RESEARCH

A qualitative analysis of knowledge graphs in recommendation scenarios through semantics-aware autoencoders

Vito Bellini[1] · Eugenio Di Sciascio[2] · Francesco Maria Donini[3] · Claudio Pomo[2] · Azzurra Ragone[4] · Angelo Schiavone[2]

Abstract

Knowledge Graphs (KGs) have already proven their strength as a source of high-quality information for different tasks such as data integration, search, text summarization, and personalization. Another prominent research field that has been benefiting from the adoption of KGs is that of Recommender Systems (RSs). Feeding a RS with data coming from a KG improves recommendation accuracy, diversity, and novelty, and paves the way to the creation of interpretable models that can be used for explanations. This possibility of combining a KG with a RS raises the question whether such an addition can be performed in a plug-and-play fashion – also with respect to the recommendation domain – or whether each combination needs a careful evaluation. To investigate such a question, we consider all possible combinations of *(i)* three recommendation tasks (books, music, movies); *(ii)* three recommendation models fed with data from a KG (and in particular, a semantics-aware deep learning model, that we discuss in detail), compared with three baseline models without KG addition; *(iii)* two main encyclopedic KGs freely available on the Web: DBpedia and Wikidata. Supported by an extensive experimental evaluation, we show the final results in terms of accuracy and diversity of the various combinations, highlighting that the injection of knowledge does not always pay off. Moreover, we show how the choice of the KG, and the form of data in it, affect the results, depending on the recommendation domain and the learning model.

Keywords Recommender systems · Autoencoder neural network · Knowledge graph · Deep learning

Vito Bellini work done prior to joining Amazon.

✉ Vito Bellini
vitob@amazon.com

✉ Claudio Pomo
claudio.pomo@poliba.it

Extended author information available on the last page of the article

1 Introduction

Nowadays, we are overwhelmed by a large amount of available information we can benefit from. In particular, e-commerce sites and entertainment Web services usually offer thousands of different items among which users are invited to find the ones they need or desire the most. In this direction, recommender systems (RSs) have proven to be a silver bullet in suggesting appropriate items to users according to their past choices and behaviors. A typical RS exploits item ratings available in a system, either implicitly or explicitly, to predict a list of unseen items which are of potential interest to the user (Ricci et al., 2011). Over the years, several strategies have been developed to build efficient recommendation algorithms. They are generally divided into three main categories: collaborative filtering (CF) techniques, content-based (CB) methods and hybrid approaches (Burke, 2002). While CF techniques exclusively rely on the feedbacks (rating, click, watch, listen) from the users on specific items without considering their description (either structured or unstructured), on the other side CB ones exploit the data associated to an item to compute relevant recommendations to a user. One of the main issues to tackle in adopting a CB approach is then getting the right amount of meaningful information/data about items, which in turn results necessary to model content-aware items and users descriptions. In fact, by gathering data about items rated by users, one can infer attributes that can be used to model users' profile and preferences.

In the last years, the technological wave related to deep learning techniques and approaches hit also the field of RSs. A variety of new approaches based on different configurations of Neural Networks (NNs) have been proposed to compute personalized lists of items to be suggested to end users (Covington et al., 2016; Wang & Wang, 2014; Elkahky et al., 2015). Among them, autoencoders have been proposed as an interesting tool to mimic the user behavior in producing ratings and by exploiting and modeling user preferences on latent item attributes (Wu et al., 2016). Autoencoders are a particular configuration of artificial NNs which turned out to be very effective especially for dimensionality reduction and feature selection tasks (Vincent et al., 2008). However, autoencoders proved useful also in a different configuration. Indeed, in their mirrored structure, neurons of the hidden layers can be interpreted as a projection of the input layer in a different space. In Bellini et al. (2019), the authors presented SEMAUTO[1], a *Semantics-Aware* autoencoder (SA-autoencoder from now on) which leverages the common graph-based structure to encode the semantics, and the structure of a knowledge graph to enhance the representational power of the underlying NN. One of the main advantages of such a hybrid structure it that of giving an explicit semantics/label to the latent dimensions of the new space (Bellini et al., 2019). In this setting, SA-autoencoders may expand the hidden layer to accommodate all features in the KG relevant for a user.

Among the various and diverse KGs freely available on the Web, for sure DBpedia (Auer et al., 2007) and Wikidata (Vrandecic & Krötzsch, 2014) play a key role due to their encyclopedic nature which makes them the ideal candidates to provide structured descriptions on items in a recommender system (Sacenti et al., 2022; Liu et al., 2023). Although there is a partial overlapping among the information sources to build DBpedia and Wikidata, the data they encode is different under various aspects, such as the amount of data and the way it is organized (Ringler & Paulhein

In this paper we test the addition of a KG to a RS through extensive experiments that variate the following dimensions:

- along a knowledge dimension, we variate the KG between DBpedia and Wikidata;

[1] An implementation is available at: https://github.com/sisinflab/SEMAUTO-2.0

- along the RS model dimension, we test several recommendation models:
  - three state-of-the-art approaches, two of which both with- and without the addition of knowledge (yielding five possible recommendation models)
  - an SA-autoencoder model, in five different configurations;
- along the choice of domain, we conduct experiments in three different ones: books, music, and movies, by employing an appropriate dataset for each one of them.

For all possible RSs built over any combination of the above characteristics, we examine its results in terms of accuracy and diversity, evidencing and discussing how the structure and coverage of information encoded on a KG may affect the accuracy and novelty of the recommendation.
The structure of the paper is as follows: in the next section, we recap how to build SA-autoencoders and their recommendation model. Then, in Sections 2 and 3 we introduce the experimental setting and the different configurations adopted in our investigation, while the results are discussed in Section 4. We then compare our results with related works in Section 5. Conclusions and future work close the paper.

2 Semantics-aware autoencoders in recommendation scenarios

Autoencoders are unsupervised NNs that learn a function capable of reconstructing the network's input at the output layer. They are built on top of two main components which are the encoder and the decoder. The former is usually responsible for compressing the input data into a lower dimensional representation, while the latter does the opposite job reconstructing the original input data starting from a lower dimensional representation (see Fig. 1). Like every NN configuration, autoencoders are structured in layers which contain neurons. Every neuron in layer i is connected through an edge to all the neurons of the following layer $i + 1$. In other words, in its standard configuration we have a fully connected network. In a recommendation scenario, we may use all the items in a catalog as representative of both

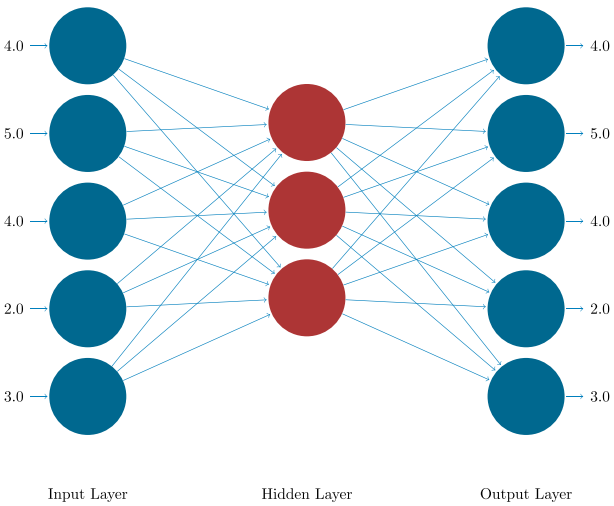


Fig. 1 Architecture of an autoencoder

the input and the output layer. We may then train the NN by using user ratings as inputs to obtain similar values produced by the output layer.

Autoencoders have been studied also in architectures that augment the input-output dimension. Figure 2 depicts an Overcomplete Autoencoder, a deep learning technique with a hidden layer wider than the input layer, offering greater representation capacity than traditional autoencoders (Ranzato et al., 2006; Rifai et al., 2011). This configuration excels both in handling noisy or missing data and in generating new data (Vincent et al., 2008). However, increased computational complexity and overfitting risks may arise. Mitigation strategies include using specialized algorithms and hardware, regularization, and dropout techniques (Ngiam et al., 2011; Lewicki & Sejnowski, 2000).

