Mitigating Data Sparsity via Neuro-Symbolic Knowledge Transfer

Tommaso Carraro[1,2,*], Alessandro Daniele[2], Fabio Aiolli[1] and Luciano Serafini[2]

[1]Department of Mathematics, University of Padova, Via Trieste, 63, 35131 Padova, Italy
[2]Data and Knowledge Management, Fondazione Bruno Kessler, Via Sommarive, 18, 38123 Povo, Italy

Abstract

Data sparsity is a well-known historical limitation of recommender systems that still impacts the performance of state-of-the-art approaches. One practical technique to mitigate this issue involves transferring information from other domains or tasks to compensate for scarcity in the target domain, where the recommendations must be performed. Following this idea, we propose a novel approach based on Neuro-Symbolic computing designed for the knowledge transfer task in recommender systems. In particular, we use a Logic Tensor Network (LTN) to train a *vanilla* Matrix Factorization (MF) model for rating prediction. We show how the LTN can be used to regularize the MF model using axiomatic knowledge that permits injecting pre-trained information learned by Collaborative Filtering on a different task or domain. Extensive experiments comparing our model with a baseline MF on two versions of a novel real-world dataset prove our proposal's potential in the knowledge transfer task. In particular, our model consistently outperforms the MF, suggesting that the knowledge is effectively transferred to the target domain via logical reasoning. Moreover, an experiment that drastically decreases the density of user-item ratings shows that the benefits of the acquired knowledge increase with the sparsity of the dataset, showing the importance of exploiting knowledge from a denser source of information when training data is scarce in the target domain.

Keywords

knowledge transfer, matrix factorization, neuro-symbolic integration, logic tensor networks, rating prediction, explicit feedback, data sparsity

1. Introduction

Recommender systems (RSs) have recently become essential for e-services (e.g., Amazon, Netflix, Spotify). Given the user's historical data, these tools mitigate information overload by suggesting novel items (e.g., products, movies, songs) that match the user's preferences [1]. Since the beginning of the RSs literature, Collaborative Filtering (CF) [2, 3, 4, 5] has been one of the most successful recommendation approaches. Latent Factor models, in particular Matrix Factorization (MF), have dominated the CF scene [6, 7, 8] for years, and this has been further emphasized with the deep learning rise [9, 10, 11, 12].

Despite their success in improving recommendation performance, state-of-the-art models still suffer from a historical issue, i.e., data sparsity, that limits their applicability in real-world scenarios. One way to address data sparsity consists of leveraging models pre-trained on other sources of information (i.e., source domains) to make the final model more accurate in the target domain, where the recommendations must be performed. This technique is called knowledge transfer [13] and belongs to the research field of transfer learning [14]. In the context of recommender systems, knowledge transfer techniques [15, 16, 17, 18] can be subdivided into two categories: *feature-based* models and *fine-tuning* models. In feature-based models, pre-trained models are used to learn features from side information for users and/or items. Then, these features are integrated into the recommendation framework. By merging side information and user-item interaction data (i.e., Collaborative Filtering), feature-based models can potentially alleviate data sparsity in recommendation datasets. Instead, fine-tuning models firstly train a deep transferable neural model on user-item interactions taken from a source domain. Then, this pre-trained model is fine-tuned on the downstream recommendation task, namely the target domain. The first category inspires the approach we propose in this paper. In particular, instead of learning features for users and items in the source domain, we learn to predict ratings via Collaborative Filtering. Then, this knowledge is transferred to the target domain via logical reasoning.

Despite Neuro-Symbolic (NeSy) [19] approaches having been successfully applied in many AI fields [20, 21, 22], including RSs [23, 24, 25, 26], they have not yet been investigated in the task of knowledge transfer for recommender systems, where we believe their application is particularly suited. In particular, NeSy aims at integrating knowledge, usually expressed using logical axioms, with neural networks. The integration has shown

Fifth Knowledge-aware and Conversational Recommender Systems (KaRS) Workshop @ RecSys 2023, September 18–22 2023, Singapore.
*Corresponding author.
tcarraro@fbk.eu (T. Carraro); daniele@fbk.eu (A. Daniele); aiolli@math.unipd.it (F. Aiolli); serafini@fbk.eu (L. Serafini)
0000-0002-3043-1456 (T. Carraro); 0000-0001-9441-0729 (A. Daniele); 0000-0002-5823-7540 (F. Aiolli); 0000-0003-4812-1031 (L. Serafini)

to be particularly beneficial in tasks with poor training data [27], giving insights that this paradigm can help deal with data sparsity in recommendation datasets.

Following this intuition, we propose using a Logic Tensor Network (LTN) [28] to encode axiomatic knowledge to enable effective knowledge transfer and injection for a *vanilla* MF model[1] trained on movie ratings. LTN is a NeSy framework that effectively integrates logical reasoning and neural networks. Our approach uses it as the interface between the model pre-trained on the source domain and the final model trained on the target domain. We called it an interface as it allows the explicit transfer of information between domains via logical reasoning. To perform our experiments, we use MindReader [30], a novel recommendation dataset containing explicit ratings from real users on movies and *non-recommendable* entities, such as movie genres, actors, and producers. In particular, we use ratings on movie genres to learn our pre-trained model via Collaborative Filtering and ratings on movies to train the final model to provide accurate recommendations in the target domain. The pipeline of the proposed approach consists of two steps. In the first step, we train a genre classifier using an MF model to learn which genres the users like and dislike in the source domain. In the second step, we use LTN to transfer the pre-trained knowledge to an MF model trained on the target domain for the movie rating prediction task.

We compare our approach with a baseline MF model to understand if knowledge transfer is successfully[2] reached thanks to Neuro-Symbolic integration. The results show that our model consistently outperforms the MF, proving its ability to transfer knowledge across domains. In addition, an experiment that drastically reduces the density of user-item ratings shows that the benefits of the knowledge increase with the sparsity of the dataset. This gives insight that our model can successfully deal with data sparsity thanks to Neuro-Symbolic reasoning. To the best of our knowledge, this is the first time a NeSy approach has been successfully applied to the knowledge transfer task for recommender systems.

2. Related works

In the last two years, the RS community has seen the emergence of some Neuro-Symbolic approaches [25, 24, 31, 32, 33]. Among them, some directly use logical formulas to learn a recommendation model [23, 26, 31]. The seminal approach that applied NeSy to RSs has been Neural Collaborative Reasoning (NCR) [26]. In NCR, the sequential recommendation task is formalized as a logical reasoning problem. In particular, the user's ratings are represented using propositional variables, and logical operators (e.g., $\wedge$, $\implies$) are used to construct propositional formulas that express sequential patterns between them. Then, NCR maps the variables to *logical* embeddings and the operators to neural networks (NNs) that act on those embeddings. The NNs are forced to behave as classical logic operators through *logical* regularization. By doing so, the formulas can be organized as a neural network to conduct logical reasoning and prediction in a continuous space. They compared NCR with many linear and deep baselines, showing it can reach state-of-the-art performance. However, this approach is not properly NeSy as neural networks implement the symbolic part (i.e., logical connectives). LTN uses fuzzy logic semantics instead, making the framework theoretically sound. Moreover, NCR uses propositional logic, which makes it impossible to encode complex and expressive knowledge due to the simplicity of the language syntax.

