Exploiting Neuro-Symbolic Graph Embeddings based on First-Order Logical Rules for Knowledge-aware Recommendations

Giuseppe Spillo[1], Cataldo Musto[1], Pasquale Lops[1], Marco de Gemmis[1] and Giovanni Semeraro[1]

[1]*University of Bari Aldo Moro, Bari, Italy*

Abstract

In this paper[1], we discuss a *knowledge-aware recommendation framework* based on *neuro-symbolic graph embeddings* that encodes *first-order logical* (FOL) rules. Our workflow starts from a *knowledge graph* (KG) encoding user preferences and item properties. Next, knowledge-aware recommendations are obtained through the combination of three modules: *(i)* a rule learner, that extracts FOL rules from the KG; *(ii)* a graph embedding module, that learns the embeddings of users and items based on the triples of the KG and the FOL rules previously extracted; *(iii)* a recommendation module, that uses the embeddings to feed a deep learning architecture. In the experimental session we evaluate the effectiveness of our strategy on two datasets. The results show that the combination of KG embeddings and FOL rules improves the predictive accuracy and the novelty of the recommendations.

Keywords

recommender systems, graph embeddings, symbolic reasoning, neuro-symbolic systems

1. Introduction

Recommender Systems (RS) are getting a crucial role in decision-making processes [2]. Differently from early RS approaches, which relied on simple *user-item* interactions and ignored descriptive information, more recent attempts showed that *knowledge-aware recommender systems* (KARS) [3] significantly improve the performance of RS [4, 5, 6] by exploiting descriptive features available in knowledge graphs (KGS), such as DBpedia [7]. In this research line, recent shreds of evidences [8, 9, 10] show how good KARS based on *KG embeddings* [11, 12] perform in recommendation tasks; however, these methods are purely *data-driven* and *non-symbolic*[1]. Instead, the current wave of *neuro-symbolic AI systems* [13], fostered the development of models that combine *data-driven* approaches with *pure symbolic* methods, in order to take the best from both the worlds. For example, methods for joint embedding of First-Order Logics (FOL) rules and graphs have been proposed [14, 15]: in this setting, graphs provide *explicit knowledge*, while FOL rules could be exploited to *explicitly* inject some *background knowledge* which is likely to

[1]An extended version of this paper has been published at the 16th ACM Conference on Recommender Systems with the title "Knowledge-aware Recommendations Based on Neuro-Symbolic Graph Embeddings and First-Order Logical Rules", as a Late Breaking Result [1].

Discussion Papers - 21st International Conference of the Italian Association for Artificial Intelligence (AIxIA 2022)

[1]https://analyticsindiamag.com/understanding-difference-symbolic-ai-non-symbolic-ai

improve the embedding process and the resulting representation. In this work, we follow these intuitions and we investigate whether RS can benefit of the integration between *symbolic* and *non-symbolic* knowledge as well. To this end, we present a *knowledge-aware recommendation framework* that relies on *neuro-symbolic graph embeddings* exploiting *first-order logical rules*. Starting from a KG encoding information about users, ratings and descriptive properties of the items, we design a modular framework based on three components: *(i) a Rule Learner*, that extracts FOL rules based on the information encoded in the KG; *(ii) a Graph Embedding* module, that jointly learns a vector-space representation of users and items based on both the explicit information encoded in the KG and the background knowledge encoded in the rules previously learned; *(iii) a Recommendation Framework*, that takes as input the embeddings and use them to feed a deep architecture that predicts the *top-k* items for the user. In the experimental session, we evaluate the effectiveness of our strategy on two datasets and results show that the combination of KG embeddings and FOL rules led to an improvement of the predictive accuracy.

The rest of the paper is organized as follows. In Section 2 we present related work in the area. Next, the different modules that compose our framework are introduced in Section 3. Results of the experiment are shown in Section 4. Finally, conclusions and future work are sketched in Section 5.