Denoising Autoencoders (DAE) also feature a wider middle layer and learn compressed, simplified representations of input data (Vincent et al., 2010; Alain & Bengio, 2014). DAEs can extract meaningful features from user interaction datasets or knowledge graphs, improving recommendation algorithm performance and recommendation quality (Liang et al., 2018).

Starting from the observation that both NNs and KGs are directed graphs, in Bellini et al. (2017) the authors propose to use the topology of the latter to model the former, in an original architecture inspired by overcomplete-autoencoders, whose aim is not dimensionality reduction. The main idea is to keep input- and output-layer nodes as representative of the items in the catalog and substitute anonymous nodes in the hidden layers by labeled resources from a knowledge graph thus inheriting their mutual semantic connections (see Fig. 2). Differently from the generic definition of autoencoder, we see here that the resulting NN is no longer fully connected since the input layer's nodes are linked to neurons in the hidden layer if and only if a corresponding connection exists in the original knowledge graph. In such a semantics-aware autoencoder, we somehow project the items in a space whose dimensions represent all the entities (features) they are connected to.

As for a generic autoencoder, also a semantics-aware autoencoder is trained with user ratings and it learns how to reconstruct them on the output layer but differently from the original autoencoder, since such a network is not fully connected, user ratings propagate only through those nodes that represent features connected in the KG to items rated by the user. According to NN models, a generic neuron outputs a value that is a non-linear function

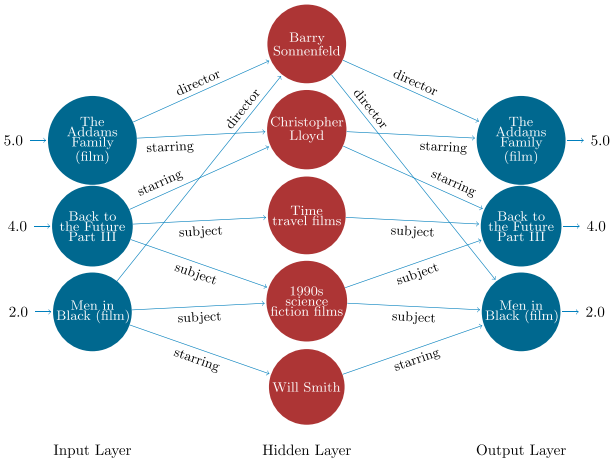


Fig. 2 Architecture of a semantics-aware autoencoder

of the weights summation over incoming edges. Then, in a recommendation scenario it turns out that positively rated items tend to have connected neurons with higher output values than those neurons connected with negatively rated items. As proposed in Bellini et al. (2019), one single autoencoder is trained per each user for 1000 epochs by using the well-known Sigmoid activation function for the hidden layer. The number of neurons in the hidden layer depends on the number of entities (features) encoded in the KG for the user's rated items. We use the autoencoder to describe a single user's ratings in a feature space that is tightly coupled with the KG, so this method, even with few rates, is able to weigh items features according to the user's interest to them. Nevertheless, since the autoencoder is fed with ratings of only one user, the overall approach can be easily parallelized by training each user/autoencoder on an autonomous thread.

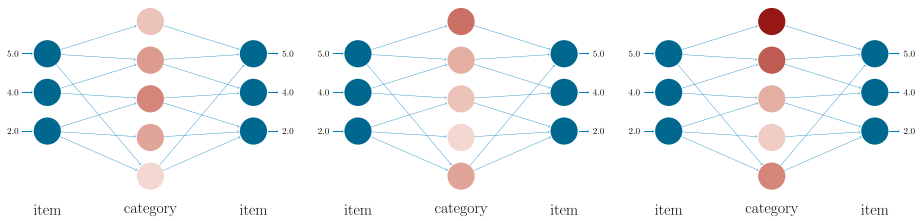
We recap the training process of the semantics-aware autoencoder in Fig. 3. At start, the model is initialized with random weights (Fig. 3a). For each epoch it is fed with user ratings and the back-propagation adjusts weights in order to reconstruct input's user ratings at the output layer (Fig. 3b). The model converges and encodes the relevance for each feature in the hidden layer Fig. (3c).

2.1 User profiles

If we train one SA-autoencoder per user, the resulting model may be interpreted as an explicit representation of the user profile on items attributes. As a matter of fact, at the end of the training, hidden nodes encode a value that represents the relevance for the user in the node's associated feature. Thus, sets of pairs $\langle feature, value \rangle$ can be defined and used as a representation of the user profile in the semantic space. An SA-autoencoder is, therefore, a model that uses deep learning techniques to extract weighted features from a KG according to user ratings in order to build users profiles for recommendation tasks. In particular, given a user u, after the training phase the weight of a feature c is the summation of the weights $w_k^u(c)$ associated to the k edges entering the hidden node representing a KG entity c (see Fig. 4).

More formally, given user u and a feature c, the weight of c for u can be computed as:

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



(a) Model initialization. (b) Training phase. (c) Model converges.

Fig. 3 Training process for a semantics-aware autoencoder for an individual user. The color nuance of a hidden unit denotes the relevance of a feature it represents according to the rating given to items connected to that hidden neuron

where $In(c)$ is the set of the edges entering the hidden node representing the feature c. As an example, if we consider the excerpt of the network in Fig. 4, for `Barry Sonnenfeld` we have:

$$\omega^u(\texttt{Barry Sonnenfeld}) = w^u_{11} + w^u_{12}$$

Having weights associated to each resource coming from KG, we can now model a user profile composed by a vector of weighted features. Given S^u as the set of all the features belonging to the items rated by the user u and $S = \bigcup_{u \in U} S^u$ as the set of all the features in the system, we have that for each user $u \in U$ and for each feature $c \in S$, the user profile $P(u)$ is represented as:

$$P(u) = \{\langle c, \omega \rangle \mid \omega = \omega^u(c) \text{ if } c \in S^u, \omega = 0 \text{ otherwise}\} \tag{1}$$

Because we train a SA for each user, the weights associated with the edges that link items to values indicate how influential that type of $\langle feature, value \rangle$ pair is for the user. Clearly, this value varies from user to user since we have a different SA for each one of them.

2.1.1 Recommendation

The vector computed with (1) is usually very sparse in the space representing the overall number of features. In other words, many values are set to 0 as the corresponding feature does not belong to any item the user rated in the past. Since this may negatively affect the final result in a recommendation scenario, we reduce the sparseness of the user profile by predicting a value to fill 0-valued features in the vector through the word2vec-like approach (Bellini et al., 2018).

Once we have a less sparse version of the user profile, we produce recommendation performing a standard user-kNN (see (2)) by computing how close are users with each other through a cosine similarity (see (3)).

$$\hat{r}(u, i) = \frac{\sum_{j=1}^{k} sim(u, v_j) \cdot r(v_j, i)}{\sum_{j=1}^{k} sim(u, v_j)} \tag{2}$$

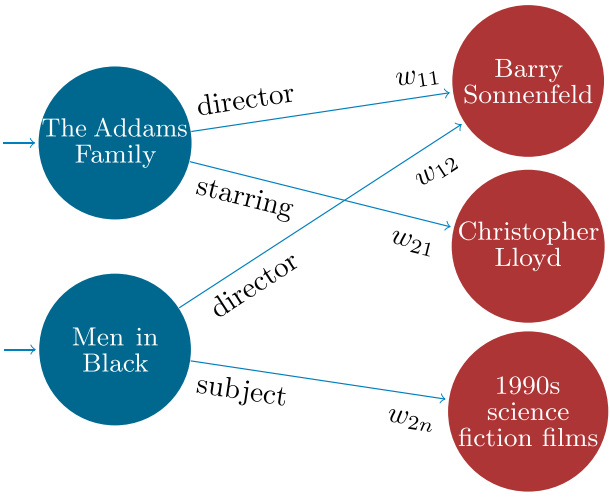


Fig. 4 An excerpt of the network in Fig. 2 after the training

$$sim(u, v) = \frac{P(u) \cdot P(v)}{||P(u)|| \cdot ||P(v)||} \tag{3}$$

In (2), given an unseen item i from user u, we predict a rating $\hat{r}(u, i)$ for u on i by considering the ratings $r(v_j, i)$ assigned to i by the k most similar users v_j.