In Graph Collaborative Reasoning (GCR) [31], NCR is extended to work with knowledge graphs. In particular, they provide a simple approach for translating the graph structure into logical expressions to convert the link prediction task into a logical reasoning problem. As in NCR, they use logically constrained neural modules to build the network architecture according to the logical expression. They conducted experiments similar to NCR, showing that GCR can improve NCR on knowledge graphs.

In Counterfactual Collaborative Reasoning (CCR) [32], NCR is used to perform data augmentation based on logical reasoning for sequential recommendation. Specifically, counterfactual logic reasoning is exploited to generate counterfactual examples for data augmentation. The examples are generated by discovering slight changes in users' explicit feedback (i.e., the sequence of purchases) by solving a counterfactual optimization problem. They showed that these new examples, together with the original examples, can alleviate scarcity and enhance the performance of sequential recommendations.

Another approach that successfully integrated logical reasoning and learning has been HYbrid Probabilistic Extensible Recommender (HyPER) [25], which is based on Probabilistic Soft Logic [34]. In particular, HyPER exploits the expressiveness of First-Order Logic (FOL) to encode knowledge from a wide range of information sources, such as multiple user and item similarity measures, content, and social information. Then, Hinge-Loss Markov Random Fields are used to learn how to balance the different information types. HyPER is highly related

[1] Note we selected Matrix Factorization for our experiments because, despite its simplicity, it is still one of the most powerful state-of-the-art approaches [29]. This is not to be intended as a limit of our approach, as any other state-of-the-art model could be used in principle.

[2] Note the objective of the experiment is not to obtain state-of-the-art performance. Instead, our goal is to show that a NeSy approach can be used for knowledge transfer in recommender systems. To this end, the only difference between the models in the comparison is the addition of knowledge via LTN.

to our work with LTN since the logical formulas we use resemble the ones used in HyPER.

In [24], they propose a NeSy approach to encode FOL formulas to enhance knowledge graph embeddings [35] and provide accurate knowledge-aware recommendations. Their approach consists of three steps: (*i*) the FOL formulas are automatically extracted from a recommendation knowledge graph, then (*ii*) the knowledge graph embeddings are learned jointly with the extracted formulas using a NeSy approach. Finally, (*iii*) the user-item embeddings are fed to a neural architecture to get predictions. Specifically, in the second step, they used KALE [36], a NeSy approach that allows learning knowledge graph embeddings jointly with FOL formulas. In particular, the learning can be unified as the graph triples can be interpreted as FOL atoms. As KALE uses fuzzy semantics to learn graph embeddings, this approach can be considered more theoretically sound compared to previous deep approaches, as the symbolic component remains symbolic during learning, as it happens with LTN and HyPER.

One [23] of the last published approaches is highly related to ours. Specifically, they try to mitigate data sparsity by using LTN to inject *content* information into an MF model. In particular, they encode FOL formulas to use side information as a regularizer for the latent factors of the MF model. They show that the proposed NeSy approach can outperform the MF. However, the improvement is poor, and the model has some scalability issues due to the number of times the formulas have to be evaluated during training. Our approach differs from [23] on how axiomatic knowledge is used. In [23], the knowledge extends the MF model using content information, while our model uses it to enable the transfer of pre-trained knowledge learned via Collaborative Filtering on another task (i.e., movie genre rating prediction). Moreover, our model provides a logical formalization for the rating prediction task[3], while [23] proposes a ranking-based method. Finally, note all the presented models use some form of knowledge (e.g., knowledge graphs, logic) to improve the recommendation performance. However, none of them have designed experiments to understand if the advantages of a NeSy system can help mitigate one or some of the historical issues of recommender systems (e.g., data sparsity, cold-start, explainability). Hence, it is unclear if the properties (e.g., *few-shot* learning, interpretability) of a NeSy approach are totally exploited.

3. Background

This section provides useful notation and terminology used in the remainder of the paper.

[3]Note our approach can be easily extended to the top-n recommendation task by changing the formalization of the knowledge base.

3.1. Notation

Bold notation differentiates vectors, e.g., $\mathbf{x} = [3.2, 2.1]$, and scalars, e.g., $x = 5$. Matrices and tensors are denoted with upper case bold notation, e.g., $\mathbf{X}$. $\mathbf{X}_i$ is used to denote the i-th row of $\mathbf{X}$, while $\mathbf{X}_{i,j}$ to denote the item at row i and column j. We refer to the set of users of an RS with $\mathcal{U}$, where $|\mathcal{U}| = n$. Similarly, the set of items is denoted as $\mathcal{I}$, such that $|\mathcal{I}| = m$. We use $\mathcal{D}$ to denote a dataset. $\mathcal{D}$ is defined as a set of N triples $\mathcal{D} = \{(u,i,r)^{(j)}\}_{j=1}^{N}$, where $u \in \mathcal{U}$, $i \in \mathcal{I}$, and $r \in \{0,1\}$ is a binary explicit rating. $\mathcal{D}$ can be reorganized in the so-called user-item matrix $\mathbf{R} \in \mathbb{N}^{n \times m}$, such that $\mathbf{R}_{u,i} = r$ if $(u,i,r) \in \mathcal{D}$, 0 otherwise. Then, since we work with binary feedback, we refer to $\mathcal{D}_+$ (resp. $\mathcal{D}_-$) as the dataset of positive (resp. negative) user-item pairs. $\mathcal{D}_+$ is defined as a set of N_+ couples $\mathcal{D}_+ = \{(u,i)^{(j)} | (u,i,r)^{(j)} \in \mathcal{D}, r^{(j)} = 1\}_{j=1}^{N}$. Similarly, $\mathcal{D}_-$ is defined as a set of N_- couples $\mathcal{D}_- = \{(u,i)^{(j)} | (u,i,r)^{(j)} \in \mathcal{D}, r^{(j)} = 0\}_{j=1}^{N}$. Finally, $\mathcal{D}_?$ denotes the dataset of user-item pairs for which the rating is unknown, i.e., $\mathcal{D}_? = \{(u,i) \in \mathcal{U} \times \mathcal{I} | (u,i,r) \notin \mathcal{D}\}$. Clearly, $N_? = n \cdot m - N$.

