SLR on Symbolic Knowledge Injection

Lorenzo Bacchiani
University of Bologna, Italy
lorenzo.bacchiani@studio.unibo.it

1 Introduction

An interesting topic in Artificial Intelligence is the the ability of the system to learn the underlying patterns in data in order to achieve some kind of human-like reasoning. Indeed, humans are continually acquiring, representing and reasoning with new facts about the world.

To make sense of the large quantity of information with which we are presented, we should compress structure and generalize from what we experience. This allows us to quickly understand new concepts and make useful predictions about them. Such an ability is useful when collecting data is hindered by:

1. **Limited data:** data available is limited in terms of feature coverage since these systems typically run in an operationally optimized settings and to collect data outside this narrow range is usually exepnsive or even unsafe.
2. **Expensive data:** in some instances, for example manufacturing facilities, collection of data may be disruptive or require destructive measurements.
3. **Poor quality data:** quality of data collected from physical infrastructure systems is usually poor (e.g., missing, corrupted, or noisy data) since they typically have old and legacy components.

The injection of prior-knowledge into learning process is a fundamental step in overcoming these limitations. Firstly, it does not require the training process to induce this knowledge from the training set, therefore reducing the number of required training data. Secondly, prior knowledge can be used to express the desired behavior of the learner on any input, providing better behavior guarantees in an adversarial or uncontrolled environment. Such a knowledge is usually represented as rules to perform human-like reasoning and inference.

Our goal is to report a systematic literature review (SLR) that investigates on symbolic knowledge injection paradigms. This work is organized as follows. Section 2 describes the method used to perform this systematic review. Section 3 reports the results obtained answering the research questions and discussing the results. Finally, conclusions are presented in Section 4.

2 Method

This Systematic Literature Review (SLR) investigates the symbolic knowledge injection. This SLR is based on guidelines described in [5] and it is performed in three main phases: planning, conducting and reporting the review. This section presents the steps followed in order to perform the review and reduce the possibility of researcher bias. These steps include:

1. Identifying the resarch questions.
2. Defining a searching strategy.

3. Defiining a study selection criteria.
4. Assessing the quality of Primary Studies (PSs).
5. Performing the corrisponging data extraction and analysis.

2.1 Research questions

The research qustions addressed by this study are:

1. What are the Symbolic Knowledge injection approaches considered in PSs?
2. What are the Symbolic Knowledge forms adopted in the considered in PSs?
3. Can the Symbolic Knowledge injection approaches considered in PSs be categorized?

To achieve the goal of this SLR we searched the literature for the corresponding PSs.

2.2 Search process

Due to the fact that the topic of this SLR, Symbolic Knowledge Injection is a convenient keywords used to indicate the process to enrich a system with prior-konwledge and being the techniques used to fulfill this purpose outside of our knowledge, we decided to add a further step to the standard PSs search process described by Kitchenham and Charters [5]. Indeed, instead of starting with the identification of the PSs we deeply analyzed the articles supplied by professor and thanks to them we were able to gain significant knowledge about the SLR topic and this made possible to build search queries that produced sensible results. On the basis of the gained knowledge and by taking into account the research questions, we have built the following research queries:

1. Symbolic knowledge in deep learning
2. Symbolic knowledge in deep logic models.
3. Joint embedding logical rules.
4. Neural ”inductive logic programming”[1]

We conducted the research by means of the search engine Google Scholar which allowed us to find a great number of documents from heterogeneous digital libraries. The search and PS selection process was performed in 4 stages, as listed in Table 1.

In stage 1, potentially useful articles, found thanks to the research queries, are collected to be analyzed later. After that, in stage 2 and stage 3, we made an initial analysis of the articles by taking into account the title and the abstract in order to remove both blatantly irrelevant - e.g. the ones that does not introduce the approach used - and duplicate ones respectively. Finally, stage 4 focuses on the full text analysis to exclude the articles that does not address the research questions.

For each article, we recorded in an Excel spreadsheet the title, the authors, year of the publication of the articles. In this way it was possible to identify and discard any duplicates. As expected, the queries alone did not get us a reasonable number of documents to analyze, indeed, at the time of research (June 2020), we obtained more than 40000 documents and of course it was necessary to introduce additional restrictions in order to exclude a priori documents that did not respect these constraints.

