

# Neuro-Symbolic Recommender Systems<sup>\*</sup>

Tommaso Carraro<sup>1,2</sup>

<sup>1</sup>Department of Mathematics, University of Padova, Via Trieste, 63, 35121, Padova (PD), Italy

<sup>2</sup>Data and Knowledge Management Unit, Fondazione Bruno Kessler, Via Sommarive, 18, 38123, Povo (TN), Italy

## Abstract

Despite being studied for over twenty years, Recommender Systems (RSs) still suffer from important issues that limit their applicability in real-world scenarios. Data sparsity, cold start, and explainability are some of the most impacting problems. Intuitively, these historical limitations can be mitigated by injecting prior knowledge into recommendation models. Neuro-Symbolic (NeSy) approaches are suitable candidates for achieving this goal. However, the application of such systems to RSs is still in its early stages, and most of the proposed architectures do not really exploit the advantages of a NeSy approach. To this end, we conducted preliminary experiments with a Logic Tensor Network (LTN), a novel NeSy framework. We used the LTN to train a *vanilla* Matrix Factorization model using a First-Order Logic knowledge base as an objective. In particular, we encoded facts to enable the regularization of the latent factors using content information, obtaining promising results. In this paper, we show our preliminary results with the LTN and propose interesting future works in this novel research area.

## Keywords

recommender systems, neuro-symbolic integration, logic tensor networks, first-order logic

## 1. Introduction

Recommender Systems (RSs) are nowadays essential tools for e-services. Specifically, they aim to mitigate information overload by suggesting novel items to users based on their preferences. Since the beginning of the RSs literature, Collaborative Filtering (CF) has been affirmed to be the *de facto* recommendation approach. Latent Factor Models, particularly Matrix Factorization (MF) [1], have been proven to be the most successful CF technique, and this has been further emphasized with the deep learning rise [2].

Despite being studied for over twenty years, state-of-the-art recommendation models still suffer from historical limitations that limit their applicability in real-world scenarios. Among the well-known issues of RSs, there are data sparsity, cold-start, and explainability.

Intuitively, these issues can be mitigated by providing additional knowledge to the model. Such knowledge should be carefully designed to compensate for data scarcity, provide additional information when no ratings are available, or enhance model transparency. We believe Neuro-Symbolic (NeSy) [3] approaches are promising candidates for injecting knowledge inside models. Specifically, they aim to integrate learning and (symbolic) reasoning to obtain the best from both worlds. In particular, (i) symbolic methods typically work well with poor training data,


---

*AIxIA'23: 22nd International Conference of the Italian Association for Artificial Intelligence, November 06–09, 2023, Rome, Italy*

<sup>\*</sup> Doctoral project supervised by Luciano Serafini (Fondazione Bruno Kessler) and Fabio Aioli (University of Padova).

✉ [tcarraro@fbk.eu](mailto:tcarraro@fbk.eu) (T. Carraro)

ORCID [0000-0002-3043-1456](https://orcid.org/0000-0002-3043-1456) (T. Carraro)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

while neural networks struggle. Then, (ii) symbolic methods are usually explainable by design, while neural networks are *black-boxes*. Finally, (iii) symbolic methods manage to work in the absence of data (i.e., *zero-shot* learning), while for neural networks, it is impossible. Ideally, by integrating these two paradigms, it could be possible to obtain a recommendation engine that can deal well with sparsity, address cold-start cases, and make its predictions less opaque.

Many Neuro-Symbolic recommenders have been proposed recently [4, 5, 6]. However, these methods do not totally exploit the advantages of the NeSy system, or they have not been applied to overcome the aforementioned limitations, where we believe they are particularly suited. For example, they implement the symbolic part using neural networks or use too simple logic (e.g., propositional logic) that is not expressive enough to model real-world problems [4]. To this end, in a preliminary experiment, we used a Logic Tensor Network (LTN) [7, 8] to encode logical axioms for enabling regularization of a *vanilla* Matrix Factorization (MF) based on logical reasoning. We obtained promising results, showing that the benefits of the encoded knowledge increase with the sparsity of the user-item ratings.