3.2. Matrix Factorization

Matrix Factorization (MF) is a Latent Factor Model that aims at factorizing the user-item matrix $\mathbf{R}$ into the product of two lower-dimensional rectangular matrices, $\mathbf{U} \in \mathbb{R}^{n \times k}$ and $\mathbf{I} \in \mathbb{R}^{m \times k}$, containing the users' and items' latent factors, respectively. k represents the number of latent factors. More formally, the objective of MF is to find $\mathbf{U}$ and $\mathbf{I}$ such that $\mathbf{R} \approx \mathbf{U} \cdot \mathbf{I}^\top$. An effective way to learn the latent factors is by using gradient-descent optimization. Given the dataset $\mathcal{D}$, an MF model seeks to minimize the Mean Squared Error (MSE) between predicted and target ratings, defined as $\frac{1}{N}\sum_{(u,i,r)\in\mathcal{D}} ||\tilde{r} - r||^2 + \lambda||\boldsymbol{\theta}||^2$. In the formulation, $\tilde{r} = \mathbf{U}_u \cdot \mathbf{I}_i^\top + \mathbf{u}_u + \mathbf{i}_i$, where $\mathbf{u}_u$ and $\mathbf{i}_i$ are bias terms for user u and item i, respectively, and $\boldsymbol{\theta} = \{\mathbf{U}, \mathbf{I}, \mathbf{u}, \mathbf{i}\}$. λ is a hyper-parameter to set the strength of the $L2$ regularization.

In our setting, we use a different implementation of MF since we treat the recommendation problem as a binary classification task. Specifically, we need to recommend if a user likes (1) or dislikes (0) an item. Hence, the focal loss is used in place of MSE for the training, and the logistic function is applied to the prediction of MF to restrict the output between 0 and 1. Focal loss is defined as

$$\frac{1}{N}\sum_{(u,i,r)\in\mathcal{D}} -\alpha_t(1-p_t)^\gamma \log p_t + \lambda||\boldsymbol{\theta}||^2$$
$$\alpha_t = \begin{cases} \alpha & \text{if } r = 1 \\ 1-\alpha & \text{if } r = 0 \end{cases} \qquad (1)$$
$$p_t = \begin{cases} p & \text{if } r = 1 \\ 1-p & \text{if } r = 0 \end{cases}$$

where α is a hyper-parameter to give different weights to the two classes, γ is a hyper-parameter that represents the penalty assigned to the examples that are hard to classify, and $p = \sigma(\mathbf{U}_u \cdot \mathbf{I}_i^\top + \mathbf{u}_u + \mathbf{i}_i)$, where σ is the logistic function.

3.3. Logic Tensor Networks

Logic Tensor Networks (LTN) [28] is a Neuro-Symbolic framework that allows using a knowledge base composed of a set of FOL axioms as the objective of a neural model. LTN uses a specific first-order language, called Real Logic, that is fully differentiable and has concrete semantics that allows mapping every symbolic expression into the domain of real numbers. We refer to the term *grounding*[4], formally denoted by $\mathcal{G}$, as the function that defines this mapping. Real Logic allows LTN to ground logical formulas into computational graphs, enabling gradient-based optimization.

In particular, $\mathcal{G}$ maps individuals (e.g., users) to tensors of real features (e.g., users' demographic information), functions (e.g., Score($user, item$)) as real functions (e.g., inner product), and predicates (e.g., Likes($user, item$)) as real functions with output in $[0, 1]$. Then, a variable x is mapped to a *sequence* of n_x individuals (e.g., some items of the dataset), with $n_x \in \mathbb{N}^+, n_x > 0$. As a consequence, a term $t(x)$ or a formula $\mathrm{P}(x)$, will be mapped to a sequence of n_x values too. Afterward, connectives are grounded using fuzzy semantics (i.e., operators dealing with fuzzy values), while quantifiers are grounded as special aggregation functions (e.g., generalized means). This paper uses the *product configuration*, best suited for gradient-based optimization [37]. In the notation, $v, z, v_1, \dots, v_n \in [0, 1]$ and $p \geq 1$.

$$\mathcal{G}(\wedge) = \mathrm{T}_{prod}(v, z) = v * z$$
$$\mathcal{G}(\implies) = \mathrm{I}_R(v, z) = 1 - v + v * z$$
$$\mathcal{G}(\neg) = \mathrm{N}_S(v) = 1 - v$$
$$\mathcal{G}(\forall) = \mathrm{ME}_p(v_1, \dots, v_n) = 1 - (\frac{1}{n}\sum_{i=1}^{n}(1 - v_i)^p)^{\frac{1}{p}}$$
$$\mathcal{G}(\exists) = \mathrm{M}_p(v_1, \dots, v_n) = (\frac{1}{n}\sum_{i=1}^{n} v_i^p)^{\frac{1}{p}})$$

The intuition behind the choice of hyper-parameter p is that the higher that p is, the more weight that M_p (resp. ME_p) will give to *true* (resp. *false*) truth-values, converging to the max (resp. min) operator. Real Logic also provides a special type of quantification, called *diagonal quantification*, denoted as $\mathrm{Diag}(x_1, \dots, x_n)$. It allows quantifying over specific tuples of individuals of variables $x_1, \dots, x_n$, such that the i-th tuple contains the i-th individual of each variable.

Given a Real Logic knowledge base $\mathcal{K} = \{\phi_1, \dots, \phi_n\}$, where $\phi_1, \dots, \phi_n$ are closed formulas, LTN allows learning the grounding of constants, functions, and predicates appearing in them. In particular, if constants are grounded as embeddings and functions/predicates onto neural networks, their grounding $\mathcal{G}$ depends on some learnable parameters $\boldsymbol{\theta}$. We denote a parametric grounding as $\mathcal{G}(\cdot|\boldsymbol{\theta})$. In LTN, the learning of parametric groundings is obtained by finding parameters $\boldsymbol{\theta}^*$ that maximize the satisfaction of $\mathcal{K}$, namely $\boldsymbol{\theta}^* = \mathrm{argmax}_{\boldsymbol{\theta}} \mathrm{SatAgg}_{\phi \in \mathcal{K}} \mathcal{G}(\phi|\boldsymbol{\theta})$, where $\mathrm{SatAgg} : [0, 1]^* \mapsto [0, 1]$ is a formula aggregating operator, generally defined using ME_p.

