, a novel approach based on an Autoencoder Neural Network that makes it possible to semantically label neurons in hidden layers, thus paving the way to the model's interpretability and consequently to the explanation of a recommendation. We tested our semantics-aware approach with respect to other state-of-the-art algorithms to prove the recommendation's accuracy. Furthermore, we performed an extensive A/B test with real users to evaluate the explanation we generate.

INDEX TERMS Autoencoder Neural Network, Cold start problem, Deep Learning, Explanation, Knowledge Graph, Recommender System.

I. INTRODUCTION

Recommender Systems (RSs) have become pervasive tools we experience in our everyday life. While browsing a catalog of items, RSs exploit users' past preferences to suggest new items they might be interested in. In a digital world where we, as users, are overwhelmed by multiple possibilities and choices, they result in a valid tool to help us finding information that fits our needs, tastes, and preferences. Many online services heavily rely on the usage of recommender systems to suggest new movies to watch, new books to read, or new songs to listen to.

Over the years, different strategies have been proposed to tackle the recommendation problem; among them, collaborative filtering (CF) has shown to be very effective in predicting the relevance of unrated items, especially if much data about users-items interactions are available. CF approaches use item ratings[1] provided by the users in a system to suggest, in a personalized way, new and unknown items to interact with. Differently from CF RS, content-based (CB) approaches exploit descriptive metadata to find items that are similar to those already available in a user profile and recommend them accordingly. CF approaches suffer from the cold start problem: when a new item is added to the catalog, it has no ratings; therefore, the recommendation engine would not be able to recommend it even if it might be of interest to some users. This problem could be mitigated by combining both CF and CB approaches into a single one, this results in a Hybrid Recommender System.

A rich and useful (free) source of content description for items is given by Knowledge Graphs, which have been recently adopted to represent items, compute their similarity and relatedness [1] as well as to feed CB and hybrid recommendation engines [2]. The publication and spread of freely available Knowledge Graphs in the form of Linked Open Data datasets, such as DBpedia [3], has paved the way to the development of knowledge-aware recommendation engines in many application domains and, still, gives the possibility to easily switch from a domain to another one by just feeding the system with a different subset of the original graph.

Over the last years, we have seen at the rising of Deep Learning models in many fields such as Computer Vision, Speech Recognition, Natural Language Processing, and more

[1]Here, with ratings we refer to whatever user interaction, both implicit and explicit, from which we can infer a like or dislike behavior.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

recently, few attempts have also been made to solve the recommendation problem [4]. Deep Learning techniques have proven their strength, thus gaining the attention of both researchers and companies, and they are widely deployed in nowadays recommender systems. While research has mainly focused on improving accuracy metrics in recommenders, under the hood, their algorithms are becoming more and more complex, thus making it extremely hard to understand the reasons behind model predictions for a particular input. Lack of interpretability recently led both researchers and companies to pay more attention to explainable models. Indeed, it has been proven that showing to users an explanation for the provided recommendation leads to better interaction with the system [5], [6]. Moreover, when users understand how the system works, they can refine their preferences to get a better recommendation according to their tastes. However, in many popular recommenders such as Amazon or Netflix, the explanation provided is still feeble, as it is mainly based on a popularity basis: it just tells that users with similar tastes have enjoyed the suggested items. It turns out that this kind of explanation is not perceived as a valid justification of the reason why the system is recommending certain items, and it hardly improves users' loyalty in the system. On the other hand, a content-based explanation proves to be more engaging from the users' point of view because it makes users aware of the item's attributes that might be relevant to them.

In this paper, we present `SemAuto`, and we show how autoencoders technology can benefit from the existence of a Knowledge Graph to create a representation of a user profile that can be eventually exploited to predict ratings for unknown items. The rationale behind the approach is that both Knowledge Graphs and Artificial Neural Networks (ANNs) behind Deep Learning expose a graph-based structure. Hence, we may imagine building the topology of the hidden layers in the ANN by mimicking that of a Knowledge Graph.

Here we show how the model built by `SemAuto`, although very effective in computing accurate recommendations, can also be adopted to compute content-based explanations to recommended items. We evaluated the effectiveness of our approach through an A/B testing platform and compared its results with respect to two baselines. We tested both a pointwise and a pairwise explanation style by exploiting different kinds of the available information in DBpedia[2] (categorical and factual), in order to investigate how the effectiveness of the proposed explanation changes according to the selected knowledge adopted to feed `SemAuto`.

This paper presents more comprehensively the contributions presented in [7]–[9] and extends them by adding an evaluation in the cold start scenario and a more detailed result discussion.

The remainder of this paper is structured as follows: in the next section, we discuss related works about recommender systems exploiting deep learning, Knowledge Graphs, and Linked Open Data. Then, the basic notions of the technologies we adopted are introduced in Section III. The proposed recommendation model, the experimental settings and evaluation are described in Sections IV and V. Conclusions and Future Work close the paper.

II. RELATED WORKS

Autoencoders and Deep Learning for RS. The adoption of deep learning techniques is undoubtedly one of the main advances of the last years in the field of recommender systems. In [10], the authors propose the usage of a denoising autoencoder performing a top-N recommendation task by exploiting a corrupted version of the input data. A pure Collaborative-Filtering (CF) model based on autoencoders is described in [11], in which the authors develop both user-based and item-based autoencoders to tackle the recommendation task. Stacked Denoising Autoencoders are combined with collaborative filtering techniques in [12] where the authors leverage autoencoders to get a smaller and non-linear representation of the users-items interactions. This representation is eventually used to feed a deep neural network, which can alleviate the cold-start problem thanks to the integration of side information. A hybrid recommender system is finally built. Moreover, in [13], it is suggested how to apply deep learning methods with side information to reduce the sparsity of the rating matrix in collaborative approaches. In [14] the authors propose a deep learning approach to build a high-dimensional semantic space based on the substitutability of items; then, a user-specific transformation is learned to get a ranking of items from such a space. Analysis of the impact of deep learning on both recommendation quality and system scalability are presented in [15], where the authors first represent users and items through a rich feature set made on different domains and then map them to a latent space. Finally, a content-based recommender system is built.

All the approaches based on deep learning models that have been proposed over the years turned out to barely leverage latent factors to which no meaning can be attached. Among them, Autoencoder Neural Networks have proven their effectiveness in CF settings, as shown in [11], in which the authors use an Autoencoder fed with user ratings to predict the missing value for users' unseen items. In other works such as [13], a stacked architecture made of Autoencoders is proposed to perform a generalization over a higher set of latent features that every stacked autoencoder is able to learn. More recently, in [16] the authors propose a hybrid architecture for Autoencoders to incorporate both users' feedbacks and content description about items. A similar approach has been proposed in [17], in which they exploit side information in a CF setting by using Stacked Autoencoders to overcome the cold start problem and data sparsity.

Knowledge Graphs and Linked Open Data for RS. Several works have been proposed exploiting side information coming from Knowledge Graphs (KGs) and Linked Open Data (LOD) to enhance the performance of recommender systems. Most of them rely on the usage of DBpedia as

[2]http://dbpedia.org

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

KG. By leveraging the knowledge encoded in DBpedia, it is possible to build an accurate content-based recommender system [18]. In [19], for the very first time, a LOD-based recommender system is proposed to alleviate some of the major problems that affect collaborative techniques, mainly the high sparsity of the user-item matrix. The effectiveness of such an approach seems to be confirmed by a large number of methods that have been proposed afterward. A detailed review of LOD-based recommender systems is presented in [20].

It is worth noticing how KGs are recently being used in lots of applications; they freely offer a large amount of structured data, which proved to be very useful also in recommendation scenarios [21]–[23]. In particular, in [7], the authors introduce the idea of a Semantics-Aware Autoencoder, which paves the way to compute explanations by leveraging deep learning techniques.

Explanation for RS. A fundamental design principle for an Explainable RS is the interpretability of its model, which leads to a recommender system transparent to users. Explainable RSs are getting more and more relevance since they may lead to users retain, as investigated in [24]. Different studies [5], [25] have pointed out that introducing transparency in the recommendation process may have lots of advantages because users appear to be more satisfied with the recommendation if they are aware of the reasons why certain items are suggested. Furthermore, the provided explanation may also convince users to try items they would have usually ignored, thus improving users' confidence in the system.

Since the explanation may be decoupled from the recommendation process, a distinction between *transparency* and *justification* has to be made [26]. The explanation brings *transparency* to the system if it makes users aware of how the recommender engine works, explaining somehow the underlying algorithm behind the proposed suggestions. This is usually the case of those explanations computed along with the recommendation. On the other hand, *justification* implies an explanation that is not directly related to the recommendation algorithm; thus, it can be generated without any constraint. Such kind of explanations may be preferred to *transparency* because of algorithms that are difficult to explain or have not to be disclosed.