## 2. Preliminary experiments

In our preliminary experiments on Neuro-Symbolic Integration, we selected Logic Tensor Networks (LTN) [7] as the NeSy framework to perform recommendations. LTN allows learning a neural model using the satisfaction of a FOL knowledge base as an objective. Specifically, it defines a FOL language, called Real Logic, that allows mapping every symbolic expression to the domain of real numbers. By doing so, the logical formulas in the knowledge base form a computational graph that can be used for gradient-based optimization. To this end, Real Logic defines the *grounding* function  $\mathcal{G}$ , which defines the mapping between the symbolic and real domains. Specifically, individuals are mapped to tensors of real values, variable symbols to sequences of individuals, functional symbols to real functions, and predicate symbols to real functions with output in  $[0., 1.]$ . Then, connectives (i.e.,  $\wedge, \vee, \neg, \implies$ ) are mapped to fuzzy semantics [9], while quantifiers (i.e.,  $\forall, \exists$ ) are mapped to special aggregation functions (e.g., generalized means).

Intuitively, functional and predicate symbols can be represented as neural networks parameterized by  $\theta$ . We refer to  $\mathcal{G}(s|\theta)$  as a parametric grounding, meaning symbol  $s$  depends on some parameters  $\theta$  that can be learned. LTN allows learning parametric groundings by maximally satisfying a specified knowledge base  $\mathcal{K} = \{\phi_1, \dots, \phi_n\}$ , where  $\phi_1, \dots, \phi_n$  are closed formulas. More formally, the objective of the LTN is  $\theta^* = \operatorname{argmax}_{\theta} \operatorname{SatAgg}_{\phi \in \mathcal{K}} \mathcal{G}(\phi|\theta)$ , where  $\mathcal{G}(\phi|\theta)$  means formula  $\phi$  includes some functional or predicate symbols parameterized by  $\theta$ .  $\operatorname{SatAgg} : [0., 1.]^* \mapsto [0., 1.]$  is a formula aggregating operator, usually defined with  $\operatorname{ME}_p$  [7], namely the fuzzy operator that represents  $\forall$ .

In what follows, we refer to  $\operatorname{Likes}(u, i)$  as a binary predicate returning whether a user  $u$  likes an item  $i$ . Note that  $\mathcal{G}(\operatorname{Likes}|\theta)$  can be any recommendation model returning the prediction for a user-item pair in the dataset. In our experiments, we implemented  $\mathcal{G}(\operatorname{Likes}|\theta)$  as a Matrix Factorization model. Specifically,  $\mathcal{G}(\operatorname{Likes}|\theta) : u, i \mapsto \sigma(\mathbf{U}_u \cdot \mathbf{I}_i^\top + \mathbf{u}_u + \mathbf{i}_i)$ , where  $\mathbf{U} \in \mathbb{R}^{n \times k}$ ,  $\mathbf{I} \in \mathbb{R}^{m \times k}$ ,  $\mathbf{u} \in \mathbb{R}^n$ , and  $\mathbf{i} \in \mathbb{R}^m$  are the users' and items' latent factors, and users' and items' biases, respectively.  $n$  denotes the number of users,  $m$  the number of items, and  $k$  the number of

latent factors.  $\sigma$  is the logistic function. Specifically,  $\sigma$  allows Likes to be interpreted as a fuzzy predicate.

## 2.1. Recommendation loss function definition

LTN allows defining the recommendation objective as a set of logical axioms. Thanks to the expressiveness of FOL, one can express fine-grained constraints that can represent complex loss functions. For example, the loss function for training an MF model could be the following.

$$\forall(u_+, i_+) \text{ Likes}(u_+, i_+) \quad (1)$$

$$\forall(u_-, i_-) \neg \text{ Likes}(u_-, i_-) \quad (2)$$

Intuitively,  $u_+$  and  $i_+$  are variable symbols denoting positive user-item pairs, while  $u_-$  and  $i_-$  denote negative user-item pairs. Axiom (1) states that for each positive user-item pair, the prediction of the MF model should be a positive truth value (i.e., the user should like the item). In contrast, Axiom (2) states that for each negative user-item pair, the prediction of the MF model should be a negative truth value since the loss imposes maximizing the negation of Likes. In other words, by satisfying this knowledge base, the LTN learns how to train the MF to factorize the user-item matrix using the ground truth (i.e., the target ratings).

