On the Aggregation of Rules for Knowledge Graph Completion

Patrick Betz [1] **Stefan Lüdtke** [2] **Christian Meilicke** [1] **Heiner Stuckenschmidt** [1]

Abstract

Rule learning approaches for knowledge graph completion are efficient, interpretable and competitive to purely neural models. The rule aggregation problem is concerned with finding one plausibility score for a candidate fact which was simultaneously predicted by multiple rules. Although the problem is ubiquitous, as data-driven rule learning can result in noisy and large rule sets, it is underrepresented in the literature and its theoretical foundations have not been studied before in this context. In this work, we demonstrate that existing aggregation approaches can be expressed as marginal inference operations over the predicting rules. In particular, we show that the common Max-aggregation strategy, which scores candidates based on the rule with the highest confidence, has a probabilistic interpretation. Finally, we propose an efficient and overlooked baseline which combines the previous strategies and is competitive to computationally more expensive approaches.

1. Introduction

A knowledge graph (KG) is a collection of *relation(subject, object)* facts which can be used to compactly describe certain domains. KGs can be utilized for various downstream applications such as drug repurposing (Liu et al., 2021) or visual relationship detection (Baier et al., 2017). Most of the real-world KGs are incomplete, which means that absent facts are not necessarily false. The problem of knowledge graph completion (KGC) aims to derive the missing facts by using the information in the existing graph (Ruffinelli et al., 2020; Rossi et al., 2021). The proposed model classes in the literature are data-driven, e.g., a model might learn the regularity that people which appear in movies tend to be actors and can use it to make new predictions. Although the dominating paradigm in the literature lies on models based on latent representation, a KG is symbolic by its nature.

Symbolic machine learning approaches for KGC employ rule mining techniques and represent the KG with the raw predicates which makes them inherently interpretable. In regard to predictive performance they are shown to be competitive to latent based approaches (Rossi et al., 2021) and can achieve state-of-the-art results on large graphs (Meilicke et al., 2023). To perform KGC with a symbolic approach, a previously learned set of rules has to be applied to the KG to derive plausibility scores for unseen target facts. Whenever multiple rules predict a candidate fact, the question arises of how to aggregate individual rules, as demonstrated in the following running example.

Example 1.1. *Consider the following clauses or rules.*

c_1 $[0.64]$: *wf(X,Y)* $\leftarrow$ *internAt(X,Y)*

c_2 $[0.44]$: *wf(X,Y)* $\leftarrow$ *studentAt(X,A), locIn(A,B), locIn(Y,B)*

c_3 $[0.41]$: *wf(X,Y)* $\leftarrow$ *studentAt(X,A), cooperatesWith(A,Y)*

Here wf represents the relation worksFor and locIn represents locatedIn. The numbers in brackets denote rule confidences, i.e., the proportion of correct predictions on a training KG. The first and third rule are quite intuitive. The second rule expresses that a person might work for a company if that company is located at the same place where this person went to university. Now assume that all three rules predict Anna to work for Google. The rule aggregation problem is concerned with finding a final score derived from the three confidence values. The aggregation will also reflect if, e.g., Anna is more likely to work at Google than a person for which only the first two rules made the prediction.

While combining logical reasoning and probabilistic uncertainty is a fundamental aspect of statistical relational learning (Muggleton et al., 1996; Kersting & De Raedt, 2001; Richardson & Domingos, 2006), the aggregation problem is often not expressed explicitly. Additionally, these approaches perform model theoretic entailment, which is too expensive in our settings, as KGs can consist of a large number of facts with millions of learned rules. Similarly, in the field of association rule mining, rule quality is often estimated for individual rules independently without considering the problem of aggregation (Galárraga et al., 2013; Chen et al., 2016; Ortona et al., 2018; Fan et al., 2022).

The predictive quality of a mined rule set depends to a large

[1]University of Mannheim, Germany [2]University of Rostock, Germany. Correspondence to: Patrick Betz <patrick@informatik.uni-mannheim.de>.

extend on the aggregation decision and surprisingly there exists a theoretical and empirical gap in the recent KGC literature between techniques to learn rules and their successful application. To the best of our knowledge, there only exist two recent works which are primarily concerned with the aggregation problem for KGC (Ott et al., 2021; Betz et al., 2022b). While they improve upon simple strategies, the approaches are computationally expensive and theoretical foundations are not discussed.