From a practitioner's perspective, this approach can scale both vertically and horizontally as one can train a single autoencoder on a single core; therefore, SA scales vertically with the number of cores on a single machine and horizontally by deploying multiple machines. Given that we train a single autoencoder for each user, the dimensionality of the overall network depends on the number of single-user interactions; this results in training relatively small networks faster in the training procedure. In industrial settings where the availability of clusters is generous, this becomes a plus since it is easy to scale.

3 Experiments

As a direct consequence of relying on a KG, it stands to reason that its structure, as well as the information it encodes, might affect how user profiles are computed. Thus, it is crucial to investigate how the KGs structure impacts the recommendation accuracy. We may believe that the better the data are engineered and curated, the more accurate a recommendation is. The point is, how to evaluate and measure the recommendation?

Given the recommendation model previously described, we performed an experimental evaluation to verify the influence of a knowledge source in the final recommendation task. In this paper, we chose the two main encyclopedic knowledge graphs freely available on the Web: DBpedia and Wikidata due to the richness of information they encode in different knowledge domains. DBpedia and Wikidata differ from each other not only by their structure but also Ringler and Paulheim (2017) by distinct fields of knowledge are covered in different ways.

We first describe the structure of the datasets used in the experiments, then we move on to the evaluation protocol for the recommendation, and finally we discuss the results.

For the sake of ensuring the reproducibility of our models we provide the link to the public repository from which to retrieve the code and datasets (https://split.to/sa-auto).

3.1 Dataset

We conducted our experiments on three different datasets as summarized in Table 1.

Table 1 Datasets characteristics and relative levels of sparsity

	#users	#items	#ratings	sparsity	#DBpedia-items	#Wikidata-items
MovieLens 1M	6040	3952	1000209	95.81%	3301 (94.70%)	3097 (93.40%)
Last.fm	1892	17632	92834	99.72%	10180 (98.22%)	9742 (96.05%)
LibraryThing	7279	37232	626000	99.77%	11695 (93.47%)	11039 (92.01%)

In addition, for each KGs, we report the number of mapped items and, in parentheses, the sparsity level in the catalog restricted to mapped items and users who rated at least one of them

MovieLens 1M[2] stores information about users-items interactions made on a 5-star scale and relates to the movie domain. Last.fm[3] contains information about music, bands and artists listenings; since in Last.fm for each user we have the number of times a user has listened to a song, we infer users' preferences by scaling it within the range [1, ..., 10] (using min-max normalization). Finally in LibraryThing[4], which is a social web application for book cataloging, rates are made on a 10-star scale with reference to books. By relying on these datasets, we have three different knowledge domains which are covered both by DBpedia and by Wikidata.

In order to map items to resources in DBpedia we adopted a freely available mapping[5] originally presented in Ostuni et al. (2013) and then refined in Anelli et al. (2017). Thus, we retain 3301 mapped items for MovieLens 1M, 10180 for Last.fm and 11695 for LibraryThing. Starting from DBpedia resources URI we obtained Wikidata entities through `owl:sameAs` links.

3.2 Knowledge graphs: DBpedia vs. Wikidata

In addition to a general evaluation on the impact of the usage of DBpedia vs. Wikidata in a recommendation setting, we also evaluated how different kinds of information from a KG impact the recommendation itself. If we look at the semantic knowledge encoded within the two knowledge graphs, we may identify **factual knowledge** where we have facts stated on a specific resource, for example[6]:

```
dbr:Men_in_Black_(Film) dbo:director dbr:Barry_Sonnenfeld .
```

and **ontological** and **categorical knowledge** which encode the semantics of an entity through classes and categories, such as:

```
dbr:Men_in_Black_(Film) dct:subject dbc:Buddy_Film .
dbr:Men_in_Black_(Film) rdf:type dbo:Film .
```

In DBpedia, categorical information is reached through the following predicates:

- http://purl.org/dc/terms/subject
- http://www.w3.org/2009/08/skos-reference/skos.html#broader

The former allows us to explore categorical resources related to an item, while the latter lets us discover a wider category in a hierarchical perspective.

As for Wikidata, we considered categorical information encoded through the predicates:

- https://www.wikidata.org/wiki/Property:P921 labeled as *main subject*
- https://www.wikidata.org/wiki/Property:P136 (whenever possible) labeled as *genre*

Since DBpedia `skos:broader` is not directly mapped in Wikidata, we used https://www.wikidata.org/wiki/Property:P279, labeled as *subclass of*, to identify hierarchical categories as well.

Regarding factual information, we used the approach proposed in Ragone et al. (2017) to automatically identify those DBpedia predicates (listed in Table 2) which turn out to be the most meaningful for a recommendation task; the corresponding Wikidata properties were properly collected through SPARQL queries.

[2] http://grouplens.org/datasets/movielens/

[3] http://www.lastfm.com

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

[5] https://github.com/sisinflab/LODrecsys-datasets

[6] All the prefixes we will use from now on are those available at http://prefix.cc

Table 2 DBpedia factual predicates selected to compute recommendations

LAST.FM	LIBRARYTHING	MOVIELENS 1M
dbo:genre	dbo:author	dbo:starring
dbo:instrument	dbo:literaryGenre	dbo:director
dbo:occupation	dbo:publisher	dbo:writer
dbo:associatedBand	dbo:mediaType	dbo:producer
dbo:associatedMusicalArtist	dbo:language	dbo:musicComposer
dbo:recordLabel	dbo:country	dbo:distributor
dbo:hometown	dbo:previousWork	dbo:language
dbo:birthPlace	dbo:subsequentWork	dbo:cinematography
dbo:country	dbo:nonFictionSubject	dbo:country
dbo:influencedBy	dbo:series	dbo:editing
dbo:voiceType	dbo:coverArtist	dbp:music
dbo:award	dbo:illustrator	dbp:studio
dbo:bandMember	dbo:translator	dbp:extra
dbo:currentMember	dbp:awards	dbp:screenplay
dbo:pastMember	dbp:writer	dbp:genre

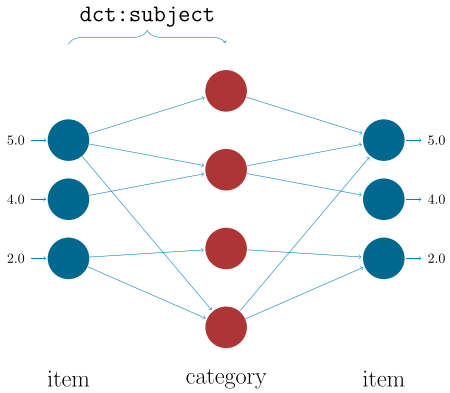
3.3 Data settings

Here we show the different configurations we adopted to inject data from DBpedia and Wikidata in our semantics-aware autoencoder. As stated in Section 2, the input and output layers are always composed by resources representing items of our recommendation setting (i.e., movies for MOVIELENS 1M, books for LIBRARYTHING, songs, bands, etc. for LAST.FM). The differences of the configurations we propose mainly rely on the information encoded in the hidden layers. In Fig. 5 we show only the case for DBpedia, as for Wikidata we have analogous configurations.