2. Related Work

Graph Embeddings for Recommender Systems. The aim of graph embedding techniques is to represent *entities* and *relations* in a KG as *dense vectors* by projecting them in a vector space, preserving the original structure of the graph [11]. Similarly to the approach proposed by Palumbo et al. [16], we applied graph embeddings over a *tripartite* graph encoding both collaborative and content-based information. Other works (such as [17, 18, 19, 12]) showed that recommendation models exploiting graph embedding techniques outperform typical baselines. Next, previous research [8], showed that *translation models*, such as TransE [20], obtain competitive performance in recommendation tasks. Accordingly, this work exploits a translation model, *i.e.* KALE [14], as graph embedding technique. KALE is considered as an extension of TransE that exploits first-order logic to merge in a unified representation logical rules and triples encoded in a knowledge graph. Up to our knowledge, the use of KALE in deep learning architectures for recommendation tasks has never been investigated in literature.

Knowledge-aware Recommender Systems (KARS). KARS foster the idea of injecting information encoded in KGs and RSs. While early works were based on similarity and Linked Open Data [21, 22, 23], more recent methods followed the wave of deep learning. Knowledge-aware Hybrid Factorization Machines (KaHFM) [5] obtained very competitive results by extending classic factorization machines by using semantics information encoded in a KG. Another interesting approach regards the application of Graph Convolutional Neural Networks [24] to KGs. Here, Wang et al. [25] present Knowledge Graph Convolutional Networks (KGCN) for RS, which learn a representation based on user-item interactions (encoded into a matrix) and descriptive properties (encoded in a KG). Similarly, KGAT (Knowledge Graph Attention Network) [25] uses an *attention mechanisms* to model the high-order connectivities in KG. Most of these work were considered as baselines in our experiments. In this case, the novelty of our work lies in the

integration of *explicit* knowledge encoded in the KG with *background knowledge* encoded as logical rules in our knowledge-aware RS. Up to our knowledge, this kind of hybridization has been poorly investigated.

3. Description of the Framework

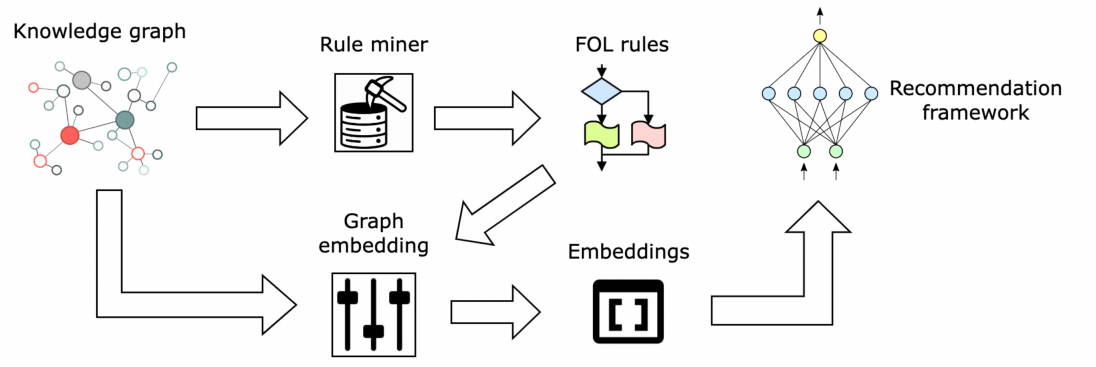

Figure 1: Workflow carried out by our framework.

In this section, we introduce the modules composing our framework; it relies on a knowledge graph (see Figure 2) encoding information about users, items and descriptive properties.

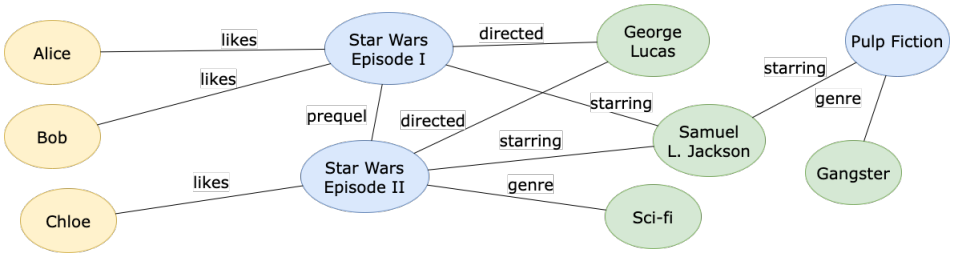

Figure 2: A tripartite knowledge graph. Different kind of entities (users, items, properties) are highlighted with different colors.