The goal of this work is to close this gap and to inspire new research in this direction. We aim to achieve this by developing the formal foundations of the problem and by empirically analysing the practicality of existing approaches. We present a probabilistic model in which the aggregation reduces to performing marginal inference over a joint distribution of the rules when rule marginals are approximated with confidences (Section 4.1 and 4.3). With this formulation we are able to show that the common Max-aggregation strategy can be recovered from the model when the correlation matrix of the rules is set to the upper Fréchet-Hoeffding bound for the correlation of random variables (Section 4.4). We then search for the simplest and most efficient way to combine the assumptions made by common aggregation strategies. This leads to an efficient baseline, Noisy-or top-h, which is competitive when taking into account the performance-runtime trade-off (Section 5). Moreover, our experiments show that the choice of the aggregation function has significant performance impacts and therefore it deserves more attention in the context of rule-based KGC.

2. Related Work

While data-driven rule learning approaches for KGC are often evaluated in comparison to embedding models, the focus of this work is rule aggregation and we therefore refer to the recent literature for an overview to latent-based KGC (Rossi et al., 2021).

Rule mining approaches learn datalog rules from a KG. In the context of **association rule mining**, AMIE (Galárraga et al., 2013) and the respective improved versions AMIE+ (Galárraga et al., 2015) and AMIE3 (Lajus et al., 2020) show how to mine rules when data is incomplete. AnyBURL (Meilicke et al., 2019) is the successor of RuleN (Meilicke et al., 2018). It is shown to be competitive to neural approaches (Rossi et al., 2021; Meilicke et al., 2023) and it can be utilized to explain predictions made by embedding models (Betz et al., 2022a). Other approaches are tailored towards large graphs (Fan et al., 2022; Chen et al., 2016) or to learn negative rules (Ortona et al., 2018). There also exist attempts to improve rule quality by providing more advanced confidence computations (Galárraga et al., 2013; Pellissier Tanon et al., 2017; Zupanc & Davis, 2018). The rule quality is evaluated by calculating the precision of the individual rules independent from the remaining rules on a gold standard KG. For the resulting metrics, the aggregation problem is irrelevant. In this work we regard rule quality from the viewpoint of the predictions made by the rules, which also allows comparisons to other model classes.

Related branches of work combine latent and symbolic models in hybrid approaches (Guo et al., 2016; 2018; García-Durán & Niepert, 2018; Wu et al., 2022; Meilicke et al., 2021). Moreover, some work propose **differentiable rule learning** i.e., learning rules by solving a smooth optimization problem (Yang et al., 2017; Sadeghian et al., 2019). Rule mining and the aggregation are arguably coalesced in one forward pass of a neural module. It has been shown, nevertheless, that the rules extracted from the models might not derive the same facts as the models themselves and achieve a lower predictive performance (Tena Cucala et al., 2022). Therefore, they might benefit from encapsulating rule learning and the aggregation. A step in this direction is made by RNNlogic (Qu et al., 2021), in which a neural rule generator and a reasoning predictor operate independently. The predictive performance of the resulting model, when not augmented with embeddings, lacks, however, in regard to purely symbolic models.

The combination of logic and uncertainty has a rich history in the **statistical relational learning** literature. For instance, Stochastic Logic Programs (Muggleton et al., 1996; Sato & Kameya, 1997) and Bayesian Logic Programs (BLP) (Kersting & De Raedt, 2001) augment inductive logic programming (Muggleton & De Raedt, 1994) with probability semantics. Rules are represented as conditional probabilities and a joint probability distribution is modelled over the least Herbrand base of the logic program. Here, the aggregation problem becomes explicit. In particular, when multiple conditionals have the same effect variable, they are collapsed into one by the use of a *combining rule*. Nevertheless, this heuristic is applied on top of the formal framework whereas in this work we model the problem directly. A difficulty for BLPs is that the probability distribution is only well defined when the underlying graph does not contain cycles which is quite unlikely in the context of KGC when millions of rules are learned. Markov Logic Networks (MLNs) (Richardson & Domingos, 2006) are proposed to overcome the cycle problem as well as the requirement to define the *ad hoc* combining rule. MLNs subsume many of the approaches from the statistical learning literature. Each possible ground fact is associated with a binary random variable and every possible grounding of every rule with a weight and a binary feature. The aggregation of clauses is performed implicit for MLNs and can not be modelled easily. We show an example regarding MLNs in the appendix of this work.

The focus of this work are settings where model theoretic

entailment is not feasible. For instance, an MLN would need to define $15k \cdot 237$ random variables on the dataset FB15k-237 (Toutanova & Chen, 2015) and a feature for every possible rule grounding with a ruleset size of 5 million. Even if we would just calculate the immediate predictions of the rules on this dataset, including storing some indices for further processing, this would already take more than 600GB of memory. A similar note can be made for neural theorem proofing, where the forward-chaining algorithm is relaxed to a smooth differentiable function (Evans & Grefenstette, 2018; Rocktäschel & Riedel, 2017; Minervini et al., 2020a;b). To the best of our knowledge, these approaches have not shown yet to scale to datasets of the size used in our experiments. This also holds for ProbLog (De Raedt et al., 2007) which combines probabilistic inference with model theoretic entailment and has the strongest resemblance to our approach. We discuss the details in Section 4.4 and in the appendix of this work.