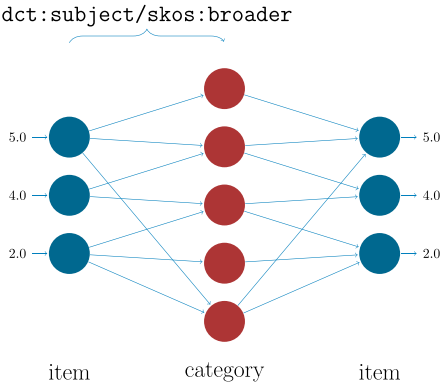
We list below the symbol of each configuration along with its explanation:

- `S` (for *Subject*) the first configuration (Fig. 5a) encodes only categories through the dct:subject property. Hence, all the nodes in the hidden layer have a one-to-one mapping with a corresponding category in DBpedia. We refer to this configuration as *Subject* since categories are structured in a hierarchical way through `skos:broader`;
- `B` (*for Broader*) We considered in the hidden layer only those categories which are one-hop distant through the `skos:broader` property from those resources directly connected to an item via `dct:subject` (Fig. 5b). In this configuration, connections between the items and the hidden layer are represented by the property chain dct:subject/skos:broader;
- `S-B` (*for Subject-Broader*) In this configuration we considered three hidden layers[7] thus mimicking the actual topology of the knowledge graph in the structure of the NN (see Fig. 5c);
- `M` (*for Mixed*) We put in the same hidden layer both the categories connected via `dct:subject` and `dct:subject/skos:broader`. We consider this configuration as flattening of `S-B`;

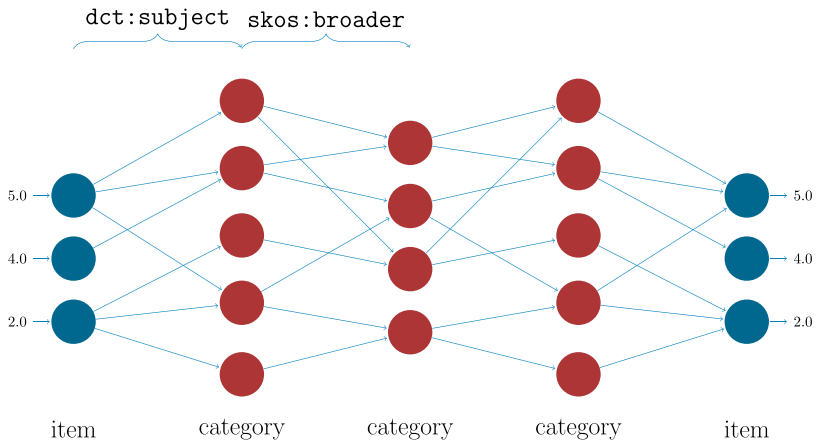
[7] Due to the mirrored structure of an autoencoder, the number of layers is always odd



(a) Categories reached by the `dct:subject` property encoded in the hidden layer.



(b) Categories reached by the `dct:subject/skos:broader` property chain encoded in the hidden layer.



(c) Subjects and broaders distributed among different hidden layers.

Fig. 5 The different configurations used in our experiments

F (*for Factual*) Herein the hidden layer is composed by all the resources which are one-hop far from the input item through the properties in Table 2.

3.4 Evaluation

Before measuring how different KGs impact on recommendation quality, we first prove the strength of the proposed method by comparing it with some state-of-the-art baselines. Then we quantify how our performances vary depending on the different subsets of the KG we use.

For the evaluation of our approach we adopted the "All Unrated Items" protocol described in Steck (2013): 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. Training and test sets are generated by splitting each dataset with Hold-Out 80/20, which ensures that user have 80% of their ratings in the training set and the remaining 20% in the test set. To carry out the experimental phase, we used a server equipped with an i7-7700K processor with 128GB RAM and GeForce GTX 960 GPU. Five full rounds (training and evaluation) were run for each model and configuration, and we reported the results of the best obtained in the individual rounds.

The produced recommendation lists are finally compared with the test set by computing performance metrics: Precision, Recall and F1-score. These metrics have been chosen to evaluate the accuracy of our model in a top-10 recommendation scenario (Cremonesi et al., 2010), using threshold values of 4 for MovieLens 1M and 8 for both LibraryThing and Last.fm.

Accuracy metrics are a valuable way to evaluate the performance of a recommender system. Nonetheless, it has been argued (Smyth & McClave, 2001) that diversity should be considered when evaluating how good a recommendation engine is. Gini index (Shani & Gunawardana, 2011, Formula 8.20) is an ideal candidate to measure the distribution/diversity of items across recommendation lists:

$$Gini = \frac{1}{n-1} \cdot \sum_{j=1}^{n} (2j - n - 1) \cdot p(i_j)$$

where n is the number of items, $p(i_j)$ is the proportion of user choices for item i_j and $i_1, ...i_n$ is the list of items ordered according to increasing $p(i_j)$. A Gini index value equal to 0 means that all items are chosen equally often, while it is 1 if a single item is always chosen.

Among the several state-of-the-art techniques used in recommender scenarios, we tested the most widely adopted: BPRSLIM (Ning & Karypis, 2011; Rendle et al., 2009), WRMF (Pan et al., 2008; Hu et al., 2008) and a single-layer autoencoder for rating prediction. Although they have been proposed a few years ago, BPRSLIM and WRMF have recently shown to have excellent performances compared to deep-learning based approaches (Dacrema et al., 2019). BPRSLIM is a Sparse Linear Method which leverages Bayesian Personalized Ranking as an objective function. WRMF is a Weighted Regularized Matrix Factorization method which exploits users' implicit feedbacks to provide recommendations. In their basic version, both strategies rely exclusively on the User-Item matrix in a pure CF approach. They can be hybridized by exploiting side information (SI) (Ning & Karypis, 2011), i.e., additional data associated with items. We used the implementations of BPRSLIM and WRMF available in MyMediaLite[8] (Gantner et al., 2011) and both our SA-autoencoder and the *classic* autoencoder are implemented with TensorFlow[9].

4 Discussion of the results

For all the recommendation models described, and for both KG, we performed experiments on three datasets: MovieLens 1M, Last.fm and LibraryThing, related to movies, music, and books domains, respectively.

[8] http://mymedialite.net

[9] https://www.tensorflow.org/

In Table 3, we present the most favorable outcomes we have obtained across three datasets by implementing the methods discussed earlier, with a specific emphasis on the level of sparsity. Regarding the SA-Autoencoder approach, we conducted tests with various numbers of neighbors (k), and we have included only the optimal results in the table. Notably, we have highlighted the overall best-performing approach in bold, while we have underscored the most effective configuration for the SA-autoencoder. We initiate our discussion by evaluating the results in terms of accuracy before delving into the aspect of diversity. Examining the table reveals that the semantics-aware autoencoder surpasses all baseline methods in the LAST.FM dataset. Conversely, in the case of the LIBRARYTHING and MOVIELENS 1M datasets, its performance closely mirrors that of the fully-connected autoencoder. To elucidate this outcome, we can posit that the semantics-aware autoencoder's performance is contingent upon the quantity of features retrieved from a knowledge graph (KG). A greater number of features corresponds to a larger number of neurons in the hidden layers. According to Hornik's Universal Approximation Theorem, a neural network with more neurons possesses a superior capacity to approximate any given function. Consequently, as indicated in Table 4, we have computed the ratio of features associated with items, denoted as $\frac{average\#features}{average\#items}$. Our findings reveal that, considering each pair of KGs (DBPedia-WikiData) for a given setting (S-B-F...), the KG with the higher ratio yields the better results on performance It is imperative to emphasize that this measure also serves as a gauge of connectivity in SA-autoencoders. Specifically, lower values of $\frac{average\#features}{average\#items}$ indicate a sparse connectivity with fewer connections, while higher values signify a denser connectivity (Pujara et al., 2017).

Nevertheless, when examining the sparsity levels presented in the final column of Table 1, it becomes evident that, despite the LIBRARYTHING and LAST.FM datasets having nearly identical sparsity values, they exhibit substantial disparities in their average features per item ratios. Specifically, the SA-Autoencoder demonstrates superior performance on the LAST.FM dataset, characterized by a higher ratio, while its performance is notably inferior on the LIBRARYTHING dataset, where the ratio is lower. Consequently, it is reasonable to infer that this method exhibits a relatively lower sensitivity to dataset sparsity in comparison to the quality of data curation within the Knowledge Graph (KG). Consequently, the comparison between the outcomes of the SA-Autoencoder and the Autoencoder can serve as a valuable means of evaluating data quality within KGs, particularly in the context of recommendation tasks.