Basics of First-Order Logic. The logical rules [26] we exploit are typically referred to as Horn clauses[2], since they are composed by several atoms connected by means of logical connectives (*e.g.,* $\vee, \wedge, \Rightarrow \ldots$), in which at most one of them is positive. Each atom is composed of *variables* (entities) and *predicates* (relations). An example of logical rules is: $\forall x, y, z : likes(x, z) \wedge prequel(y, z) \Rightarrow likes(x, y)$. A specific formula is called *ground* when every variable is replaced by a suitable entity in the graph. An example of *ground* rule based on the graph in Figure 2 is: $likes(Chloe, StarWarsII) \wedge prequel(StarWarsI, StarWarsII) \Rightarrow likes(Chloe, StarWarsI)$.

[2]https://en.wikipedia.org/wiki/Horn_clause

Mining First-Order Logical Rules. The first task carried out by our framework consists in mining FOL rules that hold in a KG to extract *background knowledge* concerning typical pattern encoded in the graph. The process of rule mining returns a *set of FOL rules in Horn form*, each with confidence score, expressing to what degree the rule holds; these scores can be used to rank the rules and encode in the model only the most promising ones. It is worth to point out that any FOL rules mining method can be used at this step. An example of the rules extracted at this step is provided in the previous paragraph.

Learning Graph Embeddings. Once the rules are extracted, a joint learning based on triples in the KG and FOL rules is carried out. In this work, we used KALE [14] as graph embedding technique, which is inspired by TransE [20] and extends it by encoding symbolic knowledge given by FOL rules.

The *neuro-symbolic nature* of KALE lies in the fact that the embeddings for each entity in the KG are learnt by exploiting: *(i) explicit* knowledge, expressed by triples encoded in KG; *(ii) background* knowledge, expressed by FOL rules. Explicit and background knowledge are represented in a *unified* framework that learns a comprehensive representation based on both information sources. Based on work by Rocktäschel et al. [27, 28], joint training is possible since triples from a KG can be seen as *atoms* in FOL (*e.g., likes(Alice,Kill Bill)*); given that also rules are expressed in a logical form, it is possible to exploit *first-order logic* as the common framework that allows to unify the representations and to carry out a joint learning. Due to space reasons, it is not possible to provide more details about KALE.

Recommendation Framework. Learnt embeddings are then used to feed a deep learning architecture that provides users with recommendations. Given a set of users and a set of items, our architecture aims at identify suitable recommendations, by predicting the interest of a user in an items and by ranking the items based on descending relevance score. As shown in Figure 3, we designed a simple yet effective architecture based on the combination of concatenation layers and dense layers which is inspired by previous work in the area [29] that obtained competitive results in the *top-k* recommendations task.

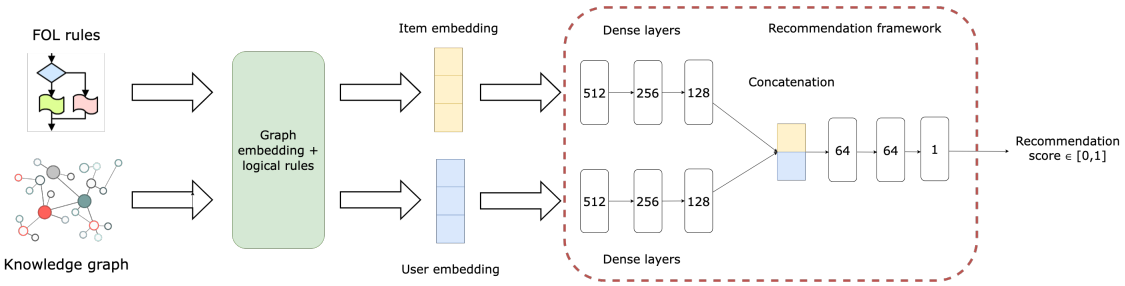

Figure 3: Recommendation framework architecture

In particular, the recommendation process starts with the embeddings which are obtained as output of the graph embedding process. Each embedding is passed through three dense layers and is then merged through concatenation layer. After a second passage through three dense layers, a sigmoid activation function is used to return a score between 0 and 1 which estimates the probability that the item i is relevant for user u. Before making predictions, the architecture

is trained by exploiting all the the ratings in the form (u, i) available in the dataset and by using *binary cross-entropy* as loss function. See next section for more details on parameters. Once the model is learned, it can predict to what extent user u would like unseen items $j \in I$. After this step, items are ranked based on predicted relevance score and *top-k* are returned as recommendations.