The main advantages users may get from the explanation are described in [27] and they include: *transparency*, *scrutability*, *trust*, *effectiveness*, *persuasiveness*, *efficiency* and *satisfaction*. In [28], the authors show how they can be exploited as evaluation metrics for explanatory services. However, providing adequate explanations is not always a trivial task; RSs have undoubtedly proven to be very accurate in accomplishing their tasks, but they usually work just like black boxes, being not transparent at all. To overcome this issue, new methods have been developed to generate an explainable recommendation ([25] provides an overview of the most successful approaches proposed over the years) such as *MoviExplain* [29], which exploits movies metadata to justify its recommendation lists. Other interesting works include: a RS based on Restricted Boltzmann Machines which looks at the rating distribution to identify the most explainable items [30]; a Latent Factor Model leveraging users reviews to compute more transparent recommendations [31]; finally, a novel approach based on movies information encoded in the LOD cloud which generates natural language explanation for the computed recommendation is presented in [32].

III. BACKGROUND

In this section, we briefly present the main technologies on which we base our model.

A. KNOWLEDGE GRAPHS

In 2012, Google announced its KG[3] as a new tool to improve the identification and retrieval of entities in return to a search query.

Since graph data are versatile so that they can model entities with connections among them, they can represent almost anything in the real world. For this reason, several big tech companies, such as Facebook[4], are spending resources on the development of their KG. Furthermore, the main advantage of the usage of graph data is that side information is easy to attach to the current graph, and this allows a company to enrich their knowledge base about a domain of interest progressively.

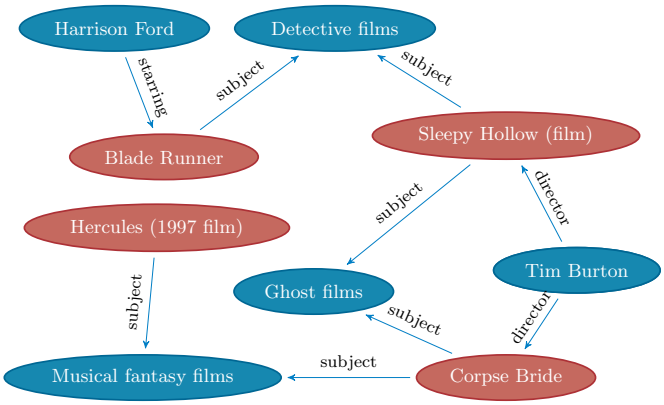


FIGURE 1: Part of a KG related to the movie domain.

Even though top tech companies have started to use KGs as knowledge bases in their products, the boost for this technology is given by some communities that began to develop KGs as well-structured graph data encoding the human knowledge.

Some prominent examples of KGs are DBpedia and Wikidata, which are community-driven projects that leverage on Wikipedia pages to automatically parse structured data. Wikipedia pages are very rich sources of information, but, unfortunately, they are human-readable documents, i.e., unstructured data that computer agents cannot easily understand. Making this information structured allows com-

[3]https://googleblog.blogspot.it/2012/05/introducing-knowledge-graph-things-not.html

[4]https://developers.facebook.com/docs/opengraph

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

puter agents to exploit this source of information. Starting from Wikipedia infoboxes, which summarize human-readable documents in tabular form, automatic tools extract entities and their relationships, which are lately stored as RDF triples[5].

Mainly, we may identify two kinds of information in DBpedia: semantics-aware and factual one. The former can be divided into categorical and ontological data. Categorical information is encoded through the `dct:subject` predicate and represents items categories parsed from Wikipedia infoboxes, such as `Detective films`[6] or `Ghost films`[7]. Categories in Wikipedia are collaboratively maintained by community editors, thus leading to a rich set of categories that reflects a human classification by encoding knowledge about classes, attributes, and other semantic relations [33]. Ontological data capture entities types (classes) and their hierarchy; it does not only represent their taxonomy but extends it by using restrictions on its relationships to other classes or on the properties a particular class is allowed to possess. Finally, factual knowledge is merely made of *facts*; it identifies items' attributes, as it can be in the movie domain that the actor `Harrison Ford` starred in the movie `Blade Runner`, as depicted in Figure 1. Differently from categorical information, factual one is identified via different attributes/predicates connecting an item to different entities as in the case of `director`, `starring`, etc..

In DBpedia, the quantity of categorical information is higher than the factual one. The former is more distributed over the items than the latter; if we consider movies, we see that they are more connected with each other via categories than via other entities. Hence, in this work, we focus on categorical information.

B. AUTOENCODER NEURAL NETWORKS

Autoencoders are a special kind of unsupervised learning ANNs that try to set the output values equal to the input ones, modeling an approximation of the identity function $\boldsymbol{y} = f(\boldsymbol{x}) = \boldsymbol{x}$. Roughly, they are forced to predict the same values they are fed with. Therefore, the number of output units and that of input nodes is the same, i.e., $|\boldsymbol{x}| = |\boldsymbol{y}|$. Such a task aims to obtain a new representation of the original data based on the values of the hidden layer neurons. Each of these layers projects the input data in a new Euclidean space whose dimensions depend on the number of the nodes in the hidden layer. Please notice that the actual meaning of each dimension in the new space is unknown since it encodes the implicit knowledge behind the original data. Hence, autoencoders are usually exploited to perform the so-called feature representation task.

IV. SEMANTICS-AWARE AUTOENCODER

Both knowledge graphs and autoencoder neural networks share a common structure: they are directed graphs. Actually, there are also differences between the two representations. In fact, in a neural network, nodes are structured in layers where two following ones are fully connected with each other; in a knowledge graph, instead, we cannot identify such a structure as each node (entity) is semantically connected to other ones. Moreover, while in a KG the semantics of connections as well as that of each node is explicit and well-defined; after the training of an autoencoder, the hidden layers encode some latent representation of the interaction between the input nodes whose meaning remains unknown.

In a Semantics-Aware Autoencoder (`SemAuto`), the hidden layers and their connections are substituted by nodes and labeled connections of a KG, thus having an explicit representation of the meaning associated both to hidden nodes and to their mutual connections [7]. This means that each neuron represents an entity in the adopted KG, and the edge between two autoencoder nodes exist if the corresponding KG entities are connected with a predicate (labeled edge).

Inspired by fully-connected Autoencoders, Semantics-Aware ones [9] try to solve the interpretability issue by labeling neurons in hidden layers, thus assigning an attribute in the explicit feature space to each of them. Considering that every hidden neuron represents a feature, it has to be stimulated only if it describes - and thus it belongs to - the associated item. Therefore a generic neuron representing an explicit feature results to be connected to those input or output neurons that describe the item. Hence, the resulting neural network is not fully-connected, as depicted in Figure 2.

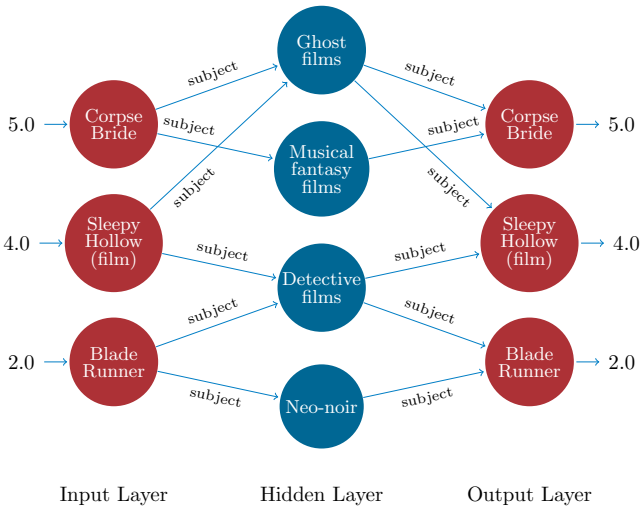


FIGURE 2: Architecture of a Semantics-Aware Autoencoder.

While DBpedia encodes different kinds of information, factual and semantics, in this work, we used only the latter for what concerns the recommendation since we found that categorical information (semantics) is better distributed among entities in the KG with respect to the factual one. A uniform distribution of item's attributes allows us to explore better the recommendation space since the user profiles we are able

[5]http://www.w3.org/TR/rdf11-concepts/
[6]https://en.wikipedia.org/wiki/Category:Detective_films
[7]https://en.wikipedia.org/wiki/Category:Ghost_films

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

to generate contain features shared across several items on which our recommendation algorithm generalizes the better.

Hence, the resulting autoencoder has three layers: an input layer, hidden layer, and output layer, where the input and output layers represent items in the catalog while the middle hidden layer contains their DBpedia categories.

Considering that the aforementioned autoencoder relies on a not fully-connected architecture and item's attributes are connected only with items they belong to, user ratings are propagating only through those hidden neurons that represent attributes related to rated items. Due to the nature of Autoencoder Neural Networks, they learn how to reconstruct input data by using a latent representation they encode in the hidden layer. Analogously, Semantics-Aware Autoencoders learn a function to reconstruct the input data by using a semantic representation of the user ratings; therefore, they reconstruct the user ratings they are fed with by using an explicit representation in terms of features.

