LoLa: Local and lazy inference on knowledge graphs

Anonymous Authors[1]

Abstract

Most state-of-the-art approaches on knowledge graph (KG) completion employ graph embeddings and sub-symbolic approaches. However, recent rule-based approaches have shown that symbolic methods can perform the same while being inherently interpretable. In this work, we present a framework for local and lazy inference on KGs using symbolic methods, LoLa. Our method focuses on query-related parts of the graph to generate crisp rules and use evidence in the local neighborhood of the query to answer it. It requires no training and can be combined with various rule learning and inference methods. Employing one such tool for local rule generation and inference within our framework, we greatly enhance the performance of the original tool and obtain state-of-the-art results in various benchmark datasets.

1. Introduction

A Knowledge Graph (KG) is a graphical representation of relational information between entities. Each relation is usually presented as a triple (s, r, o), where s and o are the subject and object entities, and r is the relation connecting them. Examples of such widely-used KGs in various domains are DBpedia (Auer et al., 2007), YAGO (Suchanek et al., 2007), NELL (Carlson et al., 2010), and Wikidata (Tanon et al., 2016). However, these massive KGs suffer from missing links for various reasons, such as their automatic generation procedure (Min et al., 2013; Dong et al., 2014).

To complete the missing links, different approaches focus on either utilizing external information regarding the data in the KG (e.g. similarity of nodes based on some textual description) or using the information already present in the KG for the same purpose. This task is known as link prediction or knowledge graph completion (Chen et al., 2020)[1] and we will be focusing on the latter approach.

Currently, the majority of the research for KG completion focuses on graph embedding procedures (Rossi et al., 2020) and other representation-learning approaches, such as graph neural networks (Arora, 2020). However, most such sub-symbolic methodologies are not transparent in the way they produce the predictions (Bianchi et al., 2020). On the other hand, symbolic approaches, such as inductive logic programming (Galárraga et al., 2013), capture patterns of the KGs using rules, which allow for decisions to be traced back and explained (Das et al., 2017). Recent symbolic approaches, such as AnyBURL (Meilicke et al., 2019b), have also been shown to achieve predictive performance that is on par with the sub-symbolic systems and are the focus of this work.

In this paper, we present a new framework for local and lazy inference, showcasing its capabilities using AnyBURL as a rule-generation and inference mechanism. Our framework is independent of the rule-generation/inference mechanism and AnyBURL has been chosen as it is fast and robust. The essence of the proposed framework is that it focuses on specific parts of the KG that contain expressive patterns and evidence to answer a specific query. The main intuition behind this approach is that we do not need all the information available in the KG to answer specific queries. Thus we can ignore the excess data to generate crisper rules efficiently and use appropriate facts as evidence. The main contributions of the proposed work are the following:

- We present a modular framework for local and lazy inference on KGs, LoLa. This framework requires *no training* and can be used as a wrapper for any tool that performs rule generation and inference.
- We showcase the effectiveness of our framework by improving the results of the original AnyBURL system and discuss the effect of the locality mechanism, especially when dealing with long-distance queries.
- We achieve state-of-the-art performance in current benchmark datasets for KG completion (WN18RR, YAGO3-10, and FB15k-237), as well as on some newly-proposed alternatives.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]In the context of adding missing facts in a KG, we will be using the terms "completion" and "inference" interchangeably throughout this work

The rest of the paper is structured as follows: Section 2 provides an overview of the related work. In Section 3 we introduce our framework in detail. Then we present our experimental results on the different datasets in Section 4, with discussion and qualitative comments. In the end, we provide conclusions and possible future work in Section 5.

2. Related Work

Representational Learning Approaches: As mentioned in the previous section, KG completion nowadays is usually based on learning representations for nodes and relations, either shallow or contextual. A recent and extensive survey on graph embedding methodologies is given in (Rossi et al., 2020), while a survey on graph neural networks (GNNs) used for link prediction can be found in (Arora, 2020). Some interesting methods extend these representation learning approaches with rules as well. For example, Ruge (Guo et al., 2018) learns rules from the KG, creates missing links according to these rules, and then utilizes them in the embedding generation procedure as new training examples with soft labels. For an overview of other such hybrid embedding approaches one could see (Wang et al., 2017). Regarding GNNs, the most interesting approaches (Zhang et al., 2020b; Qu & Tang, 2019; Qu et al., 2020) are based on Markov Logic Networks (Richardson & Domingos, 2006), which combine them with GNNs to increase efficiency in reasoning. An interesting insight that aligns with the intuition behind our own framework is noted in the SEAL method (Zhang & Chen, 2018) (and its more recent extension (Teru et al., 2020)). There, GNNs are used for extracting features to answer a given query and the importance of the local subgraphs around each query predicate is highlighted.

Rule-based Approaches: Symbolic approaches have a rich history in the context of KG completion. Inductive Logic Programming (Muggleton, 1991) has been used since the early days of KGs, while Statistical Relational Learning, in particular Markov Logic Networks (Richardson & Domingos, 2006) have been used more recently. Beyond these generic approaches, novel systems such as AMIE+ (Galárraga et al., 2015), MINERVA (Das et al., 2017), RuleN (Meilicke et al., 2018), and AnyBURL have been proposed and many of them achieve state-of-the-art performance on different benchmarks and application datasets. All of these systems can be incorporated into our framework for rule-generation and inference.