4. Experimental Evaluation

In the experimental session we evaluated the effectiveness of our methodology in the task of item recommendation to answer to the following research questions:
RQ1: Which is the best strategy to *select FOL rules* to be included in the embedding process? Is there any *difference in the accuracy, novelty and diversity* when logical rules are encoded in the model?
RQ2: How does our approach based on *neuro-symbolic graph embeddings* perform w.r.t. *competitive baselines*?

4.1. Experimental Design

Datasets. Experiments were carried out in a *movie recommendation* and in a *book recommendation* scenario. In the former case, MovieLens 1M (ML1M)[3] was exploited as dataset, while in the latter we used DBbook dataset[4]. As KG, we exploited DBpedia. To extract information from DBpedia and populate our KG, we exploited a mapping already available online[5]. Table 1 depicts some statistics about the datasets.

	Users	Items	Ratings	%Positive	Sparsity	Entities	Relations	Triples	Avg. Links
ML1M	6,040	3,883	1,000.209	57.51%	96.42%	26,858	13	828,119	30.83
DBbook	6,181	6,733	72,372	45.85%	99.83%	17,505	36	85,549	4.94

Table 1
Statistics of the datasets

Protocol. For both the datasets, we used a 80%-20% training-test split. Data were split in order to maintain the ratio between positive and negative ratings. As for *MovieLens-1M* we considered as *positive* only the ratings equal to 4 and 5 out of 5. As for *DBbook*, ratings were provided in a binary format (positive/negative). The predictive accuracy of the algorithms was evaluated on top-5 recommendation list, calculated by following the *TestRatings* strategy [30].
Source Code and Parameters. Rule mining was carried out by exploiting the latest version of AMIE[6], while a recent Java implementation of KALE[7] was used to learn graph embeddings. Finally, the source code of our deep recommendation framework is available on GitHub[8] and is

[3] http://grouplens.org/datasets/movielens/
[4] http://challenges.2014.eswc-conferences.org/index.php/RecSys
[5] https://github.com/sisinflab/LODrecsys-datasets
[6] https://github.com/lajus/amie
[7] https://github.com/iieir-km/KALE
[8] https://github.com/giuspillo/RepoNeSyRecSys2022

inspired by the implementation made available by the authors of [29] released on Github[9]. As regards the parameters of the tools, as for AMIE, we set maximum length rules to 4 atoms, while for each rule minimum confidence is set equal to to 0.001 and minimum coverage equal to 0.1. As regards KALE, we learnt embeddings having size=$512, 768$ and we learnt the representation with mini batches equal to 100 for both the datasets. Margin separating positive and negative examples $= 0.1$, entity learning rate $= 0.05$, relations learning rate = 0.05, iterations $= 1000$. All these parameters were set through a grid search. Finally, as regards our deep architecture, our models were trained for 25 epochs, by setting batch sizes to 512 for ML1M and to 1536 for DBbook, respectively. The parameter α is set to 0.9 and learning rate is set to 0.001. As optimizer, we used ADAM for ML1M and RMSprop for for DBbook. All the parameters were tuned through a grid search.

Configurations. Throughout the experimental protocol, we compared *five* different variant of our framework: two *basic* configurations (which do not invole FOL rules) and three *rule-based* configurations. In particular, we compared the embeddings learnt on the simple *user-item* and *user-item-properties* graphs (*i.e.,* graph encoding only ratings and graph encoding descriptive properties too). Next, rule-based configuration were run over the user-item-properties graph, and we defined three different heuristics to rank the rules returned by AMIE: (i) LIKE RULES, having a *like* relation in the head (*e.g.,* $likes(x, z) \wedge prequel(y, z) \Rightarrow likes(x, y)$); (ii) TOP-K LIKE RULES, having a like relation in the head based on the coverage of the rules; (iii) HIGH CONFIDENCE RULES, having a confidence score higher than 0.75. Of course, other strategies to select FOL rules can be applied in this step.