It turns out that features belonging to positively rated items tend to have a higher weight, differently from those of negatively rated items. This behavior is quite understandable considering that a rating feeding an input node (representing an item in the catalog) flows throughout the neural network by crossing only features/nodes connected to it in the KG. We want to stress here that, although each autoencoder is trained over a not huge number of samples, in [9] we prove that recommendation results have very good performance in terms of accuracy and diversity also compared to state-of-the-art algorithms[8].

To train such kinds of autoencoder, we inhibit the feed-forward and backpropagation step for those neurons which result to be not connected in the KG by using a masking multiplier matrix M where rows and columns represent respectively items and features.

$$M = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \tag{1}$$

The matrix in Equation (1) represents the adjacency matrix of the KG where a generic entry is a binary value indicating whether a connection among entities exists in it. In other words, we have

$$a_{i,j} \in M = \begin{cases} 1, & \text{if item } i \text{ is connected to entity } j \\ 0, & \text{otherwise} \end{cases}$$

Hence, hidden (h) and output (o) layers are computed by the following two equations:

$$\begin{aligned} h &= g(X \times (W_1 \circ M)) \\ o &= g(h \times (W_2 \circ M^T)) \end{aligned} \tag{2}$$

During the backpropagation step, gradients are computed as usually for W_1 and W_2 with respect to a mean squared error loss $E = \frac{1}{2}\sum_i \parallel x_i - y_i \parallel^2$ being x_i and y_i the elements of the input and output vector respectively.

The weights update step in SGD (Stochastic Gradient Descent) backpropagation has been modified according to Equations (3) in order to take into account the masking matrix M:

$$\begin{aligned} W_1 &= (W_1 \circ M) - r \cdot \tfrac{\partial E}{\partial W_1} \\ W_2 &= (W_2 \circ M^T) - r \cdot \tfrac{\partial E}{\partial W_2} \end{aligned} \tag{3}$$

Where E is the mean squared error loss while W_1 and W_2 represent the weight matrices for the connections between the input and hidden layer (W_1) and between the hidden layer and the output layer (W_2). They are both initialized randomly using Xavier initialization [34]. In our experiments, we trained the model for 1000 epochs with a learning rate $r = 0.03$ and we used the well-known sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$ as activation function. Since we train one autoencoder per user, and we want it to overfit on user ratings, we did not use any form of regularization. According to equations (2), bias terms are missing since, in this model, they do not represent any information from the KG.

Computing user profiles. After training the autoencoder for each user u, we extract the weights of the hidden neurons and use them to build a user profile $P(u)$:

$$P(u) = \{\langle f_{u1}, w_{u1}\rangle, \ldots, \langle f_{um}, w_{um}\rangle\} \tag{4}$$

being f_u the label associated to the neuron and w_u its corresponding weight for u. Indeed, as each hidden neuron represents an entity in DBpedia, we may assume that its weight after the training is an indicator of the importance of the corresponding entity for u.

V. EXPERIMENTS

Here, we describe the experimental settings we used to test our approach. In this work, we focused on three main aspects: the recommendation accuracy and diversity, the cold start problem, and finally, the explanation, which is a direct consequence of our method since it is based upon an interpretable model that leads to explainability of the recommendation.

In order to validate our approach we performed experiments on the three datasets summarized in Table 1.

TABLE 1: Datasets.

	#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%

In our experiments, we referred to the freely available KG of DBpedia[9]. The mapping contains 22,959 mapped items for

[8]The code implementing `SemAuto` has been developed by using TensorFlow and is available at **https://github.com/sisinflab/SEMAUTO-2.0**.

[9]https://dbpedia.org

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

MovieLens 20M[10], 4,077 items mapped for Amazon Digital Music[11] and 9,926 items mapped for LibraryThing[12]. For our experiments, we removed from the datasets all the items without a mapping in DBpedia.

A. RECOMMENDATION

In this section, we show how we used our approach to generate accurate top-N recommendations (which also turn to be easily explainable as we will demonstrate in Section V-D).

Since the datasets used in the experimental settings are very sparse, the resulting user profiles (see Equation 4) are still sparse because users rated a few items with respect to all the items in the catalog. To reduce the sparsity of user profiles, inspired by [35], we use a *word2vec* based approach, which lets us infer a score for missing features. *Word2vec* is an efficient technique originally conceived to compute word embeddings (i.e., numerical representations of words) by capturing the semantic distribution of textual words in a latent space starting from their distribution within the sentences composing the original text. Given a corpus, e.g., an excerpt from a book, it projects each word in a multidimensional space such that words which are similar from a semantic point of view result close to each other. In this way, we can evaluate the semantic similarity between two words even if they never appear in the same sentence. Given a sequence of words $[x_1, \ldots, x_n]$ within a window, *word2vec* compute the probability for a new word x' to be the next one in the sequence. More formally, it computes $p(x' \mid [x_1, \ldots, x_n])$.

UserProfile2Vec. In our scenario, we may imagine replacing sentences represented by sequences of words with user profiles represented by sequences of features; given $F^u = \{f_{u1}, ..., f_{um}\}$ as the set of categories belonging to all the items rated by u, we use the *word2vec* approach to compute the weight of missing feature $f \notin F^u$.

Starting from $P(u)$, we first generate a corpus made of sequences of ordered features sorted by ω. Sorting each user profile is meant to give a pattern among features in different user profiles as they should appear nearby in word2vec's window according to their features ranking. This process lets the learned pattern to infer the mostly like missing features within the word2vec window for each user since a feature $f \in F^u$ results coherently for all $u \in U$.

Then, for each $\langle f, \omega \rangle \in P(u)$ we create a corresponding pair $\langle f, norm(\omega) \rangle$ with $norm$ being the mapping function

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

that linearly maps[13] a value in the interval $[0, 1]$ to a discrete value in the set $\{0.1, 0.2, 0.3, \ldots, 1\}$. The new pairs we obtain from this discretization process form the normalized set

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

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

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

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

Once we have the set $S = \{s(u) \mid u \in U\}$ we can feed the *word2vec* algorithm with this corpus to find patterns of features according to their distribution across all users. In the prediction phase, by using each user's sequence of features $s(u)$ as input for the trained *word2vec* model, we estimate the probability of $\langle f', norm(\omega') \rangle \in \bigcup_{v \in U} P^{norm}(v) - P^{norm}(u)$ to belong to the given context, or rather to be relevant for u. In other words, we compute $p(\langle f', norm(\omega') \rangle \mid s(u))$.

It is worth noticing that given $f' \in F^u$ we may have multiple pairs with f' as first element in $\bigcup_{v \in U} P^{norm}(v) - P^{norm}(u)$. For instance, given the feature `dbc:Ghost_films` we may have both $\langle$`dbc:Ghost_films`$, 0.2\rangle$ and $\langle$`dbc:Ghost_films`$, 0.5\rangle$, with the corresponding probabilities:

$$p(\langle \texttt{dbc:Ghost\_films}, 0.2 \rangle \mid s(u))$$
$$p(\langle \texttt{dbc:Ghost\_films}, 0.5 \rangle \mid s(u))$$

As we want to add the feature `dbc:Ghost_films` and its corresponding weight only once in the user profile, we select the pair with the highest probability. The new user profile is then

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

We point out that while the original $P(u)$ is built by exploiting only content-based information, the enhanced user profile $\hat{P}(u)$ also considers collaborative information as it is based on the set S containing a representation for the profiles of all the users in U.

Recommendations. Given the user profiles represented as vectors of weighted features, recommendations are then computed by using a well-known k-nearest neighbors approach [36]. User vectors are projected into a Vector Space Model to find user similarities, which are later exploited to compute, for each user, her neighborhood. Therefore, for each pair of users u and v we calculate their cosine similarity.

Given the users' similarity matrix, for each user u we find her top-k similar neighbors to infer the rate r for the item i as the weighted average rate that the neighborhood gave to it:

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

[10] https://grouplens.org/datasets/movielens/20m/

[11] http://jmcauley.ucsd.edu/data/amazon/

[12] https://www.librarything.com

[13] In our current implementation we use a standard `minmax` normalization.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

where $r(v_j, i)$ is the rating assigned to i by the user v_j. We use then ratings from Equation (5) to provide top-N recommendation for each user.

B. EVALUATION PROTOCOL

In this section, we show how we evaluated the performances of our methods in recommending items. For the evaluation of our approach we adopted the "all unrated items" protocol described in [37]: for each user u, a top-N recommendation list is provided by computing a score for every item i not rated by u, whether i appears in the user test set or not. Using the Hold-Out 80/20 split protocol, we assure that every user has 80% of their ratings in the training set and the remaining 20% in the test set. Then, recommendation lists are compared with the test set by computing both accuracy and diversity [38] metrics. More specifically, we evaluate Precision, Recall, F-1 score, nDCG [39], and aggregate diversity as a measure of how much diversified the recommendations we are able to generate are.