In (Liu et al., 2020), the authors compare most of the above and other sub-symbolic methodologies on the task of multi-hop reasoning in a biomedical KG. An interesting approach for KG completion is seen in NeuralLP (Yang et al., 2017), and its successor DRUM (Sadeghian et al., 2019), where differentiable models are utilized to generate the logical rules. Two other recent approaches that are worth mentioning are GPFL (Gu et al., 2020) and EARDict (Zhang et al., 2020a). Both focus on ending anchored rules (i.e. rules that contain a variable and a constant in their head) and especially EARDict provides theoretic results for KG reasoning and impressive state-of-the-art performance. Finally, an idea that is similar to our local framework for rule generation is presented in (Das et al., 2020). The authors sample different query-related paths in the graph to extract simple patterns and provide an answer to the query at hand. In our approach, we utilize subgraphs instead of paths and our framework is more generic as it is agnostic towards the underlying rule-generation and inference mechanisms used.

3. Methodology

Firstly, we will introduce some notation that we use in the paper and a summary of the AnyBURL system that will be used as the core rule generation and inference module. Then, we will describe the LoLa framework in detail.

3.1. Preliminaries

Notation: Let $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{F}\}$ denote a multi-relational KG where $\mathcal{E}$ denotes the set of *entities* (nodes), $\mathcal{R}$ the set of *relations/predicates* (relation types) and $\mathcal{F}$ the set of facts (triples) found in the KG. Each fact is in the form: $(h, r, t) \in \mathcal{F}$ (or equally $r(h, t)$) and $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. The h entity is usually referred to as the *start* or *head*, while t is the *end* or the *tail* of the fact. The known KG will be denoted as G_{Train} and a corresponding test set as G_{Test}, comprising facts that do not belong in G_{Train}. We will also denote a rule as: $h(A, B) \Leftarrow b_1(C, D) \wedge b_2(E, F) \wedge ... \wedge b_N(Y, Z)$. The head of the rule is $h(A, B)$ and its body of N conjuncts is $b_1(A, B)$ through $b_N(Y, Z)$. Each triple in the rule is also called an *atom* and it may contain a mix of variables and constants.

AnyBURL: In the context of the proposed LoLa framework, we utilize AnyBURL (Meilicke et al., 2019b; 2020) as a rule generation and inference tool. AnyBURL stands for Anytime Bottom-Up Rule Learning and focuses on efficiently learning logical rules from KGs inspired by bottom-up rule learning, such as Aleph (Srinivasan, 2001). Specifically, the system samples triples from the KG and subsequently selects paths that start or end on these triples. Then, it uses each path as the body of a prototype rule and the corresponding triple as the head of the rule. An example of this procedure is illustrated in Fig. 1, where a part of a KG is shown, with the sampled triple being *speaks(ed, d)*, and three differently-colored paths are sampled. These paths generate the three rules on the left of the figure. Next, AnyBURL generalizes the rules and calculates confidence scores for each one based on how many times the rules is satisfied in the KG.

$speaks(ed, d) \leftarrow born(ed, a)$
$speaks(ed, d) \leftarrow mar(ed, lisa), born(lisa, a)$
$speaks(ed, d) \leftarrow lives(ed, nl), lang(nl, d)$

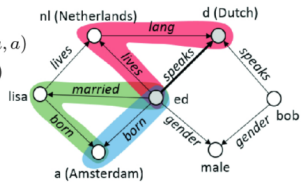

Figure 1. Example of the AnyBURL sampling procedure. We are interested in finding rules that can support the *speaks(ed,d)* fact. Three corresponding paths are sampled and colored accordingly, as shown in the right. These generate the three prototype rules on the left. The image is taken from (Meilicke et al., 2019a).

AnyBURL continues to generate rules in this manner, until a pre-set time limit is reached. The rules that are extracted in this way can be used to complete queries of the form $(h, r, ?)$ with the most probable entities, according to the confidence of the rules. In (Meilicke et al., 2020), the authors add reinforcement learning in the path sampling procedure, among other improvements, to enhance the performance of the system. An important note here is that AnyBURL starts by sampling the shortest paths first and then moves on to the longer ones when specific criteria are met. As such, the first rules that are considered have a body of only one atom (e.g. the first rule on the left of Fig. 1), next 2-atom rules are generated, then the 3-atom rules, etc.

AnyBURL has been shown to perform better or on-par with many state-of-the-art embedding methods in different datasets and with much shorter execution time (Meilicke et al., 2019b; Rossi et al., 2020). This fact and the "anytime" aspect of the methodology (i.e. the user specifies the time limit for the rule generation process), were the main reasons that we selected this system for the rule generation and inference procedure in the current work.

3.2. The LoLa framework

The key idea of the proposed framework is to answer queries one at a time, focusing on specific parts of the KG that contain information to answer each query. This local focus enhances the performance of both the rule generation and the inference procedures.

During rule generation, the *lazy* mechanism of action dictates the extraction of patterns only for the specific query relation r_q, instead of general rules to support the whole KG. Furthermore, only facts related to the query need to be considered, disregarding the rest of the KG. This leads to a much smaller number of rules that are also more relevant, leading to succinct models.

Then, during inference, the facts used as evidence to satisfy the rules are the ones located in the vicinity of the query edge (i.e. the close neighborhood around the entities in the query). Taking into account only *local facts* greatly reduces the possibility of a redundant or misleading rule to be satisfied, as opposed to using all the facts available in the KG. In summary, the LoLa framework does not train a general model to capture global patterns, but rather focuses on answering each query individually.

3.3. The LoLa inference process

The LoLa framework for answering a specific query $q : (h_q, r_q, t_q)$ is outlined in Algorithm 1[2]. In addition to the query at hand, the framework requires as input the known facts from G_{Train} and tools to generate the rules (*RuleTool*) and infer the outcome (*InferenceTool*). In the current implementation, AnyBURL acts as both the *RuleTool* and the *InferenceTool*.