[1] The “” forces search engine to look for the words between quotation marks in the defined order.

Stages	Included (+) or removed (-) articles	Total articles
Stage 1: Identifying potentially relevant article	+400	400
Stage 2: Excluding articles based on titles and abstract	-340	60
Stage 3: Excluding duplicate article	-21	39
Stage 4: Excluding articles based on full text	-19	20

Table 1: Stages of the paper selection process.

2.3 Study selection

The next step in the studies selection process is to decide the additional constraints mentioned above, in particular for this SLR we decided to take into account:

1. All documents published starting from the year 2018.
2. All documents in English.
3. All free access documents or university supplied access documents.

Using these constraints, we were able to significantly reduce the number of documents but, as expected, the imposed constraints were not enough to obtain a manageable number of articles so we decided to analyze the first 100 results for each query. Moreover, following the canonical SLR process, we screened the paper obtained using inclusion and exclusion criteria to answer the research questions. This screening was based on excluding studies that were obviously irrelevant, or duplicates. To select PSs from the potentially relevant studies, we applied the following inclusion and exclusion criteria.

Inclusion criteria:

1. Full paper (not a PowerPoint presentation or extended abstract).
2. Articles published in peer-reviewed journals or conference proceedings.
3. Sources describing in depth a knowledge injection approach.

Exclusion criteria:

1. Papers that do not supply in the abstract any informations about the characteristics of the injection method used.
2. Papers that do not address the research questions or in other words that do not supply any elements for characterizing and understanding in depth the approaches used.
3. Duplicate reports of the same study (when several reports of a study exist in different journals the most complete version of the study was included in the review).

The articles supplied by course's professor are not excluded by means of the restrictions added above, for example documents published starting from the year 2018, but they were analyzed in the 4th stage presented in Table1.

2.4 Quality Assesment

We followed the guidelines suggested in [5] to construct the quality checklist given in Table2. The checklist was applied to asses the quality of each of the PSs considered. Each question was evaluated as "Yes", "Partially" or "No" with a corresponding score of 1, 0.5 or 0 respectively. The results of the quality analysis show that all studies but S3, S8, S10 and S14 score 4 points, while the aforementioned 3.5 points in that some aspects are not clearly stated. Quality assesments are showed in Table3.

ID	Question
Q1	Are the aims clearly stated?
Q2	Are the methodologies adequately described?
Q3	Are there any elements that allow methods categorization?
Q4	Is at least one research question answered?

Table 2: Quality checklist.

ID	Reference	Year	Q1	Q2	Q3	Q4	Total
S1	[18]	2017	Y	Y	Y	Y	4
S2	[14]	2019	Y	Y	Y	Y	4
S3	[12]	2018	Y	P	Y	Y	3.5
S4	[16]	2006	Y	Y	Y	Y	4
S5	[20]	2006	Y	Y	Y	Y	4
S6	[8]	2013	Y	Y	Y	Y	4
S7	[7]	2013	Y	Y	Y	Y	4
S8	[6]	2019	Y	P	Y	Y	3.5
S9	[11]	2016	Y	Y	Y	Y	4
S10	[22]	2020	Y	P	Y	Y	3.5
S11	[17]	2015	Y	Y	Y	Y	4
S12	[9]	2016	Y	Y	Y	Y	4
S13	[23]	2015	Y	Y	Y	Y	4
S14	[19]	2018	Y	P	Y	Y	3.5
S15	[10]	2019	Y	Y	Y	Y	4
S16	[21]	2018	Y	Y	Y	Y	4
S17	[13]	2020	Y	Y	Y	Y	4
S18	[26]	2018	Y	Y	Y	Y	4
S19	[27]	2019	Y	Y	Y	Y	4
S20	[28]	2020	Y	Y	Y	Y	4

Table 3: Quality assesments results.

2.5 Data collection and analisys

The data extracted from each study were:

1. Full reference (title, authors, journal/conference name, year, volume and page numbers).
2. Framework for injecting symbolic knowledge.
3. Approaches exploiting symbolic knowledge.