In our investigations, we compared our method with three different states of the art techniques widely used in recommendation scenarios: BPRMF [40], WRMF [41], [42] and a single-layer autoencoder for rating prediction. BPRMF is a Matrix Factorization algorithm that leverages Bayesian Personalized Ranking as the objective function. WRMF is a Weighted Regularized Matrix Factorization method that exploits users' implicit feedback to provide recommendations. In their basic version, both strategies rely exclusively on the User-Item matrix in a pure CF approach. Since our approach relies on hybrid techniques, we exploited side information (additional data associated with items) within the aforementioned baselines. In our experiments, we leveraged categorical information found on the DBpedia KG as side information and used the implementations of BPRMF and WRMF available in MyMediaLite[14] and implemented the autoencoder in TensorFlow[15]. Moreover, we did not limit to run the baselines with their default values, but we performed a hyperparameters optimization to find the best parameters for each baseline; the corresponding values are reported in Table 2. For what concerns our method, we tuned some hyperparameters such as the ones related to the *word2vec* approach (window and embedding sizes), and we gathered different results by varying the number k for the neighborhood size. We found that our method works better with a window size of 500 and an embedding size of 50.

C. COLD START

The cold start problem affects every CF-based RS when a new item is added since, in that case, it has not received any ratings yet; hence, CF algorithms are unable to recommend a fresh item. On the other hand, when a user is new to the system, she has no ratings; therefore, both CF and CB techniques are unable to accurately predict any interesting item since the system knows nothing about the user's preferences. As a consequence, in RSs, we may identify cold-item and cold-user problems.

To evaluate the effectiveness of our approach in such situations, we simulated the cold start scenario by preprocessing the datasets using the following protocol inspired by [43]. We made the candidate users cold by removing their ratings from the training set. We tested our approach with profiles reduced to 2, 5 and 10 ratings.

The procedure we adopted during the evaluation is detailed in the following.

1) Setup the cold start user scenario
   - Randomly choose at most 25% of users (whether they exist) from cold candidates and put them into set $\mathbb{U}_c$
   - $\forall u \in \mathbb{U}_c$ move out their ratings from the training set to $\mathbb{F}_c$
2) Evaluate the cold start user scenario
   - Create an empty set $\mathbb{R}_c$
   - For $n \in \{2, 5, 10\}$ do
     - $\forall u \in \mathbb{U}_c$ do:
       - randomly pick up n of his ratings from $\mathbb{F}_c$ and move them to the training set
     - Train the model
     - $\forall u \in \mathbb{U}_c$ generate recommendation for all unrated items
     - Evaluate recommendations for cold-users only

The method we propose is evaluated against all the baselines with respect to the same cold start splits. Hence it evaluated on the same users we sampled as cold start candidates.

D. EXPLANATION

The strength of `SemAuto` is its explainability since the model, as previously said, is interpretable. To validate the explanation we are able to furnish users, we set up an online experiment leveraging on an A/B test platform we built ad-hoc [8]; thanks to 892 volunteers, we evaluated the effectiveness of our approach and compared its results to two baselines. Hence, we primarily focus on the following research questions:

RQ1 Can we assume that the information encoded in the hidden layer of the `SemAuto` autoencoder is representative of user preferences?

RQ2 Given a content-based explanation built upon the `SemAuto` model, is a pairwise explanation better than a simple pointwise one for the user?

In our experimental setting, we build the structure of the `SemAuto` autoencoder by using those KG entities that are reachable through the predicate `dbo:subject` as item categories. To select the top-3 factual movie properties we used the approach originally proposed in [44], retaining the following properties: `dbo:starring`, `dbo:director`,

[14]http://mymedialite.net
[15]https://www.tensorflow.org

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

TABLE 2: Optimal hyperparameters adopted for each method.

MOVIELENS 20M	
AUTOREC	**rate:** 0.03, **epochs:** 1000, **hidden_units:** 15, **keep_prob:** 1
BPRMF	**factors:** 50, **rate:** 0.05, **iters:** 50, **reg_u:** 0.0025, **reg_i:** 0.001, **reg_j:** 0.0025
BPRMF + SI	**factors:** 5, **rate:** 0.05, **iters:** 15, **reg_u:** 0.0025, **reg_i:** 0.0025, **reg_j:** 0.001
WRMF	**factors:** 50, **iters:** 50, **regularization:** 0.1, **alpha:** 1
WRMF + SI	**factors:** 50, **iters:** 50, **regularization:** 0.1, **alpha:** 1
AMAZON DIGITAL MUSIC	
AUTOREC	**rate:** 0.01, **epochs:** 1000, **hidden_units:** 64, **keep_prob:** 1
BPRMF	**factors:** 10, **rate:** 0.05, **iters:** 50, **reg_u:** 0.02, **reg_i:** 0.02, **reg_j:** 0.00025
BPRMF + SI	**factors:** 100, **rate:** 0.05, **iters:** 100, **reg_u:** 0.0025, **reg_i:** 0.0025, **reg_j:** 0.002
WRMF	**factors:** 10, **iters:** 100, **regularization:** 0.1, **alpha:** 1
WRMF + SI	**factors:** 10, **iters:** 100, **regularization:** 0.1, **alpha:** 1
LIBRARYTHING	
AUTOREC	**rate:** 0.05, **epochs:** 1000, **hidden_units:** 64, **keep_prob:** 0.75
BPRMF	**factors:** 100, **rate:** 0.05, **iters:** 100, **reg_u:** 0.01, **reg_i:** 0.0025, **reg_j:** 0.001
BPRMF + SI	**factors:** 100, **rate:** 0.05, **iters:** 100, **reg_u:** 0.01, **reg_i:** 0.0025, **reg_j:** 0.001
WRMF	**factors:** 100, **iters:** 100, **regularization:** 0.1, **alpha:** 1
WRMF + SI	**factors:** 100, **iters:** 30, **regularization:** 0.075, **alpha:** 1

`dct:writer`[16].

Explaining the provided recommendation to users is not only a matter of model's interpretability: it allows us to catch the user's attention, hence, an appropriate style should have been taken into account. For this reason, in this work, we evaluate different explanation styles we generate through user profiles we compute with `SemAuto`.

In order to formulate a human-understandable explanation for the provided results, we rely on the weights associated with features in the user profile, which also appear in the description of the recommended items. In particular, given a user u and a recommendation list $rec(u) = [\langle i_1, \tilde{r}_1^u \rangle, \ldots, \langle i_n, \tilde{r}_n^u \rangle]$, with $\tilde{r}_k^u$ being a score/rating computed for the item i_k by a recommendation engine, we may compute a pointwise and a pairwise personalized explanation.

pointwise personalized. Given an item $i = \{f_{1i}, f_{2i}, \ldots, f_{ni}\}$ described by a set of features f_i, the pointwise explanation $e1^k(i)$ is computed by considering the set of top-k highest weighted features in $P(u)$ which also appear in i.

pairwise personalized. Given two items i and j such that $\tilde{r}_i^u > \tilde{r}_j^u$, the pairwise explanation $e2^k(i,j)$ is computed by evaluating both $e1^k(i)$ and $e1^k(j)$. In case m features are in common between $e1^k(i)$ and $e1^k(j)$, we compute $e1^{k+m}(j)$ and leave them only in $e1^k(i)$ thus avoiding any overlap between the explanation for i and that for j.

To verify that the explanation generated through a Semantics-Aware Autoencoder is able to satisfy the main explanatory criteria of *transparency*, *persuasiveness*, *effectiveness*, *trust* and *satisfaction*, we built a web platform that returns the top-5 recommendations and then asks for users' feedback about the provided explanation.

[16]We selected only the top-3 properties to reduce the dimension of the feature space and then minimize the noise in the provided explanation. Finding the best number of properties to compute explanations is not in the scope of this paper and is part of our future work.