Algorithm 1 LoLa

1: **Input:** Query triple $q : (h_q, r_q, t_q)$, G_{Train}, *RuleTool*, *InferenceTool*
2: **Output:** Result on specific query triple
3: $relatedEnts = similarEnts(G_{Train}, h_q)$
4: $relatedEnts \cup = similarEnts(G_{Train}, t_q)$
5: $relatedEnts \cup = (h_q, t_q)$
6: $rulesSubgraph = \emptyset$
7: **for** ent in $relatedEnts$ **do**
8: $\quad rulesSubgraph \cup = egoGraph(G_{Train}, ent)$
9: **end for**
10: $localRules = RuleTool(rulesSubgraph)$
11: $localEvidence = egoGraph(G_{Train}, h_q)$
12: $localEvidence \cup = egoGraph(G_{Train}, t_q)$
13: **RETURN** $InferenceTool(q, localEvidence, localRules)$

1. **Fetch similar entities** (lines 3-5)
Given the query (h_q, r_q, t_q) our methodology first retrieves the k most-similar entities for h_q and t_q separately. Entity (or node) similarity is calculated by the function $similarEnts$ in the algorithm. The specific function that is used in this work is based on co-expression of relations and is presented in more detail in section 3.4. Having such a node-similarity procedure in place, we simply fetch the k most-similar entities for both the head and tail of the query. Then we also add the h, t entities to the previously fetched set of similar entities. The result of this process is a set of $2 \times k + 2$ entities related to the query. These will act as focal points on the KG. Their neighborhood will be used to generate the rules for the query.

2. **Aggregate the rule-generation subgraph** (lines 6-9)
Having generated the query-related entities, we then

[2]The $\cup =$ symbol denotes the set union of the left-hand variable with the right-hand variable

move on to aggregate the facts involving each of these entities. The $egoGraph$ function simply generates the N-hop ego-graph around each entity (graph node). We repeat this process for all query-related entities merging their ego-graphs in a common graph, the $rulesSubgraph$. The $rulesSubgraph$ is typically a very small part of the whole G_{Train} and will be used to generate the rules that can answer the query at hand. Intuitively, this pruning process allows the *RuleTool* to extract patterns from facts related to the query at hand, disregarding the rest of the KG.

3. **Generate the query-specific rules** (line 10)
   Next, the *RuleTool* utilizes the $rulesSubgraph$ created in the previous step to generate rules. An important advantage of this query-based inference procedure is that we are only interested in answering a specific query involving predicate r_q. Thus, we can constraint the *RuleTool* to generate rules only about the predicate r_q.

   For example, going back to Fig. 1, where the test query was $(ed, speaks, d)$, we can focus the search on rules that only have the relation $speaks$ in their head. This has a number of advantages over the global rule-learning procedure. First, all resources are dedicated to finding patterns that answer the specific query predicate r_q. Second, focusing on the $rulesSubgraph$ ensures that the rules generated will be much more relevant to the query at hand. Third, the rule-set generated is much smaller, leading to faster inference.

4. **Generate the local evidence** (lines 11-12)
   Then, we move on to isolate the parts of the graph that will act as evidence for answering the query. This process is straightforward and we simply create the $localEvidence$ subgraph by merging the N-hop ego-graphs of h_q, t_q. This pruning process leaves a very small number of facts to be used as evidence for each query. Thus, this process lowers the probability of an undesired rule being satisfied, by minimizing the support group of the facts to the ones in the vicinity of the query.

5. **Inference** (line 13)
   Finally, using the assigned *InferenceTool*, which takes as input a collection of facts ($localEvidence$), a collection of rules ($localRules$), and the query (h_q, r_q, t_q), we generate the result. Once again, the framework is agnostic of the inference system and this abstraction allows for flexible modeling. For example, one could utilize a Markov Logic Network (Richardson & Domingos, 2006) as the *InferenceTool* and either perform MAP inference, viewing the task as triple classification, or perform probabilistic inference through MC-SAT (Poon & Domingos, 2006), or Lifted Belief propagation (Singla & Domingos, 2008). As such, the LoLa framework can utilize any inference methodology according to the needs of the application. In this work, we use AnyBURL as the *InferenceTool*. Given a query triple (h_q, r_q, t_q), AnyBURL creates two sub-completion tasks in the form of $r_q(h_q, ?)$ and $r_q(?, t_q)$ and provides top-k candidates for substituting the corresponding question marks. So, it generates top-k predictions along with confidence scores for the most probable heads and tails for the query.

3.4. Details of the framework

Similarity of entities: LoLa allows for any kind of node similarity to be used. All we need is a process that takes an entity in the graph and returns the k most-similar entities found. We propose a similarity based on the co-expression of relations. The idea is that "similar entities" will generally exhibit the same kind of relations. For example, entities that are CEOs of companies could exhibit relations like "$ceoOf$" or "$boardMember$" , while singers would more probably exhibit relations like "$sungAt$" or "$inBand$". Drawing from this insight, we opt for the *Jaccard/Tanimoto* coefficient (Tanimoto, 1968) of the set of relations that the entities express. We also take into account the position of the entities in a fact. This is because an entity exhibits different relations as a head and as a tail. As such, we calculate the similarity matrix between all entities when they exist in the head of the facts and once more for when they are in the tail. Using these similarity matrices, we simply fetch the k most-similar entities for the head and tail entity of each query.

