KlevR and DeepR : a benchmark for exploring deductive reasoning functionalities of variational autoencoders

Laurea Magistrale in Computer science and Engineering - Ingegneria Informatica

Author: Amedeo Pachera

Advisor: Prof. Emanuele Della Valle

Co-advisor: Prof. Riccardo Tommasini

Academic year: 2021-2022

1. Introduction

Approximate Reasoning (AR) aims at improving the efficiency of deductive artificial intelligence by trading inference correctness for performance (e.g., execution time). Prominent examples of approximate reasoning include, but are not limited to consistency checking with linear classifiers [6], relational learning with Machine Learning models [4], and Knowledge Graph completion using Deep Learning techniques [1]. To the best of our knowledge, none of the existing techniques was used for materialization, i.e., the reasoning task of computing all the entailed assertions given a knowledge base. A major obstacle is the lack of a benchmark for the task that enables the investigation of expressiveness and efficiency tradeoff. In this thesis work, we address this problem by *designing a benchmark for approximated materialization.* The benchmark, named *KlevR*, is based on an accepted deep learning dataset called CLEVR [2]. It consists of a knowledge graph that counts three OWL ontologies (one for each OWL profile [7]) that captures CLEVR's abstractions, scenes, and queries. We tested *KlevR* using a baseline model set named *DeepR*, which performs the approximate materialization task. In particular, DeepR is a set of deep learning models performing approximate materialization for each OWL profile using a tensor representation for the *KlevR*.

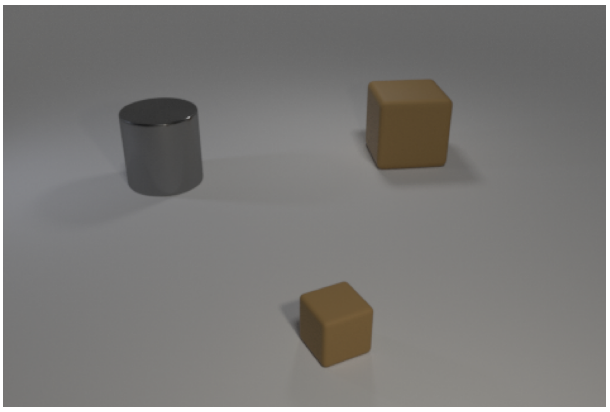


Figure 1: Example of a CLEVR scene.

2. CLEVR

CLEVR consists of a diagnostic dataset for analyzing VQA systems on different visual reasoning tasks [2]. A CLEVR **scene** is a collection of spatially related objects on the ground plane. Objects in CLEVR can are characterized by *{Shape, Size, Material, Color}*. Objects are spatially related, i.e., "left","right","behind","front". Figure 1 shows an example of a scene with three objects, a small brown cube that is behind and on the right of a small gray cylinder, which has a small brown cube on front-right. For each *scene*, there is a set of questions to answer, generated using different templates.

Each scene is associated with a **scene graph**, which is a direct graph with a set of nodes that represent objects and a set of edges that represents relationships among the objects. The nodes are labeled with the attributes of the object they represent. Scene graphs **are agnostic from the point of view**. The agnosticism is essential to have a standard and invariant description of each scene. On the other hand, a scene includes a point of view, which is randomly changed over the scenes.

3. Benchmark Task: KlevR

KlevR is a KB that represents CLEVR abstractions, scenes, and queries using Semantic Web Technologies.

In *KlevR*, each object belongs to the class *object*, while a complex hierarchy of classes captures the semantics of the attributes. Conveying the combinatorial nature of CLEVR attributes, we have organized the numerous classes in two nested hierarchies. In particular, the first hierarchy contains the classes that represent a type of CLEVR attribute, namely *ColoredObject*, *MaterialObject*, *ShapedObject* and *SizedObject*, their children instead are the intersection of the father with another attribute type (the combinations follow the alphabetical order of the attributes). The second hierarchy consists of attribute values. In particular *KlevR* contains a class for each attribute value and for each possible combination. For example, the attribute *Blue* is encoded with the class *BlueObject*, *Cube* with *CubeObject*. Combinations are made with both two values (e.g. *RedSphereObject*), three values (e.g. *GrayMetallicBigObject*) and four values (e.g. *GreenRubberSmallSphereObject*). The classes that describe an attribute value belong to both hierarchies. For example *BlueMetallicObject* is a subclass of both *BlueObject*, *MetallicObject* and *ColoredMaterialObject*.