3.4. Neuro-Symbolic knowledge transfer

Our model is inspired by feature-based knowledge transfer models [13]. In these approaches, a pre-trained model is learned on some additional source of information, usually side information. Then, the acquired knowledge is transferred to the target domain during the training of the final model to help deal with the data sparsity of user-item interactions. More formally, given the external source of information (e.g., content information, additional ratings), a pre-trained model $f_{source}(u, g|\boldsymbol{\theta}_1)$ is learned to generate features[5] for users and items in the source domain. In the notation, u is a user index, g is an item index, and $\boldsymbol{\theta}_1$ are the parameters of the pre-trained model. Instead of generating features, our approach learns f_{source} to predict user-genre preferences in the source domain via Matrix Factorization; hence, the output of this model is a score for the given user-genre pair in the source domain. Then, we assume $f_{target}(u, i|\boldsymbol{\theta}_2)$ is a model that learns to predict user-movie preferences in the target domain. In the notation, u is a user index, i is an item index, and $\boldsymbol{\theta}_2$ are the parameters of the final recommendation model. Specifically, f_{target} outputs a score for the given user-movie pair in the target domain. In our approach, f_{target} is learned while transferring[6] knowledge from f_{source} via logical reasoning thanks to Neuro-Symbolic computing. The objective of our model is to maximize function

$$\mathcal{F}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = l(f_{source}(u, g|\boldsymbol{\theta}_1), f_{target}(u, i|\boldsymbol{\theta}_2), content(i, g))$$

where l is a logic-based aggregation function, u is a user index, g is a movie genre index, and i is a movie index. Note *content* is a function that relates movies and genres

[4]Notice that this is different from the common use of the term *grounding* in logic, which indicates the process of replacing the variables of a term or formula with constants or terms containing no variables.

[5]In feature-based models, the pre-trained model is usually used to obtain features for users and items from the content or side information. In our scenario, it learns to predict user-item ratings via Collaborative Filtering.

[6]Note that parameters $\boldsymbol{\theta}_1$ (of f_{source}) are frozen during the training of parameters $\boldsymbol{\theta}_2$ (of f_{target}).

using content information. It can be seen as a lookup table denoting which movies belong to a specific genre in the dataset. In our approach, this function is used by l in combination with the predictions of f_{source} and f_{target} to regularize the training of f_{target} via logical reasoning. In particular, the aggregation function l can be seen as the logical knowledge base that LTN seeks to maximally satisfy during training. Specifically, l is a composition of logical formulas that defines how the source and target domain interact during training. In other words, l makes it possible to transfer knowledge between domains via logical reasoning and is carefully formalized in Section 4.

4. Method

Our approach uses an LTN to enable domain adaptation for effective knowledge transfer. Specifically, the LTN is trained using a Real Logic knowledge base containing facts designed to intuitively transfer information about movie genre preferences (i.e., the source domain) to a Matrix Factorization model trained on movie ratings (i.e., the target domain). In the next subsections, we will present our knowledge base, how $\mathcal{G}$ is used to convert it into a computational graph suitable for gradient-based optimization, and how the learning of the LTN takes place.

4.1. Real Logic knowledge base

The objective of our LTN model is the satisfaction of the following Real Logic knowledge base.

$$\forall \text{Diag}(user_+, movie_+)\, \text{Likes}(user_+, movie_+) \tag{2}$$

$$\forall \text{Diag}(user_-, movie_-) \neg \text{Likes}(user_-, movie_-) \tag{3}$$

$$\forall \text{Diag}(user_?, movie_?)(\exists genre \neg \text{LikesGenre}(user_?, genre) \wedge \text{HasGenre}(movie_?, genre)) \implies \neg \text{Likes}(user_?, movie_?) \tag{4}$$

Specifically, $user_+$ and $movie_+$ are variable symbols denoting positive user-item pairs, $user_-$ and $movie_-$ are variable symbols denoting negative user-item pairs, $user_?$ and $movie_?$ are variable symbols denoting user-item pairs for which the rating is unknown, and $genre$ is a variable symbol denoting the genres of the movies. Then, Likes(u, i) is a predicate symbol denoting whether a user u likes a movie i, LikesGenre(u, g) is a predicate symbol denoting whether a user u likes a movie genre g, and HasGenre(i, g) is a predicate symbol denoting whether a movie i belongs to genre g.

Intuitively, Axiom (2), Axiom (3), and Axiom (4) are applied to user-item pairs in $\mathcal{D}_+$, $\mathcal{D}_-$, and $\mathcal{D}_?$, respectively. Diag is used to quantify over the desired user-item pairs rather than quantifying over all possible combinations of user and item indexes in the dataset.

4.2. Grounding of symbols

The grounding $\mathcal{G}$ defines how logical symbols are mapped onto the real field and hence how the axioms in the knowledge base define the computational graph of the LTN model. In this work, $\mathcal{G}(user_*) = \langle u^{(j)} | (u,i)^{(j)} \in \mathcal{D}_* \rangle_{j=1}^{N_*}$ and $\mathcal{G}(movie_*) = \langle i^{(j)} | (u,i)^{(j)} \in \mathcal{D}_* \rangle_{j=1}^{N_*}$, namely $user_*$ and $movie_*$ are grounded as a sequence of the N_* user and movie indexes in $\mathcal{D}_*$, with $* \in \{+, -, ?\}$. Instead, $\mathcal{G}(genre) = \langle 1, \dots, N_g \rangle$, namely $genre$ is grounded as a sequence of N_g genre indexes, where N_g is the number of movie genres in the dataset. Afterward, $\mathcal{G}(\text{Likes} | \mathbf{U}, \mathbf{I}, \mathbf{u}, \mathbf{i}) : u, i \mapsto \sigma(\mathbf{U}_u \cdot \mathbf{I}_i^\top + \mathbf{u}_u + \mathbf{i}_i)$, namely Likes is grounded onto a function that takes as input a user index u and a movie index i and returns the prediction in $[0, 1]$ of the MF[7] model for the given user-item pair. $\mathbf{U} \in \mathbb{R}^{n \times k}$, $\mathbf{I} \in \mathbb{R}^{m \times k}$, $\mathbf{u} \in \mathbb{R}^n$, and $\mathbf{i} \in \mathbb{R}^m$ are the matrices of the users' and items' latent factors, and the vectors of users' and items' biases, respectively. Notice Likes is the predicate that implements f_{target}. Then, $\mathcal{G}(\text{LikesGenre}) : u, g \mapsto \mathbf{G}_{u,g}$, where $\mathbf{G} \in \{0, 1\}^{n \times N_g}$, namely LikesGenre is grounded onto a function that takes as input a user index u and a genre index g and returns the prediction contained in matrix $\mathbf{G}$ for user u and genre g. In particular, $\mathbf{G}$ can be seen as a *lookup* table containing the *binarized*[8] predictions of a pre-trained genre classifier. LTN has shown to work better with *binarized* outputs as the classifier was returning predictions too near the decision boundary for LTN to understand[9] the difference between *like* and *dislike*. Note LikesGenre is the predicate that implements f_{source}. Finally, $\mathcal{G}(HasGenre) : i, g \mapsto \{0, 1\}$, namely HasGenre is grounded onto a function that takes as input a movie index i and a genre index g and returns one if the movie i belongs to genre g, zero otherwise. Note HasGenre is the predicate that implements *content*. Intuitively, $\mathcal{G}(\text{LikesGenre})$ contains the knowledge pre-trained on the source domain. In contrast, $\mathcal{G}(\text{Likes} | \mathbf{U}, \mathbf{I}, \mathbf{u}, \mathbf{i})$ represents the MF model we need to train on the target domain.