Baselines and Evaluation Metrics. To ensure the complete reproducibility, we calculated our evaluation metrics and we selected our baselines by using the Elliot[10] framework [31]. As regards the metrics, to assess the accuracy of the recommendations we used *F1 score* [32], *Mean Average Precision* (*MAP*), and the normalized discounted cumulative gain (*nDGC*) [33, 34] scores; we also considered *diversity* and *novelty* of the recommendations by calculating Gini Index and Expected Popularity Complement (EPC) [35]. Finally, we also compared our methodology to ten baselines available in Elliot: three *matrix factorization* techniques (SLIM [36], BPRMF [37] and PureSVD), three methods based on *deep learning models*, (MultiVAE [38], CFGAN [39] and NGCF [40]) and four algorithms implementing *knowledge-aware* techniques (Item-KNN and User-KNN [41], KaHFM [5] and LightGCN [42]). All the algorithms are run with their optimal parameters, selected by the Elliot framework through a grid search.

4.2. Results

(RQ1) Performance of the Framework and Selection of Logical Rules. For RQ1, we compared performances obtained by basic configurations (*user-item* and *user-item-properties*), treated as baselines, with those using FOL rules. As shown in Table 2, baselines are almost always outperformed by the other configurations. For ML1M, for $k = 512$, *all* rule-based configuration overcome the baselines with a statistically significant gap; configurations based on *like* rules got the best results for all *accuracy* and *novelty* metrics. For $k = 768$ we got good results too, but with a less significant gap. For DBbook we got similar findings: the overall

[9] https://github.com/swapUniba/Deep_CBRS_Amar

[10] https://github.com/sisinflab/elliot

Configuration	ML1M					DBbook				
	Accuracy			Diversity	Novelty	Accuracy			Diversity	Novelty
	MAP	F1	nDCG	Gini	EPC	MAP	F1	nDCG	Gini	EPC
k = 512										
USER-ITEM	0.8106	0.4824	0.8410	*0.1626*	0.6479	0.6393	0.5140	0.6779	0.2792	0.6351
USER-ITEM-PROPERTIES (UIP)	0.8357	0.4881	0.8632	0.1525	0.6656	0.6361	0.5124	0.6757	0.1566	0.6340
UIP + LIKE RULES	**0.8410****	**0.4902****	**0.8680****	0.1481	**0.6698****	0.6357	**0.5126**	0.6755	**0.1573**	0.6337
UIP + TOP-K LIKE	**0.8411****	**0.4898****	**0.8673****	0.1429	0.6644	**0.6410***	**0.5142**	**0.6797***	**0.1567**	**0.6372***
UIP + HIGH CONFIDENCE	**0.8372**	**0.4893**	**0.8650**	**0.1558**	0.6748	**0.6407***	**0.5132**	**0.6790**	**0.1572**	**0.6367**
k = 768										
USER-ITEM	0.8118	0.4831	0.8426	*0.1657*	0.6478	0.6417	0.5114	0.6781	0.2793	0.6349
USER-ITEM-PROPERTIES (UIP)	0.8383	0.4890	0.8649	0.1461	0.6671	0.6332	0.5107	0.6726	0.1575	0.6312
UIP + LIKE RULES	**0.8395**	**0.4902**	**0.8667**	0.1455	**0.6690***	**0.6372***	0.5144	0.6779	0.1580	**0.6356***
UIP + TOP-K LIKE	**0.8401**	**0.4892**	**0.8664**	0.1433	**0.6687****	0.6408	0.5144	0.6794	0.1569	0.6366
UIP + HIGH CONFIDENCE	**0.8394**	**0.4897**	**0.8659**	**0.1539**	**0.6708****	0.6361	0.5114	**0.6755**	0.1569	0.6337

Table 2
Results of the Experiment on ML1M and DBbook data. For each size of the embeddings k, configurations overcoming the baseline (UIP) are highlighted in bold. Significant improvements are emphasized with * (for $p < 0.05$) and ** (for $p < 0.01$).