The rule aggregation problem is discussed explicitly by SAFRAN (Ott et al., 2021) where a clustering of the rules is learned and by Betz et al. (2022b) who represent rules with embeddings. These works show improvements in regard to simple strategies but they do not consider a fundamental treatment of the problem and the models are inefficient to use, which will be demonstrated in the experimental section.

3. Background

3.1. Knowledge Graph Completion

A KG $\mathcal{G}$ is a set of *relation(subject, object)* triples or facts with $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{P} \times \mathcal{E}$ where $\mathcal{E}$ denotes a set of entities and $\mathcal{P}$ a set of binary predicates which we term relations. KGC is concerned with finding unknown facts, given an input or training KG $\mathcal{G}$. In this work, we focus on the mostly used evaluation protocols which are defined by ranking based evaluation metrics. The derivations of this work are, however, independent of the evaluation protocol as long as scalar scores for candidate predictions are required.

The common practice is to split the graph into disjoint training, validation, and testing sets. After the training or mining phase a model is evaluated by proposing answers to queries formed from the facts in the test set. For each of these evaluation facts a head query and a tail query are formed. For example, from $worksFor(Anna, Google)$ the queries $worksFor(Anna, ?)$ and $worksFor(?, Google)$ are formed, where $worksFor$ is a relation and $Anna$ and $Google$ are entities. A model has to propose candidate facts for the tail query, e.g., $worksFor(Anna, e_1)$ and candidate facts for the head query $worksFor(e_2, Google)$ for multiple $e_1, e_2 \in \mathcal{E}$. Each candidate fact is assigned with a score such that for each direction a ranking of answers can be formed. The metrics usually are presented with their filtered versions, e.g., if $e_2 \neq Anna$ but $worksFor(e_2, Google)$ exists in one of the data splits, then it is removed from from the ranking of the current query to not penalize the model when it correctly ranks true answers on top positions. Performance is measured by the ranking position of the respective true candidate $worksFor(Anna, Google)$ in both directions where the mean reciprocal rank (MRR) and Hits@X being the most common evaluation metrics. The definitions of the metrics can be found in the appendix.

3.2. Rules and Application

We let a $c \in \tilde{\mathcal{C}}$ denote a logical clause, which we will term rule throughout the work, where $\tilde{\mathcal{C}}$ is a collection of clauses. The c will later be indexed and represented by separate random variables. The rules that we consider in this work are of the form as given in the running example. They are composed of variables and relations and they additionally can contain entities as shown in the following example.

$$speaks(X, English) \leftarrow livesIn(X, London)$$

We call $speaks(X, English)$ the head of the rule and $livesIn(X, London)$ the body of the rule. The rules and the KG can be described with a subset of Prolog, where entities are constants, relations are predicates, rules are clauses, and the facts of the KG are ground atoms where we do not consider negation. We will use the rule learners AnyBURL (Meilicke et al., 2019) and AMIE3 (Lajus et al., 2020) in our experimental section and we refer to the respective works for further details, nevertheless, the descriptions and derivations in this work are independent of the particular syntax.

We define a substitution to be the expression obtained when replacing the variables of the rules with entities from $\mathcal{E}$. For instance, for the first rule from the running example with $(X{=}Anna, Y{=}Google)$ we obtain the substitution $worksFor(Anna, Google) \leftarrow internsAt(Anna, Google)$. A detailed formalization is suppressed here for brevity.

Rule application refers to predicting previously unseen facts given a set of rules and the input or training KG. We can describe it compactly with the recently introduced concept of one-step-entailment (Betz et al., 2022a). Let $\tilde{\mathcal{C}}$ be a set of rules and $\mathcal{G}$ a KG.

Definition 3.1 (One-step entailment $\models_1$)**.** *The fact t is **one-step entailed** by $\tilde{\mathcal{C}} \cup \mathcal{G}$, written as $\tilde{\mathcal{C}} \cup \mathcal{G} \models_1 t$, iff there is a rule in $\tilde{\mathcal{C}}$ for which a substitution exists such that the resulting body facts are in $\mathcal{G}$ and the head is equal to t.*

Clearly, one-step entailment is weaker but more efficient than model theoretic entailment. As mentioned before, we focus on settings where general entailment is not feasible. One-step-entailment implies entailment but not vice versa.[1]

[1]Note that $\models_1$ is different to $\bar{k}$-entailment which limits the

In the context of KGC often the less formal notion of an individual rule predicting a candidate is used which we can now describe precisely.[2]

Definition 3.2 (Prediction)**.** *A rule $c \in \tilde{\mathcal{C}}$ **predicts** a fact t iff it individually one-step entails t, i.e., iff $\{c\} \cup \mathcal{G} \models_1 t$.*

For simplicity, we will write $c \models_1 t$ instead of $\{c\} \cup \mathcal{G} \models_1 t$, where from the context the reference to the facts $\mathcal{G}$ will be clear. The section concludes with an example.