Intuitively, Axiom (2) forces Likes to be true for each positive user-item pair in $\mathcal{D}_+$, while Axiom (3) forces Likes to be false for each negative user-item pair in $\mathcal{D}_-$. In other words, by maximizing the satisfaction of Axiom (2) and Axiom (3), the model learns to factorize the user-item matrix using the ground truth. In contrast, Axiom (4) is designed to transfer knowledge from the

[7] Notice that Likes can be any function returning a score for a user-item pair. In this work, we use an MF model. This has not to be intended as a limit of our approach as any other state-of-the-art model could be used in principle.

[8] A *binarized* prediction is obtained by using the decision boundary of the classifier on the output of the model to get values in $\{0, 1\}$.

[9] For a binary classifier, 0.45 and 0.55 are predictions belonging to different classes. For a logical framework, those values represent similar truth values.

source domain to the target domain through logical reasoning. Specifically, it forces Likes to be false whenever a user u does not[10] like at least one genre g of a movie i. Note this axiom is applied only to *unknown* user-item pairs in $\mathscr{D}_?$. In fact, when no movie ratings are available on the target domain, knowing something about movie genre preferences is better than knowing nothing. In other words, we believe transferring knowledge from the source domain is crucial when data is missing in the target domain.

4.3. Learning of the LTN

The objective of our model is to learn $\mathscr{G}(\text{Likes}\,|\mathbf{U},\mathbf{I},\mathbf{u},\mathbf{i})$ by maximizing the satisfaction of the knowledge base. In other words, LTN seeks to minimize the following loss function:

$$\boldsymbol{L}(\boldsymbol{\theta}) = (1 - \text{SatAgg}_{\phi\in\mathscr{K}} \mathscr{G}_{\substack{(user_+,movie_+)\leftarrow\mathscr{B}_+ \\ (user_-,movie_-)\leftarrow\mathscr{B}_- \\ (user_?,movie_?)\leftarrow\mathscr{B}_?}}(\phi|\boldsymbol{\theta})) + \lambda||\boldsymbol{\theta}||^2 \tag{5}$$

where $\mathscr{B}_*$ denotes a batch of training examples randomly sampled from $\mathscr{D}_*$. The notation $(user_*, movie_*) \leftarrow \mathscr{B}_*$ denotes that variables $user_*$ and $movie_*$ are grounded with actual user-movie pairs coming from the corresponding batch $\mathscr{B}_*$, where $* \in \{+,-,?\}$. Notice the loss does not specify how the variable *genre* is grounded. At each training step, we ground it with the sequence of all the movie genre indexes in the dataset. Note $\mathscr{B}_?$ is created by uniformly sampling user-item pairs from $\mathscr{D}_?$ at each training step. While all the user-item pairs in $\mathscr{D}_+$ and $\mathscr{D}_-$ are iterated at each epoch, going through all the possible *unknown* pairs is unnecessary and would be unfeasible. In this sense, $\mathscr{B}_?$ has not to be considered a *mini-batch* in the usual sense.

5. Experiments

This section presents the experiments we performed with our method. They have been executed on an Apple MacBook Pro (2019) with a 2,6 GHz 6-Core Intel Core i7. The models have been implemented in Python using PyTorch. In particular, we used the `LTNtorch`[11] library [38]. Moreover, we used Weights and Biases (WandB) for hyperparameter optimization. Our source code is freely available[12].

[10]Note the negated formula is used on purpose, as it is likely that a user dislikes the majority of movies belonging to a genre she dislikes. For example, if u does not like *horror*, likely, she will not like all the horror movies. Moreover, it is more critical to avoid recommending something users do not like than not recommending something they like. This will restrict the recommendation to a few positive movies.

[11]https://github.com/logictensornetworks/LTNtorch

[12]https://github.com/tommasocarraro/NESYKnowledgeTransfer

5.1. Datasets

To perform our experiments, we selected MindReader[13] (`MR`) [30], a novel dataset containing ratings from real users for movies and *non-recommendable* entities, such as movie genres, actors, and producers. We performed experiments on both `MR-100k` and `MR-200k`, the two available versions of the dataset. We used the ratings on movie genres as the source domain[14] and the movie ratings as the target domain. To guarantee the users in the source and target domains totally overlapped, we removed the users that only rated movie genres or movies. After this pre-processing, `MR-100k` (resp. `MR-200k`) comprised 962 (resp. 2,182) users, 3,034 (resp. 3,806) movies, and 140 (resp. 159) movie genres. The density of user-movie ratings was 0.62% (resp. 0.58%), while for user-genre ratings was 8.09% (resp. 6.37%). Selecting ratings on movie genres as the source domain allowed us to use particularly dense information[15] for knowledge transfer. When $density(source) \gg density(target)$, knowledge transfer is more likely to be effective [39].

`MR` provides three types of ratings: *likes* (1), *unknown* (0), and *dislikes* (-1). As in [30], we removed the unknown ratings. After that, we changed the label for negative ratings from -1 to 0. As the dataset provides binary explicit feedback, we treated the recommendation problem as a binary classification task[16], where one has to predict whether a user likes or dislikes an item. This choice allowed us to use the focal loss (Equation (1)) to train the MF models and the *F-measure* as an evaluation metric. This helped in dealing with class imbalance. The class imbalance ratio in `MR-100k` (resp. `MR-200k`) is 21%(-)/79%(+) (resp. 20%(-)/80%(+)) for movie genres, and 38%(-)/62%(+) (resp. 36%(-)/64%(+)) for movies. In both cases, the negative class is the minority one. Hence, we used it as the positive one to compute evaluation metrics in Table 2.

As the *splitting* strategy for the target domain, we randomly sampled 20% of the movie ratings from each user to construct the test set. Then, we randomly sampled 10% of the remaining movie ratings from each user to construct the validation set. Instead, for the source domain, we only created the validation set by randomly sampling 20% of the movie genre ratings from each user. The test

[13]https://mindreader.tech/dataset/

[14]Notice that our approach is flexible on the type of knowledge that has to be transferred. In this work, we use movie genre ratings, but every type of rating (e.g., ratings on books, actors) or side information can be used in principle. One has just to change the knowledge base formalization accordingly.

[15]Notice ratings on movie genres are dense as they are easily obtainable. It is more likely a user will provide a rating about some genre over hundreds rather than some movie (or actor) over thousands.

[16]MindReader provides binary explicit ratings rather than usual 1-5 star ratings. For this reason, the recommendation task can be interpreted as a binary classification problem rather than a regression one. Finally, we work in a rating prediction task rather than ranking as the feedback is clearly explicit.

set is not needed in the source domain, as we only need a validation set to find the optimal hyper-parameters for the pre-trained model.