The data considered shows:

1. All possible interpretations of the term "injection" (addressing RQ1).
2. All the symbolic knowledge possible forms (addressing RQ2).
3. The categorization of the extracted symbolic knowledge injection methodologies (addressing RQ3).

3 Results and Discussion

This section summarizes the results of the PSs analysis. In the sections below are presented in depth all the informations obtained by the analysed PSs. In particular section 3.1 describes the symbolic knowledge injection approaches found, section 3.2 provides a description of the forms that symbolic knowledge takes in order to be fully exploited and finally section 3.3 shows how such approaches can be categorized.

3.1 RQ1. What are the Symbolic Knowledge injection approaches considered in PSs?

As can be easily perceived by intuition, there are several methods of injecting background knowledge to increase systems' performance, or in other word, the term injection can be interpreted in different ways. The purpose of this section is to present all the possible interpretation extracted from the PSs. Basically, in the PSs can be extracted 7 different approaches which will be shown in the subsections below.

3.1.1 Symbolic knowledge injection as best satisfiability problem

The most widely spread, found in [18], [14], [16], [7] and [13], is to intend symbolic knowledge injection as a best satisfiability problem.

The boolean satisfiability problem (SAT problem) is the problem of determining if there exists an interpretation that satisfies a given boolean formula, or in other words, it asks whether the variables of a given boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE [24].

Given the above definition, it is easy to understand how the above mentioned PSs work: the optimization process is framed as finding the unknown elements which maximize the satisfaction of the set of logical formulas.

3.1.2 Symbolic knowledge injection as template model

Another kind of symbolic knowledge injection technique found in [20], is to pump in logical rules by creating a template model which can be trained and filled with ground facts and use it to make prediction on specific examples. In practice, we start from a set of rules P and some labelled examples, used to learn weights for the general rules in P. This process leads to a trained relational network N which contains weighted rules, but does not contain any ground facts. Every time we want to use N we can simply add a set of ground facts M that describe the example we want to make a prediction.

3.1.3 Symbolic knowledge injection as propositionalization

Noteworthy is injection by means of propositionalization which is the process of explicitly transforming first-order logic (or symbolic knowledge) into a propositional (feature-based, attribute-value) representation which can be found in [8]. Logic rules are converted onto features on attribute-value tables and numerical vectors in order to be exploited from an Artificial Neural Network. See [8] for further details on Bottom Clauses Propositionalization algorithm.

3.1.4 Symbolic knowledge injection as semantic loss

Another kind of injection method is the semantic loss ([26]) that allows to translate the desired output structure of a learner via the definition of a loss, which can also accommodate symbolic knowledge. However, the loss and the resulting reasoning process does not define a probabilistic reasoning process, thus limiting the flexibility of the approach.

3.1.5 Symbolic knowledge injection as knowledge distillation

A completely different approach described in [11], is a combination of knowledge distillation and posterior regularization which enables a neural network to learn not only from labeled instances but also from logic rules through an iterative procedure that transfers the structured information in the logic rules into the network parameters.

The posterior regularization method separates model complexity from the complexity of structural constraints it is desired to satisfy by imposing regularization on latent variables. Given that, the approach followed in [11] can be summarized as exploiting posterior regularization to build a rule-regularized *teacher*, and train the *student* network of interest to imitate the predictions of the teacher network.

The main limitation of this work is that the logic is distilled into the network weights but there is no guarantee that the logic will be properly generalized.

3.1.6 Symbolic knowledge injection as joint embeddings

In works like [9], [23], [10], [21], [27] and [28], symbolic knowledge is combined with knowledge graph in different ways:

1. [9] and [27] symbolic knowledge is injected as complex formulae putting in relation different triples e.g. $\forall x, y : (x, r_s, y) \implies (x, r_t, y)$, where the triples are then instantiated with concrete entities. Joint learning is performed by building a training set F containing all positive formulae, including observed triples and ground rules in whitch at least one constituent triple is observed and finally a loss function is minimized.