Example 3.3 (cont.)**.** *Let e_d, e_u, and e_g be entities in $\mathcal{E}$. Let $t = wf(e_d, e_g)$ and assume that*

$$\mathcal{G} = \left\{ \begin{array}{l} cooperatesWith(e_g, e_u) \\ internAt(e_d, e_g) \\ studentAt(e_d, e_u) \end{array} \right\}.$$

Consider the three rules from the running example. Then the joint set of rules and every pairwise set of rules one-step entail t while only the first and the third rule predict t.

3.3. Rule Aggregation

For the remainder of the work we assume that $\tilde{\mathcal{C}}$ is a given rulset that has been learned from the training graph $\mathcal{G}$. Furthermore, for a target triple $t \notin \mathcal{G}$ we let $\mathcal{C}_t(\mathcal{G})$ denote the set of rules that predicted t with respect to the KG $\mathcal{G}$. For performing KGC under any evaluation protocol a model has to assign plausibility scores to candidate facts. For rule-based KGC this requires the introduction of two additional concepts, rule confidences and aggregation strategies.

3.3.1. Confidences

Rule confidences originate from the context of association rule mining and we will now assume that each rule in $\tilde{\mathcal{C}}$ is assigned with a confidence which can be calculated as follows.

$$conf(c) = \frac{\left|\{t' \mid c \models_1 t' \wedge t' \in \mathcal{G}\}\right|}{\left|\{t' \mid c \models_1 t'\}\right|} \tag{1}$$

Equation (1) is the vanilla confidence definition described in many works (e.g., Galárraga et al., (2013)). The confidence divides the number of all true predictions a rule makes by the number of all predictions of the rule. Intuitively, we could interpret this as the probability that the rule is true, which will be discussed in later sections.

3.3.2. Aggregation Strategies

In practical scenarios it rarely occurs that a candidate fact is predicted by only one rule, i.e., then $|\mathcal{C}_t(\mathcal{G})| > 1$. The rule aggregation problem, also termed joint prediction (Galárraga et al., 2015), is concerned with defining a function that maps the confidences of the rules that predicted the candidate to a real valued score.

Note that the number of rules that predict a candidate fact simultaneously can be large, as mentioned before, such that rules are to some extend redundant. For instance, if the second rule from the running example predicts *Anna* to work for *Google*, the question arises whether the third rule provides additional evidence for this prediction. The rules make the prediction for seemingly similar reasons, as it is more likely for an university and a company to cooperate when they are located in the same location. In the following the two most common aggregation strategies are defined.

Definition 3.4 (Max-Aggregation)**.** *The Max-Aggregation score s^M is calculated according to the rule with the highest confidence from the rules that predicted the candidate, $s^M(t) = \max\{conf(c) \mid c \in \mathcal{C}_t(\mathcal{G})\}$.*

Max-aggregation was first used in the context of KGC by Galárraga et al. (2015) and it was later adapted to *Max+ aggregation* (Meilicke et al., 2019) which allows for tie handling. When the two predicting rules with the highest confidences for two candidates are identical the candidates are compared according to the rules with the second highest confidence which is continued until the candidates can be discriminated.

Definition 3.5 (Noisy-or aggregation)**.** *The Noisy-or score s^{NO} is calculated as the noisy-or product over the predicting rules, $s^{NO}(t) = 1 - \prod_{c \in \mathcal{C}_t(\mathcal{G})}(1 - conf(c))$.*

The Noisy-or product originates from Bayesian networks where it is used to express independent causes (Pearl, 1988) and it was proposed by Galárraga et al. (2015) for KGC.

Example 3.6 (cont)**.** *Let us assume that $Anna$ is predicted by all rules from the starting example to work for $Google$, while $Lisa$ is predicted by only the second and third rule to work for $Google$. The Max-aggregation and Noisy-or scores for $Anna$ are 0.64 and 0.88, respectively. For $Lisa$ they are 0.44 and 0.67.*

While the aggregation functions have the purpose of merging the various confidences into a final score, this value also should be meaningful in the sense that a higher value for one prediction should mean it is more likely than another prediction.

4. Probabilistic and Efficient Rule Aggregation

In the following section we present the notation for the probabilistic representation, subsequently we introduce the inference model and show how the introduced rule aggregation functions can be recovered from the framework when making certain dependency assumptions. Finally, we will

number of constants used in entailment (Kuzelka et al., 2018).

[2] A formalization with the immediate consequence operator in the logic programming context is likewise possible.

present an efficient baseline, that combines these assumptions.