In *KlevR*, the spatial relationships between the objects are sub-property of *hasNear* or *hasDirectlyNear*. Direct and indirect properties allow the definition of property chains, increasing the reasoning effort of the reasoner. Such design aspects are common to all the *KlevR* ontologies. However, according to existing standards, we implemented a version of KleveR's Tbox using the OWL profiles.

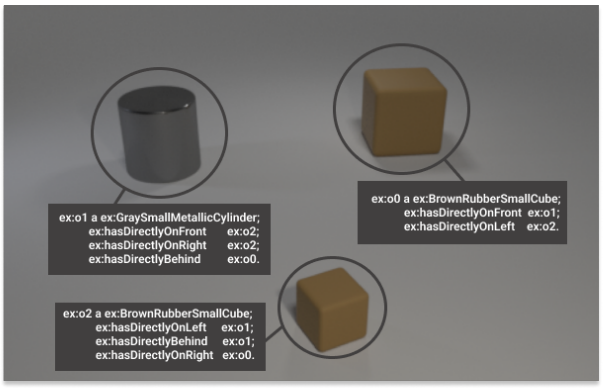
KlevR_RL uses OWL RL as representation language. It icludes existential quantification and *someValuesFrom*. Thus, we captured the semantic of the attribute values using classes. For example, the class *Red* stands for "a red Thing" and is subclass of *Color*. Their instances will be anonymous nodes, and are used in combination with four new properties whose semantic is "having that attribute", namely *hasColor*, *hasMaterial*, *hasShape* and *hasSize*. Each class of the first level of the hierarchy is restricted with an existential quantification. For example

BlueObject equivalentTo hasColor some Blue

defines the class of blue objects as the set of objects having blue color. *KlevR_RL* also supports transitive and symmetric properties, resulting in the most expressive profile.

KlevR_QL and **KlevR_EL** use the OWL QL and OWL EL profiles respectively. In particular, the QL profile supports all the RL expressions and axioms except for existential quantification and transitive property, while the EL profile does not support inverse and symmetric properties, but in addition, it supports the transitivity and disjunction. For this reason, the semantics of the attributes lies only in the class names in the hierarchy. Indeed, *KlevR_QL* and *KlevR_EL* were modeled starting from *KlevR_RL* and then removing the attribute-type classes (*Color*, *Material*, *Shape* and *Size*), their subclasses, the property *hasColor*, *hasMaterial*, *hasShape* and *hasSize* and the not supported axioms.

In populating the KBs, our goal was to fed the reasoner with individuals belonging to the lower level classes (e.g. *BlueMetallicSmallCubeObject*), forcing it to scale up the hierarchy using the *subclass* entailment rule. Moreover, we wanted the objects to be related only with their direct positional relationship to enforce the reasoner to infer about sub-properties and transitivity. With the dataset annotation, we have obtained an RDF graph for each scene, which we call *KlevR scene graph*. Figure 2 shows an example of a CLEVR scene annotated with a *KlevR scene graph*.



```
ex:o1 a ex:GraySmallMetallicCylinder;
        ex:hasDirectlyOnFront     ex:o2;
        ex:hasDirectlyOnRight     ex:o2;
        ex:hasDirectlyBehind      ex:o0.
```

```
ex:o0 a ex:BrownRubberSmallCube;
        ex:hasDirectlyOnFront  ex:o1;
        ex:hasDirectlyOnLeft   ex:o2.
```

```
ex:o2 a ex:BrownRubberSmallCube;
        ex:hasDirectlyOnLeft    ex:o1;
        ex:hasDirectlyBehind    ex:o1;
        ex:hasDirectlyOnRight   ex:o0.
```

Figure 2: Example of a KlevR scene graph.

To obtain the ground truth for the materialization task, we have proceeded with the materialization. The reasoner of our choice was HermiT[1].

4. Baseline solution: DeepR

With the materialization, we have obtained three datasets. In particular, for each OWL profile of *KlevR* we have a training and a testing set $\mathcal{D} = \{< x_i, t_i >\}$ where x_i is a *KlevR scene graph* and t_i is a *materialized KlevR scene graph*. The first step to test a baseline on the benchmark is the choice of the embedding model for the RDF graphs which is described in Subsection 4.1. To guide our choice, we elicited some requirements: (R1) : The semantic (Tbox) shall be taken into account in the vector representation. (R2) : The model shall take as input the entire ABox. (R3) : A unique mapping function between the OWL representation and the vector representation used as input an output of the model shall exist. The second step, reported in Subsection 4.2, consists of the Deep Learning model design. The generative nature of the materialization task (adding new assertions), has led us towards generative models and, in particular, Variational Autoencoders (VAE [3]).