2. [23] and [28] extracts logic rules from a knowledge graph and performs inference thanks to a MLN built with such rules. They adopt an algorithm working in two phases, the first one called E-step, known as *inference* step, perform inference over posterior distribution of the hidden variables while M-step, known as *learning* step, is used to learn the weights of logic fomulae in MLN.

3. [21] is slightly different from [9] and [27] in that the ground rules are modeled into the knowledge graph embedding process. Indeed, in [21] logical rules directly interact in the vector space with other components e.g. triples, instead of being encoded in the form of true value.

3.1.7 Symbolic knowledge injection as matrix factorization

In [17] knowledge is injected by means of a matrix factorization method where given a sparse binary matrix consisting of observed facts over entity-pairs P and predicates/relations R, matrix factorization is used to learn k-dimensional relation and entity-pair embeddings that approximate the observed matrix. Matrix factorization can be seen as a learning task where low-dimensional embeddings are estimated for all constant pairs and predicates, given a collection of ground atoms.

3.2 RQ2. What are the Symbolic Knowledge forms adopted in the considered in PSs?

At first symbolic knowledge is represented as a set of rules in order to enable the system for reasoning and inference, but clearly this knowledge should be translated in some other forms in order to be injected and fully exploited. The purpose of this section is to investigate what these forms are. Below will be presented in each subsection one of the form extracted from the PSs.

3.2.1 Symbolic knowledge as logical constraints

This method of representing symbolic knowledge is widely used among those extracted from the PSs. In [18], [14], [7] and [13] symbolic knowledge is represented by translating logical expressions into constraints.

In general, machine learning algorithm behaviour is determined by means of a set of parameters θ, e.g. connection weights in a neural network, and the learning task is performed by optimizing a function $f(Train, \theta)$. Logical constraints combine linear constraints by means of logical operators, such as logical-and, logical-or, negation or conditional statements to express complex relations between linear constraints. In the above-mentioned works, these constraints are used to determine the maximization of the objective function that expresses the truth degree of a given logical formula. Integration of learning and logical reasoning is achieved by translating logical expressions into continuous real-valued constraints, by evaluating all its possible groundings and using them to force the desired behaviour.

3.2.2 Symbolic Knowledge as sentential decision diagram

In DeepProbLog presented in [12], the logic formulas are compiled into a Sentential Decision Diagram, a form that allows for efficient evaluation of the query using knowledge compilation technology.

SDDs are a strict superset of ordered binary decision diagrams, which are one of the most popular, tractable representations of Boolean functions. Despite their generality, SDDs still maintain some key properties behind the success of OBDDs in practice. This includes canonicity, which leads to unique representations of Boolean functions. The SDD are evaluated bottom-up to calculate the success probability of the given query, starting with the probability labels of the leaves as given by the program and performing addition in every or-node and multiplication in every and-node.

3.2.3 Symbolic knowledge as template model

In [16], [20] have been spotted that a set of logic rules P is combined with some labelled examples leading to a trained network N (in this particular case it is called lifted relational neural network) which contains weighted rules, but does not contain any ground facts, and, every time someone wants to use this network he/she should first add a set of weighted ground facts M that describe the specific example about the prediction that should be made. Since in [23], [10] and [28] knowledge base completion is

modeled by a Markov Logic Network, it is therefore possible to make the same assumptions made for the MLN.

3.2.4 Symbolic Knowledge as bottom clause

In [8] symbolic knoweldge is converted into bottom clauses where Bottom clauses are boundaries on the hypothesis search space as part of the Progol system, and are built from one random positive example, background knowledge (a set of clauses that describe what is known) and language bias (a set of clauses that define how clauses can be built). A bottom clause is the most specific clause (with most literals) that can be considered as a candidate hypothesis.

3.2.5 Symbolic Knowledge as regularization terms

As stated by the authors of [11], in their approach symbolic knowledge is represented as regularization terms. Also [26] can be considered representing symbolic knowledge as regularization terms in that semantic loss is simply another regularization term that can directly be plugged into an existing loss function. More specifically with some weight w the new overall loss becomes existing loss + w $\times$ semantic loss.

Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term, regularization term or regularizer, controls the excessively fluctuating function such that the coefficients don't take extreme values [15].

