Recommender Systems based on Neuro-Symbolic Knowledge Graph Embeddings Encoding First-Order Logic Rules

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

[1*]Dipartimento di Informatica, University of Bari Aldo Moro, Via Edoardo Orabona, 4, Bari, 70125, Italy.

*Corresponding author(s). E-mail(s): giuseppe.spillo@uniba.it;
Contributing authors: cataldo.musto@uniba.it;
marco.degemmis@uniba.it; pasquale.lops@uniba.it;
giovanni.semeraro@uniba.it;

Abstract

In this paper, we present a *knowledge-aware recommendation model* based on *neuro-symbolic graph embeddings* that encode *first-order logic rules*. Our approach is based on the intuition that is the basis of neuro-symbolic AI systems: to combine deep learning and symbolic reasoning in one single model, in order to take the best out of both the paradigms. To this end, we start from a knowledge graph (KG) encoding information about users, ratings, and descriptive properties of the items and we design a model that combines *background* knowledge encoded in logical rules mined from the KG with *explicit* knowledge encoded in the triples of the KG itself to obtain a more precise representation of users and items. Specifically, our model is based on the combination of: *(i)* a rule learner, that extracts first-order logic rules based on the information encoded in the knowledge graph; *(ii)* a graph embedding module, that jointly learns a vector-space representation of users and items based on the triples encoded in the knowledge graph and the rules previously extracted; *(iii)* a recommendation module that uses the embeddings to feed a deep learning architecture that provides users with top-k recommendations. In the experimental section, we evaluate the effectiveness of our strategy on three datasets and the results show that the combination of knowledge graph embeddings and first-order logic rules led

to an improvement in the predictive accuracy and in the novelty of the recommendations. Moreover, our approach overcomes several competitive baselines, thus confirming the validity of our intuitions.

Keywords: Recommender Systems, Neuro-Symbolic AI, Knowledge-Aware Recommender Systems, Knowledge Graph, Knowledge Graph Embedding, First-Order Logic, First-Order Logic Rules

Declarations

This paper or a similar version is not currently under review by a journal or conference. This paper is void of plagiarism or self-plagiarism as defined by the Committee on Publication Ethics and Springer Guidelines. A previous and shorter version of the article has been published in the Proceedings of the 16th ACM Conference on Recommender Systems (RecSys 2022)[1], as a Late Breaking Result. The distinctive trait of the current paper is highlighted in the document of self-assessment submitted together with the manuscript.

Author Contributions. Spillo and Musto contributed to the writing of the entire manuscript, supervised the design of the knowledge-aware recommender system based on neuro-symbolic graph embedding with FOL rules, and were responsible for the experimental setting. Spillo was responsible for the development of the entire pipeline and followed the execution of the experiments. De Gemmis, Lops, and Semeraro helped with the finalization of the paper, contributed to the writing, and supported the initial idea development.

[1] https://dl.acm.org/doi/abs/10.1145/3523227.3551484

1 Introduction

Currently, the huge amount of data available on the Internet is too much to be handled by our brains. As stated in Borchers et al (1998), in several situations the users are not able to carry out *decision-making* tasks in an efficient way, and this problem is called *information overload*.

Recommender Systems (RSs) represent the class of algorithms proposed to solve this problem: they are able to build user profiles to match human needs with the information available online, thus supporting the *decision-making* process. We use RSs every day in several aspects of our life, for example, to get suggestions about movies to watch, music to listen to, books and news to read, and so on and so forth.

The earlier approaches to implement these systems relied on either *user-item interactions* and *content-based features*. The first group of approaches, defined as *collaborative-filtering* RSs (Schafer et al (2007)), is based on the intuition that similar users will tend to appreciate and enjoy similar items. Although quite precise in recommending items, this approach is affected by *cold-start* (Lika et al (2014)). In parallel, *content-based* RSs (De Gemmis et al (2015); Musto et al (2022)) emerged since the early 90s. These RSs exploit item descriptions and structured properties (such as the genre of a movie, or the author of a book) to generate representations of the items and build user profiles including their preferences (such as, *horror movies* for a user who expressed this preference); then, a matching module finds the most suitable item for the user by exploiting the representations of the item and the user profile, and finally the user gets the recommendation.

Both approaches evolved over the years: deep learning has been applied to collaborative filtering RSs, and now deep learning matrix factorization models represent the state of the art in this area (Xue et al (2017)); on the other hand, the rich semantics provided by plain text and the advances in natural language processing improved the performance of content-based approaches Lops et al (2009, 2011); Musto et al (2014). Nowadays, one of the most investigated research lines regards *hybrid* RSs, which are RSs that combine more paradigms (e.g., combining collaborative filtering and content-based approaches), in order to take the advantages of both the worlds. The first attempts in this direction, discussed in Sun et al (2019), regard the integration of content-based features (defined as *side information*) in collaborative filtering methods. More recently, this research line has evolved into *Knowledge-Aware Recommender Systems* (KARSs) (Tarus et al (2018)).

KARS are RSs which aim at using several *knowledge sources* to represent users and items and provide effective recommendations by exploiting several kind of knowledge. These methods put their roots in the area of knowledge-based recommender systems (Felfernig et al (2015)), where the knowledge used to drive the recommendation process was typically encoded in the form of constraints. Today, the landscape in the area is much more heterogeneous since KARS can rely on diverse knowledge sources, both structured, such as *Knowledge Graphs* (KGs) (He et al (2020); Vashishth et al (2019)), unstructured, such

as plain text and user reviews (Polignano et al (2021); Asani et al (2021)), and multimodal (Bonnin and Jannach (2014); Andjelkovic et al (2019); Van den Oord et al (2013); Deldjoo et al (2016)). The main intuition behind KARSs is that enriching item representations with the provided knowledge improves the performance of RSs, since the resulting representations are more precise and meaningful. Several works, such as Musto et al (2017), Sun et al (2018), Anelli et al (2019) and Petruzzelli et al (2024) confirm this intuition, generally showing that performance of RSs are improved when knowledge-derived features are encoded in the representation learning process.

In this scenario, a promising research line regards the exploitation of *Knowledge Graph Embedding* (KGE) techniques. These techniques, discussed in Cai et al (2018), aim at representing nodes and relations that exist in a KG into a vector space in which the structured properites of the original KG are still preserved; during the last decade, several approaches have been proposed to effectively learn graph embeddings, starting from the simplest ones, the *geometric* models, such as TransE proposed in Bordes et al (2013), to the most sophisticated ones, based on neural networks, such as CompGCN proposed in Vashishth et al (2019). Such techniques obtained very good performance in different scenarios and tasks where data can be modeled as a graph, including biology and social networks (Goyal and Ferrara (2018)), or RSs (Palumbo et al (2018); Sun et al (2018); Song et al (2019)).

As we said, KGEs showed very good performance in RSs, but our conjecture is that pure *data-driven* and *non-symbolic*[2] representations can be further improved. As stated in Sarker et al (2021), the current wave of *neuro-symbolic AI systems* aim at overcoming the historical dichotomy between *pure symbolic* AI and *machine learning*, by proposing approaches which take the advantages of both the worlds, such as the easy *explainability* of symbolic AI systems, and generalization power of machine learning.

In spite of these considerations, the adoption and implementation of neuro-symbolic AI systems in RSs is currently overlooked. However, this is not surprising since most of RS research has investigated *data-driven* approaches, without considering at all *symbolic* approaches. However, some symbolic representations, such as those based on First-Order Logic (FOL) rules (Smullyan (1995)), could be exploited for RSs as well, to improve the embedding learning process by injecting the *explicit knowledge* provided by them, and obtain more precise users and item representations. For example, let us consider the following FOL rule: *"If you like a movie and the movie has a sequel, then you will probably enjoy the sequel"*[3]; this kind of statement, expressed with the formalism of FOL, might provide some semantics which can improve the embeddings, but these mechanisms are overlooked in the literature related to RSs. Guo et al (2016) and Guo et al (2018) proposed two models to tackle this issue: these models can jointly learn embeddings based on both *explicit knowledge* encoded in a KG, and some *background knowledge* expressed in terms

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

[3] By using FOL formalism, this rule can be written as follows: $like(user, movieA) \wedge sequel(movieB, movieA) \Rightarrow like(user, movieB)$

of FOL rules involving the entities of the KG itself; this is made possible by exploiting a loss function that considers both the knowledge sources.

In this paper, we present a *knowledge-aware recommendation model* leveraging on *neuro-symbolic graph embeddings* exploiting *first-order logic rules*. We start from a KG encoding both user preferences and structured item properties; next, we propose a framework composed of:

1. a Rule Miner, which aims at mining FOL rules from the KG, to obtain background knowledge.
2. a Graph Embedding Learner, which is able to represent the nodes in the KG (in particular, users and items) in a vector space; the key point of this graph embedding learner is that it is able to encode not only explicit information provided by the KG, but it also encodes the background knowledge provided by FOL rules.
3. a Neural Recommender System which uses the embeddings learned by the graph embedding learner to feed a deep architecture to generate recommendation lists for users.

The *novelty* of the work we propose is related to the injection of FOL rules in KGE learning process, which brings into the game information previously not considered, such as constraints and background knowledge. According to the classification of *neuro-symbolic AI systems* presented in Kautz (2022), this recommendation model can be labeled as a *neuro-symbolic* approach: first, it starts with a *symbolic* input (a KG); then, *neural* reasoning is exploited to jointly embed the KG and the symbolic elements (the FOL rules).

In the experimental section, we evaluate the effectiveness of our strategy on three state of the art datasets by analyzing both accuracy and non-accuracy metrics, such as the novelty and the diversity of the recommendations. The results show that the combination of KG embeddings and FOL rules led to an improvement in prediction accuracy and in the novelty of the recommendations. In addition, our approach overcomes several baselines, thus supporting the validity of our intuitions. To sum up, the main contributions of this paper are the following:

- We introduce a recommendation model inspired by *neuro-symbolic AI systems*, that learns a representation of users and items based on the combination of graph embeddings and FOL rules;
- We design different strategies to select FOL rules to be injected in the embeddings;
- We evaluate our strategy against three datasets and we ensured the reproducibility of the experimental protocol.

The rest of the paper is organized as follows. In Section 2 we present related works in the area. Next, the different components of our model are introduced in Section 3. The experimental methodology is presented in Section 4, while the results are shown and discussed in Section 5. Finally, conclusions and future works are sketched in Section 7.

2 Related Work

In this section, we discuss related works. In the following, we first provide some basics of neuro-symbolic AI systems, then we identify relevant literature in the area of RS based on graph embeddings and KARS. For all these works, the distinctive aspects of our approach are highlighted.

2.1 Neuro-Symbolic AI Systems

As discussed in Sarker et al (2021), Neuro-Symbolic Artificial Intelligence, abbreviated as NeSy AI or NSAI, is a research area in the field of Artificial Intelligence (AI) that aims to unify and integrate the *neural* processing and the *symbolic* reasoning. The term *neural* refers to neural networks, which are also said to be *connectionist* systems, while *symbolic* refers to the explicit symbol manipulation, and regards tasks related to graphs (such as those related to this work, in particular with knowledge graphs, natural language question answering, and logic as well).

The increasing interest for this kind of systems is mainly due to the fact that they try to combine the advantages symbolic and neural systems offer. Indeed, as discussed in Sarker et al (2021), neural systems offer the ability to learn patterns from raw data; symbolic systems offer the high explainability, which is one of the most important issues in neural systems (indeed, explainability of neural networks is a very investigated field). The two approaches differ on data representation too: symbolic systems use *explicit* representations, which are easily understandable by humans. As an example, we can consider the following FOL rule: $human(x) \Rightarrow mammal(x)$. Humans can easily understand the meaning of this rule (if x is a human, then x is a mammal), and symbolic systems can easily handle this kind of information. On the other hand, in neural systems the information is represented in the form of *embeddings*, that are *vectors* of real numbers, representing entities in a high-dimensional, continuous, differentiable vector space, and are not at all interpretable by humans; in order to obtain this representaion, a neural network exploits weighed connections between neurons and activation functions which change the values of the embeddings. In order to emphasize the difference in the way neural and symbolic systems represent the information, embeddings are said to be a *sub-symbolic* representations (and, accordingly, neural systems are said to be *sub-symbolic* systems). Finally, while neural systems *learns* pattern in raw data, symbolic systems *reason* over manipulated data. According to Valiant (2003), the future research in the fields of AI ans NeSy AI should investigate how to integrate learning and reasoning mechanisms; in this way, that the AI systems will be able to both learn from experience and reason about what the system has learned, that is a key point highlighted also in Garcez et al (2019).

According to Kautz (2022), neuro-symbolic AI systems can be classified into 5 categories:

- *Symbolic − Neural − Symbolic*: AI systems in which the input and the output are symbolic information, while the processing is neural.

- $Symbolic[Neural]$: AI systems having a symbolic problem solver which uses a neural component as a subroutine.
- $Neural \cup compile(Symbolic)$: AI systems in which the symbolic information (*e.g.,* a FOL rule) is compiled away during the neural training; in this way, the rule $A \rightarrow B$ becomes an input-output training pair (A, B).
- $Neural \rightarrow Symbolic$: AI system having a neural module and, cascadingly, a symbolic one.
- $Symbolic[Neural]$: AI systems embedding true symbolic reasoning inside a neural engine.

In this work, we present a *Neuro-Symbolic Knowledge Graph Embedding Learner* which injects First-Order Logic Rules into the Knowledge Graph Embedding Learning process, and the resulting embeddings are used to train a neural recommender system; according to this classification, and given that we used First-Order Logic as a unified framework to inject FOL rules during the neural graph embedding learning process, we believe that our model falls in the $Neural \cup compile(Symbolic)$ category. Several other categorizations of NeSy AI systems can be made; for further readings, we suggest referring to Kautz (2022).

2.2 Graph Embeddings for Recommender Systems

Although the exploitation of graphs and KGs for the recommendation task was already proposed in the early 2000s by Huang et al (2002), the adoption of graph embedding techniques in this field is more recent. These techniques aim at representing *nodes* and *relations* of a graph into a vector space in which the structured properties of the original graph are still preserved; in addition, this latent representation can be easily treated by computers due to the continuous and differentiable nature of the vector space in which embeddings are represented.