On the other hand, we can observe that factual information (F) brings more accurate results for a recommendation task than ontological/categorical knowledge when the datasets are very sparse (as for MOVIELENS 1M and LIBRARYTHING). This is a quite interesting result. A possible explanation is that categorical information introduces fewer connections among item descriptions than factual information. Hence, factual statements result more useful in making denser connections among items (and then users), which are eventually exploited by the latent collaborative part of the semantics-aware autoencoder.

If we just focus on the absolute numbers, one may argue that SA-Autoencoder is not competitive as it is slightly beaten in terms of accuracy by state-of-the-art algorithms such as BPRSLIM and WRMF (although it is the second best performing approach). Nevertheless, we point out that, differently from the other approaches based on matrix factorization (or any deep learning technique) with SA-Autoencoder we compute a meaningful and explicit user profile which contains user preferences on single features. This may result extremely useful in case we want to automatically generate a content-based explanation for the ranking computed with the recommendation list as also shown in Bellini et al. (2018). Then, although we rely

Table 3 Experimental results over DBpedia and Wikidata KGs

	setting	KG	k	F1@10	Prec@10	Recall@10	1−Gini
LAST.FM							
AUTOENCODER	–	–	–	0.00048	0.00027	0.00240	**0.00190**
BPRSLIM	–	–	–	0.00077	0.00043	0.00400	0.02867
BPRSLIM + SI	–	DBpedia	–	0.00113	0.00064	0.00476	0.05360
	–	Wikidata	–	0.00107	0.00039	0.00393	0.02479
WRMF	–	–	–	0.00077	0.00043	0.00400	0.01073
WRMF + SI	–	DBpedia	–	0.00058	0.00032	0.00293	0.00877
	–	Wikidata	–	0.00050	0.00027	0.00132	0.00701
SA-AUTOENCODER	S	DBpedia	5	0.00151	0.00085	0.00644	0.05783
	S	Wikidata	10	0.00124	0.00069	0.00587	0.03815
	B	DBpedia	10	0.00113	0.00064	0.00484	0.03812
	B	Wikidata	30	0.00086	0.00048	0.00391	0.01430
	M	DBpedia	5	0.00151	0.00085	0.00644	0.05783
	M	Wikidata	5	0.00067	0.00037	0.00320	0.05317
	S-B	DBpedia	5	0.00111	0.00064	0.00422	0.05624
	S-B	Wikidata	5	**0.00172**	**0.00096**	**0.00844**	0.05742
	F	DBpedia	5	0.00169	0.00096	0.00689	0.05878
	F	Wikidata	10	0.00143	0.00080	0.00693	0.03689
LIBRARYTHING							
AUTOENCODER	–	–	–	0.01562	0.01375	0.01808	0.07628
BPRSLIM	–	–	–	0.01874	0.01577	**0.02309**	0.09338
BPRSLIM + SI	–	DBpedia	–	**0.01939**	**0.01685**	0.02284	0.17915
	–	Wikidata	–	0.01715	0.01502	0.01894	0.07915
WRMF	–	–	–	0.01142	0.01071	0.01223	0.00864
WRMF + SI	–	DBpedia	–	0.01136	0.01043	0.01247	0.00832
	–	Wikidata	–	0.01003	0.00975	0.01021	0.00922
SA-AUTOENCODER	S	DBpedia	100	0.01293	0.01168	0.01447	0.01855
	S	Wikidata	100	0.00993	0.00888	0.01125	0.01124
	B	DBpedia	45	0.01264	0.01139	0.01420	0.02475
	B	Wikidata	100	0.00791	0.00741	0.00848	0.00983
	M	DBpedia	45	0.01299	0.01164	0.01469	0.02938
	M	Wikidata	150	0.00867	0.00805	0.00941	**0.00770**
	S-B	DBpedia	100	0.01390	0.01247	0.01570	0.01205
	S-B	Wikidata	100	0.00995	0.00892	0.01123	0.00950
	F	DBpedia	40	0.01468	0.01306	0.01677	0.02888
	F	Wikidata	45	0.01278	0.01153	0.01435	0.02237
MOVIELENS 1M							
AUTOENCODER	–	–	–	**0.22969**	**0.28416**	0.19274	0.04536
BPRSLIM	–	–	–	0.17106	0.19581	0.15187	0.14060
BPRSLIM + SI	–	DBpedia	–	0.14986	0.17113	0.13329	0.17294
	–	Wikidata	–	0.14163	0.16872	0.12899	0.01644

Table 3 continued

	setting	KG	k	F1@10	Prec@10	Recall@10	1−Gini
WRMF	–	–	–	0.20336	0.25343	0.16981	0.03758
WRMF + SI	–	DBpedia	–	0.20373	0.25371	0.17020	0.03750
	–	Wikidata	–	0.19003	0.24714	0.15361	0.02105
SA-AUTOENCODER	S	DBpedia	50	0.18582	0.22419	0.15867	0.02298
	S	Wikidata	100	0.16809	0.21619	0.13749	0.01712
	B	DBpedia	45	0.17640	0.21369	0.15019	0.02207
	B	Wikidata	100	0.15487	0.20555	0.12424	0.01611
	M	DBpedia	45	0.18633	0.22430	0.15935	0.02421
	M	Wikidata	100	0.15592	0.20046	0.12757	**0.01378**
	S-B	DBpedia	50	0.22001	0.26616	0.18749	0.03653
	S-B	Wikidata	100	0.15800	0.20394	0.12896	0.01574
	F	DBpedia	50	0.22447	0.26788	**0.19317**	0.04446
	F	Wikidata	50	0.17150	0.21149	0.14423	0.01872

For BPRSLIM + SI and WRMF + SI in KG column, we indicate only the KG for which we have the best performance when used to get side information. Best column values are in bold, best column values among SA-Autoencoder configuration underlined. Complete results about *SA-AUTOENCODER* are reported here: https://split.to/sa-auto-res

on a deep learning approach, we can go beyond the pure black-box method and provide a human-understandable explanation for a recommendation list.

As for diversity, we can observe that when using Wikidata, values for Gini index are in general lower than the DBpedia case. This means that using Wikidata we are able to better diversify the items in a catalog. This result somehow reinforces the one obtained in Nguyen et al. (2015). Lack of diversity in recommendation strongly depends on the so-called "popularity bias": popular items tend to be recommended more than those in the long tail. This observation leads us to a possible interpretation on diversity results we obtain in our experiments if we consider the popularity of entities in a KG as the number of connections they have. In DBpedia we have that popular resources (e.g. movies) are more connected to other nodes than unpopular ones. This is not the case with Wikidata where there is a less biased distribution of connections among resources in the graph. Hence, when the adopted KG suffers from a popularity bias in terms of connections among resources, this is inherited by the recommendation dataset, thus affecting the final recommended list of results.