3.2.6 Symbolic Knowledge as embeddings

Embedding means representing each entity in KB as a low-dimension numeric vector, and different dimensions of the vector may implicitly represent different aspects of an entity. Relations in KB usually have relevant representations, such as vectors, matrixes and tensors. Entities interact under a specific relation by performing arithmetical operations between entity embeddings and relation's representation. In the considered PSs, [22], [17] and [21] represent logical rules as low-ranked logical embeddings.

3.2.7 Other forms

In [9] logical rules are first instantiated into ground rules and then interpreted as complex formulae constructed by combining ground atom with logical connectives, e.g. $\vee$, and modeled by t-norm fuzzy logics which are a family of non-classical logics, informally delimited by having a semantics that takes the real unit interval $[0, 1]$ for the system of truth.

In [27], logical rules are modeled using the semantic of Web Ontology Language (OWL2), which is an ontology language for the Semantic Web with formally defined meaning.

In [6], logical rules are taken in input as tensors which are algebraic objects that describes a multi-linear relationship between stes of algebraic objects related to a vector space.

Each rule combination in [19] is associated with a set of derivation trees wich are ordered rooted trees that represent the syntactic structure of the possible inference traces.

3.3 RQ3. Can the Symbolic Knowledge injection approaches considered in PSs be categorized?

This section will provide an in depth description of a (non exhaustive) of the categorization dimensions extracted in the PSs and, after that, put each PS into the dimensions found in order to categorize them. The categorization results are shown in Table 4

3.3.1 Statistical Relational Learning vs Knowledge Graph Embedding

This is the first and grossest categorization possible, the approaches are divided on the basis of the type of method used to inject knowledge. In particular, the statistical relational learning (a.k.a. probabilistic inductive logic programming) approach is based on the idea that combining and integrating probabilistic reasoning, first-order logic and machine learning it is possible to build models that can reason and learn in complex and uncertain domains.

As the name suggest, knowledge graph embedding approaches deal with knowledge graph. In particular, a knowledge graph (KG) is a directed heterogeneous multigraph whose node and relation types have domain-specific semantics. A knowledge graph embedding (KGE) is a low-dimensional representations of the entities and relations in a KG, providing a genarizable context about the overall KG that can be used to infer relations following different models. In the PSs examined, [18], [14], [12], [16], [20], [8], [7], [6], [11], [22], [17] and [19] use a pure statistical relational learning approach while [9], [23], [10], [21], [13], [26], [27], [28] include a knowledge graph in the process of symbolic knowledge injection. These two different approaches can be further divided into subcategories along different dimensions which will be presented in the subsections below and the categorization results will be tabled in Table 4.

3.3.2 Type of inference

From a logical point of view it is possible to distinguish a backward chaining based and a ground-based inference. The idea at the basis of the backward chaining based approach is that a trace will form the basis for probabilistic inference of the elemente to be deducted. On the other side, in the ground-based model one substitutes each free variables with constants or terms and then calls a solver. The connection between elements is formalized by the notion of *grounding* which indicates the operation of replacing the variables of a term/formula with constants or terms that do not contain other variables. Since it is clearly beneficial to keep the networks as simple as possible, it is thus important to avoid including any ground clauses that are not relevant.

In particular, [12], [20], [7], [22] and [19] are backward chaining based models in that the probabilities at the basis of the prediction made by these systems, found their roots in the various traces obtained performing a set of inference steps. On the contrary, [18], [14], [16], [8], [6], [11], [17], [9], [21], [13], [26] and [27] are ground-based models in that all of them learn through the parameters over the ground model and inference is based on possible groundings of the model. [23], [10] and [28] can be considered to use the ground-base approach, in that they use *Markov Logic Network* to perform inference, which is a ground-based model.

The backward chaining based approach is very flexible and expressive where human-readable rules can be directly extracted, but has limitations rooted in the high computational complexity. In particular, in the considered works, this can be found in exact inference and weak unification. Regarding exact inference, as multiple proofs for a query can be true in a possible world, we cannot calculate the success probability of a query based on the probabilities of the independent proofs and this, makes the system

less scalable to complex formulae. Weak unification instead, which is a soften of Prolog unification that allows to unify two different symbols s_1, s_2 by comparing their representations using a differentiable similarity function, could lead to a significant increment of proof paths, which can result in computational issues for larger programs.