Baseline	ML1M					DBbook				
	Accuracy			Diversity	Novelty	Accuracy			Diversity	Novelty
	MAP	F1	nDCG	Gini	EPC	MAP	F1	nDCG	Gini	EPC
BPRMF	0.7486	0.4636	0.7861	0.0779	0.558	*0.6392*	0.4472	0.6372	0.0360	0.6626
PureSVD	0.7553	0.4615	0.7896	0.113	0.5856	0.6365	0.4468	0.6349	0.0399	0.6622
Slim	*0.7852*	*0.4743*	*0.8189*	0.1897	*0.6521*	0.6240	0.4437	0.6285	0.0482	0.6556
MultiVAE	0.7358	0.4552	0.7714	0.1445	0.5845	0.6222	0.4400	0.6245	0.0451	0.6519
CFGAN	0.6229	0.414	0.6677	**0.2842**	0.5396	0.6104	0.4391	0.6166	0.0507	0.6458
NGCF	0.5946	0.4057	0.6442	0.192	0.5229	0.6074	0.4377	0.6150	*0.0511*	0.6448
AttributeItemKnn	0.7302	0.449	0.765	0.2257	0.6127	0.6316	0.4443	0.6333	0.0496	0.6602
AttributeUserKnn	0.7499	0.4557	0.784	0.1556	0.5957	0.6371	0.4463	0.6363	0.0452	0.6629
KaHFM	0.7403	0.4565	0.7763	0.1468	0.5865	0.6384	*0.4493*	*0.6381*	0.0354	**0.6633**
LightGCN	0.5946	0.4057	0.6442	0.192	0.5229	0.6147	0.4420	0.6212	0.0462	0.6490
Our best	**0.8406****	**0.4903****	**0.8676****	0.1517	**0.6739****	**0.6410**	**0.5142****	**0.6797****	**0.1567**	0.6372

Table 3
Comparison to baselines on ML1M and DBbook data. Overall best performing configuration is highlighted in **bold**. Significant improvements are emphasized with ** (for $p < 0.01$).

best results are obtained for $k = 512$, and the best-performing configuration is *top-k likes* rules. Moreover, our best-performing configuration improves the baselines in terms of *diversity* and *novelty* as well. As regards $k = 768$, the behavior we noted is in line with that we observed for ML1M. We can also notice that *user-item* often overcomes *user-item-properites*: this means that descriptive properties are poorly informative; however, injecting *background* knowledge encoded as FOL rules is able to overcome this issue and leads to the overall best results.

(RQ2) Comparison to Baselines. To answer RQ2, we compared our best-performing configuration with some competitive baselines. Table 3 show that our framework significantly overcomes those baselines. For ML1M (Table 3), all metrics but *diversity* have been outperformed with a significant gap ($p < 0.01$). It is worth to notice that KARS have been outperformed by methods for matrix factorization, differently by our expectations. Results on DBbook confirm these findings: all metrics, except for *novelty*, are overcame by our framework. The best baseline in this case is KaHFM [5], and this should not surprise since the high sparsity of the dataset

emphasized the importance of the descriptive features of the items. We can also note that our approach obtained the best results for *novelty* and *diversity* on ML1M and DBbook, respectively, so further analyses will be carried out to better understand this behavior.

5. Conclusions and Future Work

In this paper we discussed a *knowledge-aware recommendation framework* based on *neuro-symbolic graph embeddings* encoding *first-order logical rules*; it combines *explicit* knowledge provided by triples in the graph and *background* knowledge provided FOL rules. Our experiments provided us with the following findings: *(i)* joint learning based on FOL rules and KGs generates more precise embeddings and more accurate recommendations, in particular using rules with only *like relation* in the head; *(ii)* comparisons with several baselines confirmed the accuracy of our framework. A good impact is also noted in terms of novelty and diversity of the recommendations. Such promising results represent a first step in the direction of developing *neuro-symbolic* RSs exploiting neural and symbolic reasoning. However, given the novelty of the current work, we are aware that several limitations exist. As an example, more accurate methods for selecting the rules are needed, together with a more extensive analysis of the informative power of different FOL rules. As future work, we will also strengthen the level of the baselines we consider in our experiments and the number of datasets, and by evaluating the system in different domains (i.e., food recommendations [43]) and by modeling user preferences by exploiting different sets of features [44].