5 Related works

Very few works exist about qualitative studies of Linked Open Data (LOD) knowledge graphs. In Ringler and Paulheim (2017) present an analysis of main KGs such as DBpedia, Wikidata, YAGO, underlining their differences in coverage, identifying overlapping and complementary parts of KGs. They assert that KGs are not easily interchangeable and each of them has its strengths and weaknesses for a domain-related task. Thus, using a specific KG that is suitable for the task to accomplish leads to better performances of the overall system. Authors in this work made a category-specific analysis, asserting that even if DBpedia and YAGO come from the same source (Wikipedia) and have a quite similar number of instances, there are

Table 4 Summary of hidden units over DBpedia and Wikidata KGs using both factual and semantics information

	setting	KG	average #features	$\frac{\text{average \#features}}{\text{average \#items}}$
		LAST.FM		
SA-AUTOENCODER	S	DBpedia	374.33	11.86
	S	Wikidata	38.48	1.26
	B	DBpedia	38.48	20.36
	B	Wikidata	17.98	0.59
	M	DBpedia	974.61	30.87
	M	Wikidata	47.95	1.57
	S-B	DBpedia	2865.35	90.76
	S-B	Wikidata	246.66	8.08
	F	DBpedia	297.20	9.41
	F	Wikidata	180.67	5.92
		LIBRARYTHING		
SA-AUTOENCODER	S	DBpedia	206.83	1.96
	S	Wikidata	28.33	0.35
	B	DBpedia	395.75	3.75
	B	Wikidata	30.96	0.38
	M	DBpedia	572.22	5.43
	M	Wikidata	52.59	0.66
	S-B	DBpedia	2578.11	24.45
	S-B	Wikidata	426.95	5.35
	F	DBpedia	137.08	1.30
	F	Wikidata	110.15	1.38
		MOVIELENS 1M		
SA-AUTOENCODER	S	DBpedia	1102.46	8.38
	S	Wikidata	100.01	0.76
	B	DBpedia	1499.39	11.39
	B	Wikidata	82.60	0.63
	M	DBpedia	2386.48	18.14
	M	Wikidata	167.34	1.27
	S-B	DBpedia	11245.00	85.46
	S-B	Wikidata	1279.82	9.73
	F	DBpedia	1097.76	8.34
	F	Wikidata	2278.99	17.32

In setting column we denote: S = subjects, B = broaders, M = merge S and B in a single hidden layer, S-B = S and B in multiple hidden layers, F = factual information

notable differences in coverage. YAGO has five times the number of events of DBpedia, while DBpedia has four times as many settlements (i.e., cities and town) as YAGO; but Wikidata contains twice as many persons as DBpedia and YAGO. They conclude their investigation by providing a coverage summarization for some popular classes. A comparative survey of some popular KGs is done in Färber et al. (2018) in which authors propose a method to

find the most suitable KG for a given task setting. To achieve this result, authors identify a set of characteristics they found relevant to describe a KG and then compare different KGs accordingly. Furthermore, in Färber et al. (2018) they provide a more detailed analysis of quantitative information stored in KGs by using several statistics such as the number of triples and classes, distribution of classes and corresponding instances, domain and classes coverage. Finally, to select the KG that best fits task requirements, a novel method that takes into account KG's quantitative assessment is proposed.

In the last few years, thanks to increasing computational resources, new techniques based on deep learning have been successfully adopted in the recommendation scenario (Cheng et al., 2016). These techniques led to the development of different neural network models, each yielding interesting results (Singhal et al., 2017). In particular, some of these models turned out to be more effective for a specific recommendation task than others; e.g. autoencoders have proved their strength in collaborative filtering, outperforming state-of-the-art approaches (Sedhain et al., 2015), while Recurrent Neural Networks seem to be more suitable in session-based recommendation (Hidasi et al., 2016). Among the several autoencoders extensions, denoising autoencoders have been efficiently used to address the recommendation problem by improving users' profile learning (Wu et al., 2016) or getting a smaller and non-linear representation of the User-Item rating matrix (Strub et al., 2016). Furthermore, Dong et al. (2017) shows how to build a hybrid RS by integrating side information in CF deep learning techniques, alleviating the sparsity problem and improving all system performances (Dacrema et al., 2019).

LOD are increasingly adopted in recommender systems because they provide more complex structured data that leverages relationships among entities in the graph. Moreover, they encode somehow semantics behind the data (Di Noia et al., 2012; de Gemmis et al., 2015). Recent works leveraged the data encoded in KGs to represent items thus achieving interesting results in recommendation scenarios (Oramas et al., 2017; Di Noia et al., 2016, 2012). Other approaches, as the one proposed by Ostuni et al. (2014) uses kernel graphs to compute item similarities by matching their local neighborhood graphs. LODs have been also exploited for measuring semantic distances between resources in order to provide top-N recommendations (Piao & Breslin, 2016). In Bellini et al. (2017) presented a novel method that combines both deep learning techniques and KGs, in which a semantics-aware neural network that explicitly computes user profiles for recommendation tasks is modeled. In particular, they focus on cold-start scenarios using DBpedia as a source of information for both user- and item modeling. In Bellini et al. (2018), authors used the aforementioned method to perform experiments in recommendation scenarios by using DBpedia KG on three different datasets and they compared their approach with state-of-the-art algorithms. Furthermore, they investigated the effectiveness of their method that leverages on a KG to provide an explanation in recommendation scenarios (Bellini et al., 2018). Another approach that uses KGs to represent the relationship between users and items is investigated in Ristoski et al. (2019), in which the authors leveraged on language models to extract features from node sequences in RDF graphs. Recently, there has also been much interest in the topic of *KG completion*, whose aim is to infer hidden relationships between entities in a KG. Taking advantage of the progress made in this area, the authors of KTUP (Cao et al., 2019) have proposed to exploit KG completion methods with respect to users' preferences towards items in a catalogue in order to jointly train a model that exploits both the interactions between users and items and the characteristics of the items in a KG.

6 Conclusions and future work

In this paper, we showed how it is possible to combine the computational predictive power of NNs (in the form of autoencoders) with the representational power of KGs such as DBpedia and Wikidata. We found that the choice of the KG affects the results of a KG-aware approach to recommendation, and that some combinations of KG, recommendation model, and task domain, yield better recommendations than others. In particular, we evaluated an SA-autoencoder on different configurations for DBpedia and Wikidata, and we tested and compared its results in terms of accuracy and diversity with other recommendation models used as baselines, in three different recommendation domains. We showed that the selection of the right information from the right KG may heavily affect recommendations both in terms of accuracy and in terms of diversity of results.

As a prominent example, Wikidata seems to be a better choice than DBpedia if we look for recommendation lists where the popularity bias is mitigated: our results show that Wikidata allows the SA-autoencoder to tackle the popularity bias and to better diversify the items in a catalog, by recommending also less popular items.

As for future work, we are developing a new version of the SA-autoencoder – one that better exploits collaborative information by building a global model encoding preferences from all users at the same time, as done by state-of-the-art CF approaches – with the aim of improving the experiments presented here.

Acknowledgements This work was partially supported by the following projects: Secure Safe Apulia, Casa delle Tecnologie Emergenti Comune di Matera, CT_FINCONS_III "Smart Rights Management Platform", LUTECH Digitale 4.0.

Author Contributions Vito Bellini, Angelo Schiavone have conceptualized and developed the framework, have written the first draft of the work presented here. Claudio Pomo, Francesco M. Donini have reworked the first draft, inserting details and contextualizing more recent literature. They revised the discussion of the results. Azzurra Ragone, Eugenio Di Sciascio supervised and reviewed the submitted work

Funding Open access funding provided by Politecnico di Bari within the CRUI-CARE Agreement.

Availability of supporting data The datasets generated during and/or analysed during the current study are available in the *SEMAUTO-2.0* repository, https://github.com/sisinflab/SEMAUTO-2.0.

Declarations

Ethical Approval Not Applicable.

Competing interests The authors declare no competing interests.

References

Alain, G., & Bengio, Y. (2014). What regularized auto-encoders learn from the data-generating distribution. *Journal of Machine Learning Research, 15*(1), 3563–3593. https://doi.org/10.5555/2627435.2750359

Anelli, V. W., Di Noia, T., Lops, P., et al. (2017). Feature factorization for top-n recommendation: From item rating to features relevance. In Y. Zheng, W. Pan, S. S. Sahebi, et al. (Eds.), *Proceedings of the 1st Workshop on Intelligent Recommender Systems by Knowledge Transfer & Learning co-located with ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, CEUR Workshop Proceedings, vol. 1887* (pp. 16–21). CEUR-WS.org. Accessed 27 Aug 2017. https://ceur-ws.org/Vol-1887/paper3.pdf.