In Xie et al (2016), the authors propose a RS for Points of Interests (POIs) exploiting graph embedding, in which a *bipartite* graph expressing users-POIs interactions was used to provide recommendation; another solution was proposed in Palumbo et al (2017), but here the graph is *tripartite*, since it encodes not only users interactions, but item properties as well: in this way, the authors are able to compute both *user-item* and *item-item* similarities, aiming at obtaining better recommendation performance. In our work, we start from a tripartite KG encoding user preferences and item properties as well. Musto et al (2019) performs a comparison between *node2vec,* proposed in Grover and Leskovec (2016), and *Laplacian Eigenmaps* (Belkin and Niyogi (2002)), showing how exploiting information from KGs can improve the recommendation models in a significant manner; these findings are also confirmed by many other shreds of evidence, including Grad-Gyenge et al (2017), Zhang et al (2020), and Forouzandeh et al (2020).

Our work relies on *translation models*. In particular, our work exploits KALE, proposed by Guo et al (2016), as graph embedding technique. KALE

extends the original TransE by including knowledge derived from FOL rules; both graph-derived and rules-derived information is expressed in terms of First-Order Logic, which acts as a unified framework between them. In this way, it is possible to jointly learn graph embeddings including external knowledge provided by FOL rules. Moreover, since we combine pure *symbolic* information (FOL rules) with classical deep learning approaches (*e.g.,* TransE), this model can be labeled as *neuro-symbolic graph embedding technique*.

Generally speaking, this is one of the first attempts investigating the introduction of FOL rules in the graph embedding learning process for the task of recommendation. Other attempts in the direction of neuro-symbolic recommender systems are represented by Carraro et al (2024), where the authors use a Logic Tensor Network to train latent Factor Models, and by Zhang et al (2022), where logical layers are introduced in a neural network to learn user-attribute rules that are exploited to make the recommendation process more explainable and transparent.

2.3 Knowledge-aware Recommender Systems

Knowledge-aware Recommender Systems (KARSs) are a class of RSs that encourage the exploitation of information encoded in knowledge bases (such as KGs or plain text), to improve users' and items' representations and provide better recommendations. The first works in this area relied on Linked Open Data (LOD) and were proposed by Musto et al (2012), Ristoski et al (2014), Piao and Breslin (2016) and Musto et al (2018). More recently, the spreading of deep learning contributed in KARSs as well. In Anelli et al (2019), the authors proposed the Knowledge-aware Hybrid Factorization Machines (KaHFM), a model which extended the classical Matrix Factorization technique by adding and exploiting the knowledge encoded in KGs. Given the effectiveness of this model, we used it as a baseline for our experiments.

Of course, all these works are strictly connected to the literature investigating recommender systems based on side information (Liu et al (2019)). However, while RS with side information originally focus on the injection of descriptive features into collaborative approach, such as the previously mentioned KaHFM, more recent attempts aim to design and develop more comprehensive architectures that encode heterogeneous kind of knowledge (collaborative, textual, structured and unstructured) into recommendation algorithms.

Current trends lie in the application of GNNs and GCNs in this field, which propagate information between nodes up a certain number of hops in an iterative manner (this is called *message passing*), and then aggregate them to update the node representation. LightGCN (He et al (2020)) relies only on collaborative information (so it cannot be defined to be a KARS), while other models exploiting GCNs and GNNs use external knowledge as well. For example, KGCN (Knowledge Graph Convolutional Network) (Wang et al (2019a)) is a model that learns a representation based on both user-item interactions and item properties encoded in a KG, then propagates this information over the

neighbors of each node, which aggregates by using proper graph convolutional filters to obtain a unique representation used for the recommendation. Similarly, KGAT (Knowledge Graph Attention Network) (Wang et al (2019a)) uses *attention mechanisms* (Vaswani et al (2017)) to aggregate neighbors' information and weight their contributions, in order to identify the most informative ones.

Other KARSs worth to mentions are those based on the combination of more information sources, such as those presented in Polignano et al (2021) and in Spillo et al (2023b): in the first case, the framework relies on graph embeddings learned with TransE (Bordes et al (2013)) or TransH (Wang et al (2014)) and word embeddings learned with BERT (Devlin et al (2019)) or USE (Wang et al (2018)); Graph and word embeddings are then combined by concatenating them, and finally they are used to feed a neural recommender system. The experimental results show how to combination improves the general performance of the recommender system, with respect to the exploitation of the single (graph or word) embeddings; Next, in Spillo et al (2023b) the authors exploit graph embeddings learned with GCNs (in particular, CompGCN (Vashishth et al (2019))), and text embeddings learned with SBERT (Reimers and Gurevych (2019)), which is an extension of BERT, adapted to handle and represent *sentences* instead of *words*. Then, these two representations are combined with *self-attention* and *cross-attention*, resulting in a more precise final representation which obtained very competitive recommendation performance. Finally, there are KARSs based on multimedia information as well: McAuley et al (2015) and Albanese et al (2013) recommend clothes and paintings based on image features, respectively; Similarly, Van den Oord et al (2013) suggests music based on spectrograms, while Deldjoo et al (2016) recommends movies based on visual-based features.

In our experimental settings, we have considered several baselines, including GCNs, GNNs, KARSs, and approaches based on matrix factorization, both exploiting deep learning and not. The peculiarity of this work lies in the adoption of FOL rules to add more external knowledge during the KGE process, with the application in the recommendation field; up to our knowledge, the combination of symbolic and non-symbolic reasoning (logic rules and deep learning) in recommendation tasks has received limited attention.

3 Methodology

In this section, we introduce the modules that compose our model for *knowledge-aware recommendations* based on *neuro-symbolic graph embeddings* exploiting FOL rules. First, we describe our KG and what kind of entities and relations it encodes; next, we discuss the mining of FOL rules from the KG; then, we describe how the KGE jointly learns the embeddings by encoding both graph-based and rule-based knowledge; finally, we describe how these embeddings are used to feed the neural recommender systems. The general workflow of this model is presented in Figure 1.

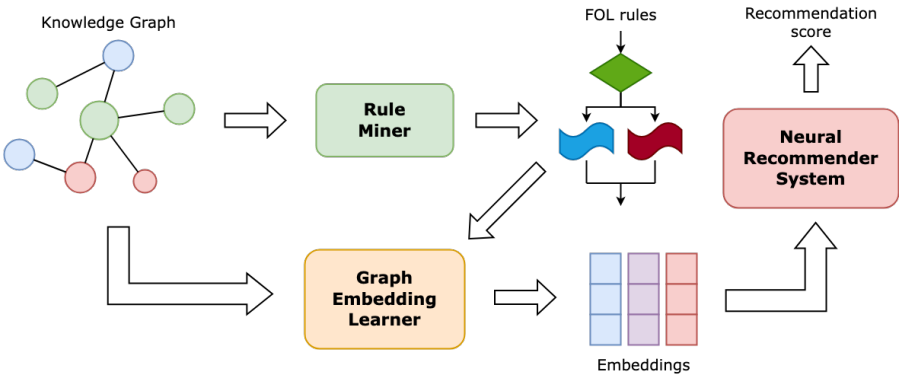

Fig. 1: Workflow carried out by our model.

3.1 Basics of the Model

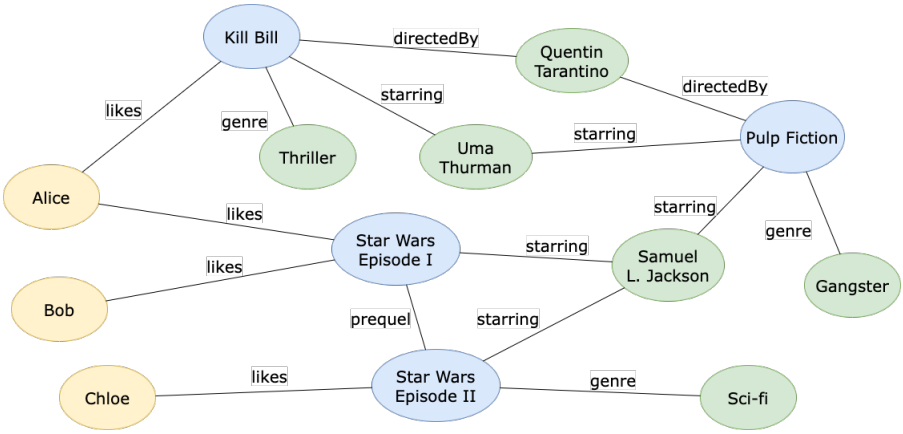

Fig. 2: A tripartite knowledge graph. Different kinds of entities (users, items, properties) are highlighted with different colors.

The main idea behind our work is to model *users*, *items*, and *items' properties* as entities. Let $\mathcal{U}$ be the set of users $\mathcal{U} = \{u_1 \dots u_n\}$, $\mathcal{I}$ a set of items $\mathcal{I} = \{i_1 \dots u_m\}$, and $\mathcal{P}$ a set of items' properties $\mathcal{P} = \{p_i \dots p_k\}$; starting from these three sets, we can define a set of entities $\mathcal{E}$ so that $\mathcal{E} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{P}$; now, let $\mathcal{R}$ be a set of relations. We can define a *knowledge graph* $\mathcal{KG}$ as follows: $\mathcal{KG} = \{(e_h, r, e_t) : e_h, e_t \in \mathcal{E}, r \in \mathcal{R}\}$. This formalism means that $\mathcal{KG}$ is a set of triples in which the head and the tail are entities (users, items, or item properties), and the relation is one of the available relations.

As for the set of relations $\mathcal{R}$, it is the set of relations that exist in the knowledge graph $\mathcal{KG}$; since entities can be users, items, or items' properties,

we have two main kind of relations: those related to *collaborative* information (user-item connections) and those related to descriptive *properties* about the items (item-entity connections). Formally, $\mathcal{R}_c = \{like, dislike\}$ are the relations connecting users and items (positive or negative, respectively), while $\mathcal{R}_p$ represent properties of the items, which are strictly dependent on the domain. As an example, for the movie domain, we might have $\mathcal{R}_p = \{director, starring, genre, subject, etc.\}$. Given $\mathcal{R}_c$ and $\mathcal{R}_p$, the set of relations $\mathcal{R} = \mathcal{R}_c \cup \mathcal{R}_p$. Figure 2 shows an example of *tripartite* KG in which user preferences and item properties are encoded in a unified representation.

First-Order Logic. In this paper, we will make extensive use of terminology borrowed from *first-order languages* and *first-order logic* (Smullyan (1995)). FOL rules exploited in our approach are formally defined as Horn clauses[4] due to the fact they are composed of several atoms connected by logical connectives (*e.g.,* $\vee, \wedge, \Rightarrow \ldots$), and only one atom appears in the *head* of the clauses (the other atoms are in the *body* of the clause). The *ariety* of a predicate represents the number of variables it takes as arguments: for example, a predicate taking one argument is said to be *unary*, while a predicate taking two arguments is said to be *binary*. Each atom is composed of *variables* (*e.g.,* $x, y, \ldots$) and *predicates* (equivalent to the *relations* previously introduced). Accordingly, the rules exploited in our model are first-order formulas in the form $\forall x, y : (x, r_s, y) \Rightarrow (x, r_t, y)$, meaning that if x and $y \in \mathcal{E}$ are linked by relation $r_s \in \mathcal{R}$, then they are linked by relation r_t too. As an example, a simple FOL rule could be the following: $\forall x, y, z : likes(x, z) \wedge prequel(z, y) \Rightarrow likes(x, y)$. This FOL rule expresses the fact that *'If a user likes a movie and that movie has a prequel, the user will like the prequel too'*. Finally, a rule is said to be *grounded* if every variable has been replaced by a suitable entity $e \in \mathcal{E}$. The *grounded rule* based on the previous logical rule can be: $likes(user, StarWarsII) \wedge prequel(StarWarsII, StarWarsIII) \Rightarrow likes(user, StarWarsIII)$. In this case, the variables y and z have been replaced with the items *StarWarsI* and *StarWarsII*, respectively. Of course, given a FOL rule, it is very likely that the same FOL rule has several grounded rules (or *groundings*); for example, other groundings for the FOL rule presented above might be $likes(user, LOTR\_TheTwoTowers) \wedge prequel(LOTR\_TheTwoTowers, LOTR\_TheReturnOfTheKing) \Rightarrow likes(user, LOTR\_TheReturnOfTheKing)$, or $likes(user, SpiderMan) \wedge prequel(SpiderMan, SpiderMan2) \Rightarrow likes(user, SpiderMan2)$. The concept of *grounded rule* (and *groundings*) is crucial to describe how our model aims at jointly learning embeddings from the KG and FOL rules works.

3.2 Mining First-Order Logical Rules

Given a knowledge graph $\mathcal{KG}$ (see Figure 2), the first task performed by our model is to mine FOL rules that exist in $\mathcal{KG}$; as already discussed, the mined rules are expressed as Horn clauses, so they are clauses in the form $\forall x, y, z :$

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

$predicate1(x, y) \wedge predicate2(y, z) \Rightarrow predicate3(x, z)$ in which there is at most one atom in the head of the rule.

As shown in Chen et al (2020) and in Shi and Weninger (2018), methodologies for mining logical rules were initially proposed for the task of *knowledge graph completion*, which aims at finding and discovering new and missing triples; then, they have been adopted for other tasks. As an example, Karimi and Kamandi (2019) use logical rules for ontology alignment and fact prediction. In our work, we used them to mine *background knowledge* encoded in the KG related to patterns which can be useful to improve the embedding learning process. More specifically, the focus of our work is not to extract *association rules* (Kotsiantis and Kanellopoulos (2006)) that connect two entities (such as, users who liked Star Wars II, also liked Star Wars III), but we are mainly interested in identifying *background knowledge* that characterize several entities and patterns, also including users' nodes.

In this work, we mined FOL rules with a strategy based on AMIE+ and AMIE 3, proposed in Galárraga et al (2013, 2015); Lajus et al (2020), which are specifically designed to mine Horn Clauses on large knowledge bases. Before describing how the rules miner works, it is worth pointing out some useful metrics that are crucial for the mining phase.

- *Length*: the length of the rule, expressed as the number of atoms the rule is composed of.
- *Standard Confidence* (or *Confidence*): it expresses how often the head of a rule is true, given its body, in the dataset.
- *Head Coverage*: it expresses the number of occurrences of the head of the rule in the data.
- *PCA-Confidence*: it expresses how often the head and body of the rule occur together and independently. In other words, it is the sum between the occurrences of rules in which both the head and the body occur, and the occurrences of rules in which only the head occurs.
- *Positive example*: it expresses the absolute number for which the FOL rule holds in the dataset (in other terms, the occurrences of the rule).