E. EXPLANATION STYLES

We provided our platform with four different explanation styles: as in [32], we used a *popularity-based explanation* and a *non-personalized* one as baselines [28] while as third and fourth style we provide our pointwise and pairwise approaches. While a user interacts with the platform, we randomly select one of the four styles and show the associated explanation generated for the top-2 recommended items in a *pairwise* fashion. Hence, the user may receive one of the following explanations:

popularity-based *We suggest these items since they are very popular among people who like the same movies as you.*

(non-/pointwise) personalized *We guess you would like to watch i and j since they are about $\tilde{f}_{u1}, \ldots \tilde{f}_{uk}$*

pairwise personalized *We guess you would like to watch i more than j because you may prefer $e1^k$(i) over $e1^{k+m}$(j)* (Example 1)

Example 1. In order to show the difference between a pointwise and a pairwise personalized explanation, hereafter we report the two explanation styles with reference to a recommendation having *Terminator 2: Judgment Day* and *Transformers: Revenge of the fallen* as the first two items in the recommendation list. The pointwise personalized explanation may look like:

We guess you would like to watch Terminator 2: Judgment Day (1991) and Transformers: Revenge of the Fallen (2009)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

because you may prefer:

- *(subject) 1990s science fiction films*
- *(subject) Science fiction adventure films*
- *(subject) Drone films*
- *(subject) Cyberpunk films*

and:

- *(subject) Science fiction adventure films*
- *(subject) Films set in Egypt*
- *(subject) Robot films*
- *(subject) Films shot in Arizona*
- *(subject) Ancient astronauts in fiction*

while the pairwise version (see also Figure 3b) is a bit different:

We guess you would like to watch Terminator 2: Judgment Day (1991) more than Transformers: Revenge of the Fallen (2009) because you may prefer:

- *(subject) 1990s science fiction films*
- *(subject) Science fiction adventure films*
- *(subject) Drone films*
- *(subject) Cyberpunk films*

over:

- *(subject) Films set in Egypt*
- *(subject) Robot films*
- *(subject) Films shot in Arizona*
- *(subject) Ancient astronauts in fiction*
- *(subject) IMAX films*

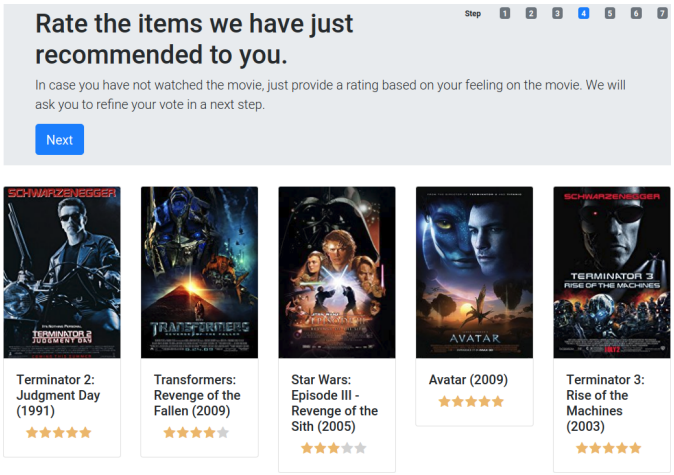
■

The *popularity-based explanation*, as its name suggests, justifies recommender choices by leveraging the popularity of suggested items among the users with similar tastes of the active user u, hence it may be considered as the less meaningful to the user. The *non-personalized explanation*, instead, tries to explain the provided recommendation by using additional information about the items. In our experiments, given the top-2 recommended items i and j, we randomly select $k = 5$ features from the set $F_{ij} = F_i \cup F_j = \{f_{1i}, f_{2i}, \ldots, f_{ni}\} \cup \{f_{1j}, f_{2j}, \ldots, f_{n'j}\}$. In a similar manner, in a *pointwise personalized explanation* we selected the top-5 features from each set F_i and F_j. The value $k = 5$ has been selected also to compute $e2^k(i, j)$ in the *pairwise personalized explanation*.

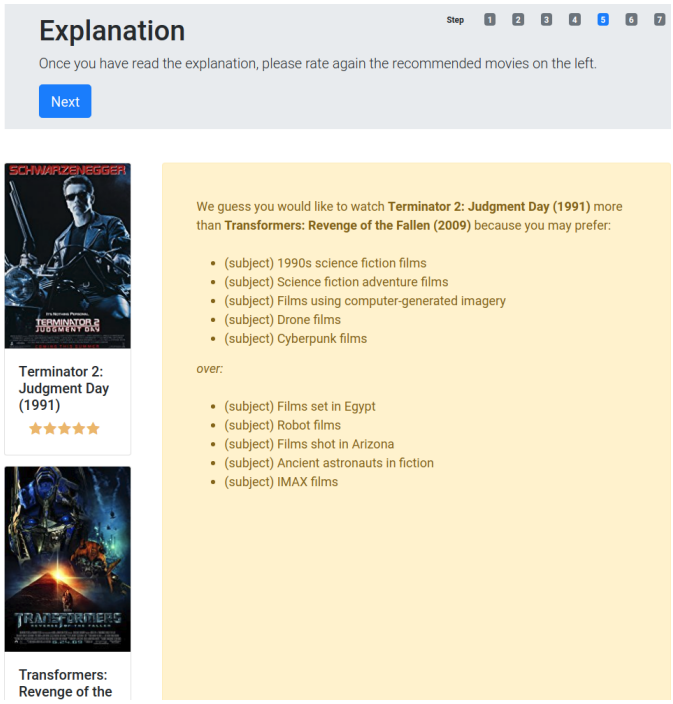
Please notice that the considered set of features per item varies according to the different configuration adopted for the `SemAuto` autoencoder; it may include just item categories, factual data or both of them.

During the online A/B testing phase, we fixed a sequence of steps in order to measure the aforementioned explanatory criteria.

Steps 1-3. At the beginning of the experiment, the user u selects at least 15 movies she has watched among the ones randomly listed by the platform. The movies belong to the well-known MovieLens 20M dataset [17]. Then, she is invited to rate each selected movie on a five-stars rating scale; data so gathered are exploited to get both the user profile computed with the semantic autoencoder and a top-5 recommendation list.



(a) Step 4. The user is asked to rate the recommended items, even if she has not watched them.



(b) Step 5. The user is asked to read the explanation and after that to rate again the top-2 recommended items.

FIGURE 3: Screenshots of the A/B testing platform.

Step 4. Once the recommendation has been generated, the user is asked to rate the suggested items, even if no explanation has been shown yet (see Figure 3a): these ratings will be relevant to determine the impact the explanation has

[17] https://grouplens.org/datasets/movielens/20m/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

on the user (*persuasiveness*).

Step 5. The next step consists of showing to u one of the four randomly selected explanation styles deployed within the application (see Figure 3b). After enjoying the explanation, the user has to re-rate the top-2 recommended items, letting us measure how different is the evaluation of the items before and after the explanation has been provided.

Step 6. Similarly, in the last part of the experiment, the user is asked to re-rate the recommended movies after watching the related trailers. This phase allows u to emulate the consumption of the items and makes her more aware of the topics of the suggested movies. In this way we can evaluate how much effective the selected explanation style was (*effectiveness*).

Step 7. Finally, the user fills a questionnaire, aimed at measuring the explanation *transparency*, *trust* and *satisfaction* (see Table 3).

When evaluating an explanation system, the main metrics to evaluate are [25]:

- *transparency*, which refers to the capability of the explanation to make users aware of how the system works;
- *trust*, or rather the confidence users have in the system;
- *satisfaction*, if users have an enjoyable experience in the usage of the system;
- *persuasiveness*, which evaluates how much convincing is the proposed explanation;
- *effectiveness*: the explanation is said to be effective if it helps users to correctly estimate items relevance before the consumption.

The first three metrics are evaluated by collecting answers from users after filling the questionnaire at Step 7. As a final score for the first and the second metrics, we used the percentage of users that answered positively to the questions while we exploited the average score assigned by users to quantify the overall *satisfaction*.

In order to evaluate the *persuasiveness* of the proposed explanation, we asked users to rate each recommended item before and after showing them the explanation: if the rating provided after looking at the explanation is higher than the original one, then the explanation has been able to persuade the user to try the suggested item. More formally we measure persuasiveness as [25]:

$$\textit{persuasiveness} = \frac{1}{|U|} \cdot \sum_{u \in U} \frac{1}{N} \cdot \sum_{i \in I_N^u} (r_{ui}^e - r_{ui})$$

where U stands for the collection of users; I_N^u represents the set of top-N recommended items for u; r_{ui} and r_{ui}^e are, respectively, the ratings u assigns to i just before and after the explanation is provided.

Analogously, we evaluated the *effectiveness* as the difference between two ratings (see Equation (V-E) [25]), where r_{ui}^t represents the rating the user gives to the suggested movie after watching the related trailer (r_{ui}^t).

$$\textit{effectiveness} = \frac{1}{|U|} \cdot \sum_{u \in U} \frac{1}{N} \cdot \sum_{i \in I_N^u} ||r_{ui}^e - r_{ui}^t||$$

The lower this value, the more effective the explanation, since it implies that users have rated each item with very similar values before and after the explanation has been provided.

TABLE 3: The final questionnaire.

METRIC	QUESTION
transparency	I understood the reason why the two movies have been ranked in the proposed order.
trust	The explanation increased my trust in the system.
satisfaction	The provided explanation: **really** captures my tastes. **partially** captures my tastes. **does not** capture my tastes.

We conducted our experiment with the help of 892 volunteers[18], with at least 73 subjects for each of the implemented settings. As stated in [45], 73 has to be considered as the minimum acceptable sample size for such kinds of experiments. This assures the significance of our experimental results. Furthermore, we verified the statistical significance of our experiment by using *Wilcoxon Rank-Sum Test*, getting $p \ll 0.01$.