Moreover, for a query (h_q, r_q, t_q), when fetching related entities (heads) to h_q (let's say e_h), we make sure that all retrieved entities are involved in a fact with r_q (i.e. there exists at least one fact (e_h, r_q, X) for all e_h). However, more complex pre-trained embeddings could be used in the similarity calculation as well. The parameter k directly affects the size of the aggregated $rulesSubgraph$ (i.e. having more related entities, leads to more ego-graphs and a larger subgraph in the end) and so we chose to keep it rather small, setting it to $k = 2$ for all tasks. However, it can be configured according to the scenario at hand.

Neighborhood size: We talked about keeping a small part of the KG around focal points when creating the $rulesSubgraph$ and $localEvidence$ regions. The bigger the N-hop neighborhood is, the larger these regions become. Intuitively, we need them to be just large enough to capture useful rules (in the $rulesSubgraph$) and contain the appropriate facts around the query (for the $localEvidence$). The current implementation with AnyBURL starts from short rules and extends them over time, as mentioned before. Alongside the fact that we only allow the tool to run for a

few seconds on each query, the resulting rules have very rarely a body with length ≥ 3. As such, we only care to sample the 2-hop neighborhood around each related entity (i.e. that can support the generation and satisfaction of 2-atom cyclic rules). However, this too can be changed according to the rule-generation methodology followed and thus we can impose a different language bias on the rules.

Multiple queries and efficiency: If we want to answer multiple queries, caching can be useful. We can pre-compute offline the similarity of the entities in the KG and also the N-hop ego-graphs of each entity and store them in memory. Then, during the call of the $similarEnts$ and $egoGraph$ functions in Alg. 1, we can fetch them from the memory store, decreasing the time needed for each query.

Complexity: Regarding the efficiency of the framework, we note that the time complexity is linear to the number of queries we want to perform. Specifically, it is $\mathcal{O}(N_q(T_{RG} + T_I) + c)$, where N_q is the number of queries, T_{RG} and T_I is the time the *RuleTool* and the *InferenceTool* need per query and c is the time needed for the remaining steps with $c \ll (T_{RG} + T_I)$ in principle. With T_{RG}, T_I and c being constants dependent on the tools used and the KG, the complexity is $\mathcal{O}(N_q)$. Currently, using the AnyBURL tool, we have set $T_{RG} = 5$ seconds and the average time per query (across all datasets) is ≈ 8 seconds. As such, very big test batches can be challenging to deal with. However, due to the laziness of the LoLa framework, it is straightforward to perform parallel or distributed processing, further reducing the complexity by a factor equal to the number of processors used.

4. Experiments

We performed experiments on three benchmark knowledge graphs used in link prediction literature: WN18RR (Dettmers et al., 2018), YAGO3-10 (Mahdisoltani et al., 2015), and FB15k-237 (Toutanova et al., 2015). We also experimented with other 2 newly proposed datasets. The first one, YAGO3-10-DR (Akrami et al., 2020), is a filtered version of YAGO3-10-DR with a near-duplicate relation removed, together with some redundant triples belonging to symmetric relations. The second one, CoDEx (Safavi & Koutra, 2020), is actually a set of 3, varying in size, KG-datasets constructed from Wikipedia and Wikidata. It is more diverse, with a wider scope, and provides a more difficult alternative to FB15k-237 for the link prediction task. We experiment with 2 out of the 3 datasets in CoDEx. The basic characteristics of these datasets can be seen in Table 1. Finally, we also performed a detailed comparison of the results of the LoLa framework against the original AnyBURL system, in order to study the impact of the various components of the framework.

Table 1. Characteristics of the evaluation datasets. $|\mathcal{E}|$ denotes the number of unique entities, $|\mathcal{R}|$ denotes the number of unique relation types and G_X presents the number of facts found in the corresponding set X.

DATASET	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	G_{Train}	G_{Eval}	G_{Test}
WN18RR	40,493	11	86,835	3,034	3,134
FB15K-237	14,541	237	272,115	17,535	20,046
YAGO3-10	123,182	37	1,079,040	5,000	5,000
YAGO3-10-DR	122,837	36	732,556	3,359	3,390
CODEX-S	2,034	42	32,888	1,827	1,828
CODEX-M	17,050	51	185,584	10,310	10,311

4.1. Evaluation Protocol

In our experiments. we have adopted the commonly-used *filtered* evaluation protocol introduced in (Bordes et al., 2013). According to this framework, given a query triple (h_q, r_q, t_q) one calculates the probability of all triples: (h_q, r_q, t') with $t' \in \mathcal{E}$, filtering out the triples found in the training, validation and test data, sorting them in descending probability order and reporting the rank of the wanted (h_q, r_q, t_q) triple. This process is repeated for predicting the head as well (i.e. by calculating all probabilities for (h', r_q, t_q) with $h' \in \mathcal{E}$). Then, one calculates the average of the commonly reported measures, such as Mean Rank (MR), Mean Reciprocal Rank (MRR), and Hits@K (K usually being $1, 3$ and 10), across the test set.

However, due to the local nature of LoLa the evidence provided contain only a few entities per query (i.e. the ones contained in the $localEvidence$ subgraph). Therefore, the calculation of the MR and MRR measures is not meaningful. Instead, we report here our results only for Hits@K, which provides a fair comparison to other methods. Regarding the hyper-parameters of our methodology, as explained in the sections above we have used $k = 2$ for the similar entities, $N = 2$ for the ego-graphs, and $T_{RG} = 5$ seconds for all datasets. We have also set the the parameters of AnyBURL to the values mentioned in (Meilicke et al., 2020), except for the threshold of the confidence of the rules that we change from 0.0001 to 0.01 (filtering out more low-confidence rules). Given the greatly reduced amount of facts used by LoLa, we expect the rules to be more robust and can afford a higher confidence threshold. Our own experiments were run on a MacBook Pro with Intel Core i7 Quad-Core @ 2.30GHz. We use 6 threads and reserve 8 GB RAM for our experiments.