On the other side ground-based approaches are in general less complex from a computational viewpoint and scale up better to complex rules, but do not allow human-readable rules to be extracted and do not perform well as the number of objects/entities or relationships grows.

3.3.3 Type of logic

Basically, there are four different kinds of logic: logic programming, propositional logic, relational logic and first-order logic. Logic programming is a programming paradigm based upon an extended first-order logic with e.g. higher order logic programming, constraints etc, [2]. Propositional logic a.k.a. *sentential logic* or *statement logic*, is the branch of logic that studies ways of joining and/or modifying entire propositions, statements or sentences in order to make them complex as well as to devies the logical relationships and properties from these methods of combining statements [3]. *Relational logic* is an alternative to propositional logic that solves inadequacy of the latter to capture the full meaning of some beliefs, for example suppose that Jack knows Jill, by means of propositional logic it is impossible to express the general fact in a way that allows to conclude that Jill knows Jack. Relational logic overcome this problem augmentig the language with two new linguistic features, *variables* and *identifiers* [4], this kind of logic can be considered as a limited first-order logic in that it lacks of function symbols. *Frist-order logic*, a.k.a. predicate logic, is a collection of formal systems used in several areas which uses quantified variables over non-logical objects and allows the usage of sentences that contain variables [25].

On the different above mentionioned logic definitions it is possible to assume that [12] and [22] belong to logic programming category in that they are examples of Prolog-based/like frameworks, S5 belong to this category too since a *Lifted Relational Neural Network* can also be considered as a logic program. As clearly stated by the authors [26] belong to propositional logic category in that they formally define their Semantic Loss by means of concepts in propositional logic. Being [7] and [19] based on Datalog, which is considered to use relational logic [1] and [6] has limitations that imposes that all arguments in the preidcates can only be variables or objects but not function symbols, it is possible to state that all of them belong to relational logic category. Finally, [18], [14], [16], [8], [11], [17], [9], [23], [10], [21], [13], [27] and [28]:

1. [18], [14], [9] and [27] use fuzzy logic to translate general first-order logic theory into a traning objective.
2. [16] and [13] use first-order logic to generate a random field.
3. [8] generates from first-order logic bottom lauses to be used as features in a truth-table.
4. [21] minimizes a global loss over first-order logics in the knowledge graph embedding.
5. [11], [17], [23], [10] and [28] use constraints imposed by some prior knowledge expressed as a set of first-order clauses to improve learning procedures.

The main limitation of propositional logic based approaches relies in the limited expressive power in that it is not possible to describe statements in terms of their properties, logical relationships or using quantifiers.

	Approach	Type of inference	Type of logic
LTN [18]	SRL	Grounding	First-order logic
SBR [14]	SRL	Grounding	First-order logic
DeepProbLog [12]	SRL	Backward chaining	Logic Programming
MLN [16]	SRL	Grounding	First-order logic
LRNN [20]	SRL	Backward chaining	Logic Programming
BCP [8]	SRL	Grounding	First-order logic
δILP [7]	SRL	Backward chaining	Logic Programming
NLM [6]	SRL	Grounding	Relational logic
KD + PR [11]	SRL	Grounding	First-order logic
NLProlog [22]	SRL	Backward chaining	Logic Programming
MF [17]	SLR	Grounding	First-order logic
KALE [9]	KGE	Grounding	First-order logic
KBC + MLN [23]	KGE	Grounding	First-order logic
DiffLog [19]	SRL	Backward chaining	Relational logic
pGAT [10]	KGE	Grounding	First-order logic
RE-E [21]	KGE	Grounding	First-order logic
RNM [13]	SRL	Grounding	First-order logic
SL [26]	SRL	Grounding	Propositional logic
iterE [27]	KGE	Grounding	First-order logic
expressGNN [28]	KGE	Grounding	First-order logic

Table 4: Categorization results.