4.1. Representation

First, we enumerate the rules in $\tilde{\mathcal{C}}$ with an index set $\tilde{I} = \{1, ..., N\}$ such that $c_i \in \tilde{\mathcal{C}}$ for $i \in \tilde{I}$. Each rule c_i is represented by a binary random variable $\tilde{R}_i$ which is also indexed by $\tilde{I}$ and has realisations $\tilde{r}_i \in \{0, 1\}$. We let $\tilde{\mathbf{R}}$ denote the random vector representing all rules and likewise $\tilde{\mathbf{r}} = (\tilde{r}_i)_{i \in \tilde{I}} \in \{0, 1\}^N$ is the vector of realisations. For brevity we write $p(\tilde{\mathbf{r}})$ for $p(\tilde{\mathbf{R}}{=}\tilde{\mathbf{r}})$, that is, the probability that $\tilde{\mathbf{R}}$ takes value $\tilde{\mathbf{r}}$.

For the rule aggregation problem the set of rules $\mathcal{C}_t(\mathcal{G}) \subseteq \tilde{\mathcal{C}}$ that predict a target fact t based on $\mathcal{G}$ are of particular relevance. Therefore, similar as above $\mathcal{C}_t(\mathcal{G})$ is enumerated by $I = \{1, ..., k\}$ and the random vector $\mathbf{R}$ with realisations $\mathbf{r} = (r_j)_{j \in I} \in \{0, 1\}^k$ represents the rules that predict the target. Note that $\mathbf{R}$ represents a subset of all the rules and this depends on t, however, to not clutter notation we not write this explicit and the reference to t will be clear from the context.

Moreover, we write p_j or p_i for the probability that a rule is true, i.e., for the marginals $p(R_j{=}1)$ or $p(\tilde{R}_i{=}1)$. We assume that index sets are ordered according to the marginals, e.g., $p_m \geq p_n$ when $m \leq n$ with m, n being indices. Facts t are likewise represented as binary variables, here we overload notation for brevity and write $p(t)$ for the probability of a query triple to be true. For an observed triple $t \in \mathcal{G}$ we set $p(t) = 1$.

4.2. Dealing with Uncertainty

To incorporate uncertainty into the prediction of new facts we take the following approach. If we are certain that a rule is true, then we deduce that a prediction it makes must be also true. We can model this for all the learned rules with a conditional distribution that conditions on the truth values of the rules and the data.

$$p(t|\tilde{\mathbf{r}}, \mathcal{G}) = \begin{cases} 1, \text{ if } L(\tilde{\mathbf{r}}) \models_1 t \\ 0, \text{ else,} \end{cases} \tag{2}$$

Here, L is a simple mapping that collects all rule objects in $\tilde{\mathcal{C}}$ whose realisation are one in $\tilde{\mathbf{r}}$ and takes the union with $\mathcal{G}$, i.e.,

$$L^{\mathcal{G}}_{\tilde{I}} : \tilde{\mathbf{r}} \mapsto L^{\mathcal{G}}_{\tilde{I}}(\tilde{\mathbf{r}}) = \{c_i \mid \tilde{r}_i = 1 \textit{ and } i \in \hat{I}\} \cup \mathcal{G}. \tag{3}$$

We drop, as shown in equation (2), the reference to the index set $\tilde{I}$ and $\mathcal{G}$ from L for readability. Clearly, if the rules would not be associated with uncertainty evaluating equation (2) would boil down to performing rule application in regard to the correct rules. However, the truth values of the rules cannot be observed from the data.

We have, on the other hand, an estimate that statistically quantifies the uncertainty of the rules, the defined rule confidences. A confidence may serve as an approximation for the marginal probability that the respective rule is true, i.e., $p(\tilde{R}_i{=}1)$. However, we have to acknowledge that it is only the marginal $\sum p(\tilde{R}_i = 1, \tilde{\mathbf{r}}_{-i})$, which sums over all realisations of the remaining rules, where $\tilde{\mathbf{r}}_{-i}$ is the vector of realisations with $\tilde{r}_i$ dropped.

The last paragraph makes the difference to the viewpoint of association rule mining explicit. In fact, we assume that $p(\tilde{R}_i{=}1)$ is potentially influenced by an underlying joint distribution. For instance, the confidence of the rule c_2 of the running example might be influenced by the confidence of c_3 through the second term in the sum $p(\tilde{R}_2{=}1) = p(\tilde{R}_2{=}1, \tilde{R}_3{=}0) + p(\tilde{R}_2{=}1, \tilde{R}_3{=}1)$. Therefore, for fact prediction associated with uncertainty we have to take into account the joint distribution over the rules which will be discussed in the next section.