The pseudo-code reported in Algorithm 1, taken from Galárraga et al (2015), describes the strategy used to mine FOL rules from the KG; it takes as input a knowledge base $\mathcal{KG}$ and some parameters: the maximum length of rules to be mined l; the minimum head coverage *minHC* of the rules; the minimum confidence *min-conf* of the rules that have to be mined. The rule miner uses the *breadth-first search*; in particular, it starts with a queue composed of all possible rules with an empty body (*e.g.,* rules of size 1, with one atom in the head and empty body, such as $\emptyset \Rightarrow likes(x, y)$; *line 1*). Next, each rule is dequeued and its *confidence* and *PCA-confidence* are computed: if the rule is *closed* (which means that its variables appear in at least two atoms), its confidence is greater than or equal to *min-conf*, and its PCA-confidence is greater than or equal to the PCA-confidence of all the previously mined rules with the same head (*line 5*), then the rule is added to the set of the rules to be

Algorithm 1 Algorithm used for mining FOL rules, taken from Galárraga et al (2015). Input: Knowledge Graph $\mathcal{KG}$; maximum length l; minimum head coverage $minHeadCov$; minimum confidence $minConf$

1: $q \leftarrow [\emptyset \Rightarrow r_1(x,y), \emptyset \Rightarrow r_2(x,y), ..., \emptyset \Rightarrow r_k(x,y)]$
2: $rules = \emptyset$
3: **while** $|q| > 0$ **do**
4: $r \leftarrow q.dequeue()$
5: **if** $closed(r) \wedge PCAconf(r) > minConf \wedge betterConf(r, rules)$ **then**
6: $rules.add(r)$
7: **end if**
8: **if** $|r| > l \wedge PCAconf(R) < 1$ **then**
9: **for** $R \in refine(r)$ **do**
10: **if** $headCov(R) \geq minHeadCov \wedge R \notin q$ **then**
11: $q.enqueue(R)$
12: **end if**
13: **end for**
14: **end if**
15: **end while**
16: **return** $rules$

returned (*line 6*); if the rule is shorter than l and its confidence can still be improved (*line 8*), then the rules is refined by adding all the possible atoms to the body, and this new rule is added to the queue of the rules to be considered for the mining (*lines 9-11*). To speed up rule mining, based on Galárraga et al (2015), several strategies have been applied, including pruning strategies, parallelization, and lazy computation of confidence scores; in this way, the system is able to scale effortlessly to large knowledge bases, and at the same time it is still able to compute, for each rule, the exact confidence and support values without approximations.

The output for this step is a *a set of logical rules* (*line 16*) with related scores expressing to what degree the rules hold: we used these scores to split the rules into several subsets and to identify which are the most promising ones. An example of the output can be found in Table 1, in which we can find some real FOL rules mined from a movie recommendation dataset. Along with FOL rules, we can also see some of the already mentioned metrics (in particular, the head coverage, the standard confidence, and the number of positive examples).

Next, based on these metrics, we then defined some *selection heuristics* to split them into subsets and inject them into the graph embedding learning process. Given the absence of a proper literature that already defined some criterion to define relevant rules, we designed our subsets by considering different heuristics. In particular, the heuristics we defined were based on both the metrics that are typically used to assess the validity of FOL rules (*i.e.,* confidence, coverage, positive examples matching the rule) as well as to some background knowledge about the recommendation tasks (*i.e.,* rules having a like predicate in the head).

Based on these initial selection heuristics, eight different subsets are defined and evaluated in this work. In particular:

1. ALL RULES (ALL), including all the FOL rules that have been mined by the rule miner;
2. LOW STANDARD CONFIDENCE (LSC), including rules with a standard confidence value greater than 0.2;
3. MEDIUM STANDARD CONFIDENCE (MSC), including rules with a standard confidence value greater than 0.5;
4. MEDIUM-HIGH STANDARD CONFIDENCE (MHSC), including rules with a standard confidence value greater than 0.75;
5. HIGH STANDARD CONFIDENCE (HSC), including rules with a standard confidence value greater than 0.9;
6. LOW HEAD COVERAGE (LHC), including rules with a head coverage value greater than 0.2;
7. HIGH POSITIVE EXAMPLES (HPE), including rules with high number of positive examples. Since different datasets may have rules with very different values related to this metric, we identified one specific threshold value for each dataset: *(i)* for LAST.FM, we set 100 as threshold value; *(ii)* for DBBOOK, we set 250 as threshold value; *(iii)* for ML1M, we set 1000 as threshold value.
8. LIKE IN THE HEAD OF THE RULE (LH), including rules that have, in their head, an atom including the relation *likes*. We believe this is a reasonable choice since we are in the task of the recommendation, and the *like* relation is crucial.

Even though these are general heuristics, we can justify our choices. In particular, rule-based configuration (1) is useful to assess whether *all* the possible FOL rules mined from the graph are useful to improve the accuracy of the model; rule-based configurations (2-5) are useful to assess if it is possible to choose a *good-for-all* standard confidence value to improve the embeddings; moreover, FOL rules with a high confidence value should, in theory, contribute to the *graph completion* task, resulting in more paths available connecting two nodes, and this might lead to more meaningful and precise embeddings. Similarly, configurations 7-8 aim at considering different metrics for the same purpose. Configuration 8 includes all the rules with the *like* relation in the head, and it is the set in which we are most interested in, since the task of the recommendation is focused on investigating user preferences, and for this reason this set has been considered. In Section 4, the effectiveness of our framework on varying of the different subsets of rules that are picked based on the different heuristics will be assessed.

3.3 Learning Graph Embeddings

Once the rules are extracted, the joint learning based on triples in the KG and FOL rules is carried out. In this work, we used a model based on KALE (Guo et al (2016)). The *neuro-symbolic nature* of this model lies in the fact that the

Rule	Head Cov.	Std Conf.	Pos. Ex.
ML1M			
$producer(a,b) \Rightarrow director(a,b)$	0.06	0.00	12
$writer(m,b) \wedge producer(a,h) \wedge basedOn(m,h) \Rightarrow writer(a,b)$	0.01	0.63	5
$like(a,n) \wedge producer(n,h) \wedge subject(b,h) \Rightarrow like(a,b)$	0.01	0.33	5255
$producer(a,h) \wedge language(m,b) \wedge musicComposer(m,h) \Rightarrow language(a,b)$	0.35	0.34	1306
DBbook			
$genre(a,b) \wedge precededBy(h,b) \Rightarrow genre(h,b)$	0.17	0.62	310
$author(a,b) \wedge precededBy(h,b) \Rightarrow author(h,b)$	0.19	0.88	328
$like(a,b) \wedge precededBy(h,b) \Rightarrow like(a,h)$	0.27	0.16	865
Last.FM			
$genre(a,g) \wedge subject(a,s) \wedge subject(b,s) \Rightarrow genre(b,s)$	0.02	0.09	63
$genre(m,b) \wedge pastMember(a,h) \wedge currentMember(m,h) \Rightarrow genre(a,b)$	0.01	0.25	30
$genre(a,h) \wedge genre(m,h) \wedge instrument(m,b) \Rightarrow instrument(a,b)$	0.27	0.03	124

Table 1: Some examples of rules mined by our framework. In the tables, we also report the metrics we used for the definition of the heuristics

embeddings for each entity in the KG are learned by exploiting: *(i) explicit* knowledge, expressed in the form of triples (e_i,r_k,e_j) that are encoded in KG; *(ii) background* knowledge, expressed as FOL rules and learned as explained in Section 3.2.

The hallmark of our approach lies in the fact that both explicit and background knowledge is represented in a *unified* framework that learns a comprehensive representation based on both information sources. Generally speaking, our model is based on KALE, a graph embedding techniques that in turn inherits the principles of TransE and extends it by introducing FOL rules in the learning process. Given the good performance of TransE in recommendation tasks, as shown by several shreds of evidence including Palumbo et al (2018) and Polignano et al (2021), the choice of exploiting a method that relies on this algorithm can be considered reasonable.

Our KGE model implements joint learning based on triples from the KG and logical rules. Based on previous work by Rocktäschel et al (2014, 2015), joint training is possible since triples from a KG can be seen as *atoms* in first-order logic (*e.g., likes(Alice,Kill Bill)* or *starring(Uma Thurman,Kill Bill))*. Indeed, each relation that connects two entities in a graph can be seen as a *binary predicate* (that is, a predicate which takes as arguments two variables), and the two entities can be seen as arguments of the predicate.

In this way, the triple of the KG $(Alice, like, KillBill)$ can be written in terms of FOL as follows: $like(Alice, KillBill)$, where *like* is a binary predicate applied to the variables *Alice* and *KillBill*. In Table 2 we report how the KG in Figure 2 can be written as a set of FOL facts.

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 such joint learning. To start the learning process, a

KG triple	FOL formalism
$(Alice, like, KillBill)$	$like(Alice, KillBill)$
$(Alice, like, StarWarsI)$	$like(Alice, StarWarsI)$
$(Bob, like, StarWarsI)$	$like(Bob, StarWarsI)$
$(Chloe, like, StarWarsII)$	$like(Chloe, StarWarsII)$
$(KillBill, genre, Thriller)$	$genre(KillBill, Thriller)$
$(KillBill, starring, UmaThurman)$	$starring(KillBill, UmaThurman)$
$(KillBill, directedBy, QuentinTarantino)$	$directedBy(KillBill, QuentinTarantino)$
$(PulpFiction, directedBy, QuentinTarantino)$	$directedBy(PulpFiction, QuentinTarantino)$
$(PulpFiction, starring, UmaThurman)$	$starring(PulpFiction, UmaThurman)$
$(PulpFiction, starring, SamuelLJackson)$	$starring(PulpFiction, SamuelLJackson)$
$(PulpFiction, genre, Gangster)$	$genre(PulpFiction, Gangster)$
$(StarWarsI, starring, SamuelLJackson)$	$starring(StarWarsI, SamuelLJackson)$
$(StarWarsII, starring, SamuelLJackson)$	$starring(StarWarsII, SamuelLJackson)$
$(StarWarsI, prequel, StarWarsII)$	$prequel(StarWarsI, StarWarsII)$
$(StarWarsII, genre, SciFi)$	$genre(StarWarsI, SciFi)$

Table 2: Mapping of triples in a KG in FOL formalism.

set $\mathcal{F}$ consisting of *positive* training elements is built. In particular, $\mathcal{F}^+$ contains: *(i)* all the atomic formulas based on the KG (*i.e.,* triples in the form (e_i, r_k, e_j)); *(ii) grounded* rules based on the logical rules previously extracted.

Given a logical rule in the form $\forall x, y : (x, r_s, y) \Rightarrow (x, r_t, y)$, a grounded rule is obtained by replacing variables x and y with real entities $e_i, e_j \in \mathcal{E}$. By referring again to the toy example in Figure 2, if the rule $\forall x, y, z : likes(x, y) \wedge prequel(y, z) \Rightarrow likes(x, z)$ is learned, the following *grounded rule* is generated: $likes(user, StarWars) \wedge prequel(StarWars, StarWarsII) \Rightarrow likes(user, StarWarsII)$. Of course, replacement shall be bound *only* to the triples that exist in the graph (*i.e.,* combination of entities that *do not exist* in the graph do not constitute a valid grounding). Once the set $\mathcal{F}^+$ is obtained, a *negative* training set $\mathcal{F}^-$ shall be provided as well. To build $\mathcal{F}^-$, for each $(e_i, r_k, e_j) \in \mathcal{F}^+$, a corresponding negative triple is obtained by replacing either e_i or e_j with a random entity $e \in \mathcal{E}$.

Finally, the joint learning takes place by minimizing a margin-based ranking loss (see Formula 1), enforcing positive training examples to have larger truth values than negative ones:

$$\min_{\{e\},\{r\}} \sum_{f^+ \in \mathcal{F}^+} \sum_{f^- \in \mathcal{F}^-} [\gamma - \mathcal{I}(f^+) + \mathcal{I}(f^-)]_+ \tag{1}$$

such that $\|e\|_2 \leq 1, \forall e \in \mathcal{E}$ and $\|r\|_2 \leq 1, \forall r \in \mathcal{R}$, with γ a margin separating positive and negative examples, and $[x]_+ \triangleq \max\{0, x\}$. Even if we are only

interested in the embeddings of the entities (nodes), it is worth to point out that Formula 1) is used to learn both entity and relationship embeddings.

To complete the training, it is also necessary a function $\mathcal{I} : \mathcal{F} \rightarrow [0,\ 1]$ that assigns to each training example (*i.e.,* atomic and complex formulas) a soft truth value, indicating how likely a triple holds or to what degree a ground rule is satisfied.

Given a triple $f^+ \in \mathcal{F}^+$, computation of $\mathcal{I}(f)$ is based on TransE (Bordes et al (2013)), since each triple is modeled such that $e_i + r_k \approx e_j$. Accordingly, each triple is scored as $\|e_i + r_k - e_j\|_1$ by using the following Equation 2:

$$\mathcal{I}(f^+) = \mathcal{I}(e_i, r_k, e_j) = 1 - \frac{1}{3\sqrt{d}} \parallel e_i + r_k - e_j \parallel_1, \tag{2}$$

where d is the dimension of the embedding space. Of course, the same holds for negative triples $f^- \in \mathcal{F}^-$.

On the other hand, rules are modeled with t-norm fuzzy logics (Hájek (2013)): the truth value of a complex formula is given by the composition of the truth values of its constituents triples through specific t-norm based logical connectives. In particular, truth values are computed recursively; supposing f_1 and f_2 are atomic or complex formulae, truth values of complex formulae are computed as follows:

$$\begin{gathered}
\mathcal{I}(f_1 \wedge f_2) = \mathcal{I}(f_1) \cdot \mathcal{I}(f_2) \\
\mathcal{I}(f_1 \vee f_2) = \mathcal{I}(f_1) + \mathcal{I}(f_2) - \mathcal{I}(f_1) \cdot \mathcal{I}(f_2) \\
\mathcal{I}(\neg f_1) = 1 - \mathcal{I}(f_1) \\
\mathcal{I}(f_1 \Rightarrow f_2) = \mathcal{I}(f_1) \cdot \mathcal{I}(f_2) - \mathcal{I}(f_1) + 1
\end{gathered}$$

where the truth values of the atomic formulae $\mathcal{I}(f_1)$, $\mathcal{I}(f_2)$, ..., $\mathcal{I}(f_n)$, these are computed according to Equation 2. Since we are most interested in FOL rules, it is worth to provide an example related to the last formulae presented, related to the *logical implication*.