VI. RESULTS DISCUSSION

We have evaluated our approach by comparing it with different state-of-the-art baselines and using three datasets in different domains, with respect to accuracy and diversity point of view. In Table 6, we report the results gathered in datasets by applying the methods discussed above. As for our approach `SemAuto`, we tested it for a different number of neighbors by varying k. In terms of accuracy, we see that `SemAuto` outperforms our baselines on both MovieLens 20M and Amazon Digital Music datasets, while on LibraryThing the achieved results are quite the same as BPRMF and WRMF baselines. In particular, we suppose that the LibraryThing dataset is highly affected by popularity bias since the fully-connected autoencoder significantly outperforms all the other baselines obtaining an aggregate diversity of only 118 items.

Moreover, focusing on the results, it seems that our approach provides very discriminative descriptions for each user, letting us identify the best neighborhood and compute both accurate and diversified recommendations. Regarding diversity, we get much better results on all the datasets. As a matter of fact, we achieve the same results in terms of accuracy as the baselines by suggesting many more items. This means that our approach really captures the real users' preferences and therefore, it provides useful recommendations that turn out to be also diversified since `SemAuto`

[18]They were recruited both among our students and via Amazon Mechanical Turk.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

extracts unusual features from user ratings that are relevant to the user. Hence, exploring the long tail allows SemAuto to provide both accurate and diversified recommendations. A further confirmation that our approach examines the long tail is given by results reported in Table 6, where at a given baselines' value for the F-1 score, our approach recommends more items than the baselines. Even if one may argue that our method is more likely a rating prediction approach, nDCG results reported in Table 6 confirm that SemAuto ranks recommended items as they appear in the test set. In other words, it means that our method not only suggests relevant items, even from the long tail and so maintaining a high level of diversity, but it does in the proper ranking order within the recommendation list it generates.

Analyzing Table 4, we can state that SemAuto performs better on those datasets whose items are described by a larger amount of categorical information, which implies the usage of many hidden units. Since ANNs can model very complex functions if enough hidden units are provided, as Universal Approximation Theorem points out, the more dataset's items are rich in features, the better SemAuto performs. For this reason, our approach proved to work better on MovieLens 20M dataset (whose related neural networks have a high number of hidden units) rather than the others. In particular, the experiments show that the performances get worse as the number of the neurons decreases, i.e., available categories are not enough.

Regarding the cold user start scenario, we can see that the same trend is confirmed. Our SemAuto approach performs better when a large number of features are associated with items since the more feature we have, the more hidden neurons our model has. Taking a look at Table 7 we can see that in a cold start scenario our method performs the best on MovieLens 20M where we have a huge amount of features in the user profiles, while we perform worst on Amazon Digital Music and LibraryThing where few features are available (Table 5).

TABLE 4: Summary of hidden units for mapped items only.

	average #features
Movielens 20M	1015.87
Amazon Digital Music	7.22
LibraryThing	206.88

TABLE 5: Average number of features in cold start user profiles.

	average #features		
	n = 2	n = 5	n = 10
Movielens 20M	760.03	775.09	793.49
Amazon Digital Music	6.57	6.72	6.89
LibraryThing	156.41	161.11	166.35

Concerning the explanation, analyzing the results shown in Figure 4, we can state that, as expected, users prefer a CB explanation, as the *popularity-based* merely tells users that the recommended item is popular among the user's neighborhood. All the explanation metrics but the *persuasiveness* one, confirm this trend; in this case, quite interestingly, the *non-personalized* explanation style with categorical information get negative values, as it happens when users rate items with lower values after looking at the explanation than before. This means that the provided explanation discourage users from watching the suggested movies, probably because prompting a random set of item's feature as an explanation is deleterious for what concerns users' persuasion to watch a movie. Interestingly, it is worthy to notice that when a *personalized* explanation is provided to users, the categorical information works better than the factual one, although combining both of them achieves the best results in users' persuasion. Considering the *satisfaction*, it is reasonable to expect that *non-personalized* style works worst with respect to other explanation styles since it uses a random set of items' feature; this behavior is confirmed when it uses categorical or factual information. However, when both of them are combined, unexpectedly *non-personalized* outperforms the categorical pairwise. We suppose that the more considerable amount of diversified item's attributes deceive the user and lead her to be satisfied with the received explanation; furthermore, we assume that a significant contribution is given by the factual, as also pointed by the *effectiveness* metric. Here, the factual information outperforms all the other explanation styles when used in the pairwise approach, but when combined with the categorical information, they perform worst with respect to the former alone. We suppose that the categorical information brings so much noise that it makes the user not correctly to estimate her expectation after she read the explanation. In detail, the factual information works better than the categorical one; we assume that users feel more comfortable with entities such as actors, directors, or real people rather than abstract concepts like the ones categorical information provides. The same goes for *trust* in which it turns out that users consider more affordable an explanation, whether it consists of factual information. Regarding *transparency*, the system is perceived as more transparent from users when supplementary information is prompted to them, therefore combining categorical and factual gets the best result.

Finally, we can state that the *personalized* style outperforms the *non-personalized* since the latter relies on a random list of features that belong to the suggested item, with no assurance that they reflect the user's preferences. Furthermore, among the personalized approaches, the pairwise is the one that performs better; we suppose that it actually captures the user's preferences, and is capable to rank items' features accordingly.

To provide an answer to **RQ1**, examining the results, it turns out that our SemAuto provides reliable users' descriptions, as evidenced by the *effectiveness* metric, which gets the lowest value by using a pairwise explanation. This can be interpreted as a strong signal that the information encoded in the autoencoder hidden layer is representative of the users' preferences because the users are less prone to change her

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

TABLE 6: Experimental Results.

	k	F1	Prec.	Recall	nDCG	aggrdiv
MOVIELENS 20M						
AUTOREC	–	0.17837	0.17840	0.17835	0.21211	372
BPRMF	–	0.16902	0.17334	0.16491	0.17106	3827
BPRMF + SI	–	0.15129	0.15025	0.13786	0.16112	1191
WRMF	–	0.23161	0.23066	0.23255	0.26664	1567
WRMF + SI	–	0.23235	0.23072	0.23248	0.26652	1566
SemAuto	5	0.18857	0.18551	0.19173	0.21941	**5214**
	10	0.21268	0.21009	0.21533	0.24945	3350
	20	0.22886	0.22684	0.23092	0.27147	2417
	40	**0.23675**	**0.23534**	**0.23818**	**0.28363**	1800
	50	**0.23827**	**0.23686**	**0.23970**	**0.28605**	1653
	100	**0.23961**	**0.23832**	**0.24090**	**0.28924**	1310
AMAZON DIGITAL MUSIC						
AUTOREC	–	0.01728	0.00981	0.07230	0.04762	534
BPRMF	–	0.01164	0.00651	0.05479	0.02363	1017
BPRMF + SI	–	0.01206	0.00677	0.05502	0.02475	539
WRMF	–	0.02261	0.01278	0.09794	0.05656	122
WRMF + SI	–	0.02151	0.01216	0.09325	0.05220	111
SemAuto	5	0.01514	0.00862	0.06233	0.04365	3378
	10	0.01920	0.01091	0.07994	0.05421	3449
	20	0.02233	0.01267	0.09385	**0.06296**	3523
	40	**0.02572**	**0.01460**	**0.10805**	**0.06980**	**3549**
	50	**0.02618**	**0.01486**	**0.10974**	**0.07032**	**3549**
	100	**0.02835**	**0.01608**	**0.11964**	**0.07471**	3448
LIBRARYTHING						
AUTOREC	–	**0.11157**	**0.15073**	**0.08856**	**0.14919**	118
BPRMF	–	0.01672	0.01464	0.01950	0.01834	2354
BPRMF + SI	–	0.01344	0.01148	0.01620	0.01588	3165
WRMF	–	0.01838	0.01648	0.02077	0.01996	1715
WRMF + SI	–	0.01385	0.01251	0.01551	0.01574	1769
SemAuto	5	0.00840	0.00764	0.00931	0.00930	**4895**
	10	0.01034	0.00930	0.01163	0.01139	3558
	20	0.01152	0.01029	0.01310	0.01248	2245
	40	0.01195	0.01073	0.01347	0.01339	1498
	50	0.01229	0.01110	0.01378	0.01374	1312
	100	0.01278	0.01136	0.01461	0.01503	873

ratings after she read the explanation.