## 2.2. Model regularization by logical reasoning

Encoding additional information to regularize the recommendation model is straightforward. This can be done by encoding additional axioms that enable logical reasoning based on side or content information. Following this intuition, we conducted a preliminary experiment [6] to see if LTN could enable an underlying MF model reasoning about some additional content information. In particular, an experiment that drastically reduces the density of user-item ratings showed that the benefits of the encoded knowledge increase with the sparsity of the dataset, proving our NeSy approach has been beneficial in dealing with sparsity. Specifically, we added the following axiom to the previous knowledge base. Note the experiment has been conducted on MindReader<sup>1</sup>, a movie recommendation dataset providing ratings for both movies and movie genres.

$$\forall(u_?, i_?) (\exists g \neg \text{ LikesGenre}(u_?, g) \wedge \text{ HasGenre}(i_?, g)) \implies \neg \text{ Likes}(u_?, i_?) \quad (3)$$

In the formalization,  $u_?$  and  $i_?$  are variable symbols denoting user-movie pairs for which the rating is unknown, while  $g$  is a variable symbol denoting the movie genres of the movies of the dataset. Then, LikesGenre is a fixed (i.e., not learnable) binary predicate returning one if user  $u$  likes genre  $g$ , zero otherwise. Similarly, HasGenre is a fixed binary predicate returning one if a movie  $i$  belongs to genre  $g$ , zero otherwise. Note these two predicates can be easily implemented as *lookup* tables filled with data from the dataset.

Intuitively, Axiom (3) states that every time there is a user-movie pair for which we do not have information (i.e., the rating is missing), if we know user  $u_?$  does not like some genre  $g$  of the

<sup>1</sup><https://mindreader.tech/dataset/>

associated movie  $i_?$ , then  $u_?$  should not like  $i_?$ . This formula enables the underlying MF model to reason about relationships between users, movies, and movie genres. In this sense, it acts as a kind of *logical* regularization for the latent factors of the MF. In particular, we designed this axiom based on the idea that when no ratings are available, knowing something about movie genres is better than knowing nothing. This intuition has been evidenced by our results [6], which prove that the addition of the formula has been crucial to deal with sparsity.

### 3. Proposed directions

This section proposes possible extensions of our NeSy model to solve different recommendation tasks.

#### 3.1. Hybrid recommendation

In Section 2.1, we showed how LTN could be used to implement a recommendation model based on Collaborative Filtering. It is well-known that CF cannot deal with cold-start cases, as collaborative information for newly added users and items is unavailable. For this reason, hybrid recommendation models have been proposed. The idea is to merge CF with content-based recommendations to deal with cold-start while maintaining all the advantages of CF. Implementing this idea in LTN is straightforward. It involves adding the following axiom to the knowledge base.

$$\forall u, i, i_{cold} \text{Likes}(u, i) \wedge \text{Sim}(i, i_{cold}) \implies \text{Likes}(u, i_{cold}) \quad (4)$$

Intuitively, Axiom (4) states that for each triple  $(u, i, i_{cold})$ , where  $u$  is a user,  $i$  is an item, and  $i_{cold}$  a cold-start item, if  $u$  likes  $i$ , and  $i$  and  $i_{cold}$  are similar, then  $u$  should like  $i_{cold}$  too. Note the predicate Sim can be implemented as a similarity measure based on content (e.g., movie genres in common) or latent (i.e., pre-trained embeddings) information. The only constraint is the output has to be in the range  $[0., 1.]$  as it has to be interpreted as a logical predicate by LTN. Clearly, this formula will help the model deal with cold-start item cases, as when no information is available about some ratings, we are using content information to compensate for this. The LTN will perform hybrid recommendations by finding a trade-off between Axiom (4), Axiom (1), and Axiom (2) in the objective. Note that a similar idea can also be used to deal with cold-start users.