4.2. Results on Datasets

The results on the benchmark datasets can be seen in Table 2. In the table, the first group of systems (first three lines) lists the state-of-the-art systems (to our knowledge) for each dataset and their respective scores (underlined). The second group of systems contains other competitive models from various families of models, such as tensor decom-

Table 2. Performance of state-of-the-art systems on WN18RR, YAGO3-10 and FB15K-237. Hits@K scores are in percentage. The best score is in **bold** and the previous state-of-the-art is underlined.

Model	WN18RR			YAGO3-10			FB15k-237		
	$H@1$	$H@3$	$H@10$	$H@1$	$H@3$	$H@10$	$H@1$	$H@3$	$H@10$
MLMLM (Clouatre et al., 2020)	43.9	54.2	61.1	-	-	-	18.7	28.2	40.3
RotH (Chami et al., 2020)	44.9	51.4	58.6	50.3	62.1	71.2	25.6	39.0	54.1
GAAT (Wang et al., 2019)	42.4	52.5	60.4	-	-	-	51.2	57.2	65.0
RotatE (Sun et al., 2019)	42.8	49.2	57.1	40.2	55.0	67.0	24.1	37.5	53.5
Complex-N3 (Lacroix et al., 2018)	43.5	49.5	57.2	49.8	60.9	70.1	26.4	39.2	54.7
KBAT (Nathani et al., 2019)	36.1	48.3	58.1	-	-	-	46.0	54.0	62.6
A2N (Bansal et al., 2019)	42.0	46.0	51.0	-	-	-	23.2	34.8	48.6
DRUM (Sadeghian et al., 2019)	42.5	51.3	58.6	-	-	-	25.5	37.8	51.6
EARDict (Zhang et al., 2020a)	**87.0**	**93.5**	**96.6**	**64.2**	**72.5**	**79.1**	33.0	45.9	60.2
AnyBURL (Meilicke et al., 2020)	45.7	-	57.7	49.4	-	69.1	27.3	-	52.2
LoLa	**54.6**	**62.1**	**67.3**	**56.5**	**67.0**	**72.8**	**67.1**	**80.23**	**84.6**

position Complex-N3 (Lacroix et al., 2018), translational models RotatE (Sun et al., 2019), and graph attention networks KBAT (Nathani et al., 2019). Special mention should be given to EARDict (Zhang et al., 2020a), presented in a very recent pre-print which reports impressive results in most datasets, especially in WN18RR. Given that the corresponding paper is not published yet, we present the results of both EARDict and the published state-of-the-art systems. Finally, we present the results for the original AnyBURL model and the LoLa framework incorporating AnyBURL. For all models except LoLa, we report the measures as presented originally by the authors.

First, we observe that our model outperforms the standing state-of-the-art system on all benchmark datasets, without taking into account the newly-proposed EARDict. Specifically, in the FB15k-237 dataset, where LoLa also outperforms the EARDict model, we obtain an absolute increase of more than 15 percentage points on all measures. Additionally, the LoLa framework greatly enhances the results of the basic AnyBURL model. For all measures and datasets, but more emphatically in FB15k-237, the performance of the system is improved significantly. These improved results are attained using the same rule-generation and inference procedure of AnyBURL, but localizing lazily, according to the LoLa framework.

Table 3 presents the results on the more difficult datasets. We report results directly from the corresponding papers (Akrami et al., 2020; Safavi & Koutra, 2020) for commonly used systems. We also experimented with EARDict, due to its exceptional performance on the benchmarks, using the basic model (which performed the best across all datasets according to the authors). Also, AnyBURL is used with the default parameters for 1000 seconds runtime. We keep the settings of LoLa unchanged.

The first observation is that the scores are much lower across the table. For instance, comparing the performance of the models in Table 2 for YAGO3-10 to the results of the same models in YAGO3-10-DR, all systems perform worse. Next, we observe that LoLa outperforms competing models in these datasets in almost all cases. It is also important to note that symbolic methodologies, such as AnyBURL and EARDict, with their default parameters perform on-par with embedding methodologies, which have undergone hyperparameter tuning for these results (Safavi & Koutra, 2020). Moreover, it is worth noting the positive effect that LoLa has on the AnyBURL system, even in cases where the vanilla system is not doing relatively well, such as YAGO3-10-DR. We can thus confirm that the same rule generation and inference method can perform drastically better when used under the local and lazy framework, achieving state-of-the-art results.

4.3. Analysis of LoLa

In this section, we provide insights on the role of different components of the LoLa framework, by comparing it against the basic AnyBURL system on WN18RR.

4.3.1. Ablation Study

In Table 4 we can see the results of a small ablation study on WN18RR. We want to emphasize the importance of the two key components of LoLa: i) the *Lazy* aspect that generates dedicated rules per query, and ii) the *Local* evidence used per query. The first column reports the average Miss@10 on the test set [3]. The second column expresses the percent-

[3]There is a slight misalignment with the corresponding Hits@10 as reported in Table 1 for WN18RR. This is because $\approx 6\%$ of the test dataset has either a head or tail not seen before in G_{Train}, which makes correct prediction impossible. These test samples are included in the results of Table 2, in order for the comparison to other systems to be fair, but they are excluded during the ablation study for clearer results.

Table 3. Performance on newly-proposed datasets. Hits@K scores are in percentage. The best score is in **bold**.