As for **RQ2**, we can assert that the pairwise approach outperforms the pointwise one in all metrics, especially in *transparency* because it provides a better justification on how the system ranks items according to the importance of the features in the user profile. This lets the user understand better how her preferences are involved in the recommendation process. This has an impact, especially for the *persuasiveness* metric, where the pairwise approach has a higher score with respect to the pointwise explanation, thus leading users to consume an item after they have read the provided explanation.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have presented a novel approach that exploits both deep learning techniques and knowledge graphs to provide accurate, diversified and explainable recommendations. Usually, a classical application of autoencoders consists of compressing the original input data into a new latent space with lower dimensions so that finding relationships and similarities among the data should be easier thanks to the reduced dimensionality. In this case, we rely on the topology of a KG to label hidden neurons for a not fully-connected Autoencoder Neural Network whose model turns out to be interpretable. We used our approach to auto-encode user ratings in a recommendation scenario via the DBpedia KG and proposed an algorithm to compute user profiles, which are exploited to provide recommendations based on the seman-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

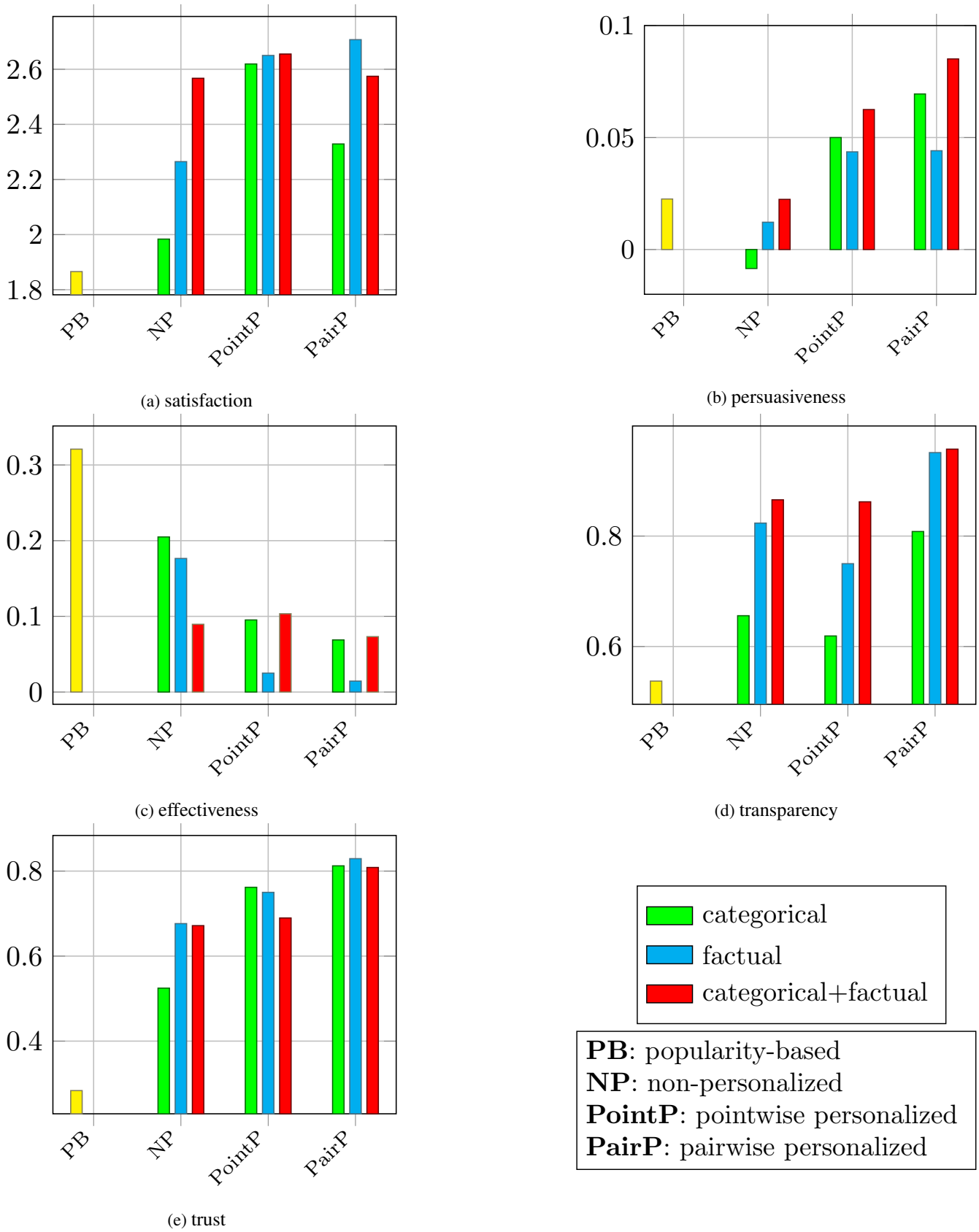


FIGURE 4: Results comparison.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953308, IEEE Access

TABLE 7: Experimental results in cold start scenario.

	#ratings	k	F1	Prec.	Recall	nDCG	aggrdiv
MOVIELENS 20M							
AUTOREC	2	–	0.00596	0.00356	0.03564	0.01824	88
BPRMF		–	0.02136	0.01175	0.11749	0.05675	492
WRMF		–	0.01647	0.00906	0.09057	0.05541	**840**
SemAuto		100	**0.02211**	**0.01216**	**0.12159**	**0.06039**	78
AUTOREC	5	–	0.00798	0.00439	0.04395	0.02068	78
BPRMF		–	0.02907	0.01599	0.15995	0.08020	**652**
WRMF		–	0.03549	0.01952	0.19517	0.09586	553
SemAuto		100	**0.03579**	**0.01969**	**0.19689**	**0.10161**	354
AUTOREC	10	–	0.01215	0.00694	0.04877	0.02648	113
BPRMF		–	0.04542	0.02598	0.18054	0.10064	**968**
WRMF		–	**0.05594**	**0.03199**	**0.22251**	**0.10953**	887
SemAuto		100	0.05093	0.02911	0.20328	0.10809	654
AMAZON DIGITAL MUSIC							
AUTOREC	2	–	0.00011	0.00006	0.00061	0.00032	**435**
BPRMF		–	0.00514	0.00283	0.02828	0.01076	45
WRMF		–	**0.01144**	**0.00629**	**0.06296**	**0.03763**	62
SemAuto		100	0.00116	0.00064	0.00634	0.00228	41
AUTOREC	5	–	0.00044	0.00024	0.00243	0.00143	**774**
BPRMF		–	0.00480	0.00264	0.0264	0.01157	14
WRMF		–	**0.01011**	**0.00556**	**0.05565**	**0.02863**	59
SemAuto		35	0.00249	0.00137	0.01369	0.00634	59
AUTOREC	10	–	0.00076	0.00045	0.00256	0.00172	**918**
BPRMF		–	0.00612	0.00356	0.02189	0.01212	14
WRMF		–	**0.01541**	**0.00903**	**0.05246**	**0.02909**	66
SemAuto		100	0.00507	0.00292	0.01916	0.00967	33
LIBRARYTHING							
AUTOREC	2	–	0.00020	0.00011	0.00111	0.00070	**2291**
BPRMF		–	0.02496	0.01373	0.13726	0.07732	437
WRMF		–	**0.02796**	**0.01538**	**0.15380**	**0.09429**	1364
SemAuto		100	0.01694	0.00932	0.09316	0.04933	707
AUTOREC	5	–	0.00178	0.00098	0.00982	0.00508	**1371**
BPRMF		–	0.02665	0.01466	0.14664	0.08040	598
WRMF		–	**0.02996**	**0.01648**	**0.16483**	**0.10044**	1270
SemAuto		100	0.01864	0.01025	0.10253	0.05391	656
AUTOREC	10	–	0.00267	0.00159	0.00830	0.00406	321
BPRMF		–	0.03373	0.02012	0.10419	0.06090	730
WRMF		–	**0.04263**	**0.02547**	**0.13065**	**0.09167**	**1254**
SemAuto		100	0.02661	0.01587	0.08214	0.05334	595

tic features we extract with our autoencoder. Experimental results show that we are able to outperform state-of-the-art recommendation algorithms on both accuracy and diversity. We tested our approach even in a cold start scenario, finding a common trend: the more categorical information (features) we have for each user profile, the better our method performs. This is quite interesting since this approach could be used to perform studies on data quality for knowledge bases in recommendation scenarios; in a future investigation, we will compare how the use of different knowledge graphs within our method will impact the quality of recommendations regarding both accuracy and diversity. Furthermore, we will compare our approach with other competitive baselines, as suggested in more recent works [46].

We also performed online experiments to validate the capability of our approach to generating an explanation for recommendation lists via the exploitation of data coming from the DBpedia knowledge graph. Experimental results show that our SemAuto can be used to generate a com-

pelling explanation for a recommendation list. In particular, a content-based explanation is preferred by users, as it outperforms other baselines concerning transparency, trust, satisfaction, persuasiveness,and effectiveness. As we can see in the *satisfaction*, *effectiveness* and *trust* plots in Figure 4, for both pointwise and pairwise approaches, an interesting point is that, in order to build an explanation, factual data works better than the semantic/categorical one, achieving the same results as when both semantic and factual data are exploited.