References

[1] G. Spillo, C. Musto, M. De Gemmis, P. Lops, G. Semeraro, Knowledge-aware recommendations based on neuro-symbolic graph embeddings and first-order logical rules, in: Proceedings of the 16th ACM Conference on Recommender Systems, 2022, pp. 616–621.

[2] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, Recommender systems: an introduction, Cambridge University Press, 2010.

[3] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He, A survey on knowledge graph-based recommender systems, IEEE Transactions on Knowledge and Data Engineering (2020).

[4] C. Musto, P. Lops, M. de Gemmis, G. Semeraro, Semantics-aware recommender systems exploiting linked open data and graph-based features, Knowledge-Based Systems 136 (2017) 1–14.

[5] V. W. Anelli, T. Di Noia, E. Di Sciascio, A. Ragone, J. Trotta, How to make latent factors interpretable by feeding factorization machines with knowledge graphs, in: International Semantic Web Conference, Springer, 2019, pp. 38–56.

[6] C. Musto, M. d. Gemmis, P. Lops, F. Narducci, G. Semeraro, Semantics and content-based recommendations, in: Recommender systems handbook, Springer, 2022, pp. 251–298.

[7] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: The semantic web, Springer, 2007, pp. 722–735.

[8] E. Palumbo, G. Rizzo, R. Troncy, E. Baralis, M. Osella, E. Ferro, Translational models

for item recommendation, in: European Semantic Web Conference, Springer, 2018, pp. 478–490.

[9] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L.-K. Huang, C. Xu, Recurrent knowledge graph embedding for effective recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 297–305.

[10] W. Song, Z. Duan, Z. Yang, H. Zhu, M. Zhang, J. Tang, Explainable knowledge graph-based recommendation via deep reinforcement learning, arXiv preprint arXiv:1906.09506 (2019).

[11] H. Cai, V. W. Zheng, K. C.-C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Transactions on Knowledge and Data Engineering 30 (2018) 1616–1637.

[12] C. Musto, P. Lops, M. de Gemmis, G. Semeraro, Context-aware graph-based recommendations exploiting personalized pagerank, Knowledge-Based Systems 216 (2021) 106806.

[13] M. K. Sarker, L. Zhou, A. Eberhart, P. Hitzler, Neuro-symbolic artificial intelligence, AI Communications (2021) 1–13.

[14] S. Guo, Q. Wang, L. Wang, B. Wang, L. Guo, Jointly embedding knowledge graphs and logical rules, in: Proceedings of the 2016 conference on empirical methods in natural language processing, 2016, pp. 192–202.

[15] S. Guo, Q. Wang, L. Wang, B. Wang, L. Guo, Knowledge graph embedding with iterative guidance from soft rules, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

[16] E. Palumbo, G. Rizzo, R. Troncy, Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, ACM, 2017, pp. 32–36.

[17] C. Musto, P. Basile, G. Semeraro, Embedding knowledge graphs for semantics-aware recommendations based on dbpedia, in: Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization, 2019, pp. 27–31.

[18] S. Forouzandeh, K. Berahmand, M. Rostami, Presentation of a recommender system with ensemble learning and graph embedding: a case on movielens, Multimedia Tools and Applications (2020) 1–28.

[19] L. Grad-Gyenge, A. Kiss, P. Filzmoser, Graph embedding based recommendation techniques on the knowledge graph, in: Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization, 2017, pp. 354–359.

[20] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Neural Information Processing Systems (NIPS), 2013, pp. 1–9.

[21] C. Musto, G. Semeraro, P. Lops, M. d. Gemmis, F. Narducci, Leveraging social media sources to generate personalized music playlists, in: International Conference on Electronic Commerce and Web Technologies, Springer, 2012, pp. 112–123.

[22] C. Musto, F. Narducci, P. Lops, G. Semeraro, M. d. Gemmis, M. Barbieri, J. Korst, V. Pronk, R. Clout, Enhanced semantic tv-show representation for personalized electronic program guides, in: International Conference on User Modeling, Adaptation, and Personalization, Springer, 2012, pp. 188–199.