4.3. Inference for Target Facts

We want to calculate the probability that an unknown target fact $t \notin \mathcal{G}$ is true, given the known triples, i.e., we seek to compute $p(t|\mathcal{G})$. However, we cannot observe the truth values $\tilde{\mathbf{r}}$ of the rules from the data and we therefore choose a standard approach regarding such settings, i.e., we marginalize over all possible rule realisations,

$$p(t|\mathcal{G}) = \sum_{\tilde{\mathbf{r}} \in \{0,1\}^N} p(t|\tilde{\mathbf{r}}, \mathcal{G}) p(\tilde{\mathbf{r}}|\mathcal{G}). \tag{4}$$

Where we set $p(t|\tilde{\mathbf{r}}, \mathcal{G})$ to equation (2). We can simply calculate $p(t|\tilde{\mathbf{r}}, \mathcal{G})$ by collecting all rules that are one in $\tilde{\mathbf{r}}$ and subsequently evaluate if one of these rules predicts the target, i.e., performing rule application. The distribution $p(\tilde{\mathbf{r}}|\mathcal{G})$ seems to be more problematic. It defines the joint distribution over all N rules, given the data, including the rules that did not predict t. Rule aggregation, however, was defined with only the k rules that predicted a candidate. We will argue in the following proposition that under one-step entailment for calculating $p(t|\mathcal{G})$ it is indeed sufficient also under the probabilistic model to exclusively take into account the rules $\mathbf{R}$ with realisations $\mathbf{r}$ that predicted t.

Proposition 4.1. *Under a one-step entailment regime, i.e., using equation* (2) *for* $p(t|\tilde{\mathbf{r}}, \mathcal{G})$*, and a global distribution* $p(\tilde{\mathbf{r}}|\mathcal{G})$ *we have that*

$$p(t|\mathcal{G}) = \sum_{\mathbf{r} \in \{0,1\}^k} p(t|\mathbf{r}, \mathcal{G}) p(\mathbf{r}|\mathcal{G}). \tag{5}$$

The proof is in the appendix. Instead of using the global distribution we can focus directly on performing marginal inference $p(\mathbf{r}|\mathcal{G})$ with respect to the rules that predicted t. Although marginal inference can equally be expensive,

the complexity can be reduced if the joint distribution is specified accordingly and if some parameters of the joint are known such as the individual rule marginals. Additionally, it might even be beneficial to model $p(\mathbf{r}|\mathcal{G})$ directly.

Note that Proposition (4.1) would not hold if we would consider general model theoretic entailment. Finally, by the definition of equation (2) and one-step entailment it is easy to see that the query probability is the probability that at least one rule from $\mathbf{R}$ is true.

Proposition 4.2. *For the query probability it holds that*

$$p(t|\mathcal{G}) = p\big(\sum_{j\in I} R_j \geq 1 \mid \mathcal{G}\big). \tag{6}$$

Proof. We write out $p(t|\mathbf{r},\mathcal{G})$ in equation (5) and then drop the one term that is zero. The proposition follows from the definition of one-step-entailment as $L(\mathbf{r})$ one-step entails the target if at least one component of $\mathbf{r}$ is one. That means the probabilities of all realisations where at least one rule is true are summed up. □

We will henceforth refer to calculating $p(t|\mathcal{G})$ under the previous derivations when mentioning the inference model and we conclude the section with an example.

Example 4.3 (cont)**.** *Lisa is predicted by the two rules c_2 and c_3 to work for Google. Assuming that we know the joint distribution over all rules, we can calculate the probability that Lisa works for Google by querying the joint distribution for the probability that at least one of c_2 and c_3 is true.*

4.4. Recovering Aggregation Functions

We will demonstrate in this section that the inference model leads to the different aggregation strategies depending on the assumed joint distribution when marginals are approximated with the rule confidences. Therefore we assume for the following derivations $p(\tilde{R}_i{=}1) = \mathit{conf}(c_i)$ for $i \in \tilde{I}$.

4.4.1. Probabilistic Max-Aggregation

Max-aggregation was introduced in the literature as a computational heuristic (Galárraga et al., 2015), it was further described as accounting for strong rule dependencies without providing a detailed treatment (Meilicke et al., 2019), or it was even described with assuming fact independence (Svatoš et al., 2020). We will now introduce the Fréchet-Hoeffding bound which will help us to achieve a formal derivation. It limits the possible association, expressed as correlation, of two random variables (Joe, 1997). Let p_i and p_j be the marginal probabilities for two Bernoulli variables, then it holds for the correlation ρ_{ij} that $\rho_{ij} \leq U(i,j)$ where

$$U(i,j) = \min\left\{\left(\frac{p_i(1-p_j)}{p_j(1-p_i)}\right)^{1/2}, \left(\frac{p_j(1-p_i)}{p_i(1-p_j)}\right)^{1/2}\right\}. \tag{7}$$

Example 4.4 (cont)**.** *Let $p_1 = 0.64$ and $p_2 = 0.44$ then $U(1,2) \approx 0.66$. Whereas for $p_3 = 0.41$, $U(2,3) \approx 0.94$.*

While the configuration of the marginals in Example 4.4 allows for complex dependencies in regard to the joint distribution, they are not compatible with complete dependence as this would require unit correlation. Interestingly, equation (7) suffices to specify a joint distribution $p(\tilde{\mathbf{r}}|\mathcal{G})$ such that the inference model from Section 4.3 performs Max-aggregation.