4.1. Embedding Model

The simpler model that could respect our requirements is the SPO [5] , a tensor representation for triples in a KB, where an entry $\chi_{i,j,k} = 1$ if the triple $<s_i,p_j,o_k>$ exists. With SPO, we can encode every triple of the KB in a single tensor, making our model able to predict based on the entire available information. In particular, each tensor slice encodes a properties p_k, while its rows and column are unique indexes for subjects and objects in the scenes. Encoding all the scenes in this way, introduces a bias for the deep learning model, but it is essential to comply with R1. R2 is also respected since an SPO tensor embeds an entire KlevR scene graph. For what concern R3, the algorithm we have used to build the embedding keeps track of the indexes of the elements, building a mapping function between the RDF and the tensor representations.

4.2. VAE Design

In order to evaluate our benchmark for the approximation of the materialization task, we have designed a baseline VAE called **DeepR**. In particular, *DeepR* has three different architectures, one for each OWL profile of Klevr. Our approach in designing the architectures was not model based. In the research community in fact there is no other studies regarding VAEs for the reasoning approximation. Thus, we have decided to plan an hyper-parameters tuning to find the best architectures that maximize the models' accuracies on the data. In particular, we have tested different number of layers, filters per layer, learning rates and optimizers. The loss function of our choice is the binary cross-entropy, which ignore the existing trade-off between the reconstruction term and the regularization term of the standard VAE loss functions. In this way the whole model contributes to the approximation, getting rid of redundant or useless information in the encoding and adding new triples both in the encoding and decoding part. The choice of a simpler loss function leads to a simplified VAE, however, in Section 6 we will discuss an extension of the model that strictly follows the design of a VAE, encoding the input in the first part of the architecture.

5. Evaluation

The hyperparameter tuning selected a single-layer architecture for the QL profile, a four-layer architecture for the EL profile and a five-layer architecture for the RL profile. To evaluate the efficiency of the models' approximation in terms of time performances, we compare the computational time of *DeepR* in its different phases (scene parsing, embedding, training and testing) with the time performances of Her-

[1]http://www.hermit-reasoner.com

Profile	Parsing	Embedding	Training	Total	Testing	HermiT
DeepR_EL	4402.6	92.2	912505.2	917000.0	14340.5	211933.0
DeepR_QL	5027.3	4141.0	555266.1	564434.4	8835.7	536749.0
DeepR_RL	5563.6	6438.1	193370.0	205371.5	3210.1	4764269.0

Table 1: Computing time (in seconds) comparison between DeepR models and HermiT.

miT on the entire dataset. Table 1 reports the results of the performance evaluation. As we could expect, the training time for the *DeepR* models is larger than the average time that HermiT spends reasoning on a KG, except for the RL profile. Instead, the *DeepR* models outperform HermiT in the testing. To evaluate the quality of the approximation, we opted for standards Deep Learning metrics (i.e., accuracy, precision, and recall), giving them an interpretation from a reasoning point of view. Thanks to the SPO embedding, we evaluated the model's performances as if it was a classification task. In particular, TPs are valid triples predicted by the model, TNs are triples not asserted both by the model and the reasoner, FPs are incorrectly asserted triples by the model, and FNs are missing triples by the model. With this interpretation, *precision* is the percentage of valid triples correctly asserted with respect to the total number of assertions, while *recall* is the percentage of valid triples asserted with respect to the total number of expected assertions. We also compared the model's predictions to the target ABoxes removing the input triples, which allowed us to distinguish if a triple is asserted by the model or if it was already present in the ABox. Due to the sparse nature of the SPO embedding, we expected the models to push their weights towards zero during the training phase. To deal with the unbalanced classification problem, we evaluated the models using the *threshold-moving* technique, which consists of comparing the confidence of a model's predictions with different thresholds. Thus, we exploited the ROC curve to understand the tradeoff between the true-positive rate (TPR) and false-positive rate (FPR) for different thresholds and the ROC AUC to compare the two approaches (with and without the input triples). Given the ROC, the highest value of the Geometric Mean between TPR and the FPR denotes the best threshold. To evaluate the assertions made by the model, however, the main focus should fall on the asserted triples (the positives). For this reason, we also relied on the Precision-Recall (PR) curve to compare the precision against the recall on different thresholds. Given the PR, the highest value of F-measure denotes the best threshold. Figure 3 shows the ROCs derived from the experiments of *DeepR QL* on *KlevR*, Figure 5 the ROCs derived from *DeepR EL*, and Figure 7 the ROCs derived from *DeepR RL*. The blue curve refers to the standard predictions, while the green one is derived by removing the triples belonging to the input from the predictions and the targets. Despite the results of the efficiency evaluation, *DeepR* does not perform well for what concerns the approximation quality. The two ROCs of each models are close to each other, denoting a similar degree of separability in both the experiments (with and without the input), and the high values of the AUC suggest good overall performances. However, the steepness of the curves reveals a high true-positive rate and a low false-positive rate, caused by the sparse nature of the SPO embedding and the resulting tendency of the model to push the weights toward zero. Unlike standard Deep Learning experiments, the high value of the accuracy in our baseline does not prove the quality of the approximation. The high number of 0s in the embedding leads to an elevated number of true negatives which increases the accuracy. To better deal with the unbalanced problem, we also derived the PR curves for the experiments, which are reported in Figure 4 for *DeepR QL*, in Figure 6 for *DeepR EL*, and in Figure 8 for *DeepR RL*. Despite the high accuracy, the model performs a poor inference in terms of both correctly predicted triples, which is pointed by the low precisions, and the number of predicted triples, which is pointed by the low recalls. Moreover, the difference between the two PR curves highlights the real model's ability to infer new triples. Lower precisions in the experiments without the input triples mean fewer TPs and/or more FPs, which translates into fewer asserted triples and/or more wrong assertions. Therefore, the majority of the TPs are due to the triples already belonging to the ABoxes before the materialization. The best thresholds selected by the threshold-moving technique are very small, increasing the risk of false positives. Table 2 reports the metrics computed for the *DeepR* models, relative to their best threshold, for the experiments that do not consider the input triples. The results highlight how the nature of the embedding impacts the evaluation of the metrics. The values of accuracy for the materialization task with the SPO do not reflect the quality of the inference, which is instead explained by the precision and the recall.