Works exploiting first-order logic have a good degree of exepressivity in rule definition, but, basically, they transofrm such rules in the propositional form by means of the grounding mechanism, since it is the only way to make them suitable for a SAT solver.

Finally, logic programming is the most powerful and expressive kind of logic but also the most complex in that exploits manifold mechanisms like backtracking and unification which are algorithmic processes for finding all (or some) solutions and solving equations between symbolic expressions respectively.

4 Conclusion

Symbolic knowlege injection, as testified by the great works found in the literature, turned out to be an interesting research field which can endow systems with powerful human-like reasoning skills. We conducted our research in 3 different directions:

1. The extraction of different interpretation of word “injection”. About this, there are several methods of injecting background knowoledge in order to make systems capable of reasoining, in particular these interpretations are: (i) best satisfiability problem, (ii) template model, (iii) propositionalization, (iv) semantic loss, (v) knowledge distillation, (vi) joint embedding and (vii) matrix factiorization.
2. The isolation the forms that symbolic knowledge should take in order to be fully exploitable, indeed, at first, it is represented as logical rules but secondly translated in a format that fits better to the characteristic of the injection method used. As an example, considering the best satisfiability

problem interpretation, logical rules are translated into a set of logical constraints which forms the satisfiability problem.

3. The identification of a (non-exhaustive) set of dimensions to categorize the approaches by reasoning and looking for similarities about them. We were able to discover 2 different dimensions, the first one refers to the inference type which can be either ground-based or backward chaining based; the second ones refers to the type of logic used in the systems which can be, from the simplest to the most complex, propositional, relational, first-order and logic programming.

In the future we would like to deepen some other directions, like the limits of the injection and how much the model can exploit the injected knowledge, as well as integrate the categorization dimensions in order to obtain an exhaustive set of them.

A Legend

1. LTN as **Logic Tensor Network.**
2. SBR as **Semantic Based Regularization.**
3. MLN as **Markov Logic Network.**
4. LRNN as **Lifted Relational Neural Network.**
5. BCP as **Bottom Clause Propositionalization.**
6. NLM as **Neural Logic Machine.**
7. PR as **Posterior Regularization.**
8. KD as **Knowledge Distillation.**
9. MF as **Matrix Factorization.**
10. RE-E as **Rule-Enhanced Embeddings.**
11. RNM as **Relational Neural Machine.**
12. SL as **Semantic Loss.**

References

[1] *Datalog*. https://clojure.github.io/clojure-contrib/doc/datalog.html.

[2] *Logic Programming*. http://www.doc.ic.ac.uk/~cclw05/topics1/first.html.

[3] *Propositional Logic*. https://www.iep.utm.edu/prop-log/.

[4] *Relational Logic*. intrologic.stanford.edu/chapters/chapter_06.html.

[5] Kitchenham BA & Stuart Charters (2007): *Guidelines for performing Systematic Literature Reviews in Software Engineering* 2.

[6] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li & Denny Zhou (2019): *Neural Logic Machines*. *CoRR* abs/1904.11694. Available at http://arxiv.org/abs/1904.11694.

[7] Richard Evans & Edward Grefenstette (2017): *Learning Explanatory Rules from Noisy Data*. *CoRR* abs/1711.04574. Available at http://arxiv.org/abs/1711.04574.

[8] Manoel Franca, Gerson Zaverucha & Artur Garcez (2014): *Fast relational learning using bottom clause propositionalization with artificial neural networks*. *Machine Learning* 94, pp. 81–104, doi:10.1007/s10994-013-5392-1.

[9] Shu Guo, Quan Wang, Lihong Wang, Bin Wang & Li Guo (2016): *Jointly Embedding Knowledge Graphs and Logical Rules*. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Austin, Texas, pp. 192–202, doi:10.18653/v1/D16-1019. Available at https://www.aclweb.org/anthology/D16-1019.

[10] L Vivek Harsha Vardhan, Guo Jia & Stanley Kok (2020): *Probabilistic Logic Graph Attention Networks for Reasoning*. In: *Companion Proceedings of the Web Conference 2020*, WWW '20, Association for Computing Machinery, New York, NY, USA, p. 669–673, doi:10.1145/3366424.3391265.