Theorem 4.5. *If for the correlation matrix $\Omega \in [-1,1]^{(N,N)}$ with entries ρ_{ij} for all i,j it holds that $\rho_{ij} = U(i,j)$ then a unique distribution for $p(\tilde{\mathbf{r}}|\mathcal{G})$ is induced such that $p(t|\mathcal{G}) = s^M(t)$.*

We will show the proof for the case where $k = 2$ rules predicted the candidate here briefly and the general case can be found in the appendix. Let $p_{\bar{i}} = 1 - p_i$ and let, e.g., $p_{\bar{i}j} = p(R_i{=}0, R_j{=}1|\mathcal{G})$ and likewise for the remaining realisations. Further note for the correlation $\rho_{ij} = \frac{p_{ij}-p_ip_j}{\tilde{\sigma}_i\tilde{\sigma}_j}$ where $\tilde{\sigma}$ is the respective standard deviation.

Proof (k=2). Following Propositions (4.1) and (4.2), $p(t|\mathcal{G})$ is equivalent to querying the joint distribution marginally for $p(r_i + r_j \geq 1)$ assuming c_i and c_j predicted the target. We here assume the global distribution exists and is unique. It therefore suffices to show that

$$\max\{p_i, p_j\} = p_{\bar{i}j} + p_{i\bar{j}} + p_{ij}\ .$$

Assume w.l.o.g. that $p_i \geq p_j$. Then after plugging in $U(i,j)$ into ρ_{ij} and solving for p_{ij}, we obtain $p_{ij} = p_j$. However, by definition of the marginal it holds that $p_j = p_{ij} + p_{\bar{i}j}$ and therefore $p_{\bar{i}j} = 0$. Then we have,

$$\begin{aligned}\max\{p_i, p_j\} &= \max\{p_{i\bar{j}} + p_{ij},\ p_{\bar{i}j} + p_{ij}\}\\ &= \max\{p_{i\bar{j}} + p_{ij},\ p_{ij}\}\\ &= p_{i\bar{j}} + p_{ij}\\ &= p_{\bar{i}j} + p_{i\bar{j}} + p_{ij}.\end{aligned}$$ □

Example 4.6 (cont)**.** *For $p_1 = 0.64$ and $p_2 = 0.44$ we obtain $p_{12} = 0.44$, $p_{\bar{1}2} = 0$, and $p_{1\bar{2}} = p_1 - p_2 = 0.2$, leading to $p(t|\mathcal{G}) = 0\cdot p_{\bar{1}\bar{2}} + 1\cdot p_{12} + 1\cdot p_{1\bar{2}} + 1\cdot p_{\bar{1}2} = 0.64$.*

We have specified a unique multivariate Bernoulli distribution $p(\tilde{\mathbf{r}}|\mathcal{G})$ by simply defining a correlation matrix. Clearly setting the N^2 values of the correlation matrix is in general not sufficient for defining a distribution that has 2^N parameters and also not every correlation matrix is admissible in the first place (Huber & Marić, 2019).

4.4.2. Noisy-or Aggregation

To derive Noisy-or aggregation we have to make an assumption about the joint distribution that goes beyond pairwise interactions.

Proposition 4.7. *If the N rules in $p(\tilde{\mathbf{R}}|\mathcal{G})$ are mutually independent then $p(t|\mathcal{G}) = s^{NO}(t)$.*

It is trivial to derive the Noisy-or product from the inference model under the independence assumption and the proof is shown in the appendix for completeness.

The independence assumption of Noisy-or aggregation reveals the connection of the model from section 4.3 to ProbLog (De Raedt et al., 2007). ProbLog assigns probabilities to logic programs and inference is performed by aggregating all programs that logically entail a query by assuming individual probabilities are independent. Two results are shown in the appendix that make the connection to the derivations here explicit. First, if the logical semantics of ProbLog would be substituted with one-step entailment than it would perform Noisy-or aggregation. Second if we setup a ProbLog program with the rules $\tilde{\mathcal{C}}$, the fact probability would be always equal or larger than the Noisy-or probability. Note that the computational complexity of reasoning, as discussed earlier, here also applies. Finally, aggregating all the predicting rules with the Noisy-or product might not optimal in the context of data-driven rule learning where millions of rules can be partially redundant, which will be shown in the experimental section.