Let us consider the following FOL rule, composed of two atoms:

$$f \triangleq (e_i, r_k, e_j) \Rightarrow (e_k, r_l, e_j)$$

The truth value of f is given by

$$\begin{gathered}
\mathcal{I}(f) = \mathcal{I}(e_i, r_k, e_j) \cdot \mathcal{I}(e_i, r_l, e_j) - \mathcal{I}(e_i, r_k, e_j) + 1 \text{ where} \\
\mathcal{I}(e_i, r_k, e_j) = 1 - \frac{1}{3\sqrt{d}} \parallel e_i + r_k - e_j \parallel_1 \text{ and} \\
\mathcal{I}(e_i, r_l, e_j) = 1 - \frac{1}{3\sqrt{d}} \parallel e_i + r_l - e_j \parallel_1 \text{ , which refer to Equation 2.}
\end{gathered}$$

Let us now consider a rule composed of three atoms:

$$f \triangleq (e_i, r_k, e_j) \wedge (e_j, r_l, e_m) \Rightarrow (e_i, r_x, e_m)$$

Its truth value will be computed as follows:

$$\mathcal{I}(f) = \mathcal{I}(e_i, r_k, e_j) \cdot \mathcal{I}(e_j, r_l, e_m) \cdot \mathcal{I}(e_i, r_x, e_m) -$$
$$\mathcal{I}(e_i, r_j, e_j) \cdot \mathcal{I}(e_j, r_l, e_m) + 1 \text{ where, according to Equation 2,}$$
$$\mathcal{I}(e_i, r_k, e_j) = 1 - \frac{1}{3\sqrt{d}} \parallel e_i + r_k - e_j \parallel_1 ,$$
$$\mathcal{I}(e_j, r_l, e_m) = 1 - \frac{1}{3\sqrt{d}} \parallel e_j + r_l - e_m \parallel_1 , \text{ and}$$
$$\mathcal{I}(e_i, r_x, e_m) = 1 - \frac{1}{3\sqrt{d}} \parallel e_i + r_x - e_m \parallel_1 .$$

Of course, the same process can be generalized to formulae with an indefinite number of atoms.

In this way, it is possible to score both triples deriving from the KG and triples derived from the FOL rules; thanks to the score, it is then possible to start the iterative training process, during which neuro-symbolic embeddings will be learned, encoding *explicit* knowledge deriving from the KG, and *backgroud* knowledge deriving from the FOL rules, thus leading to a different representation of users and items.

3.4 Recommendation Framework

Once the neuro-symbolic embeddings are learned, the neural recommender system can be trained by feeding it with such embeddings. Given a user $u \in \mathcal{U}$ and an item $i \in \mathcal{I}$, the aim of this RS architecture is to predict to what extent the user u will be interested in the item i by providing a recommendation score; these scores are used to generate a recommendation list, which is computed by ordering in descending order the items by the recommendation score.

In this way, it is possible to generate a ranking of items, or a *recommendation list*. Figure 3 showed the architecture of our deep neural network: starting from the embeddings representing the user u and item i (previously learned by our graph embedding learner), these are fed into three dense layers with decreasing dimensions and ReLU as activation function. Then, the two embeddings are concatenated to obtain a unique representation, and this new embedding is fed again into three dense layers with decreasing dimensions and ReLU as activation function. The last layer has size 1 and uses a *sigmoid* as activation function, to produce a recommendation score $score \in [0, 1]$, which estimates the probability that the item i is relevant for user u; these scores are finally used to generate the recommendation list. The number of dense layers we used in the architecture is based on design choices that are common in literature (i.e., Spillo et al (2023b); Gu et al (2018); Liu et al (2020)) and

it is the result of a tuning of the architecture. Our design choice relies on the intuition that stacking more layers of gradually smaller dimensions allows the model to better capture non-linearities between the input and the output. In turn, this allows to better approximate a function (in our case, the recommendation function). This is in line with recent advances in the area of automatica dimensioning De Filippo et al (2022).

Formally, let $\vec{u}$ and $\vec{i}$ be the neuro-symbolic embeddings related to user u and item i, respectively; then, they are fed into three dense layers with reducing dimensions (with sizes 512, 256, and 128) and ReLU as activation functions:

$$u_{dense} = ReLU\left(W_{u\_dense} * \vec{u} + b_{u\_dense}\right)$$
$$i_{dense} = ReLU\left(W_{i\_dense} * \vec{i} + b_{i\_dense}\right)$$

where u_{dense} and i_{dense} are the reduced embeddings related u and i, respectively, W_{u_dense} and W_{i_dense} are the learnable parameters of the layers, and b_{u_dense} and b_{i_dense} are the biases. For the sake of brevity, we report here just one application of this layer.

Then, the two embeddings are concatenated, and the resulting embedding is fed into two layers with reducing dimensions (64 and 32) and ReLU activation function, and one layer with size 1 and sigmoid activation function. The output is the recommendation score between user u and item i.

$$concat_{dense} = ReLU\left(W_{concat\_dense} * (u_{dense} \oplus i_{dense}) + b_{concat\_dense}\right)$$
$$score = sigmoid\left(W_{score} * concat_{dense} + b_{score}\right)$$

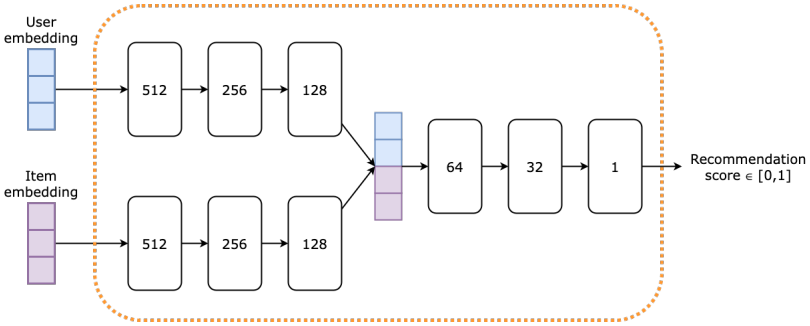

Fig. 3: Recommendation framework architecture

This neural architecture is trained by exploiting the user ratings in the form (u, i) available in our datasets; moreover, since we are training an architecture to understand which are *positive* and *negative* elements for some users, this model has been trained by using the *binary cross-entropy* as loss function, and the *accuracy* as evaluation metric to be optimized. More details about the

Dataset	Users	Items	Ratings	Sparsity	% pos.	% neg.	Rat./user	Rat./item
LAST.FM	1881	2828	71426	98.66%	53.64%	46.36%	37.97	25.26
DBBOOK	5660	6698	129513	99.66%	52.65%	47.35%	22.88	19.34
ML1M	6036	3081	946120	94.91%	57.26%	42.74%	156.75	307.08

Table 3: Statistics of the datasets

Dataset	Mapped items	KG Entities	KG Relations	KG Properties	KG Prop./item
LAST.FM	1217	7418	30	19513	16.03
DBBOOK	1931	5656	37	16589	8.59
ML1M	2839	17614	11	70668	24.89

Table 4: Statistics of the KGs of the datasets

training, the size of the layers, the optimizer and the other hyperparameters will be provided in the next section. After the training of the model, it is able to predict to what extent user u would like unseen items $i \in I$, from which a recommendation list consisting of the *top-k* items for each user is generated.

4 Experimental Evaluation

In the experimental section, we carried out experiments aiming at assessing the effectiveness of our methodology in the task of item recommendation. In particular, experiments were designed to answer the following research questions:

- **RQ1:** Does the introduction of logical rules *changes* the elements that are included in the recommendation lists?
- **RQ2:** How does the different *selection heuristics* to pick FOL rules impact on the *accuracy, novelty, and diversity* of the recommendation?
- **RQ3:** How does our approach based on *neuro-symbolic graph embeddings* perform w.r.t. *competitive baselines*?

4.1 Experimental Design

4.1.1 Datasets

Experiments were carried out in a *movie, book and music recommendation* scenarios. We used state-of-the-art datasets: MovieLens 1M (ML1M)[5] for the movie domain, DBBOOK for the book domain, and LAST.FM[6] for music domain; these datasets can also be found on our public repository[7]. In Table 3 some statistics about the datasets we used are reported; in general, we can

[5] http://grouplens.org/datasets/movielens/
[6] http://millionsongdataset.com/lastfm/
[7] https://github.com/swapUniba/RecSysDatasets

observe how ML1M is the dataset recording the highest number of interactions, followed by DBbook; the smallest one is Last.FM. We can also notice how DBbook and Last.FM are the most sparse datasets, while ML1M is more dense. As for the side information encoded in the KGs, we used knowledge encoded in DBpedia; this knowledge has been obtained by exploiting an online available mapping [8]: thanks to a SPARQL query we have obtained side information about the items (e.g., the name of a book, or the genre of a movie or an album). The mapping information has been exploited to obtain a KG, whose statistics are reported in Table 4.

4.1.2 Protocol

The three datasets have been split into training and test sets, and for all of them we adopted an 80%-20% ratio. While ratings from DBbook already came into binary form (*positive* and *negative* represented by 1s and 0s, respectively), for Last.FM and ML1M ratings were expressed as integers ranging from 1 to 5; in order to handle them, we considered *positive* ratings those equal to 4 and 5, while *negative* those from 1 to 3. As for the prediction accuracy evaluation, we generated *top-5* and *top-10* recommendation lists by following the *TestRatings* strategy proposed in Bellogin et al (2011). Such strategy is often used to evaluate the effectiveness of recommendation algorithms, as shown in Polignano et al (2021) and Spillo et al (2023b). The lists of *top-k* items have been obtained by scoring (by exploiting our deep architecture described in Section 3) all the items i in the Test Set for each user u; then these rankings have been sorted in descending order, and cut at the first k entries. Given that the findings we obtained with *top-5* and *top-10* recommendations were similar, for the sake of brevity we just report here the results for *top-5* recommendations. Results for top-10 recommendations are available in our repository[9].

4.1.3 Source Code and Parameters

Source Code. The source code of this model, including used datasets, rule mining, neuro-symbolic KGE model, and deep recommender systems, is publicly available on GitHub[10]. The source code of the rule miner is based on AMIE[11], while the implementation of our KGE model injecting FOL rules is based on KALE[12]. As for the source code of the deep recommendation architecture, it extends the implementation made available by the authors of Polignano et al (2021) released on Github[13].

First-Order Logic Rules. FOL rules mining has been performed by in order to mine rules with at most 4 atoms (in other words, the maximum length of the rules), and we set as minimum confidence a value equal to 0.001, and 0.1 as minimum coverage value. This choice comes after several run of the strategy,

[8] https://github.com/sisinflab/LODrecsys-datasets
[9] https://tinyurl.com/top10recsysresults
[10] https://github.com/swapUniba/KARS_NeSy_KGE_with_FOL_rules
[11] https://github.com/lajus/amie
[12] https://github.com/iieir-km/KALE
[13] https://github.com/swapUniba/Deep_CBRS_Amar

based on a trade-off analysis between algorithmic complexity and knowledge provided by the FOL rules. Experiments with different numbers of atoms are left as future work.

Embeddings. Embeddings were learnt at three different sizes: $k = 256$, $k = 512$, and $k = 768$ dimensions; the learning process has been carried out in mini-batches of size 100 for all the three datasets, with a separating margin (between positive and negative examples) equals to 0.1, entity and relation learning rate equal to 0.05, and 1000 epochs.

Recommender System. the training of the neural recommender system has been carried out for 25 epochs, with a batch size equal to 512; we used Adam as optimizer, a learning rate equal to 0.001, and default values set by Keras for beta parameters (which control the moving average; default values are 0.9 for *beta_1*, and 0.999 for *beta_2*). As loss function, we used the *binary cross-entropy*. Other optimizers De Filippo et al (2021) will be evaluated as future work.

4.1.4 Experimental Configurations

In this work, we carried out experiments aimed at comparing *ten* different configurations of our recommendation model based on *neuro-symbolic graph embeddings combining FOL rules*. In particular, we considered *two* basic configurations, relying only on information from the graph, and *eight* configurations based on the injection of FOL rules in the KGE process.

For the first *basic* configuration, we learned graph embeddings encoding only information derived from *user-item* interactions: for example, referring to Figure 2, we only encode the triples $(Alice, like, KillBill), (Alice, like, StarWarsI), (Bob, like, StarWarsII)$, $(Chloe, like, StarWarsII)$. For the second *basic* configuration, we considered not only user-item interactions but also all the *structured properties* deriving from the KG: for example, again referring to Figure 2, we encode the triples related to the genre of the movies or the actors starring in them, such as $(KillBill, genre, Thriller)$, $(PulpFiction, starring, SamuelLJackson)$, $(StarWarsII, starring, SamuelLJackson)$. Next, we evaluated *eight* configurations based on the injection of FOL rules in the KGE learning. The configurations have been presented in Section **??**. In Table 5, we report some statistics about the number of rules and groundings found for each dataset; let us recall that, given a FOL rule, a *grounding* for that rule is given by the same rule in which all variables have been replaced with real entities in the KG. It is necessary to observe that, for LAST.FM, the set of LIKE rules is empty, and for this reason, it will not appear in the analysis and discussion of the results.

Heuristic	Last.FM		DBbook		ML1M	
	Rules	Groundings	Rules	Groundings	Rules	Groundings
LSC	17	3652	28	2265	505	4M
MSC	4	2371	20	1472	152	75M
MHSC	3	147	12	362	62	3817
HSC	2	14	7	78	39	3029
LHC	9	864	5	2195	63	292K
HPE	9	864	10	17989	70	10M
LH	0		2	8189	27	5M

Table 5: Statistics about the used subsets of FOL rules

4.2 Baselines and Evaluation Metrics

Evaluation Metrics. The evaluation has been performed by using the Elliot[14] framework (Anelli et al (2021)), to guarantee reproducibility of the experimental protocol; as for the metrics, we naturally considered *accuracy* metrics, such as *Precision*, *Recall*, *F1 score* and *Mean Average Precision* (*MAP*), and we also considered *nDCG*. Moreover, we also evaluated non-accuracy metrics, such as *diversity* and *novelty*: in particular, we considered Gini Index for diversity, and Expected Free Discovery (EFD) and Expected Popularity Complement (EPC) for the novelty, defined as in Vargas and Castells (2011). In all these cases, metrics were calculated based on the implementations available in Elliot. In particular, it is worth to emphasize that the value reported as 'Gini' in the tables is calculated as $1 - Gini$ (so, the higher the better).