Model	CoDEx-S			CoDEx-M			YAGO3-10-DR		
	$H@1$	$H@3$	$H@10$	$H@1$	$H@3$	$H@10$	$H@1$	$H@3$	$H@10$
TransE (Bordes et al., 2013)	21.9	42.2	63.4	22.3	33.6	45.4	11.7	-	32.3
Complex-N3 (Lacroix et al., 2018)	37.2	50.4	64.6	26.2	37.0	47.6	14.3	-	31.5
ConvE (Dettmers et al., 2018)	34.3	49.3	63.5	23.9	35.5	46.4	14.7	-	31.5
Tucker (Balazevic et al., 2019)	33.9	49.8	63.8	25.9	36.0	45.8	14.8	-	32.0
EARDict (Zhang et al., 2020a)	32.2	46.0	65.5	32.2	46.0	**65.6**	29.7	37.4	46.9
AnyBURL (Meilicke et al., 2020)	33.8	47.2	62.1	24.7	34.4	45.1	16.1	23.1	32.8
LoLa	**37.3**	**57.9**	**67.3**	**38.4**	**48.9**	59.9	**43.1**	**58.1**	**71.9**

Table 4. Ablation study on WN18RR. The first column corresponds to the total Miss@10. The second column presents the number of no prediction responses given by the system. The third corresponds to actually incorrect predictions. All numbers are in percentages.

MODEL	MISS@10	NO PRED.	INCORRECT PRED.
LOLA	25.3	6.7	18.6
- W/O LOCAL EVIDENCE	26.2	6.8	19.4
- W/O LOCAL RULES	34.4	5.2	29.2
ANYBURL	38.2	3.4	34.8

age of cases for which no prediction was made, typically because no appropriate pattern was generated. The third column corresponds to the cases where predictions were made but the correct answer was not found in the top-10 entities predicted. The figures in the second and third columns sum up to the total misses in the first column.

The first row of Table 4 presents the results of LoLa, while in the second the local evidence mechanism is disabled. This means that at inference time the whole G_{Train} is used as evidence. We observe a small deterioration in the results, mainly due to the increase in incorrect predictions. When more irrelevant facts are used as evidence, we get more ground atoms, increasing the chance that an inaccurate (with respect to the test query) rule is satisfied.

In the second ablation experiment, we re-evaluate the framework, but this time we remove the local rule-generation mechanism (third row of Table 4). That is, for inference we now use the global rules generated by the vanilla AnyBURL after running for 1000 seconds, but we keep the local evidence mechanism. In this case, we observe a large drop in performance ($\approx$ 10 percentage points) due to the increase of wrong predictions. This confirms the superiority of locally-generated rules in inferring missing links, as compared to rules generated to support the whole KG. At the same time, we observe a slight improvement in the percentage of no-prediction cases, which is due to the fact that the global rules are many more than the average number of local rules per query ($60,000$ global rules vs 92 local rules on average). This provides higher coverage over the test queries, at the expense of much lower precision, as can be seen in the wrong predictions column.

The final evaluation is done by removing both the local evidence and the local rule-generation mechanism, essentially degenerating back to the basic AnyBURL system (fourth row). Compared to row 3, we observe a drop in performance, due to even more incorrect predictions made, caused by the lack of local-only evidence to satisfy the appropriate rules. Here again, global rules and global evidence lead to an increased coverage and a small drop in no predictions rate, at the expense of much lower precision.

To summarize, the main findings of the ablation study are the following:

- The main improvement factor of the LoLa framework is the lazy mechanism that leads to dedicated rules for a specific test predicate. This greatly reduces the rate of incorrect predictions, because the rules generated are specialized to a specific query, capturing patterns from neighborhoods that are similar to the query neighborhood.
- The use of local evidence also helps, as even when used with global rules we observe improved results. This happens because the local facts around a test query are very focused and limited in number, thus indirectly weeding out inaccurate rules and only satisfying useful local patterns.
- There is a trade-off between *precision* and *coverage* with the global or local procedures. The basic AnyBURL method has the highest coverage and therefore the lowest rate of non-predictions. However, this comes at a high cost in precision, leading to the worst overall performance. On the other hand, the LoLa framework achieves a much lower incorrect predictions rate, in the expense of a higher no predictions rate. However, the improvement on precision greatly out-values the increase of no prediction cases, striking a better balance between precision and coverage and leading to the best overall performance.

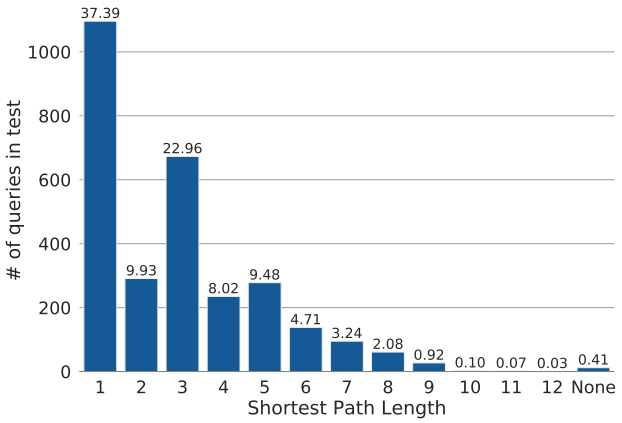

Figure 2. Distribution of the shortest path length between entities in the queries of the WN18RR testset. The number on top of each bar corresponds to the percentage of the test set. "None" corresponds to disconnected pairs.