5.2. Experimental setting

Our experiment compares the proposed Neuro-Symbolic approach, denoted as NESY_{MF}, with a baseline MF model, denoted as MF, to check if NESY_{MF} can effectively transfer knowledge from source to target domain and improve the performance when training data becomes scarce. Specifically, the experiment consists of the following pipeline: (1) additional training sets are generated by randomly sampling the 50%, 20%, 10%, and 5% of the movie ratings[17] from the entire training set, referred to as 100%. Notice ratings are sampled independently from the user, differently from the splitting strategy explained previously. Then, (2) for each training set $Tr \in \{100\%, 50\%, 20\%, 10\%, 5\%\}$ and for each model $m \in \{\text{MF}, \text{NESY}_{\text{MF}}\}$: (2*a*) m is trained on Tr using hyper-parameters found through a bayesian search. Finally, (2*b*) m is evaluated on the test set. Note that for NESY_{MF}, step (2*a*) consists of two steps: (*i*) a standard MF model is pre-trained on the source domain to populate matrix $\mathbf{G}$, then (*ii*) NESY_{MF} is trained on the target domain, namely Tr. We repeated the entire procedure 30 times using seeds from 0 to 29. The test metrics have been averaged across these runs and reported in Table 2.

5.3. Training details

All the models have been trained for 500 epochs using the `Adam` optimizer. *Early stopping* has been used to stop the training if no improvements were found on the validation set for ten epochs. For all the models, the user and item latent factors, $\mathbf{U}$ and $\mathbf{I}$, and the user and item biases, $\mathbf{u}$ and $\mathbf{i}$, have been randomly initialized using the *Glorot* initialization. The MF models have been trained using Equation (1), while NESY_{MF} using Equation (5). For NESY_{MF}, Axiom (4) has been added to the loss from epoch five[18], allowing LTN to learn something about the latent factors before starting reasoning on the acquired knowledge.

We used Bayesian optimization to find the optimal hyper-parameters for our models. We executed every hyper-parameter search for 150 runs and selected the configuration that led to the best validation score. Due to computational time, the searches have been conducted only for the first seed of the experiment and just for the complete dataset (i.e., 100% ratings). The best hyper-parameters found for the models have been then used in the rest of the experiment. For all the models, we tried a number of latent factors $k \in \{5, 10, 25, 50\}$, regularization coefficient $\lambda \in \{0.01, 0.001, 0.0001, 0.00005\}$, learning rate $\eta \in \{0.01, 0.001, 0.0001\}$, training batch size $\beta \in \{64, 128, 256\}$. For the MF models, we additionally tried focal loss hyper-parameters $\alpha \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and $\gamma \in \{0, 1, 2, 3\}$. For the MF model trained on the source domain, we also tried different thresholds for the decision boundary $t \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$. Due to the huge class imbalance in the source domain, we preferred finding a threshold to maximize *precision* rather than *recall*[19]. Finally, for NESY_{MF}, we additionally tried different values for hyper-parameter $p \in \{2, 4, 6, 8, 10\}$ of ME_p and M_p. Table 1 presents the best hyper-parameters found for the MF model trained on the source domain (pre-trained model) and the models trained on the target domain (i.e., MF and NESY_{MF}).

For the MF model trained on the source domain, we used *F0.5-measure*[20] as the validation metric, while in all the other cases, we used *F1-measure*. Clearly, the performance of NESY_{MF} depends on the quality of $\mathbf{G}$ (i.e., the pre-trained model). Using *F0.5-measure* allowed us to obtain more precise predictions for $\mathbf{G}$. In particular, we reduced the number of *false positives*, namely cases in which $\mathbf{G}$ erroneously predicts that a user dislikes a genre. In such cases, Axiom (4) would have been unreliably applied.

6. Results

The results obtained with our experiments are summarized in Table 2. The discussion is limited to `MR-200k` as for `MR-100k` we obtained similar results. Note the results for `MR-100k` are better since the dataset is slightly denser. By looking at the *F1-measure*, it is possible to observe that NESY_{MF} outperforms MF on all five folds. In particular, the performance gap increases with the sparsity of the user-item ratings, starting from a 1.04% improvement on the full dataset (i.e., 100% fold) and ending with a 6.69% improvement on the most sparse dataset (i.e., 5% fold). This shows the benefits of transferring knowledge from a denser domain when training data is poor. Moreover, it suggests our proposal can be effectively used in the task of knowledge transfer for recommendation.

Interestingly, by looking at *recall*, it is possible to observe that the addition of knowledge helps NESY_{MF} in

[17] Notice the ratings on movie genres are kept untouched as they are used for accurate pre-training.

[18] Notice this is an arbitrary choice. The idea of using knowledge transfer is to correct the misclassifications made by the MF model when training data are sparse. The more the data sparsity, the more likely the MF will erroneously classify user-item pairs. During the first steps of learning, the MF has not learned enough information to predict accurately. For this reason, it is likely knowledge transfer is applied to random predictions, hence useless.

[19] By maximizing *precision*, we obtained a more accurate pre-trained model for predicting user-genre preferences. This helped in transferring knowledge more effectively.

[20] *F0.5-measure* gives more weight to *precision* than *recall*. It is used when avoiding *false positives* is particularly important.

Table 1
Best hyper-parameters (referred to as h-p) found through Bayesian optimization for the pre-trained model (left) and the models learned on the target domain (right). The hyper-parameters are subdivided by dataset.

(a) Source domain

h-p	MF	
	MR-100k	MR-200k
k	50	25
λ	0.0001	0.00005
η	0.01	0.01
β	128	64
α	0.4	0.3
γ	1	1
t	0.5	0.4

(b) Target domain

h-p	MF		NESY_{MF}	
	MR-100k	MR-200k	MR-100k	MR-200k
k	50	50	50	50
λ	0.00005	0.00005	0.0001	0.00005
η	0.01	0.01	0.001	0.0001
β	256	256	256	128
α	0.3	0.3	-	-
γ	2	0	-	-
p	-	-	10	10

Table 2
Comparison of MF and NESY_{MF} on the selected datasets. The test metrics are averaged across 30 runs.