6. Conclusion and future works

KlevR proves how a trivial model based on a VAE is not able to successfully perform the approximation of the materialization. This result manifests the importance of the benchmark for the task, which lies the foundation for future assessments. The quality of the benchmark lies on the scalability of the dataset and on the reasoning effort proved by the ground-truth given by HermiT. However, its implementation is not exempt from improvements regarding the portability and usability of the data. The poor results obtained from the baseline proposed by us proves the complexity of the task. In particular, the SPO embedding is not the best choice for encoding the Knowledge graphs. The sparse nature of SPO forces the network to very low thresholds, increasing the risk of false positives. Also, classic Deep Learning metrics can be misleading due to the high presence of true negatives. The accuracy of *DeepR*, in fact, is excellent on the test set, but it does not reflect the accuracy of the task objective, which instead resides in the true positives/negatives and the precision. For reasons of time, we did not have the opportunity to perform other complex experiments. An important extension that represents a future work consists of testing different embedding models on *KlevR* and verifying how different representations deals with our benchmark. To solve the problem of encoding the KGs, the design of an embedding model that takes into consideration the semantics of the ontology should be tackled. Moreover, other experiments could focus on the scalability property of *KlevR*. Future investigations include the design of a more efficient baseline model for the approximate materialization task with more complex VAE architectures.

Model	Accuracy	Precision	Recall	Threshold
DeepR EL	0.99957	0.08532	0.09276	0.105182
DeepR QL	0.99955	0.08532	0.09276	0.089300
DeepR RL	0.99930	0.06418	0.14723	0.029293

Table 2: Accuracy, precision, recall and threshold comparison of DeepR models.

References

[1] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1811–1818. AAAI Press, 2018.

[2] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016.

[3] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Found. Trends Mach. Learn.*, 12(4):307–392, 2019.

[4] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104(1):11–33, 2016.

[5] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 809–816. Omnipress, 2011.

[6] Heiko Paulheim and Heiner Stuckenschmidt. Fast approximate a-box consistency checking using machine learning. In *The 13th, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, volume 9678 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 2016.

[7] W3C. Owl 2 web ontology language profiles, 2012.

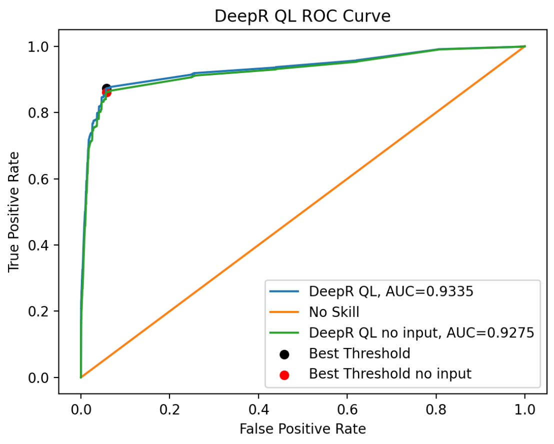


Figure 3: DeepR QL ROC curve.