Baselines. As regards the baselines, we compared our results to ten different baselines available in the Elliot framework[15]: three *matrix factorization* techniques, such as SLIM (Ning and Karypis (2011)), BPRMF (Rendle et al (2012)) and PureSVD, three methods based on *deep learning models*, such as MultiVAE (Liang et al (2018)), CFGAN (Chae et al (2018)) and NGCF (Wang et al (2019b)) and four algorithms implementing *knowledge-aware* techniques, such as Item and User-KNN encoding side information (Gantner et al (2011)), the previously mentioned KaHFM proposed in Anelli et al (2019), LightGCN (He et al (2020)) and the basic Vector Space Model (VSM). All the algorithms are run with their optimal parameters, selected by the Elliot framework. In our repository[16], we have added a folder containing all the scripts to find the optimal hyper-parameters and to run the baselines with those values. To assess statistical significance, we used the *t-test* implementation available in the Elliot framework. In particular, we compared the recommendation lists

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

[15] We adapted the Elliot framework to the TestRating modality; our implementation is available here: https://github.com/giuspillo/elliot_testratings

[16] https://github.com/swapUniba/KARS_NeSy_KGE_with_FOL_rules/tree/main/baselines

provided by both our model and those generated by the baselines. These tests are computed for each considered metrics.

5 Results and Discussion

In this section, we discuss the results of the experimental section, by first providing a qualitative analysis on the representations of the items, and then we answer the research questions.

5.1 RQ1: Impact of FOL rules on recommendation lists

In this section, we first present some results assessing the impact of FOL rules on the lists of the recommendations. In other terms, we want to investigate whether and how the introduction of FOL rules changes the items recommended to the users.

First, we present some qualitative results obtained by considering some real recommendation lists generated by our recommendation model based on neuro-symbolic graph embeddings with FOL rules. Results are related to the movie recommendation domain and are reported in Table 6; we considered two users and compared the recommendation lists generated for the two *basic* configurations (*user-item* and *user-item-properties*, shortened as *UI* and *UIP*), and one configuration based on FOL rules (those related to the rules with the relation *like* in the head). It is worth observing that different information encoded can have a huge impact on the recommendation list: for example, for *User 6039*, only one movie appears in all the three recommendation lists (*"North by Northwest"*), while only two movies are shared by the UIP and the UIP + LH recommendation lists (the already mentioned *"North by Northwest"*, and *"Citizen Kane"*). This confirms what we depicted in the previous section, that is, different encoded knowledge, although deriving from the same source (the KG), leads to different representations. In addition, other interesting observations can be made by analyzing the recommendation lists for *User 5743*; in this case, we can better assess the impact of FOL rules: while the two *basic* configurations share *four* movies, on a total of *five*, we can say they are almost the same list, except for one element and the permutation of the others, the UIP + LH list has only two movies shared with the *basic* configurations (*"Wings"* and *"Roman Holiday"*), while the other three movies are totally new. This shows very well how injecting FOL rules in the graph embedding process can impact not only the representation of items but the task of recommendation too. In any case, it is necessary to point out that not always encoding different information leads to that differences, as the recommendation lists related to *User 12* show: in this case, the three recommendation lists are very similar to each other, even if some differences in the ordering can be observed, and movies not occurring in the other lists can be observed as well.

Next, together with the above presented *qualitative* results, we also carried out a more systematic analysis of the variations in the recommendations lists based on Kendall-Tau (KT) coefficient and NDCG. In both cases, we

User 6039		
UI	***UIP***	***UIP + LH***
Fanny and Alexander	The Usual Suspects	Citizen Kane
Heathers	The Shawshank Redemption	The Treasure of the Sierra Madre
The Pillow Book	North by Northwest	North by Northwest
North by Northwest	Alien	Graduate
Citizen Kane	Monty Python and the Holy Grail	Notorious (1946 film)
User 5743		
UI	***UIP***	***UIP + LH***
Sabrina	Roman Holiday	Wings
Some Like It Hot	Wings	Suspicion
Wings	Sabrina	Winnie the Pooh and the Blustery Day
Roman Holiday	Some Like It Hot	Roman Holiday
Them!	Being John Malkovich	Lady and the Tramp
User 12		
UI	***UIP***	***UIP + LH***
The Silence of the Lambs	Star Wars Episode IV	Star Wars Episode IV
Star Wars Episode IV	The Silence of the Lambs	The Silence of the Lambs
Kolya	Kolya	Jurassic Park
Jurassic Park	Galaxy Quest	Kolya
Men in Black	Men in Black	Men in Black

Table 6: Top-k real recommendations provided by different configurations of our model in the movie field, for Users *6039*, *5743, and 12*; for this example, we set k=*5*.

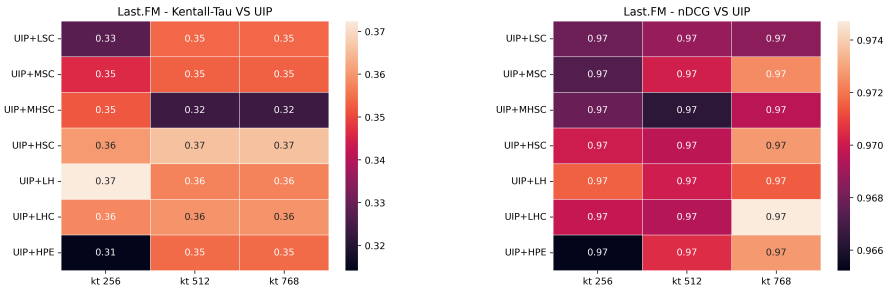

Fig. 4: Values of Kendall-Tau's test and nDCG metric for the LAST.FM dataset

comparedthe top-5 recommendation lists (*i.e.,* for all the users) obtained on varying of all the different subsets of FOL rules, based on the implementation of KT available in Sci-Kit Learn[17]. In both cases, *lower is better*, since KT coefficient measures the overlap in terms of elements in the recommendation lists, while nDCG score measures how similar a recommendation lists is w.r.t.

[17]The implementation was also released on our repository: https://github.com/swapUniba/KARS_NeSy_KGE_with_FOL_rules/blob/main/kendall_tau.py

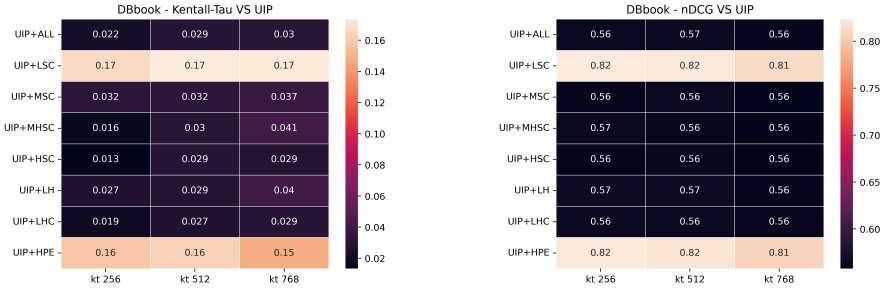

Fig. 5: Values of Kendall-Tau's test and nDCG metric for the DBbook dataset

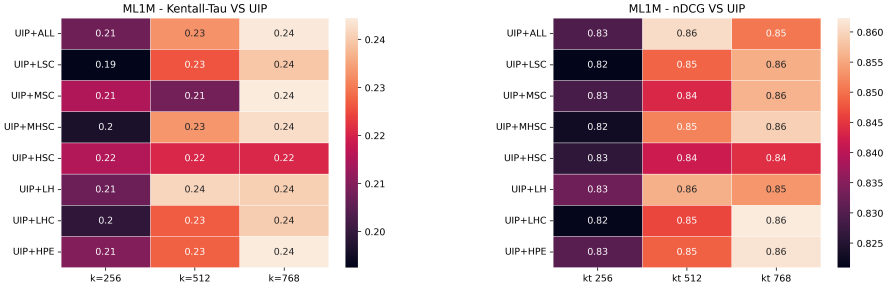

Fig. 6: Values of Kendall-Tau's test and nDCG metric for the ML1M dataset

a *ground truth*. In our case, the ground truth is represented by the recommendation lists obtained by running the simple UIP configuration, that is to say, the one based on the user-item-properties graph without any FOL rule. The results are presented in Figures 4, 5 and 6.

As for *Last.fm*, the results show that KT scores range between 0.31 and 0.37. It means that all the configurations based on FOL change some of the items in the recommendation lists w.r.t. the basic UIP configuration. However, the analysis of the nDCG scores allows to assess that the change is usually tiny, since these score range from 0.966 to 0.974. Given that nDCG ranges from 0 to 1, it means that the change is very limited. However, by considering the statistics presented in Table 5, this is not surprising since the number or rules (and grounding) mined from the data is very low. Accordingly, it makes sense that this leads to a mininum change in the recommendation lists.

Next, by considering *DBbook* data, we note a more significant change since nDCG scores range from 0.56 to 0.82. In particular, 6 out of 8 configurations show lower nDCG scores, so it means that many different items are modified due to the introduction of FOL rules. Similar outcomes also emerge from *ML1M* data, since all the configurations obtain an nDCG score around 0.82. While the score does not different on varying of the different configurations, we note that lower dimension of the embeddings ($k = 256$) leads to a lower

nDCG score (so, more change in the items). In the experimental evaluation we will try to assess whether the change in the lists leads to an increase in terms of both accuracy and non-accuracy metrics.

5.2 RQ2: FOL Heuristics Comparisons

To answer RQ2, we compared the performance of the *eight* configuration based on neuro-symbolic knowledge graph embeddings to the two *basic* configurations based only on *user-item* and *user-item-prop* information (shortened, again, as *UI* and *UIP*), respectively. Results for the music domain are reported in Tables 7, while those related to the book domain are reported in Table 8, and finally, results related to the movie domain are reported in Table 9. In these tables, we report the performance obtained by our model considering not only the different heuristics used to select FOL rules to be injected during the graph embedding learning process, but also considering and comparing the performance at the variation of the dimension k of the embeddings we learned, as we mentioned in Section 3.

In particular, as for the music domain (see Table 7), we can observe that in most of the cases, the configuration based on the injection of FOL rules overcome the two *basic* configurations, especially when considering the configurations with $k = 256$ and $k = 768$. For the $k = 256$ setting, the best configuration is clearly UIP + LSC, that is the configuration obtained by injecting all FOL rules with a Standard Confidence > 0.2, whose precision, recall, F1 score, Gini index, and EFD score overcame the baselines and the other configurations; this means that this configuration is not only accurate, achieving best performance in several accuracy metrics, but also provided more diverse and novel recommendation lists. The configuration UIP + HSC, which is obtained by injecting FOL rules with a Standard Confidence > 0.9, obtained the best results in the other accuracy metrics, MAP, MAR, and nDCG, and achieved the best performance in terms of EPC as well. The setting $k = 512$ shows a more varied scenario: the *basic* configuration based only on *user-item* interactions (UI) obtained the best results in terms of precision, recall, and F1 score, being the lead for the accuracy metrics; best MAP and best nDCG have been obtained by the configuration UIP + ALL (configuration injecting *all* the available FOL rules), while the configurations UIP + MHSC and UIP + HSC provided the more diverse recommendation lists; moreover, the latter configuration obtained the best F1 score among the configurations with FOL rules.As for the setting $k = 768$, the best configuration is for sure UIP + LHC (rules with head coverage > 0.2), which is not only accurate (best MAP, MAR, and nDCG), but also provide recommendation lists with the best novelty. UIP + MHSC obtained the best results in terms of precision and F1, while UIP + ALL achieved the best F1 score. Finally, the *basic* configuration UIP provided the most diverse recommendation lists.

Let us now move to the book domain (Table 8). As for the setting $k = 256$, the overall best configuration is UIP + HSC (rules with standard confidence > 0.75) got the best performance in terms of MAP, MAR, and nDCG for the

Heuristic	Accuracy						Diversity	Novelty	
	MAP	Precision	MAR	Recall	F1	nDCG	Gini	EFD	EPC
256									
UI	0.5779	0.4524	0.7227	0.9534	0.5739	0.8460	0.2902	6.87	0.6200
UIP	0.5801	0.4523	0.7276	0.9536	0.5739	0.8507	0.2890	6.87	0.6213
UIP + ALL	0.5809	0.4512	0.7277	0.9532	0.5728	0.8496	0.2889	6.87	0.6202
UIP + LSC	0.5760	**0.4528**	0.7237	**0.9549**	**0.5744**	0.8470	**0.2923**	6.89	0.6202
UIP + MSC	0.5816	0.4517	0.7278	0.9512	0.5728	0.8496	0.2897	**6.89**	**0.6215**
UIP + MHSC	0.5802	0.4518	0.7265	0.9532	0.5733	0.8490	0.2881	6.87	0.6203
UIP + HSC	**0.5841**	0.4513	**0.7286**	0.9529	0.5728	**0.8515**	0.2875	6.87	0.6215
UIP + LHC	0.5783	0.4514	0.7271	0.9521	0.5727	0.8487	0.2913	6.88	0.6202
UIP + HPE	0.5755	0.4513	0.7244	0.9528	0.5728	0.8462	0.2894	6.86	0.6189
512									
UI	0.5776	**0.4532**	0.7248	**0.9550**	**0.5751**	0.8481	0.2894	6.87	0.6202
UIP	0.5765	0.4515	0.7239	0.9533	0.5732	0.8471	0.2911	6.87	0.6192
UIP + ALL	**0.5799**	0.4510	0.7258	0.9525	0.5726	**0.8502**	0.2889	6.86	0.6204
UIP + LSC	**0.5772**	0.4497	0.7242	0.9482	0.5705	0.8461	0.2914	6.87	0.6196
UIP + MSC	0.5728	0.4490	0.7210	0.9484	0.5700	0.8394	0.2903	6.84	0.6156
UIP + MHSC	0.5791	0.4528	**0.7279**	0.9523	0.5740	0.8495	0.2901	**6.89**	**0.6210**
UIP + HSC	0.5752	0.4509	0.7242	0.9536	0.5728	0.8463	**0.2915**	6.85	0.6182
UIP + LHC	0.5776	0.4500	0.7241	0.9505	0.5712	0.8455	0.2906	6.86	0.6184
UIP + HPE	0.5770	0.4497	0.7248	0.9506	0.5710	0.8454	0.2892	6.85	0.6182
768									
UI	0.5746	0.4522	0.7228	0.9530	0.5737	0.8442	0.2908	6.86	0.6183
UIP	0.5807	0.4524	0.7262	0.9525	0.5738	0.8489	**0.2911**	6.89	0.6216
UIP + ALL	0.5771	0.4519	0.7238	**0.9547**	0.5737	0.8473	0.2886	6.86	0.6193
UIP + LSC	0.5837	0.4517	0.7285	0.9529	0.5732	0.8530	0.2875	6.90	0.6227
UIP + MSC	0.5755	0.4510	0.7233	0.9518	0.5725	0.8443	0.2900	6.84	0.6180
UIP + MHSC	0.5780	**0.4528**	0.7250	0.9538	**0.5743**	0.8484	0.2902	6.87	0.6203
UIP + HSC	0.5769	0.4524	0.7222	0.9546	0.5742	0.8438	0.2895	6.86	0.6184
UIP + LHC	**0.5849**	0.4524	**0.7310**	0.9536	0.5740	**0.8555**	0.2896	**6.92**	**0.6246**
UIP + HPE	0.5824	0.4521	0.7293	0.9527	0.5733	0.8531	0.2899	6.88	0.6219

Table 7: Results of our model on the Last.FM dataset. In **bold**, the best overall score, while the best score obtained by a configuration based on FOL rules is underlined.

accuracy metrics; moreover, it resulted to be the configuration with the most diverse recommendation lists, and obtained good performance in terms of novelty as well. The best precision has been reached by the configuration UIP + LSC, while the best recall and F1 score have been achieved by UIP + LH (rules having the relation *like* in their head), which also got the highest EPC score. Generally speaking, and according to the setting $k = 256$ of the previous domain, the *basic* configurations have been outperformed by the configurations including FOL rules. As for $k = 512$, similarly to the previous case, the *basic*

configurations obtained the best performance in almost all metrics: in particular, UIP resulted to be the best configuration in terms of MAP (equal score to UIP + HPE, rules with high number of positive examples), MAR, recall, F1, nDCG, EFD, and EPC; the highest MAR has been obtained by UIP + LHC, while the best diversity of the recommendation lists has been achieved by the configurations UIP + HPE and UIP + LH. Finally, for the setting $k = 768$, we can easily notice that the configurations with rules outperformed the two baselines, and the UIP + LSC is the configuration that achieved the best results. In particular, it got the best MAR, NDCG, Novelty and Diversity, while UIP + HSC got the best results in terms of accuracy (precision, recall an F1).