(a) MR-100k

Fold	Metric	MF	NESY_{MF}
100%	Precision	**0.5595**$_{(0.0592)}$	0.5432$_{(0.0559)}$
	Recall	0.7377$_{(0.0452)}$	**0.7848**$_{(0.0395)}$
	F1-measure	0.6328$_{(0.0322)}$	**0.6392**$_{(0.0349)}$
50%	Precision	**0.5410**$_{(0.0580)}$	0.5297$_{(0.0512)}$
	Recall	0.6723$_{(0.0459)}$	**0.7054**$_{(0.0260)}$
	F1-measure	0.5956$_{(0.0276)}$	**0.6030**$_{(0.0307)}$
20%	Precision	**0.5081**$_{(0.0558)}$	0.4946$_{(0.0532)}$
	Recall	0.5948$_{(0.0411)}$	**0.6529**$_{(0.0249)}$
	F1-measure	0.5444$_{(0.0261)}$	**0.5605**$_{(0.0311)}$
10%	Precision	**0.4856**$_{(0.0545)}$	0.4689$_{(0.0545)}$
	Recall	0.5468$_{(0.0357)}$	**0.6456**$_{(0.0322)}$
	F1-measure	0.5112$_{(0.0261)}$	**0.5407**$_{(0.0349)}$
5%	Precision	**0.4606**$_{(0.0476)}$	0.4474$_{(0.0522)}$
	Recall	0.5284$_{(0.0250)}$	**0.6584**$_{(0.0286)}$
	F1-measure	0.4897$_{(0.0213)}$	**0.5304**$_{(0.0358)}$

(b) MR-200k

Fold	Metric	MF	NESY_{MF}
100%	Precision	**0.5640**$_{(0.0560)}$	0.5531$_{(0.0503)}$
	Recall	0.7555$_{(0.0333)}$	**0.7931**$_{(0.0207)}$
	F1-measure	0.6431$_{(0.0341)}$	**0.6498**$_{(0.0334)}$
50%	Precision	**0.5510**$_{(0.0554)}$	0.5415$_{(0.0491)}$
	Recall	0.6893$_{(0.0332)}$	**0.7240**$_{(0.0195)}$
	F1-measure	0.6095$_{(0.0300)}$	**0.6178**$_{(0.0306)}$
20%	Precision	**0.5093**$_{(0.0499)}$	0.5050$_{(0.0474)}$
	Recall	0.5838$_{(0.0302)}$	**0.6459**$_{(0.0171)}$
	F1-measure	0.5413$_{(0.0231)}$	**0.5650**$_{(0.0286)}$
10%	Precision	**0.4806**$_{(0.0469)}$	0.4725$_{(0.0441)}$
	Recall	0.5359$_{(0.0283)}$	**0.6067**$_{(0.0193)}$
	F1-measure	0.5041$_{(0.0202)}$	**0.5295**$_{(0.0258)}$
5%	Precision	**0.4508**$_{(0.0438)}$	0.4422$_{(0.0428)}$
	Recall	0.5049$_{(0.0257)}$	**0.5953**$_{(0.0203)}$
	F1-measure	0.4740$_{(0.0193)}$	**0.5057**$_{(0.0273)}$

finding more negative items compared to MF. In particular, the more the dataset is sparse, the more the additional knowledge affects *recall*. In fact, for the 100% fold, the gap is minimal (i.e., 4.98% improvement), meaning the dataset is dense enough for the MF model to learn movie-genre preferences autonomously. Instead, in the 5% fold, MF does not have enough data to capture these patterns. Hence, the acquired knowledge becomes crucial (i.e., 17.90% improvement) to find more negative items.

To conclude, when recommending, due to the huge number of items available in the catalog, one has to filter out as many negative items as possible to focus the model's attention on a small number of positive items that can be confidently recommended. Our model exhibits this behavior by drastically improving the *recall* (i.e., 17.90% improvement on 5% fold) at a minimal cost in *precision* (i.e., 1.91% decrease). This decrease in *precision* could be explained by the fact that when using Axiom (4), the model cannot be completely sure (i.e., having high confidence) that a user really dislikes an item, only knowing she disliked one genre of that item. It is likely, but not certain.

7. Conclusions and future works

In this paper, we presented a Neuro-Symbolic approach to knowledge transfer for RSs. Specifically, we used a Logic Tensor Network to encode axiomatic knowledge suitable for transferring information from source to target domain via logical reasoning. We showed that our model outperforms a standard MF model on all the presented tasks, proving its potential in the knowledge transfer task. Moreover, an experiment that drastically reduced the user-item ratings in the dataset showed the ability of our proposal to deal with data sparsity. In the future, we would like to extend our model to the cross-domain recommendation task. In particular, instead of using movie-genre preferences, one could use more usual source domains (e.g., books, songs).

References

[1] F. Ricci, L. Rokach, B. Shapira, Recommender Systems: Introduction and Challenges, Springer US, Boston, MA, 2015, pp. 1–34. doi:10.1007/978-1-4899-7637-6_1.

[2] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, WWW '01, Association for Computing Machinery, New York, NY, USA, 2001, p. 285–295. URL: https://doi.org/10.1145/371920.372071. doi:10.1145/371920.372071.

[3] Y. Koren, R. Bell, Advances in Collaborative Filtering, Springer, Boston, MA, 2011, pp. 145–186. doi:10.1007/978-0-387-85820-3_5.

[4] F. Aiolli, Efficient top-n recommendation for very large scale binary rated datasets, in: Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 273–280. doi:10.1145/2507157.2507189.

[5] M. Polato, F. Aiolli, Boolean kernels for collaborative filtering in top-n item recommendation, Neurocomput. 286 (2018) 214–225. doi:10.1016/j.neucom.2018.01.057.

[6] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37. doi:10.1109/MC.2009.263.

[7] S. Rendle, Factorization machines, in: 2010 IEEE International Conference on Data Mining, 2010, pp. 995–1000. doi:10.1109/ICDM.2010.127.

[8] X. Ning, G. Karypis, Slim: Sparse linear methods for top-n recommender systems, in: 2011 IEEE 11th International Conference on Data Mining, 2011, pp. 497–506. doi:10.1109/ICDM.2011.134.

[9] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444. doi:10.1038/nature14539.

[10] H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, AAAI Press, 2017, p. 3203–3209. doi:10.24963/ijcai.2017/447.

[11] T. Carraro, M. Polato, L. Bergamin, F. Aiolli, Conditioned variational autoencoder for top-n item recommendation, in: E. Pimenidis, P. Angelov, C. Jayne, A. Papaleonidas, M. Aydin (Eds.), Artificial Neural Networks and Machine Learning – ICANN 2022, Springer Nature Switzerland, Cham, 2022, pp. 785–796. doi:10.1007/978-3-031-15931-2_64.

[12] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, p. 689–698. doi:10.1145/3178876.3186150.

[13] Z. Zeng, C. Xiao, Y. Yao, R. Xie, Z. Liu, F. Lin, L. Lin, M. Sun, Knowledge transfer via pre-training for recommendation: A review and prospect, Frontiers in Big Data 4 (2021). URL: https://www.frontiersin.org/articles/10.3389/fdata.2021.602071. doi:10.3389/fdata.2021.602071.

[14] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (2010) 1345–1359. doi:10.1109/TKDE.2009.191.