4.3.2. Effect of Shortest Path Length in test queries

Delving deeper into the cases where AnyBURL misses the correct answer, while LoLa finds it, we assessed the importance of the distance between the entities in the test query. Specifically, we are interested in the Shortest Path Length (SPL) between the head and tail of a query, as measured on G_{Train}. In WN18RR[4] the SPL distribution of the test queries is shown in Fig. 2. As we can see, in most of the queries ($\approx$ 70%) the tail is at most 3 hops away from the head, while longer pairs are much rarer. There is even a small number of pairs for which no such path exists, meaning the graph is disconnected ("None" value in Fig. 2). Taking into account the language bias of the rules generated by AnyBURL, we expect it won't easily cope with long-distance pairs. Especially when one of the 3 type of rules it can generate are cyclic path rules; rules of the form: $r_q(h_q, t_q) \Leftarrow r_1(h_q, x_1) \wedge r_2(x_1, x_2) \wedge ...r_N(x_{N-1}, t_q)$ where a complete path is formed in the body of the rule between the head and the tail. One such example is the 3rd rule on the left of Fig. 1, which corresponds to the red-highlighted cyclic path on the same figure.

The intuition about the difficulty with long pairs is verified by the results in Fig. 3. This figure presents the Hits@10 of both systems across different SPL levels. As we can see, both systems perform their best when dealing with pairs that are connected with few hops (low SPL). AnyBURL suffers a quick drop in performance up to SPL$=7$, when the Hits@10 becomes 0. On the other hand, LoLa retains a higher level of Hits@10 across almost all SPL levels and performs much better on the longer-distance pairs where the SPL is ≥ 5. However, this result is not due to LoLa creating cyclic rules of such body length. After all, it inherits the same language bias from AnyBURL. Despite this fact, the *locality* mechanism allows the rule-generation tool to allocate all resources on finding patterns that directly apply to the query at hand.

[4]We chose WN18RR to study this effect because in YAGO3-10 and FB15k-237 the maximum SPL is 5 and the effect is less apparent.

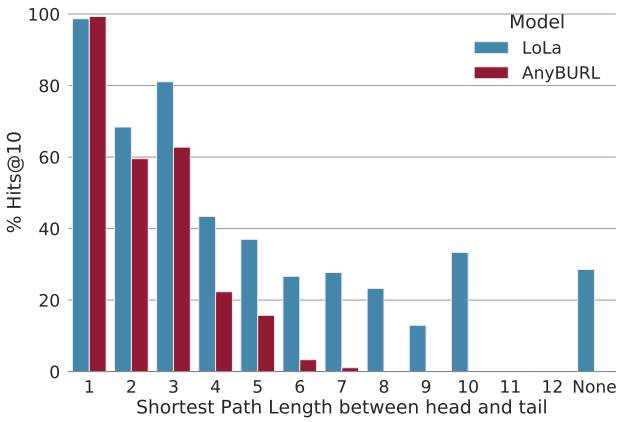

Figure 3. Average Hits@10 calculated over WN18RR test queries, grouped by their shortest path length. "None" corresponds to disconnected pairs.

5. Conclusions

In this work, we propose a lazy and local framework for knowledge graph inference (LoLa). LoLa is a modular framework that incorporates rule-generation and inference tools to provide predictions on test queries in a lazy fashion, focusing on specific parts of the KG each time. It requires no pre-training and can be readily used as a wrapper around most existing tools. We showcased, using AnyBURL at its core, that it outperforms the state-of-the-art systems on most benchmark datasets, as well as in some more challenging ones. Moreover, we analyzed the importance of the local and lazy components in the framework. In particular, we provided empirical results on the increased precision of the framework, alongside a discussion on how this is achieved.

Next, we aim at validating the impact of LoLa using other symbolic tools similar to AnyBURL. Moreover, we will seek to incorporate methods such as Markov Logic Networks and PSL (Bach et al., 2017) that can support probabilistic inference. We would also like to examine the effect of local inference in sparse-data scenarios, where data-intensive methods struggle. Another line of research would be to incorporate more complex node-similarity measures and also try to incorporate sub-symbolic approaches in the framework. Also, it would be interesting to extensively study the effect of the SPL in test queries across multiple datasets and evaluate embedding methodologies when considering SPL levels. Finally, we aim at assessing the scalability of the method in parallel settings, using large query sets.

References

Akrami, F., Saeef, M. S., Zhang, Q., Hu, W., and Li, C. Realistic re-evaluation of knowledge graph completion methods: An experimental study, 2020.

Arora, S. A survey on graph neural networks for knowledge graph completion. *arXiv preprint arXiv:2007.12374*, 2020.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. Dbpedia: A nucleus for a web of open data. In Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudré-Mauroux, P. (eds.), *The Semantic Web*, pp. 722–735, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-76298-0.

Bach, S. H., Broecheler, M., Huang, B., and Getoor, L. Hinge-loss markov random fields and probabilistic soft logic. *The Journal of Machine Learning Research*, 18(1): 3846–3912, 2017.

Balazevic, I., Allen, C., and Hospedales, T. Tucker: Tensor factorization for knowledge graph completion. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Bansal, T., Juan, D.-C., Ravi, S., and McCallum, A. A2n: attending to neighbors for knowledge graph inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4387–4392, 2019.

Bianchi, F., Rossiello, G., Costabello, L., Palmonari, M., and Minervini, P. Knowledge graph embeddings and explainable ai. *arXiv preprint arXiv:2004.14843*, 2020.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26:2787–2795, 2013.

Carlson, A., Betteridge, J., Wang, R. C., Hruschka, Jr., E. R., and Mitchell, T. M. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pp. 101–110, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6.

Chami, I., Wolf, A., Juan, D.-C., Sala, F., Ravi, S., and Ré, C. Low-dimensional hyperbolic knowledge graph embeddings. *arXiv preprint arXiv:2005.00545*, 2020.