Auer, S., Bizer, C., Kobilarov, G., et al. (2007) Dbpedia: A nucleus for a web of open data. In K. Aberer, K. Choi, N. F. Noy, et al. (Eds.), *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, Lecture Notes in Computer Science, vol. 4825* (pp. 722–735). Springer. Accessed 11-15 Nov 2007. https://doi.org/10.1007/978-3-540-76298-0_52.

Bellini, V., Anelli, V. W., Noia, T. D., et al. (2017). Auto-encoding user ratings via knowledge graphs in recommendation scenarios. In B. Hidasi, A. Karatzoglou, O. S. Shalom, et al. (Eds.), *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2017, Como, Italy* (pp. 60–66). ACM. Accessed 27 Aug 2017. https://doi.org/10.1145/3125486.3125496.

Bellini, V., Schiavone, A., Di Noia, T., et al. (2018). Computing recommendations via a knowledge graph-aware autoencoder. In V. W. Anelli, T. D. Noia, P. Lops, et al. (Eds.), *Proceedings of the Workshop on Knowledge-aware and Conversational Recommender Systems 2018 co-located with 12th ACM Conference on Recommender Systems, KaRS@RecSys 2018, Vancouver, Canada, CEUR Workshop Proceedings, vol. 2290* (pp. 9–15). CEUR-WS.org. Accessed 7 Oct 2018. https://ceur-ws.org/Vol-2290/kars2018_paper3.pdf.

Bellini, V., Schiavone, A., Di Noia, T., et al. (2018). Knowledge-aware autoencoders for explainable recommender systems. In B. Hidasi, A. Karatzoglou, O. S. Shalom, et al. (Eds.), *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2018, Vancouver, BC, Canada* (pp. 24–31). ACM. Accessed 6 Oct 2018. https://doi.org/10.1145/3270323.3270327.

Bellini, V., Di Noia, T., Di Sciascio, E., et al. (2019). Semantics-aware autoencoder. *IEEE Access, 7*, 166122–166137. https://doi.org/10.1109/ACCESS.2019.2953308

Burke, R. D. (2002). Hybrid recommender systems: Survey and experiments. *User Model User Adapt Interact, 12*(4), 331–370. https://doi.org/10.1023/A:1021240730564

Cao, Y., Wang, X., He, X., et al. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In L. Liu, R. W. White, A. Mantrach, et al. (Eds.), *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA* (pp. 151–161). ACM. Accessed 13-17 May 2019. https://doi.org/10.1145/3308558.3313705.

Cheng, H., Koc, L., Harmsen, J., et al. (2016). Wide & deep learning for recommender systems. In A. Karatzoglou, B. Hidasi, D. Tikk, et al. (Eds.), *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA* (pp. 7–10). ACM. Accessed 15 Sept 2016. https://doi.org/10.1145/2988450.2988454.

Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for youtube recommendations. In S. Sen, W. Geyer, J. Freyne, et al. (Eds.), *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA* (pp. 191–198). ACM. Accessed 15-19 Sept 2016. https://doi.org/10.1145/2959100.2959190.

Cremonesi, P., Koren, Y., & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In X. Amatriain, M. Torrens, P. Resnick, et al. (Eds.), *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain* (pp. 39–46). ACM. Accessed 26-30 Sept 2010. https://doi.org/10.1145/1864708.1864721.

Dacrema, M. F., Cremonesi, P., & Jannach, D. (2019). Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In T. Bogers, A. Said, P. Brusilovsky, et al. (Eds.), *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark* (pp. 101–109). ACM. Accessed 16-20 Sept 2019. https://doi.org/10.1145/3298689.3347058.

de Gemmis, M., Lops, P., Musto, C., et al. (2015). Semantics-aware content-based recommender systems. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender Systems Handbook* (p. 119–159). Springer. https://doi.org/10.1007/978-1-4899-7637-6_4.

Di Noia, T., Mirizzi, R., Ostuni, V. C., et al. (2012). Linked open data to support content-based recommender systems. In V. Presutti, & H. S. Pinto (Eds.), *I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria* (pp. 1–8). ACM. Accessed 5-7 Sept 2012. https://doi.org/10.1145/2362499.2362501.

Di Noia, T., Ostuni, V. C., Tomeo, P., et al. (2016). Sprank: Semantic path-based ranking for top-*N* recommendations using linked open data. *ACM Transactions on Intelligent Systems and Technology, 8*(1), 9:1-9:34. https://doi.org/10.1145/2899005

Dong, X., Yu, L., Wu, Z., et al. (2017). A hybrid collaborative filtering model with deep structure for recommender systems. In S. Singh, & Markovitch S. (Eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, California, USA* (pp. 1309–1315). AAAI Press. Accessed 4-9 Feb 2017. https://doi.org/10.1609/AAAI.V31I1.10747.

Elkahky, A. M., Song, Y., & He, X. (2015). A multi-view deep learning approach for cross domain user modeling in recommendation systems. In A. Gangemi, S. Leonardi, & A. Panconesi (Eds.), *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy* (pp. 278–288). ACM. Accessed 18-22 May 2015. https://doi.org/10.1145/2736277.2741667.

Färber, M., Bartscherer, F., Menne, C., et al. (2018). Linked data quality of dbpedia, freebase, opencyc, wikidata, and YAGO. *Semantic Web, 9*(1), 77–129. https://doi.org/10.3233/SW-170275

Gantner, Z., Rendle, S., Freudenthaler, C., et al. (2011). Mymedialite: A free recommender system library. In B. Mobasher, R. D. Burke, D. Jannach, et al. (Eds.), *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA* (pp. 305–308). ACM. Accessed 23-27 Oct 2011. https://doi.org/10.1145/2043932.2043989.

Hidasi, B., Karatzoglou, A., Baltrunas, L., et al. (2016). Session-based recommendations with recurrent neural networks. In Y. Bengio, & Y. LeCun (Eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, Conference Track Proceedings*. Accessed 2-4 May 2016 http://arxiv.org/abs/1511.06939.

Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008) Pisa, Italy* (pp. 263–272). IEEE Computer Society. Accessed 15-19 Dec 2008. https://doi.org/10.1109/ICDM.2008.22.

Lewicki, M. S., & Sejnowski, T. J. (2000). Learning overcomplete representations. *Neural Computation, 12*(2), 337–365. https://doi.org/10.1162/089976600300015826

Liang, D., Krishnan, R. G., Hoffman, M. D., et al. (2018). Variational autoencoders for collaborative filtering. In P. Champin, F. Gandon, M. Lalmas, et al. (Eds.), *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France* (pp. 689–698). ACM. Accessed 23-27 April 2018. https://doi.org/10.1145/3178876.3186150.

Liu, Y., Xuan, H., & Li, B. (2023). Bi-knowledge views recommendation based on user-oriented contrastive learning. *Journal of Intelligent Information System, 61*(2), 611–630. https://doi.org/10.1007/S10844-023-00778-0

Ngiam, J., Khosla, A., Kim, M., et al. (2011). Multimodal deep learning. In L. Getoor, & T. Scheffer (Eds.), *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA* (pp. 689–696). Omnipress. Accessed 28 June - 2 July 2011. https://icml.cc/2011/papers/399_icmlpaper.pdf

Nguyen, P. T., Tomeo, P., Di Noia, T., et al. (2015). Content-based recommendations via dbpedia and freebase: A case study in the music domain. In M. Arenas, Ó. Corcho, E. Simperl, et al. (Eds.), *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, Proceedings, Part I, Lecture Notes in Computer Science, vol. 9366* (pp. 605–621). Springer. Accessed 11-15 Oct 2015. https://doi.org/10.1007/978-3-319-25007-6_35.

Ning, X., & Karypis, G. (2011). SLIM: Sparse linear methods for top-n recommender systems. In D. J. Cook, J. Pei, W. Wang, et al. (Eds.), *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada* (pp. 497–506). IEEE Computer Society. Accessed 11-14 Dec 2011. https://doi.org/10.1109/ICDM.2011.134.