[15] T. Man, H. Shen, X. Jin, X. Cheng, Cross-domain recommendation: An embedding and mapping approach, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, 2017, pp. 2464–2470. URL: https://doi.org/10.24963/ijcai.2017/343. doi:10.24963/ijcai.2017/343.

[16] H. Kanagawa, H. Kobayashi, N. Shimizu, Y. Tagami, T. Suzuki, Cross-domain recommendation via deep domain adaptation, in: L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, D. Hiemstra (Eds.), Advances in Information Retrieval, Springer International Publishing, Cham, 2019, pp. 20–29. doi:10.1007/978-3-030-15719-7_3.

[17] F. Yuan, L. Yao, B. Benatallah, Darec: Deep domain adaptation for cross-domain recommendation via transferring rating patterns, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 4227–4233. URL: https://doi.org/10.24963/ijcai.2019/587. doi:10.24963/ijcai.2019/587.

[18] C. Gao, X. Chen, F. Feng, K. Zhao, X. He, Y. Li, D. Jin, Cross-domain recommendation without sharing user-relevant data, in: The World Wide Web Conference, WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 491–502. URL: https://doi.org/10.1145/3308558.3313538. doi:10.1145/3308558.3313538.

[19] A. S. d'Avila Garcez, K. Broda, D. M. Gabbay, Neural-symbolic learning systems - foundations and applications, in: Perspectives in Neural Computing, 2012. doi:10.1007/978-1-4471-0211-3.

[20] I. Donadello, L. Serafini, A. d'Avila Garcez, Logic tensor networks for semantic image interpretation, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, 2017, pp. 1596–1602. URL: https://doi.org/10.24963/ijcai.2017/221. doi:10.24963/ijcai.2017/221.

[21] M. K. Sarker, L. Zhou, A. Eberhart, P. Hitzler, Neuro-symbolic artificial intelligence: Current trends,

ArXiv abs/2105.05330 (2021).

[22] T. Campari, L. Lamanna, P. Traverso, L. Serafini, L. Ballan, Online learning of reusable abstract models for object goal navigation, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2022, pp. 14850–14859. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01445. doi:10.1109/CVPR52688.2022.01445.

[23] T. Carraro, A. Daniele, F. Aiolli, L. Serafini, Logic tensor networks for top-n recommendation, in: AIxIA 2022 – Advances in Artificial Intelligence: XXIst International Conference of the Italian Association for Artificial Intelligence, AIxIA 2022, Udine, Italy, November 28 – December 2, 2022, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2023, p. 110–123. URL: https://doi.org/10.1007/978-3-031-27181-6_8. doi:10.1007/978-3-031-27181-6_8.

[24] 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, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 616–621. URL: https://doi.org/10.1145/3523227.3551484. doi:10.1145/3523227.3551484.

[25] P. Kouki, S. Fakhraei, J. Foulds, M. Eirinaki, L. Getoor, Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems, in: Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 99–106. URL: https://doi.org/10.1145/2792838.2800175. doi:10.1145/2792838.2800175.

[26] H. Chen, S. Shi, Y. Li, Y. Zhang, Neural collaborative reasoning, in: Proceedings of the Web Conference 2021, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 1516–1527. URL: https://doi.org/10.1145/3442381.3449973. doi:10.1145/3442381.3449973.

[27] A. Daniele, L. Serafini, Knowledge enhanced neural networks, in: A. C. Nayak, A. Sharma (Eds.), PRICAI 2019: Trends in Artificial Intelligence, Springer International Publishing, Cham, 2019, pp. 542–554. doi:10.1007/978-3-030-29908-8_43.

[28] S. Badreddine, A. d'Avila Garcez, L. Serafini, M. Spranger, Logic tensor networks, Artificial Intelligence 303 (2022) 103649. URL: https://www.sciencedirect.com/science/article/pii/S0004370221002009. doi:https://doi.org/10.1016/j.artint.2021.103649.

[29] M. Ferrari Dacrema, P. Cremonesi, D. Jannach, Are we really making much progress? a worrying analysis of recent neural recommendation approaches, in: Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 101–109. URL: https://doi.org/10.1145/3298689.3347058. doi:10.1145/3298689.3347058.

[30] A. H. Brams, A. L. Jakobsen, T. E. Jendal, M. Lissandrini, P. Dolog, K. Hose, Mindreader: Recommendation over knowledge graph entities with explicit user ratings, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 2975–2982. URL: https://doi.org/10.1145/3340531.3412759. doi:10.1145/3340531.3412759.

[31] H. Chen, Y. Li, S. Shi, S. Liu, H. Zhu, Y. Zhang, Graph collaborative reasoning, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, WSDM '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 75–84. URL: https://doi.org/10.1145/3488560.3498410. doi:10.1145/3488560.3498410.

[32] J. Ji, Z. Li, S. Xu, M. Xiong, J. Tan, Y. Ge, H. Wang, Y. Zhang, Counterfactual collaborative reasoning, in: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 249–257. URL: https://doi.org/10.1145/3539597.3570464. doi:10.1145/3539597.3570464.

[33] Y. Xian, Z. Fu, H. Zhao, Y. Ge, X. Chen, Q. Huang, S. Geng, Z. Qin, G. de Melo, S. Muthukrishnan, Y. Zhang, Cafe: Coarse-to-fine neural symbolic reasoning for explainable recommendation, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 1645–1654. URL: https://doi.org/10.1145/3340531.3412038. doi:10.1145/3340531.3412038.

[34] A. Kimmig, S. Bach, M. Broecheler, B. Huang, L. Getoor, A short introduction to probabilistic soft logic, Mansinghka, Vikash, 2012, pp. 1–4. URL: https://lirias.kuleuven.be/retrieve/204697.

[35] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, Proceedings of the AAAI Conference on Artificial Intelligence 28 (2014). URL: https://ojs.aaai.org/index.php/AAAI/article/view/8870. doi:10.1609/aaai.v28i1.8870.

[36] 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, Associa-

tion for Computational Linguistics, Austin, Texas, 2016, pp. 192–202. URL: https://aclanthology.org/D16-1019. doi:10.18653/v1/D16-1019.

[37] E. van Krieken, E. Acar, F. van Harmelen, Analyzing differentiable fuzzy logic operators, Artificial Intelligence 302 (2022) 103602. URL: https://www.sciencedirect.com/science/article/pii/S0004370221001533. doi:https://doi.org/10.1016/j.artint.2021.103602.

[38] T. Carraro, LTNtorch: PyTorch implementation of Logic Tensor Networks, 2023. URL: https://doi.org/10.5281/zenodo.7778157. doi:10.5281/zenodo.7778157.

[39] F. Zhu, Y. Wang, C. Chen, J. Zhou, L. Li, G. Liu, Cross-domain recommendation: Challenges, progress, and prospects, in: Z.-H. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 4721–4728. URL: https://doi.org/10.24963/ijcai.2021/639. doi:10.24963/ijcai.2021/639, survey Track.