Finally, let us discuss the results obtained in the movie domain (Table 9). As for $k = 256$, the configurations using FOL rules overcame the two baselines. In particular, the overall best configuration is UIP + LH, which achieved the best accuracy performance in terms of precision, MAR, Precision, and nDCG, while UIP + HSC got the highest MAP and UIP + MHSC got the highest recall and F1. The highest novelty has been obtained by UIP + MHSC for the EFD and UIP + LSC for the EPC, while the basic configuration UI resulted to be the configuration providing the most diverse recommendation lists. UIP + MHSC achieved the highest diversity among the configurations based on rules. In the setting $k = 512$, unlike the previous cases, the configurations with FOL rules overcome the two *basic* configurations in almost all metrics. In fact, UIP + LHC and UIP + MHSC dominate the accuracy, with the highest precision, recall, F1, and nDCG for UIP + LHC, and the highest MAP and MAP for UIP + MHSC. The configuration UIP + LSC is the one that provided the most novel recommendation lists, while the highest diversity has been obtained again by UI, while UIP + MSC provided the highest novelty among the configurations using rules. The setting $k = 768$ follows the others: the highest accuracy has been obtained by UIP + LSC and UIP + MSC (MAP, MAR, and nDCG for UIP + LSC, while precision, recall, and F1 score for UIP + MSC); the configuration UIP + HSC provided the most novel recommendation lists; it also achieved the best diversity among the configurations using FOL rules, but the highest diversity has been obtained again by UIP. These results bring us to think that, in the case of the dataset ML1M, which is denser than the previous two and has much more interactions (about 1M), the properties might influence negatively the diversity, while the FOL rules tend to increase it, without bringing it back to the best value obtained by UI.

Take-Home Messages. Generally speaking, the exploitation of FOL rules in the knowledge graph embedding learning process influences positively the performance of the recommender systems in almost all cases. As for Last.fm and DBbook, the injection of FOL rules improves *all* the metrics for $k = 256$ and $k = 768$ and 6 out of 9 metrics (with the exception of Precision, Recall an F1) on $k = 512$. As for MovieLens, the results are even better since the only exception is represented by the Gini index whose score decreases when FOL

Heuristic	Accuracy						Diversity	Novelty	
	MAP	Precision	MAR	Recall	F1	nDCG	Gini	EFD	EPC
256									
UI	0.6378	0.6362	0.3013	0.4982	0.5145	0.6787	0.2797	7.4064	0.6337
UIP	0.6374	0.6362	0.2989	0.4970	0.5138	0.6770	0.2796	7.4124	0.6337
UIP + ALL	0.6360	0.6328	0.2978	0.4937	0.5108	0.6747	0.2813	7.3670	0.6308
UIP + LSC	0.6380	**0.6385**	0.2986	0.4978	0.5155	0.6777	0.2791	7.4289	0.6349
UIP + MSC	0.6384	0.6367	0.3000	0.4983	0.5148	0.6785	0.2809	7.4132	0.6342
UIP + MHSC	0.6377	0.6351	0.2998	0.4953	0.5128	0.6771	0.2798	7.3972	0.6330
UIP + HSC	**0.6414**	0.6376	**0.3015**	0.4980	0.5150	**0.6806**	**0.2835**	**7.4473**	0.6361
UIP + LHC	0.6380	0.6350	0.2991	0.4979	0.5135	0.6773	0.2806	7.4070	0.6331
UIP + HPE	0.6388	0.6362	0.2995	0.4958	0.5133	0.6780	0.2810	7.4210	0.6343
UIP + LH	0.6413	0.6381	0.3009	**0.4985**	**0.5157**	0.6803	0.2787	7.4381	**0.6362**
512									
UI	0.6370	0.6362	0.2997	0.4972	0.5141	0.6771	0.2786	7.3964	0.6333
UIP	**0.6397**	**0.6376**	0.2993	**0.4992**	**0.5156**	**0.6791**	0.2796	**7.4234**	**0.6355**
UIP + ALL	0.6378	0.6365	0.2998	0.4979	0.5143	0.6783	0.2819	7.4120	0.6336
UIP + LSC	0.6392	0.6364	0.2987	0.4957	0.5134	0.6779	0.2808	7.4128	0.6348
UIP + MSC	0.6380	0.6358	0.2998	0.4967	0.5137	0.6776	0.2786	7.4116	0.6336
UIP + MHSC	0.6380	0.6352	0.3001	0.4966	0.5135	0.6773	0.2795	7.3928	0.6330
UIP + HSC	0.6372	0.6353	0.2992	0.4945	0.5128	0.6763	0.2800	7.3957	0.6326
UIP + LHC	0.6376	0.6355	**0.3002**	0.4981	0.5138	0.6779	0.2790	7.3970	0.6331
UIP + HPE	**0.6397**	0.6358	0.2991	0.4954	0.5133	0.6774	0.2822	7.4230	0.6344
UIP + LH	0.6369	0.6338	0.2996	0.4964	0.5124	0.6759	**0.2825**	7.3893	0.6318
768									
UI	0.6351	0.6346	0.2975	0.4947	0.5126	0.6743	0.2807	7.3802	0.6315
UIP	0.6392	0.6372	0.2996	0.4979	0.5148	0.6785	0.2805	7.4192	0.6351
UIP + ALL	0.6375	0.6334	0.2982	0.4926	0.5110	0.6756	0.2800	7.3785	0.6320
UIP + LSC	**0.6404**	0.6375	0.2998	0.4978	0.5150	**0.6795**	**0.2822**	**7.4345**	**0.6358**
UIP + MSC	0.6387	0.6371	0.2992	0.4974	0.5147	0.6784	0.2795	7.4094	0.6345
UIP + MHSC	0.6389	0.6363	0.2991	0.4960	0.5137	0.6781	0.2793	7.4192	0.6345
UIP + HSC	0.6382	**0.6379**	0.2997	**0.4993**	**0.5155**	0.6795	0.2796	7.4110	0.6346
UIP + LHC	0.6378	0.6354	0.2992	0.4965	0.5131	0.6778	0.2805	7.4015	0.6333
UIP + HPE	0.6396	0.6356	**0.2999**	0.4963	0.5135	0.6780	0.2812	7.4174	0.6343
UIP + LH	0.6387	0.6371	0.2998	0.4973	0.5148	0.6783	0.2786	7.4165	0.6345

Table 8: Results of our model on the DBbook dataset. In **bold**, the best overall score, while the best score obtained by a configuration based on FOL rules is underlined.

rules are injected. Overall, the best-performing configuration is obtained by exploiting a configuration that also encodes rules, with the only exception of Precision, Recall and F1 on Last.fm. In all the other comparisons, the overall best results exploit the FOL rules mined by our framework.

As regards the performance of the different heuristics, a very heterogeneous behavior emerges. First, it is important to point out that the idea of inject *all* the available rules is not the optimal solution. Indeed, the best rule-based

Heuristic	Accuracy						Diversity	Novelty	
	MAP	Precision	MAR	Recall	F1	nDCG	Gini	EFD	EPC
256									
UI	0.6880	0.6763	0.2548	0.4089	0.4278	0.7242	**0.2618**	7.0021	0.5789
UIP	0.8386	0.7953	0.2990	0.4581	0.4884	0.8648	0.1482	7.9823	0.6621
UIP + ALL	0.8399	0.7955	0.2992	0.4571	0.4878	0.8655	0.1492	8.0102	0.6650
UIP + LSC	0.8373	0.7950	0.2975	0.4572	0.4879	0.8634	0.1568	8.0754	**0.6728**
UIP + MSC	0.8380	0.7958	0.2986	0.4579	0.4884	0.8645	0.1566	8.0463	0.6679
UIP + MHSC	0.8386	0.7976	0.2996	**0.4593**	**0.4901**	0.8659	0.1586	**8.0883**	0.6714
UIP + HSC	**0.8404**	0.7970	0.2995	0.4579	0.4886	0.8661	0.1549	8.0669	0.6700
UIP + LHC	0.8378	0.7971	0.2991	0.4587	0.4892	0.8653	0.1502	8.0352	0.6682
UIP + HPE	0.8391	0.7968	0.2989	0.4581	0.4887	0.8657	0.1504	8.0282	0.6669
UIP + LH	0.8403	**0.7988**	**0.2998**	0.4593	0.4901	**0.8674**	0.1505	8.0496	0.6690
512									
UI	0.6426	0.6405	0.2466	0.3994	0.4141	0.6827	**0.2948**	6.6859	0.5527
UIP	0.8361	0.7954	0.2986	0.4581	0.4884	0.8632	0.1587	8.0673	0.6712
UIP + ALL	0.8338	0.7942	0.2981	0.4573	0.4874	0.8619	0.1572	8.0348	0.6684
UIP + LSC	0.8375	0.7960	0.2990	0.4585	0.4889	0.8647	0.1592	**8.0889**	**0.6718**
UIP + MSC	0.8381	0.7951	0.2983	0.4578	0.4884	0.8638	0.1615	8.0756	0.6709
UIP + MHSC	**0.8408**	0.7974	**0.2999**	0.4585	0.4891	0.8669	0.1489	8.0424	0.6682
UIP + HSC	0.8391	0.7947	0.2993	0.4586	0.4886	0.8646	0.1591	8.0432	0.6672
UIP + LHC	0.8404	**0.7994**	0.2996	**0.4601**	**0.4909**	**0.8675**	0.1481	8.0541	0.6704
UIP + HPE	0.8402	0.7965	0.2995	0.4582	0.4889	0.8660	0.1457	8.0154	0.6665
UIP + LH	0.8390	0.7960	0.2997	0.4581	0.4885	0.8657	0.1504	8.0260	0.6686
768									
UI	0.6937	0.6796	0.2583	0.4121	0.4307	0.7292	**0.2503**	7.0112	0.5815
UIP	0.8383	0.7971	0.2986	0.4574	0.4886	0.8648	0.1508	8.0427	0.6687
UIP + ALL	0.8370	0.7961	0.2986	0.4581	0.4886	0.8642	0.1582	8.0670	0.6698
UIP + LSC	**0.8419**	0.7975	**0.3002**	0.4589	0.4894	**0.8677**	0.1504	8.0540	0.6685
UIP + MSC	0.8386	**0.7977**	0.2998	**0.4598**	**0.4899**	0.8662	0.1504	8.0166	0.6658
UIP + MHSC	0.8377	0.7961	0.2986	0.4582	0.4887	0.8644	0.1545	8.0393	0.6672
UIP + HSC	0.8381	0.7960	0.2985	0.4576	0.4883	0.8645	0.1633	**8.1166**	**0.6750**
UIP + LHC	0.8373	0.7971	0.2991	0.4586	0.4893	0.8647	0.1570	8.0582	0.6689
UIP + HPE	0.8391	0.7972	0.2993	0.4583	0.4892	0.8659	0.1502	8.0239	0.6667
UIP + LH	0.8362	0.7936	0.2975	0.4569	0.4870	0.8619	0.1601	8.0532	0.6703

Table 9: Results of our model on the ML1M dataset. In **bold**, the best overall score, while the best score obtained by a configuration based on FOL rules is underlined.

configuration is never the one labeled as UIP+ALL. Similarly, the idea of only using the rules that have a *like* in the head led to satisfying results. Indeed, while the UIP+LH configuration got the best partial results on some comparisons (*i.e.*, Recall and F1 on DBbook with $k = 256$, or some metrics on ML1M with $k = 256$) other subsets of rules tend to obtain better results. Accordingly, it seems that the injection of FOL can be useful also to identify

patterns in the data that do not necessarily identify or clarify the preferences of the users.

In particular, the configurations based on the use of *confidence* scores as a means to select the rules are the most promising ones. As for Last.fm, the UIP+LSC configuration (based on rules with a low confidence threshold) got the best results on Precision, Recall, F1 and Gini Index. Good performance was also obtained by UIP+LHC that got the best results on the remaining five metrics. In this case, the *coverage* is used as criterion to select the rules.

Next, by considering DBbook data, the best results are obtained with the UIP+HSC configuration. In this case, a lower number of rules are injected since the system only exploits rules having a confidence higher than 0.9. The adoption of this subset allowed to beat all the other configurations on all the metrics, with the exception of Precision and F1. Accordingly, this confirms the good impact of the rules to better rank the items and to make the recommendation more novel and diverse. A similar behavior emerged on ML1M data, since the exploitation of rules with *high confidence* led to the best results in terms of novelty and diversity, while the adoption of the *coverage* as heuristic improves the accuracy metrics.