Oramas, S., Ostuni, V. C., Noia, T. D., et al. (2017). Sound and music recommendation with knowledge graphs. *ACM Transactions on Intelligent Systems and Technology, 8*(2), 21:1-21:21. https://doi.org/10.1145/2926718

Ostuni, V. C., Di Noia, T., Di Sciascio E., et al. (2013). Top-n recommendations from implicit feedback leveraging linked open data. In Q. Yang, I. King, Q. Li, et al. (Eds.), *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China* (pp. 85–92). ACM. Accessed 12-16 Oct 2013. https://doi.org/10.1145/2507157.2507172.

Ostuni, V. C., Noia, T. D., Mirizzi, R., et al. (2014). A linked data recommender system using a neighborhood-based graph kernel. In M. Hepp, & Y. Hoffner (Eds.), *E-Commerce and Web Technologies - 15th International Conference, EC-Web 2014, Munich, Germany. Proceedings, Lecture Notes in Business Information Processing, vol. 188* (pp. 89–100). Springer. Accessed 1-4 Sept 2014. https://doi.org/10.1007/978-3-319-10491-1_10

Pan, R., Zhou, Y., Cao, B., et al. (2008). One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), Pisa, Italy* (pp. 502–511). IEEE Computer Society. Accessed 15-19 Dec 2008. https://doi.org/10.1109/ICDM.2008.16.

Piao, G., & Breslin, J. G. (2016). Measuring semantic distance for linked open data-enabled recommender systems. In S. Ossowski (Ed.), *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy* (pp. 315–320). ACM. Accessed 4-8 April 2016. https://doi.org/10.1145/2851613.2851839.

Pujara, J., Augustine, E., & Getoor, L. (2017). Sparsity and noise: Where knowledge graph embeddings fall short. In M. Palmer, R. Hwa, & S. Riedel (Eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark* (pp. 1751–1756). Association for Computational Linguistics. Accessed 9-11 Sept 2017. https://doi.org/10.18653/V1/D17-1184.

Ragone, A., Tomeo, P., Magarelli, C., et al. (2017). Schema-summarization in linked-data-based feature selection for recommender systems. In A. Seffah, B. Penzenstadler, C. Alves, et al. (Eds.), *Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco* (pp. 330–335). ACM. Accessed 3-7 April 2017. https://doi.org/10.1145/3019612.3019837.

Ranzato, M., Poultney, C. S., Chopra, S., et al. (2006) Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. C. Platt, & T. Hofmann (Eds.), *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada* (pp. 1137–1144). MIT Press. Accessed 4-7 Dec 2006. https://proceedings.neurips.cc/paper/2006/hash/87f4d79e36d68c3031ccf6c55e9bbd39-Abstract.html.

Rendle, S., Freudenthaler, C., Gantner, Z., et al. (2009). BPR: Bayesian personalized ranking from implicit feedback. In J. A. Bilmes, & A. Y. Ng (Eds.), *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada* (pp. 452–461). AUAI Press. Accessed 18-21 June 2009. https://www.auai.org/uai2009/papers/UAI2009_0139_48141db02b9f0b02bc7158819ebfa2c7.pdf.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, et al. (Eds.), *Recommender Systems Handbook* (p. 1–35). Springer. https://doi.org/10.1007/978-0-387-85820-3_1.

Rifai, S., Vincent, P., Muller, X., et al. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In L. Getoor, & T. Scheffer (Eds.), *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA* (pp. 833–840). Omnipress. Accessed 28 June - 2 July 2011. https://icml.cc/2011/papers/455_icmlpaper.pdf.

Ringler, D., & Paulheim, H. (2017). One knowledge graph to rule them all? Analyzing the differences between dbpedia, yago, wikidata & co. In G. Kern-Isberner, J. Fürnkranz, & M. Thimm (Eds.), *KI 2017: Advances in Artificial Intelligence - 40th Annual German Conference on AI, Dortmund, Germany, Proceedings, Lecture Notes in Computer Science, vol. 10505* (pp. 366–372). Springer. Accessed 25-29 Sept 2017. https://doi.org/10.1007/978-3-319-67190-1_33.

Ristoski, P., Rosati, J., Di Noia, T., et al. (2019). Rdf2vec: RDF graph embeddings and their applications. *Semantic Web, 10*(4), 721–752. https://doi.org/10.3233/SW-180317

Sacenti, J. A. P., Fileto, R., & Willrich, R. (2022). Knowledge graph summarization impacts on movie recommendations. *Journal of Intelligent Information System, 58*(1), 43–66. https://doi.org/10.1007/S10844-021-00650-Z

Sedhain, S., Menon, A. K., Sanner, S., et al. (2015). Autorec: Autoencoders meet collaborative filtering. In A. Gangemi, S. Leonardi, & A. Panconesi (Eds.) *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy - Companion Volume* (pp. 111–112). ACM. Accessed 18-22 May 2015. https://doi.org/10.1145/2740908.2742726.

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, et al. (Eds.), *Recommender Systems Handbook* (p. 257–297). Springer. https://doi.org/10.1007/978-0-387-85820-3_8.

Singhal, A., Sinha, P., & Pant, R. (2017). Use of deep learning in modern recommendation system: A summary of recent works. *International Journal of Computer Applications, 180*(7), 17–22. https://doi.org/10.5120/ijca2017916055

Smyth, B., & McClave, P. (2001). Similarity vs. diversity. In D. W. Aha, & I. D. Watson (Eds.) *Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, Proceedings, Lecture Notes in Computer Science, vol. 2080* (pp. 347–361). Springer. Accessed 30 July - 2 August 2001. https://doi.org/10.1007/3-540-44593-5_25.

Steck, H. (2013). Evaluation of recommendations: Rating-prediction and ranking. In Q. Yang, I. King, Q. Li, et al. (Eds.), *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China* (pp. 213–220). ACM. Accessed 12-16 Oct 2013. https://doi.org/10.1145/2507157.2507160.

Strub, F., Gaudel, R., & Mary, J. (2016). Hybrid recommender system based on autoencoders. In A. Karatzoglou, B. Hidasi, D. Tikk, et al. (Eds.), *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA* (pp. 11–16). ACM. Accessed 15 Sept 2016 https://doi.org/10.1145/2988450.2988456.

Vincent, P., Larochelle, H., Bengio, Y., et al. (2008). Extracting and composing robust features with denoising autoencoders. In W. W. Cohen, A. McCallum, & S. T. Roweis (Eds.), *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, ACM International Conference Proceeding Series, vol. 307* (pp. 1096–1103). ACM. Accessed 5-9 June 2008. https://doi.org/10.1145/1390156.1390294.

Vincent, P., Larochelle, H., Lajoie, I., et al. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research, 11*, 3371–3408. https://doi.org/10.5555/1756006.1953039

Vrandecic, D., & Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Communications of the ACM, 57*(10), 78–85. https://doi.org/10.1145/2629489

Wang, X., & Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. In K. A. Hua, Y. Rui, R. Steinmetz, et al. (Eds.), *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA* (pp. 627–636). ACM. Accessed 03 - 07 Nov 2014. https://doi.org/10.1145/2647868.2654940.

Wu, Y., DuBois, C., Zheng, A. X., et al. (2016). Collaborative denoising auto-encoders for top-n recommender systems. In P. N. Bennett, V. Josifovski, J. Neville, et al. (Eds.), *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA* (pp. 153–162). ACM. Accessed 22-25 Feb 2016. https://doi.org/10.1145/2835776.2835837.

Authors and Affiliations

Vito Bellini[1] · Eugenio Di Sciascio[2] · Francesco Maria Donini[3] · Claudio Pomo[2] · Azzurra Ragone[4] · Angelo Schiavone[2]

Eugenio Di Sciascio
eugenio.disciascio@poliba.it

Francesco Maria Donini
donini@unitus.it

Azzurra Ragone
azzura.ragone@uniba.it

Angelo Schiavone
angelo.schiavone@poliba.it

1 Amazon Music ML, Berlin, Germany

2 Politecnico di Bari, Bari, Italy

3 Università della Tuscia, Viterbo, Italy

4 Università degli Studi di Bari, Bari, Italy