[11] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy & Eric Xing (2016): *Harnessing Deep Neural Networks with Logic Rules*. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Berlin, Germany, pp. 2410–2420, doi:10.18653/v1/P16-1228. Available at https://www.aclweb.org/anthology/P16-1228.

[12] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester & Luc De Raedt (2018): *DeepProbLog: Neural Probabilistic Logic Programming*. *CoRR* abs/1805.10872. Available at http://arxiv.org/abs/1805.10872.

[13] Giuseppe Marra, Michelangelo Diligenti, Francesco Giannini, Marco Gori & Marco Maggini (2020): *Relational Neural Machines*.

[14] Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti & Marco Gori (2019): *LYRICS: a General Interface Layer to Integrate AI and Deep Learning*. *CoRR* abs/1903.07534. Available at http://arxiv.org/abs/1903.07534.

[15] Megha Mishra (2018): *REGULARIZATION: An important concept in Machine Learning*. https://towardsdatascience.com/regularization-an-important-concept-in-machine-learning-5891628907ea.

[16] Matthew Richardson & Pedro Domingos (2006): *Markov Logic Networks*. *Mach. Learn.* 62(1–2), p. 107–136, doi:10.1007/s10994-006-5833-1.

[17] Tim Rocktäschel, Sameer Singh & Sebastian Riedel (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*, Association for Computational Linguistics, Denver, Colorado, pp. 1119–1129, doi:10.3115/v1/N15-1118. Available at https://www.aclweb.org/anthology/N15-1118.

[18] Luciano Serafini, Ivan Donadello & Artur Garcez (2017): *Learning and reasoning in logic tensor networks: theory and application to semantic image interpretation*. pp. 125–130, doi:10.1145/3019612.3019642.

[19] Xujie Si, Mukund Raghothaman, Kihong Heo & Mayur Naik (2019): *Synthesizing Datalog Programs Using Numerical Relaxation*. *CoRR* abs/1906.00163. Available at http://arxiv.org/abs/1906.00163.

[20] Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezný & Ondrej Kuzelka (2015): *Lifted Relational Neural Networks*. *CoRR* abs/1508.05128. Available at http://arxiv.org/abs/1508.05128.

[21] Pengwei Wang, Dejing Dou, Fangzhao Wu, Nisansa de Silva & Lianwen Jin (2019): *Logic Rules Powered Knowledge Graph Embedding*. *CoRR* abs/1903.03772. Available at http://arxiv.org/abs/1903.03772.

[22] Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser & Tim Rocktäschel (2019): *NLProlog: Reasoning with Weak Unification for Question Answering in Natural Language*. *CoRR* abs/1906.06187. Available at http://arxiv.org/abs/1906.06187.

[23] Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun & Guanhua Tian (2015): *Large-Scale Knowledge Base Completion: Inferring via Grounding Network Sampling over Selected Instances*. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, Association for Computing Machinery, New York, NY, USA, p. 1331–1340, doi:10.1145/2806416.2806513. Available at https://doi.org/10.1145/2806416.2806513.

[24] Wikipedia contributors (2020): *Boolean satisfiability problem — Wikipedia, The Free Encyclopedia*. Available at https://en.wikipedia.org/w/index.php?title=Boolean_satisfiability_problem&oldid=966122596. [Online; accessed 20-July-2020].

[25] Wikipedia contributors (2020): *First-order logic — Wikipedia, The Free Encyclopedia*. Available at https://en.wikipedia.org/w/index.php?title=First-order_logic&oldid=968333186. [Online; accessed 20-July-2020].

[26] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang & Guy Van den Broeck (2017): *A Semantic Loss Function for Deep Learning with Symbolic Knowledge*. *CoRR* abs/1711.11157. Available at http://arxiv.org/abs/1711.11157.

[27] Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein & Huajun Chen (2019): *Iteratively Learning Embeddings and Rules for Knowledge Graph Reasoning*. *CoRR* abs/1903.08948. Available at http://arxiv.org/abs/1903.08948.

[28] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi & Le Song (2020): *Efficient Probabilistic Logic Reasoning with Graph Neural Networks*.