#### 3.2. Cross-domain recommendation

Cross-domain recommendation aims at mitigating data sparsity by transferring knowledge acquired from other domains (e.g., books, songs) to the target domain (e.g., movies). The source domain is usually denser than the target domain, as the objective is to compensate for sparsity in the latter. To this end, LTN can be used as an interface for transferring knowledge between domains. To do that, the following axiom can be added to the knowledge base.

$$\forall(u, b, m) \text{Likes}_{\text{source}}(u, b) \wedge \text{Sim}(b, m) \implies \text{Likes}_{\text{target}}(u, m) \quad (5)$$

In the formalization,  $u$ ,  $b$ , and  $m$  are variable symbols for denoting users, books, and movies, respectively. Then,  $\text{Likes}_{\text{source}}$  is a pre-trained recommendation model on book ratings (i.e., the source domain), while  $\text{Likes}_{\text{target}}$  represents the recommendation model we want to train on the target domain (i.e., movie ratings in this example). Again,  $\text{Sim}$  is a predicate that can be implemented using a similarity measure based on content information (e.g., storyline) or latent structure. Intuitively, Axiom (5) states that every time a user  $u$  likes a book  $b$  in the source domain, if the book is similar to a movie  $m$  in the target domain, then  $u$  should like  $m$ . Clearly, this axiom allows the transfer of information between domains by logical reasoning. We conducted some preliminary experiments based on this idea and obtained promising results. Note one could apply Axiom (5) only when user-movie ratings are missing on the target domain. In such cases, knowing something about the source domain acts as a kind of data augmentation for the target domain and, hence, helps mitigate data sparsity.

### 3.3. Explainable recommendation

LTN has not been designed just for learning. In particular, it also provides the possibility to query the knowledge base after the training phase. Querying is the process of grounding the knowledge base with novel data and checking the satisfaction level of its formulas. We believe this feature can be used to provide explanations for recommendations. For example, after we train a model with Axiom (1), Axiom (2), and Axiom (3), we could check the satisfaction level of the following formula for a test user  $user_{test}$  to which we recommended the movie *Avatar*.

$$\exists g \text{LikesGenre}(user_{test}, g) \wedge \text{HasGenre}(Avatar, g)$$

Intuitively, if the formula is evaluated with a high truth value, it means our test user likes at least one movie genre of *Avatar*. Note that one can go through the computational graph of the formula to understand precisely which genre it is and provide this finding as an explanation.

## 4. Conclusions

In this paper, we proposed different ways to use a Neuro-Symbolic approach to mitigate important recommender systems' limitations and solve interesting recommendation tasks. In particular, we showed the flexibility of LTN in designing heterogeneous recommendation objectives.

## 5. Acknowledgments

I would like to thank my supervisors, Luciano Serafini and Fabio Aiolli, for having proposed working on this stimulating and ambitious research area. Then, I thank my colleague Alessandro Daniele, who provided additional supervision during my Ph.D. journey. Furthermore, I am grateful to the University of Padova for the delivery of this interesting doctoral program and for the excellent formation that was provided during my journey. Finally, I thank Fondazione Bruno Kessler, who sponsored my Ph.D. expeditions worldwide and supported my research career.

## References

- [1] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: 2008 Eighth IEEE International Conference on Data Mining, 2008, pp. 263–272. doi:10.1109/ICDM.2008.22.
- [2] T. Carraro, M. Polato, L. Bergamin, F. Aioli, 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.
- [3] 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.
- [4] 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. doi:10.1145/3442381.3449973.
- [5] 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. doi:10.1145/3523227.3551484.
- [6] T. Carraro, A. Daniele, F. Aioli, 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. doi:10.1007/978-3-031-27181-6\\_8.
- [7] S. Badreddine, A. d’Avila Garcez, L. Serafini, M. Spranger, Logic tensor networks, *Artificial Intelligence* 303 (2022) 103649. doi:https://doi.org/10.1016/j.artint.2021.103649.
- [8] T. Carraro, LTNtorch: PyTorch implementation of Logic Tensor Networks, 2023. doi:10.5281/zenodo.7778157.
- [9] E. van Krieken, E. Acar, F. van Harmelen, Analyzing differentiable fuzzy logic operators, *Artificial Intelligence* 302 (2022) 103602. doi:https://doi.org/10.1016/j.artint.2021.103602.