Chen, Z., Wang, Y., Zhao, B., Cheng, J., Zhao, X., and Duan, Z. Knowledge graph completion: A review. *IEEE Access*, 8:192435–192456, 2020.

Clouatre, L., Trempe, P., Zouaq, A., and Chandar, S. Mlmlm: Link prediction with mean likelihood masked language model. *arXiv preprint arXiv:2009.07058*, 2020.

Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., and McCallum, A. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*, 2017.

Das, R., Godbole, A., Dhuliawala, S., Zaheer, M., and McCallum, A. A simple approach to case-based reasoning in knowledge bases. *arXiv preprint arXiv:2006.14198*, 2020.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 601–610, 2014.

Galárraga, L., Teflioudi, C., Hose, K., and Suchanek, F. M. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 24(6):707–730, 2015.

Galárraga, L. A., Teflioudi, C., Hose, K., and Suchanek, F. 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, 2013.

Gu, Y., Guan, Y., and Missier, P. Towards learning instantiated logical rules from knowledge graphs. *arXiv preprint arXiv:2003.06071*, 2020.

Guo, S., Wang, Q., Wang, L., Wang, B., and Guo, L. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Lacroix, T., Usunier, N., and Obozinski, G. Canonical tensor decomposition for knowledge base completion. *arXiv preprint arXiv:1806.07297*, 2018.

Liu, Y., Hildebrandt, M., Joblin, M., Ringsquandl, M., Raissouni, R., and Tresp, V. Neural multi-hop reasoning with logical rules on biomedical knowledge graphs, 2020.

Mahdisoltani, F., Biega, J., and Suchanek, F. M. YAGO3: A knowledge base from multilingual wikipedias. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org, 2015.

Meilicke, C., Fink, M., Wang, Y., Ruffinelli, D., Gemulla, R., and Stuckenschmidt, H. Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In *International Semantic Web Conference*, pp. 3–20. Springer, 2018.

Meilicke, C., Chekol, M. W., Ruffinelli, D., and Stuckenschmidt, H. An introduction to anyburl. In Benzmüller, C. and Stuckenschmidt, H. (eds.), *KI 2019: Advances in Artificial Intelligence*, pp. 244–248, Cham, 2019a. Springer International Publishing. ISBN 978-3-030-30179-8.

Meilicke, C., Chekol, M. W., Ruffinelli, D., and Stuckenschmidt, H. Anytime bottom-up rule learning for knowledge graph completion. In *IJCAI*, pp. 3137–3143, 2019b.

Meilicke, C., Chekol, M. W., Fink, M., and Stuckenschmidt, H. Reinforced anytime bottom up rule learning for knowledge graph completion, 2020.

Min, B., Grishman, R., Wan, L., Wang, C., and Gondek, D. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 777–782, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Muggleton, S. Inductive logic programming. *New generation computing*, 8(4):295–318, 1991.

Nathani, D., Chauhan, J., Sharma, C., and Kaul, M. Learning attention-based embeddings for relation prediction in knowledge graphs. *arXiv preprint arXiv:1906.01195*, 2019.

Poon, H. and Domingos, P. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, volume 6, pp. 458–463, 2006.

Qu, M. and Tang, J. Probabilistic logic neural networks for reasoning, 2019.

Qu, M., Bengio, Y., and Tang, J. Gmnn: Graph markov neural networks, 2020.

Richardson, M. and Domingos, P. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

Rossi, A., Firmani, D., Matinata, A., Merialdo, P., and Barbosa, D. Knowledge graph embedding for link prediction: A comparative analysis. *arXiv preprint arXiv:2002.00819*, 2020.

Sadeghian, A., Armandpour, M., Ding, P., and Wang, D. Z. Drum: End-to-end differentiable rule mining on knowledge graphs. In *Advances in Neural Information Processing Systems*, pp. 15347–15357, 2019.

Safavi, T. and Koutra, D. Codex: A comprehensive knowledge graph completion benchmark, 2020.

Singla, P. and Domingos, P. M. Lifted first-order belief propagation. In *AAAI*, volume 8, pp. 1094–1099, 2008.

Srinivasan, A. The aleph manual, 2001.

Suchanek, F. M., Kasneci, G., and Weikum, G. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pp. 697–706, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7.

Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

Tanimoto, T. An elementary mathematical theory of classification and prediction, ibm report (november, 1958), cited in: G. salton, automatic information organization and retrieval, 1968.

Tanon, T. P., Vrandečić, D., Schaffert, S., Steiner, T., and Pintscher, L. From freebase to wikidata: The great migration. In *World Wide Web Conference*, 2016.

Teru, K., Denis, E., and Hamilton, W. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pp. 9448–9457. PMLR, 2020.

Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1499–1509, 2015.

Wang, Q., Mao, Z., Wang, B., and Guo, L. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

Wang, R., Li, B., Hu, S., Du, W., and Zhang, M. Knowledge graph embedding via graph attenuated attention networks. *IEEE Access*, 8:5212–5224, 2019.

Yang, F., Yang, Z., and Cohen, W. W. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pp. 2319–2328, 2017.

Zhang, C., Katsis, Y., Vazquez-Baeza, Y., Bartko, A., Kim, H.-C., and Hsu, C.-N. Theoretical knowledge graph reasoning via ending anchored rules, 2020a.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 5165–5175, 2018.

Zhang, Y., Chen, X., Yang, Y., Ramamurthy, A., Li, B., Qi, Y., and Song, L. Efficient probabilistic logic reasoning with graph neural networks, 2020b.