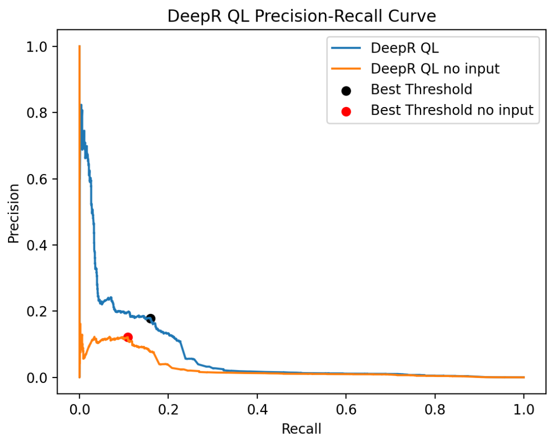


Figure 4: DeepR QL PR curve.

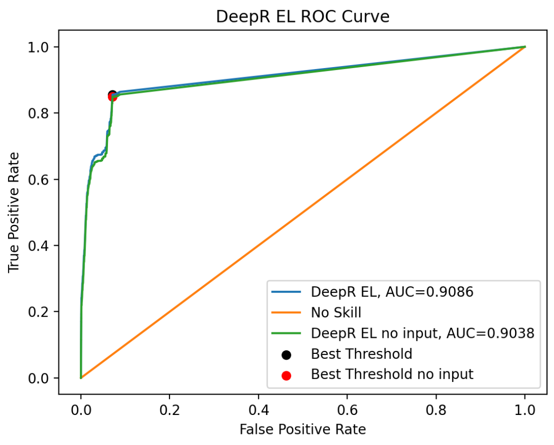


Figure 5: DeepR EL ROC curve.

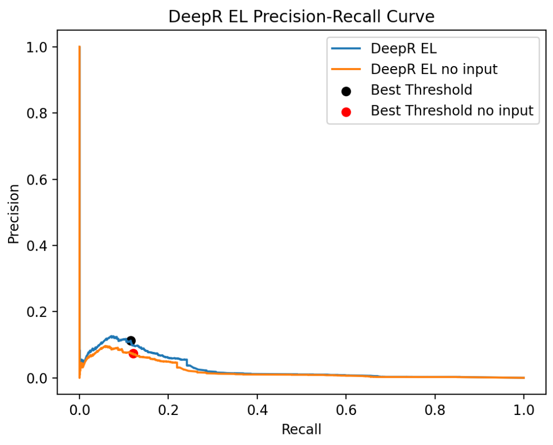


Figure 6: DeepR EL PR curve.

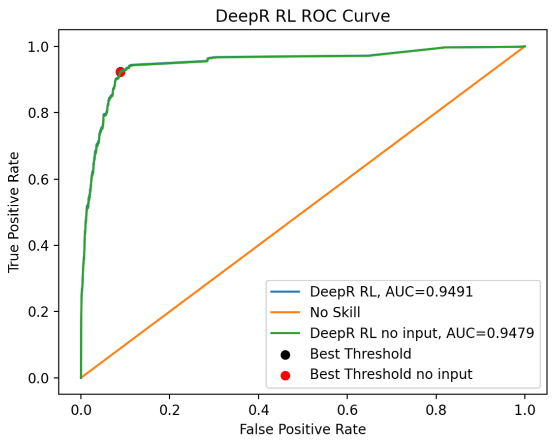


Figure 7: DeepR RL ROC curve.

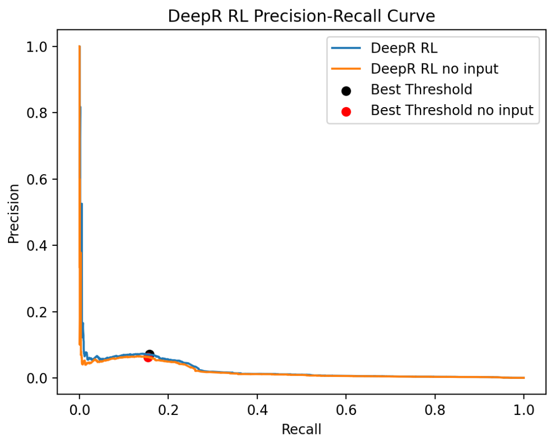


Figure 8: DeepR RL PR curve.