Overall, we can state that the most promising subsets are those that use *confidence* (LSC and HSC) and *coverage* (LHC) as heuristics to select the rules. As regards the threshold to be adopted to cut the selection of the rules, some relationship between the structure of the graph and the amount of rules to be injected seems to exist. Indeed, with smaller KGs (such as DBbook), with a lower number of entities and a lower number of properties per item (see Table 3), it is better to inject less rules. Conversely, when more data are available, more rules can be exploited (so, LSC configuration). For example, this holds for ML1M where a huge number of groundings are injected in the model (see Table 5). This can give a general guideline that can both: *(i)* guide the design of further implementations based on our subsets; *(ii)* provide a deeper understanding of the results of our model.

To conclude, our analysis showed that the heuristics we designed are able to automatically identify a subset of the rules that can be useful to improve the accuracy of the recommendations. Of course, further qualitative and quantitative analyses (*i.e.,* based on the kind of properties mentioned in the rules) will be carried out to go into details of the single rules injected in the model, in order to better assess the kind of information which is needed or useful to improve the quality of the recommendation process. These analyses are left as future work.

5.3 RQ3: Comparisons to Baselines

To answer the RQ3, we carried out experiments aimed at comparing the results obtained by our recommender system based on neuro-symbolic graph embeddings with some competitive state-of-the-art recommendation models. The results of the comparison can be found in Table 10 for the music domain (Last.fm), in Table 11 for book domain (DBbook), and finally for the movie

Baseline	Accuracy						Diversity	Novelty	
	MAP	**Precision**	**MAR**	**Recall**	**F1**	**nDCG**	**Gini**	**EFD**	**EPC**
MultiVAE	0.5531	0.4467	0.6955	0.9406	0.5666	0.8173	**0.3050***	6.7488	0.6079
VSM	0.5617	0.4440	0.7043	0.9381	0.5638	0.8257	0.3043	6.7989	0.6121
CFGAN	0.5669	0.4501	0.7111	0.9484	0.5712	0.8314	0.2819	6.7171	0.6115
LightGCN	0.5665	0.4500	0.7125	0.9483	0.5711	0.8311	0.2792	6.6889	0.6100
NGCF	0.5475	0.4437	0.6921	0.9361	0.5632	0.8080	0.3004	6.6980	0.6026
KaHFM	0.5460	0.3921	0.6509	0.8176	0.4949	0.7653	0.1600	6.3085	0.6099
BPRMF	0.5854	0.4504	0.7277	0.9507	0.5718	0.8519	0.2682	6.6775	0.6175
PureSVD	0.5923	0.4488	0.7292	0.9421	0.5686	0.8555	0.2696	6.7875	0.6273
Slim	0.5118	0.2875	0.5410	0.5866	0.3595	0.6521	0.1366	6.7781	0.6257
ItemKnn	0.5940	0.4463	0.7275	0.9359	0.5654	0.8549	0.2575	6.7666	0.6269
UserKnn	0.5813	0.4230	0.6963	0.8756	0.5332	0.8203	0.2340	6.7974	**0.6340***
Attr.ItemKnn	0.5064	0.3849	0.5977	0.7784	0.4814	0.7097	0.2769	6.5825	0.5953
Attr.UserKnn	0.5681	0.4130	0.6733	0.8524	0.5205	0.7968	0.2218	6.6102	0.6198
EASE^R	**0.6028**	**0.4549**	**0.7451***	**0.9586**	**0.5772**	**0.8726***	0.2714	6.8311	0.6305
Our best	0.5760	0.4528	0.7237	0.9549	0.5744	0.8470	0.2923	**6.8858***	0.6202

Table 10: Comparisons to baselines in the music field (Last.fm dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant.

domain in Table 12 (ML1M). In particular, we compared the performance of the best configuration in each domain with all the baselines considered. Generally speaking, it is easy to observe that our model outperformed all baselines for almost all the considered metrics, with the only exception represented by EASE^R. Moreover, most of these improvements are statistically significant ($p < 0.05$).

In the music domain, the configuration of our model that got the best overall performance is the one injecting FOL rules with a standard confidence score greater than 0.2 (the UIP + LSC configuration reported in Table 7). From Table 10, we can observe that our model obtains satisfying results, since they are comparable with those obtained by the best baseline (*i.e.*, EASE^R) in terms of precision, recall and F1 measure. Moreover, we got the best results in terms of novelty (measured through EFD). As for ranking measures, such as NDCG, we obtain lower results w.r.t. EASE^R, but we are still in line with the next best-performing techniques such as BPRMF, ItemKNN and PureSVD. In our opinion, to better comprehend the nature of the results it is necessary to consult the statistics presented in Table 4 and Table 5. Indeed, both the tables show that the number of items mapped to the knowledge graph and the number of FOL rules are generally low, especially when compared to ML1M. Accordingly, it is likely that the sparsity of the graph and the low quality of information that is mined from the graph contributed to these results.

As for the book domain, Table 11 shows that the our approach does not overcome all the baselines we considered. Unfortunately, this holds for all the metrics since we have at least one baseline that beats our approach. As we already noted for LastFM, EASE^R got the best results. However, in this case

Baseline	Accuracy						Diversity	Novelty	
	MAP	Precision	MAR	Recall	F1	nDCG	Gini	EFD	EPC
MultiVAE	0.6407	0.6389	0.3022	0.4997	0.5163	0.6803	0.2728	7.3677	0.6331
VSM	0.6388	0.6315	0.3011	0.4951	0.5107	0.6766	**0.2923***	**7.5134***	0.6320
CFGAN	0.6380	0.6341	0.2988	0.4948	0.5122	0.6766	0.2551	7.3709	0.6318
LightGCN	0.6351	0.6352	0.2976	0.4947	0.5127	0.6752	0.2102	7.3602	0.6307
NGCF	0.6298	0.6297	0.2960	0.4915	0.5084	0.6699	0.2295	7.4636	0.6305
KaHFM	0.6719	0.6424	0.3067	0.4864	0.5127	0.6943	0.0791	6.7468	0.6673
BPRMF	0.6711	0.6593	0.3137	0.5141	0.5333	0.7071	0.1879	6.9947	0.6547
PureSVD	0.6638	0.6364	0.3062	0.4878	0.5097	0.6894	0.1432	7.0166	0.6576
Slim	0.6352	0.5604	0.2724	0.4053	0.4360	0.6283	0.0868	7.1835	0.6620
ItemKnn	0.6211	0.5177	0.2741	0.3936	0.4111	0.6061	0.1028	7.3233	0.6911
UserKnn	0.6312	0.5270	0.2738	0.3906	0.4138	0.6141	0.0930	7.2967	**0.6998**
Attr.ItemKnn	0.5107	0.3731	0.2171	0.2887	0.2984	0.4720	0.0995	7.1571	0.6495
Attr.UserKnn	0.5963	0.4880	0.2465	0.3474	0.3759	0.5700	0.0853	6.9958	0.6799
EASE^R	**0.7096***	**0.6781**	**0.3351***	**0.5339***	**0.5502***	**0.7415***	0.1887	7.4077	0.6841
Our best	0.6413	0.6381	0.3009	0.4985	0.5157	0.6803	0.2787	7.4381	0.6362

Table 11: Comparisons to baselines in the book field (DBbook dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant.

the gap is statistically significant. This behavior can have multiple reasons, that are independent from the choice of exploiting FOL in the model (indeed, as shown in Table 8, the use of FOL rules overcomes the basic configurations, whose performance would have been even worse). In our opinion, it is very likely that the characteristics of the dataset as shown in Table 4, only 1931 out of 6698 items in the dataset are mapped to the knowledge graph. Accordingly, most of the users and most of the items do not exploit the information coming either from descriptive properties or logical rules. Accordingly, it is not surprising that a knowledge-aware method as the one we propose does not beat advanced data-driven approaches such as BPRMF, that do not exploit exogenous knowledge and only rely on the available items. However, further investigations will be carried out to better understand the behavior of the model on this dataset.

Finally, let us discuss the results obtained in the movie domain (ML1M), which are reported in Table 12. In this case, these results better show the effectiveness of our approach since the best configuration (the one injecting FOL rules with head coverage scores greater than 0.2, labeled as UIP + LHC, as reported in Table 9)) obtained the best results for all the accuracy metrics. Our model also obtained the best results in terms of novelty, with pretty wide gaps with respect to the baselines. All these differences are statistically significant with $p < 0.01$ (with the only exception of the MAR). Also in this case, EASE^R emerged as the best baseline, since only the best diversity is obtained by CFGAN. However, these results are generally not surprising. ML1M is a denser and larger dataset than the previous two, where a larger number

Baseline	Accuracy						Diversity	Novelty	
	MAP	**Precision**	**MAR**	**Recall**	**F1**	**nDCG**	**Gini**	**EFD**	**EPC**
MultiVAE	0.7586	0.7245	0.2743	0.4275	0.4515	0.7861	0.0713	6.6728	0.5385
VSM	0.7036	0.6729	0.2559	0.4041	0.4242	0.7294	0.1560	6.7085	0.6433
CFGAN	0.6311	0.6229	0.2404	0.3883	0.4024	0.6670	**0.2801***	6.3929	0.5751
LightGCN	0.5997	0.6016	0.2349	0.3835	0.3942	0.6412	0.1781	6.1745	0.5141
NGCF	0.6573	0.6377	0.2413	0.3895	0.4051	0.6873	0.2133	6.4227	0.5802
KaHFM	0.7616	0.7259	0.2717	0.4244	0.4492	0.7877	0.1237	6.9823	0.5356
BPRMF	0.7622	0.7291	0.2729	0.4264	0.4512	0.7894	0.1038	6.8912	0.6120
PureSVD	0.7683	0.7280	0.2750	0.4275	0.4518	0.7927	0.1040	6.9338	0.5677
Slim	0.7569	0.7241	0.2717	0.4256	0.4504	0.7842	0.2563	7.6629	0.5968
ItemKnn	0.8239	0.7767	0.2929	0.4487	0.4773	0.8472	0.0957	7.3958	0.5141
UserKnn	0.8182	0.7750	0.2888	0.4467	0.4758	0.8425	0.0959	7.3426	0.6047
Attr.ItemKnn	0.7322	0.6982	0.2630	0.4117	0.4341	0.7581	0.1693	7.0742	0.5807
Attr.UserKnn	0.7802	0.7456	0.2741	0.4313	0.4587	0.8056	0.1069	7.0611	0.5651
EASE^R	0.8328	0.7865	0.2936	0.4508	0.4811	0.8563	0.0995	7.5592	0.626
Our best	**0.8404***	**0.7994***	**0.2996**	**0.4601***	**0.4909***	**0.8675***	0.1481	**8.0541***	**0.6704***

Table 12: Comparisons to baselines in the movie field (ML1M dataset). The best overall value is highlighted in bold, while the best baseline value is underlined. (*) means the differences are statistically significant with $p < 0.05$.

of FOL rules are mined and exploited by our model. Accordingly, this naturally improves the performance w.r.t. other methods; moreover, due to the same reason, the impact of external knowledge is less pronounced, and this improves methods based on matrix factorization and, more in general, collaborative filtering. To conclude, we can state that the comparison to the state of the art showed that our approach is particularly suitable and effective when the graph is large enough, and when a large amount of rules can be mined and extracted from the triples. In these scenarios, we are able to even beat current state-of-the-art approaches.

Overall, we can say that our model outperformed almost all baselines in almost all accuracy metrics, and with statistically significant improvements with few exceptions; good and significant improvements have been observed also for non-accuracy metrics, such as diversity (Gini index) and novelty (EFD, EPC).

6 Discussion and Limitations

Although these are good aspects of our model, we are aware there are some limitations, that we will tackle in future works:

- *Choice of the embedding learning model*: although TransE got good performance in the task of graph embedding for the recommendation, there are more recent and sophisticated methods to embed graph-derived information, including GNNs and GCNs; exploiting again the First-Order Logic as a unified framework for both triples and FOL rules could be a good strategy

to combine more effective graph embedding models and keeping injecting FOL rules, to obtain again *neuro-symbolic graph embeddings*.
- *Identification of the FOL rules*: in this work, we mined FOL from the KG itself. Another option could be adding rules provided by domain experts to discover new knowledge that is not even encoded in the original KG as *background knowledge*.
- *A weighting mechanism for FOL rules*: in this work, we treated all FOL rules with the same importance, even when they had different confidence values. An extension of this work might be a *weighting mechanism* for the rules, aiming to lower the impact of rules with low confidence scores while amplifying the impact of rules with higher confidence scores.
- *An automated FOL rules selection strategy*: in this work, we considered eight different subsets of FOL rules, related to eight different heuristics. Although the standard confidence seems to be the best metric to select the most prominent heuristic, an automated mechanism to select the best threshold value is currently lacking, and will be considered as a future work.
- *A user study*: in this work, we only performed *in-vitro* experiments; executing *in-vivo* experiments by evaluating how this model affects the recommendation lists for real users in necessary as well.
- *More sophisticated methodologies to provide recommendation*: in this work, we used simple concatenation to combine the embeddings related to users and items, but more sophisticated methods might be exploited, such as the *attention mechanisms*. In addition, even though this is beyond the scope of this paper, combining information related to other information sources (e.g., plain text) could even improve the performance of the recommender system.

7 Conclusions and Future Work

In this paper, we presented a model aiming at *(i):* mining FOL rules from a KG; *(ii):* learning graph embedding by injecting the knowledge provided by the FOL rules; *(iii):* providing users with effective recommendations. We tested this methodology on three datasets related to different domains (music, books, and movies) and carried out extensive experiments to assess the effectiveness of the model.

The experiments carried out proved that the initial intuition was correct: first, the introduction of FOL rules has a decisive impact on both item representation and recommendation lists generation; then, our model improved the performance with respect to diverse competitive baselines; last but not the least, we have been able to derive some guidelines choose the most promising subset of FOL rules to be injected, with particular focus on the standard confidence score.

This work represents a first step in the adoption of neuro-symbolic AI in the field of graph embedding for the recommendation, and we believe that much more research can be done in this field. As future works, we will work at overcoming the sketched limitations, by also comparing the model in terms

of trade-off Spillo et al (2023a) between sustainability and effectiveness of the approach.

Acknowledgement

This research is partially funded by PNRR project FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI and Spoke 8 - Pervasive AI (CUP H97G22000210007) under the NRRP MUR program funded by the NextGenerationEU, by the project PHaSE (CUP H53D23003530006) - Promoting Healthy and Sustainable Eating through Interactive and Explainable AI Methods, funded by MUR under the PRIN program, and by the Apulia Region under the Program "RIPARTI (assegni di Ricerca per riPARTire con le Imprese)", POC PUGLIA FESR- FSE 2014 / 2020 - CUP: H93C22000520002, project: "Recommendation systems and Digital Storytelling in Tourism". We acknowledge the CINECA award under the ISCRA initiative, for the availability of high-performance computing resources and support. We want to thank the student Gianmarco Turchiano for his technical support and for carrying out some of the experiments.