The results presented in this paper pave the way for various further investigations in different directions. From a methodological and algorithmic point of view, we can surely investigate the augmentation of further deep learning techniques via the injection of explicit and structured knowledge coming from external sources of information. Giving an explicit meaning to neurons in an ANN as well as to their connections can fill the semantic gap in describing models trained via deep learning algorithms. Moreover, having an explicit representation of latent features opens the door to better and explicit user modeling. We are currently investigating how to exploit the structure of a KG-enabled autoencoder to infer qualitative preferences represented by means of expressive languages such as CP-theories [47]. Providing such a powerful representation may also result in being a key factor in the automatic generation of explanation to recommendation results.

REFERENCES

[1] T. Di Noia, V. Ostuni, J. Rosati, P. Tomeo, E. Di Sciascio, R. Mirizzi, and C. Bartolini, "Building a relatedness graph from linked open data: A case study in the it domain," Expert Systems with Applications, vol. 44, pp. 354–366, 2016.

[2] S. Oramas, V. Ostuni, T. Di Noia, X. Serra, and E. Di Sciascio, "Sound and music recommendation with knowledge graphs," ACM Transactions on Intelligent Systems and Technology, vol. 8, no. 2, 2016.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference, ser. ISWC'07/ASWC'07. Springer-Verlag, 2007, pp. 722–735.

[4] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," ACM Comput. Surv., vol. 52, no. 1, pp. 5:1–5:38, Feb. 2019. [Online]. Available: http://doi.acm.org/10.1145/3285029

[5] R. Sinha and K. Swearingen, "The role of transparency in recommender systems," in CHI '02 Extended Abstracts on Human Factors in Computing Systems, ser. CHI EA '02. New York, NY, USA: ACM, 2002, pp. 830–831.

[6] N. Tintarev, "Explanations of recommendations," in Proceedings of the 2007 ACM Conference on Recommender Systems, ser. RecSys '07. New York, NY, USA: ACM, 2007, pp. 203–206. [Online]. Available: http://doi.acm.org/10.1145/1297231.1297275

[7] V. Bellini, V. W. Anelli, T. Di Noia, and E. Di Sciascio, "Auto-encoding user ratings via knowledge graphs in recommendation scenarios," in Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems. ACM, 2017, pp. 60–66.

[8] V. Bellini, A. Schiavone, T. Di Noia, A. Ragone, and E. Di Sciascio, "Knowledge-aware autoencoders for explainable recommender systems," in Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems, ser. DLRS 2018. New York, NY, USA: ACM, 2018, pp. 24–31.

[9] V. Bellini, A. Schiavone, T. Di Noia, A. Ragone, and E. Di Sciascio, "Computing recommendations via a knowledge graph-aware autoencoder," in Proceedings of the RecSys 2018 Workshop on Knowledge-aware and Conversational Recommender Systems (KaRS) co-located with 12th ACM Conference on Recommender Systems (RecSys 2018), Vancouver, Canada, October 7, 2018., 2018.

[10] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, ser. WSDM '16. New York, NY, USA: ACM, 2016, pp. 153–162.

[11] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in Proceedings of the 24th International Conference on World Wide Web, ser. WWW '15 Companion. New York, NY, USA: ACM, 2015, pp. 111–112.

[12] F. Strub, R. Gaudel, and J. Mary, "Hybrid recommender system based on autoencoders," in Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, ser. DLRS 2016. New York, NY, USA: ACM, 2016, pp. 11–16.

[13] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '15. New York, NY, USA: ACM, 2015, pp. 1235–1244.

[14] J. B. P. Vuurens, M. Larson, and A. P. de Vries, "Exploring deep space: Learning personalized ranking in a semantic space," in Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, ser. DLRS 2016. New York, NY, USA: ACM, 2016, pp. 23–28.

[15] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in Proceedings of the 24th International Conference on World Wide Web, ser. WWW '15. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015, pp. 278–288.

[16] H. Wang, X. SHI, and D.-Y. Yeung, "Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks," in Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 415–423.

[17] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, "A hybrid collaborative filtering model with deep structure for recommender systems." in AAAI, 2017, pp. 1309–1315.

[18] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker, "Linked open data to support content-based recommender systems," in Proceedings of the 8th International Conference on Semantic Systems, ser. I-SEMANTICS '12. New York, NY, USA: ACM, 2012, pp. 1–8.

[19] B. Heitmann and C. Hayes, "C.: Using linked data to build open, collaborative recommender systems," in In: AAAI Spring Symposium: Linked Data Meets Artificial IntelligenceâĂŹ. (2010, 2010.

[20] M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro, "Semantics-aware content-based recommender systems," in Recommender Systems Handbook, 2015, pp. 119–159.

[21] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker, "Linked open data to support content-based recommender systems," in Proceedings of the 8th International Conference on Semantic Systems. ACM, 2012, pp. 1–8.

[22] M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro, Semantics-Aware Content-Based Recommender Systems. Boston, MA: Springer US, 2015, pp. 119–159.

[23] S. Oramas, V. C. Ostuni, T. D. Noia, X. Serra, and E. D. Sciascio, "Sound and music recommendation with knowledge graphs," ACM Trans. Intell. Syst. Technol., vol. 8, no. 2, pp. 21:1–21:21, Oct. 2016.

[24] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, ser. CSCW '00. New York, NY, USA: ACM, 2000, pp. 241–250.

[25] N. Tintarev and J. Masthoff, Designing and Evaluating Explanations for Recommender Systems. Boston, MA: Springer US, 2011, pp. 479–510.

[26] J. Vig, S. Sen, and J. Riedl, "Tagsplanations: Explaining recommendations using tags," in Proceedings of the 14th International Conference on Intelligent User Interfaces, ser. IUI '09. New York, NY, USA: ACM, 2009, pp. 47–56.

[27] N. Tintarev and J. Masthoff, "A survey of explanations in recommender systems," in Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop, ser. ICDEW '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 801–810.

[28] N. Tintarev and J. Masthoff, "Evaluating the effectiveness of explanations for recommender systems," User Modeling and User-Adapted Interaction, vol. 22, no. 4-5, pp. 399–439, 2012.

[29] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Moviexplain: A recommender system with explanations," in Proceedings of the Third

ACM Conference on Recommender Systems, ser. RecSys '09. New York, NY, USA: ACM, 2009, pp. 317–320.

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

[31] 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 Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, ser. SIGIR '14. New York, NY, USA: ACM, 2014, pp. 83–92.

[32] C. Musto, F. Narducci, P. Lops, M. De Gemmis, and G. Semeraro, "Explod: A framework for explaining recommendations based on the linked open data cloud," in Proceedings of the 10th ACM Conference on Recommender Systems, ser. RecSys '16. New York, NY, USA: ACM, 2016, pp. 151–154.

[33] V. Nastase and M. Strube, "Decoding wikipedia categories for knowledge acquisition." in AAAI, vol. 8, 2008, pp. 1219–1224.

[34] 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, 2010, pp. 249–256.

[35] 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., 2013, pp. 3111–3119.

[36] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," IEEE Internet computing, no. 1, pp. 76–80, 2003.

[37] H. Steck, "Evaluation of recommendations: Rating-prediction and ranking," in Proceedings of the 7th ACM Conference on Recommender Systems, ser. RecSys '13. New York, NY, USA: ACM, 2013, pp. 213–220.

[38] P. Castells, N. J. Hurley, and S. Vargas, "Novelty and diversity in recommender systems," in Recommender Systems Handbook. Springer, 2015, pp. 881–918.

[39] K. Järvelin and J. Kekäläinen, "Ir evaluation methods for retrieving highly relevant documents," in Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR '00. New York, NY, USA: ACM, 2000, pp. 41–48.

[40] 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, ser. UAI '09. Arlington, Virginia, United States: AUAI Press, 2009, pp. 452–461.

[41] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, ser. ICDM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 502–511.

[42] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, ser. ICDM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 263–272.

[43] J. Xu, Y. Yao, H. Tong, X. Tao, and J. Lu, Ice-Breaking: Mitigating cold-start recommendation problem by rating comparison, 2015, pp. 3981–3987.

[44] A. Ragone, P. Tomeo, C. Magarelli, T. Di Noia, M. Palmonari, A. Maurino, and E. Di Sciascio, "Schema-summarization in linked-data-based feature selection for recommender systems," in Proceedings of the Symposium on Applied Computing, ser. SAC '17. New York, NY, USA: ACM, 2017, pp. 330–335.

[45] B. P. Knijnenburg and M. C. Willemsen, Evaluating Recommender Systems with User Experiments. Boston, MA: Springer US, 2015, pp. 309–352.

[46] C. Musto, T. Franza, G. Semeraro, M. de Gemmis, and P. Lops, "Deep content-based recommender systems exploiting recurrent neural networks and linked open data," in Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization, ser. UMAP '18. New York, NY, USA: ACM, 2018, pp. 239–244.

[47] T. Di Noia, T. Lukasiewicz, M. V. Martínez, G. I. Simari, and O. Tifrea-Marciuska, "Combining existential rules with the power of cp-theories," in Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, 2015, pp. 2918–2925.