[23] G. Piao, J. G. Breslin, Measuring semantic distance for linked open data-enabled recommender systems, in: Proceedings of the 31st Annual ACM Symposium on Applied

Computing, 2016, pp. 315–320.
[24] S. Zhang, H. Tong, J. Xu, R. Maciejewski, Graph convolutional networks: a comprehensive review, Computational Social Networks 6 (2019) 1–23.
[25] X. Wang, X. He, Y. Cao, M. Liu, T.-S. Chua, KGAT: Knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 950–958.
[26] R. M. Smullyan, First-order logic, Courier Corporation, 1995.
[27] T. Rocktäschel, M. Bosnjak, S. Singh, S. Riedel, Low-dimensional embeddings of logic, in: Proceedings of the ACL 2014 workshop on semantic parsing, 2014, pp. 45–49.
[28] T. Rocktäschel, S. Singh, S. Riedel, Injecting logical background knowledge into embeddings for relation extraction, in: Proceedings of the 2015 conference of the north American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2015, pp. 1119–1129.
[29] M. Polignano, C. Musto, M. de Gemmis, P. Lops, G. Semeraro, Together is better: Hybrid recommendations combining graph embeddings and contextualized word representations, in: Fifteenth ACM Conference on Recommender Systems, 2021, pp. 187–198.
[30] A. Bellogin, P. Castells, I. Cantador, Precision-oriented evaluation of recommender systems: an algorithmic comparison, in: Proceedings of the fifth ACM conference on Recommender systems, 2011, pp. 333–336.
[31] V. W. Anelli, A. Bellogín, A. Ferrara, D. Malitesta, F. A. Merra, C. Pomo, F. M. Donini, T. D. Noia, Elliot: a comprehensive and rigorous framework for reproducible recommender systems evaluation, CoRR abs/2103.02590 (2021).
[32] F. H. Del Olmo, E. Gaudioso, Evaluation of recommender systems: A new approach, Expert Systems with Applications 35 (2008) 790–804.
[33] K. Järvelin, J. Kekäläinen, Ir evaluation methods for retrieving highly relevant documents, in: ACM SIGIR Forum, volume 51, ACM New York, NY, USA, 2017, pp. 243–250.
[34] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, T.-Y. Liu, A theoretical analysis of ndcg ranking measures, in: Proceedings of the 26th annual conference on learning theory (COLT 2013), volume 8, 2013, p. 6.
[35] S. Vargas, P. Castells, Rank and relevance in novelty and diversity metrics for recommender systems, in: Proceedings of the fifth ACM conference on Recommender systems, 2011, pp. 109–116.
[36] X. Ning, G. Karypis, Slim: Sparse linear methods for top-n recommender systems, in: 2011 IEEE 11th International Conference on Data Mining, IEEE, 2011, pp. 497–506.
[37] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, arXiv preprint arXiv:1205.2618 (2012).
[38] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Proceedings of the 2018 world wide web conference, 2018, pp. 689–698.
[39] D.-K. Chae, J.-S. Kang, S.-W. Kim, J.-T. Lee, Cfgan: A generic collaborative filtering framework based on generative adversarial networks, in: Proceedings of the 27th ACM international conference on information and knowledge management, 2018, pp. 137–146.
[40] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd international ACM SIGIR conference on Research and development

in Information Retrieval, 2019, pp. 165–174.
[41] Z. Gantner, S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Mymedialite: A free recommender system library, in: Proceedings of the fifth ACM conference on Recommender systems, 2011, pp. 305–308.
[42] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, LightGCN: Simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020, pp. 639–648.
[43] C. Musto, C. Trattner, A. Starke, G. Semeraro, Towards a knowledge-aware food recommender system exploiting holistic user models, in: Proceedings of the 28th ACM conference on user modeling, adaptation and personalization, 2020, pp. 333–337.
[44] C. Musto, M. Polignano, G. Semeraro, M. de Gemmis, P. Lops, Myrror: a platform for holistic user modeling, User Modeling and User-Adapted Interaction 30 (2020) 477–511.