References

Albanese M, d'Acierno A, Moscato V, et al (2013) A multimedia recommender system. ACM Transactions on Internet Technology (TOIT) 13(1):1–32

Andjelkovic I, Parra D, O'Donovan J (2019) Moodplay: interactive music recommendation based on artists' mood similarity. International Journal of Human-Computer Studies 121:142–159

Anelli VW, Di Noia T, Di Sciascio E, et al (2019) How to make latent factors interpretable by feeding factorization machines with knowledge graphs. In: International Semantic Web Conference, Springer, pp 38–56

Anelli VW, Bellogín A, Ferrara A, et al (2021) Elliot: a comprehensive and rigorous framework for reproducible recommender systems evaluation. CoRR abs/2103.02590

Asani E, Vahdat-Nejad H, Sadri J (2021) Restaurant recommender system based on sentiment analysis. Machine Learning with Applications 6:100,114

Belkin M, Niyogi P (2002) Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, pp 585–591

Bellogin A, Castells P, Cantador I (2011) Precision-oriented evaluation of recommender systems: an algorithmic comparison. In: Proceedings of the fifth ACM conference on Recommender systems, pp 333–336

Bonnin G, Jannach D (2014) Automated generation of music playlists: Survey and experiments. ACM Computing Surveys (CSUR) 47(2):1–35

Borchers A, Herlocker J, Konstan J, et al (1998) Ganging up on information overload. Computer 31(4):106–108

Bordes A, Usunier N, Garcia-Duran A, et al (2013) Translating embeddings for modeling multi-relational data. Advances in Neural Information Processing Systems 26:2787–2795

Cai H, Zheng VW, Chang KCC (2018) A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Transactions on Knowledge and Data Engineering 30(9):1616–1637

Carraro T, Daniele A, Aiolli F, et al (2024) Mitigating data sparsity via neuro-symbolic knowledge transfer. In: European Conference on Information Retrieval, Springer, pp 226–242

Chae DK, Kang JS, Kim SW, et al (2018) CFGAN: a generic collaborative filtering framework based on generative adversarial networks. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp 137–146

Chen Z, Wang Y, Zhao B, et al (2020) Knowledge graph completion: A review. IEEE Access 8:192,435–192,456

De Filippo A, Lombardi M, Milano M (2021) Integrated offline and online decision making under uncertainty. J Artif Intell Res 70:77–117. https://doi.org/10.1613/JAIR.1.12333, URL https://doi.org/10.1613/jair.1.12333

De Filippo A, Borghesi A, Boscarino A, et al (2022) HADA: an automated tool for hardware dimensioning of AI applications. Knowl Based Syst 251:109,199. https://doi.org/10.1016/J.KNOSYS.2022.109199, URL https://doi.org/10.1016/j.knosys.2022.109199

De Gemmis M, Lops P, Musto C, et al (2015) Semantics-aware content-based recommender systems. In: Recommender systems handbook. Springer, p 119–159

Deldjoo Y, Elahi M, Cremonesi P, et al (2016) Content-based video recommendation system based on stylistic visual features. Journal on Data Semantics 5(2):99–113

Devlin J, Chang MW, Lee K, et al (2019) BERT: pre-training of deep bidirectional transformers for language understanding

Felfernig A, Friedrich G, Jannach D, et al (2015) Constraint-based recommender systems. Recommender Systems Handbook pp 161–190

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

Galárraga L, Teflioudi C, Hose K, et al (2015) Fast rule mining in ontological knowledge bases with AMIEPlus. The VLDB Journal 24(6):707–730

Galárraga LA, Teflioudi C, Hose K, et al (2013) Amie: association rule mining under incomplete evidence in ontological knowledge bases. In: Proceedings of the 22nd International Conference on World Wide Web, pp 413–422

Gantner Z, Rendle S, Freudenthaler C, et al (2011) MyMediaLite: A free recommender system library. In: Proceedings of the Fifth ACM conference on Recommender Systems, pp 305–308

Garcez Ad, Gori M, Lamb LC, et al (2019) Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. arXiv preprint arXiv:190506088

Goyal P, Ferrara E (2018) Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems 151:78–94

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

Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, Acm, pp 855–864

Gu Y, Yang K, Fu S, et al (2018) Hybrid attention based multimodal network for spoken language classification. In: Proceedings of the Conference. association for Computational Linguistics. meeting, NIH Public Access, p 2379

Guo S, Wang Q, Wang L, et al (2016) Jointly embedding knowledge graphs and logical rules. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp 192–202

Guo S, Wang Q, Wang L, et al (2018) Knowledge graph embedding with iterative guidance from soft rules. In: Proceedings of the AAAI Conference on Artificial Intelligence

Hájek P (2013) Metamathematics of fuzzy logic, vol 4. Springer Science & Business Media

He X, Deng K, Wang X, et al (2020) 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, pp 639–648

Huang Z, Chung W, Ong TH, et al (2002) A graph-based recommender system for digital library. In: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, Acm, pp 65–73

Karimi H, Kamandi A (2019) A learning-based ontology alignment approach using inductive logic programming. Expert Systems With Applications 125:412–424

Kautz H (2022) The third AI summer: AAAI Robert S. Engelmore Memorial Lecture. AI Magazine 43(1):93–104

Kotsiantis S, Kanellopoulos D (2006) Association rules mining: A recent overview. GESTS International Transactions on Computer Science and Engineering 32(1):71–82

Lajus J, Galárraga L, Suchanek F (2020) Fast and exact rule mining with AMIE 3. In: European Semantic Web Conference, Springer, pp 36–52

Liang D, Krishnan RG, Hoffman MD, et al (2018) Variational autoencoders for collaborative filtering. In: Proceedings of the 2018 world wide web conference, pp 689–698

Lika B, Kolomvatsos K, Hadjiefthymiades S (2014) Facing the cold start problem in recommender systems. Expert Systems With Applications 41(4):2065–2073

Liu P, Zhang L, Gulla JA (2020) Dynamic attention-based explainable recommendation with textual and visual fusion. Information Processing & Management 57(6):102,099

Liu T, Wang Z, Tang J, et al (2019) Recommender systems with heterogeneous side information. In: The World Wide Web Conference, pp 3027–3033

Lops P, de Gemmis M, Semeraro G, et al (2009) A semantic content-based recommender system integrating folksonomies for personalized access. In: Castellano G, Jain LC, Fanelli AM (eds) Web Personalization in Intelligent Environments, Studies in Computational Intelligence, vol 229. p 27–47, https://doi.org/10.1007/978-3-642-02794-9_2, URL https://doi.org/10.1007/978-3-642-02794-9_2

Lops P, de Gemmis M, Semeraro G, et al (2011) Leveraging the linkedin social network data for extracting content-based user profiles. In: Mobasher B, Burke RD, Jannach D, et al (eds) Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011. ACM, pp 293–296, https://doi.org/10.1145/2043932.2043986, URL https://doi.org/10.1145/2043932.2043986

McAuley J, Targett C, Shi Q, et al (2015) Image-based recommendations on styles and substitutes. In: Proceedings of the 38th international ACM SIGIR Conference on Research and Development in Information Retrieval, pp 43–52

Musto C, Semeraro G, Lops P, et al (2012) Leveraging social media sources to generate personalized music playlists. In: Huemer C, Lops P (eds) E-Commerce and Web Technologies - 13th International Conference, EC-Web 2012, Vienna, Austria, September 4-5, 2012. Proceedings, Lecture Notes in Business Information Processing, vol 123. Springer, pp 112–123, https://doi.org/10.1007/978-3-642-32273-0_10, URL https://doi.org/10.1007/978-3-642-32273-0_10

Musto C, Semeraro G, Lops P, et al (2014) Combining distributional semantics and entity linking for context-aware content-based recommendation. In: Dimitrova V, Kuflik T, Chin D, et al (eds) User Modeling, Adaptation, and Personalization - 22nd International Conference, UMAP 2014, Aalborg, Denmark, July 7-11, 2014. Proceedings, Lecture Notes in Computer Science, vol 8538. Springer, pp 381–392, https://doi.org/10.1007/978-3-319-08786-3_34, URL https://doi.org/10.1007/978-3-319-08786-3_34

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

Musto C, Franza T, Semeraro G, et al (2018) Deep content-based recommender systems exploiting recurrent neural networks and linked open data. In: Adjunct Publication of the 26th conference on user modeling, adaptation and personalization, pp 239–244

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

Musto C, de Gemmis M, Lops P, et al (2022) Semantics and content-based recommendations. In: Ricci F, Rokach L, Shapira B (eds) Recommender Systems Handbook. Springer US, p 251–298, https://doi.org/10.1007/978-1-0716-2197-4_7, URL https://doi.org/10.1007/978-1-0716-2197-4_7

Ning X, Karypis G (2011) Slim: Sparse linear methods for top-n recommender systems. In: 2011 IEEE 11th International Conference on Data Mining, IEEE, pp 497–506

Van den Oord A, Dieleman S, Schrauwen B (2013) Deep content-based music recommendation. Advances in Neural Information Processing Systems 26

Palumbo E, Rizzo G, Troncy R (2017) Entity2rec: Learning user-item relatedness from knowledge graphs for Top-N item recommendation. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, Acm, pp 32–36

Palumbo E, Rizzo G, Troncy R, et al (2018) Translational models for item recommendation. In: European Semantic Web Conference, Springer, pp 478–490

Petruzzelli A, Martina AFM, Spillo G, et al (2024) Improving transformer-based sequential conversational recommendations through knowledge graph embeddings. In: Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2024, Cagliari, Italy, July 1-4, 2024. ACM, pp 172–182, https://doi.org/10.1145/3627043.3659565, URL https://doi.org/10.1145/3627043.3659565

Piao G, Breslin JG (2016) Measuring semantic distance for linked open data-enabled recommender systems. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp 315–320

Polignano M, Musto C, de Gemmis M, et al (2021) Together is better: Hybrid recommendations combining graph embeddings and contextualized word representations. In: Fifteenth ACM Conference on Recommender Systems, pp 187–198

Reimers N, Gurevych I (2019) Sentence-BERT: sentence embeddings using siamese bert-networks. arXiv preprint arXiv:190810084

Rendle S, Freudenthaler C, Gantner Z, et al (2012) BPR: bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:12052618

Ristoski P, Loza Mencía E, Paulheim H (2014) A hybrid multi-strategy recommender system using linked open data. In: Semantic Web Evaluation Challenge, Springer, pp 150–156

Rocktäschel T, Bosnjak M, Singh S, et al (2014) Low-dimensional embeddings of logic. In: Proceedings of the ACL 2014 Workshop on Semantic Parsing, pp 45–49

Rocktäschel T, Singh S, Riedel S (2015) 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, pp 1119–1129

Sarker MK, Zhou L, Eberhart A, et al (2021) Neuro-symbolic artificial intelligence. AI Communications (Preprint):1–13

Schafer JB, Frankowski D, Herlocker J, et al (2007) Collaborative filtering recommender systems. In: The Adaptive Web: Methods and Strategies of Web Personalization. Springer, p 291–324

Shi B, Weninger T (2018) Open-world knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence

Smullyan RM (1995) First-order logic. Courier Corporation

Song W, Duan Z, Yang Z, et al (2019) Explainable knowledge graph-based recommendation via deep reinforcement learning. arXiv preprint arXiv:190609506

Spillo G, Filippo AD, Musto C, et al (2023a) Towards sustainability-aware recommender systems: Analyzing the trade-off between algorithms performance and carbon footprint. In: Zhang J, Chen L, Berkovsky S, et al (eds) Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023, Singapore, Singapore, September 18-22, 2023. ACM, pp 856–862, https://doi.org/10.1145/3604915.3608840, URL https://doi.org/10.1145/3604915.3608840

Spillo G, Musto C, Polignano M, et al (2023b) Combining graph neural networks and sentence encoders for knowledge-aware recommendations. In: Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2023, Limassol, Cyprus, June 26-29, 2023. ACM, pp 1–12, https://doi.org/10.1145/3565472.3592965, URL https://doi.org/10.1145/3565472.3592965

Sun Z, Yang J, Zhang J, et al (2018) Recurrent knowledge graph embedding for effective recommendation. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp 297–305

Sun Z, Guo Q, Yang J, et al (2019) Research commentary on recommendations with side information: A survey and research directions. Electronic Commerce Research and Applications 37:100,879

Tarus JK, Niu Z, Mustafa G (2018) Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. Artificial Intelligence Review 50:21–48

Valiant LG (2003) Three problems in computer science. Journal of ACM 50(1):96–99. https://doi.org/10.1145/602382.602410, URL https://doi.org/10.1145/602382.602410

Vargas S, Castells P (2011) Rank and relevance in novelty and diversity metrics for recommender systems. In: Proceedings of the Fifth ACM conference on Recommender systems, pp 109–116

Vashishth S, Sanyal S, Nitin V, et al (2019) Composition-based multi-relational graph convolutional networks. arXiv preprint arXiv:191103082

Vaswani A, Shazeer N, Parmar N, et al (2017) Attention is all you need. In: Advances in Neural Information Processing Systems, pp 5998–6008

Wang A, Singh A, Michael J, et al (2018) GLUE: A multi-task benchmark and analysis platform for natural language understanding. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Association for Computational Linguistics, Brussels, Belgium, pp 353–355, https://doi.org/10.18653/v1/W18-5446, URL https://aclanthology.org/W18-5446

Wang X, He X, Cao Y, et al (2019a) KGAT: Knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 950–958

Wang X, He X, Wang M, et al (2019b) Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, pp 165–174

Wang Z, Zhang J, Feng J, et al (2014) Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI conference on Artificial Intelligence

Xie M, Yin H, Wang H, et al (2016) Learning graph-based poi embedding for location-based recommendation. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp 15–24

Xue HJ, Dai X, Zhang J, et al (2017) Deep matrix factorization models for recommender systems. In: IJCAI, Melbourne, Australia, pp 3203–3209

Zhang W, Yan J, Wang Z, et al (2022) Neuro-symbolic interpretable collaborative filtering for attribute-based recommendation. In: Proceedings of the ACM Web Conference 2022, pp 3229–3238

Zhang Y, Wang J, Luo J (2020) Knowledge graph embedding based collaborative filtering. IEEE Access 8:134,553–134,562