4.5. Mixing Assumptions

Both of the aggregation approaches derived in Section 4.4 make strong assumptions in regard to the dependence structure of the joint distribution over the rules. Clearly this can lead to an overestimation or underestimation of the final probability when the assumptions fail. Intuitively, this gives rise to mixture distributions that make assumptions between mutual independence and maximal correlation. Along these lines, previous work proposes models that can express both approaches as their special cases. These models are expensive to use, however, as they learn a clustering of all rules (Ott et al., 2021) or represent rules with latent embeddings (Betz et al., 2022b). We will now present a simple approach that is overlooked in the literature so far which likewise operates in between both assumptions.

Definition 4.8. *(Noisy-or top-h) Let $I^* \subseteq I$ be the subset of indices for the h predicting rules with the highest marginals. The Noisy-or top-h aggregation strategy calculates the final score according to $s(t)^{NO_h} = 1 - \prod_{j \in I^*}(1 - conf(c_j))$.*

The correlation assumption is revealed when considering that for decreasing h the approach converges to Max-aggregation which is stated more compactly in the final proposition of this section.

Proposition 4.9. *For the score calculated with noisy-or top-h we have that $s^M(t) \leq s^{NO_h}(t) \leq s^{NO}(t)$ where the equalities are achieved for $h = 1$ and $h = k$, respectively.*

The proposition immediately follows from the definitions of the approaches. Furthermore, instead of setting one value for h we can exploit the mixture property more finegrained and set the value independently for relations and query-directions which will be discussed in the next section.

5. Experiments

The goal of our experimental section is to analyse the predictive performance of the existing aggregation approaches, to evaluate how to efficiently exploit the overlooked Noisy-or top-h approach, and to give a potential user an overview about the performance-speed trade-off regarding more complex approaches. We abstain from comparing against the general KGC literature which is not the focus of this work. The competitiveness of rule-based approaches is discussed in many works and we refer to the recent literature for a summary (Rossi et al., 2021; Sadeghian et al., 2019; Meilicke et al., 2023).

5.1. Experimental Settings

We evaluate the aggregation techniques on the most common KGs from the KGC community. We use FB15k-237 (Toutanova & Chen, 2015), WNRR (Dettmers et al., 2018), Codex-M (Safavi & Koutra, 2020), and Yago3-10 (Dettmers et al., 2018). The datasets are downloaded from the LibKGE library (Broscheit et al., 2020) and we use the same train, valid, testing splits as used throughout the literature as well as the exact same evaluation protocol (Rossi et al., 2021) which is described in Section 3.1.

We use AnyBURL (Meilicke et al., 2019) and AMIE3 (Lajus et al., 2020) to mine the rulsets $\tilde{\mathcal{C}}$. For AnyBURL we use the same rulesets as used by Meilicke et al. (2021). For AMIE3 we tried to find the best possible hyperparameter configuration regarding the results (see appendix).

We compare Max (MAX), Max+ (MAX+), Noisy-or (NO), and Noisy-or top-h aggregation (NO top-h). For Noisy-or top-h we investigate how one global value $h = 5$ performs over all datasets and we additionally search for the best parameter on the validation set for the relations and query directions independently (NO top-h^*) as described in Section 4.5. For AnyBURL we search over the values $h \in \{1, 4 \dots 10\}$ where for h=1 we use MAX+. For AMIE3 we additionally include h=k as AMIE3 learned smaller rulesets and overall a smaller number of rules predict the query candidates. We also include the two works concerned with the aggregation problem, SAFRAN (Ott et al., 2021) and the supervised sparse aggregator (SV) proposed by Betz et al. (2022b). We provide wall-clock times (Table 2) of the approaches for the larger datasets and the rulesets of AnyBURL. Further experimental details, the used server architecture, dataset statistics, and the overall

		FB15k-237			WNRR			Codex-M			Yago3-10		
	Approach	h@1	h@10	MRR	h@1	h@10	MRR	h@1	h@10	MRR	h@1	h@10	MRR
AnyBURL	MAX	0.236	0.496	0.321	0.442	0.561	0.482	0.240	0.443	0.309	0.394	0.640	0.477
	MAX+	0.246	0.506	0.331	0.457	0.574	0.497	0.248	0.452	0.317	0.498	0.691	0.566
	NO	0.251	0.499	0.333	0.391	0.560	0.446	0.219	0.427	0.290	0.367	0.628	0.456
	NO top-5	0.260	0.524	0.347	0.458	0.578	0.499	0.243	0.461	0.317	0.486	0.697	0.560
	NO top-h^*	0.263	0.524	0.349	0.459	0.578	0.499	0.253	0.464	0.326	0.498	0.698	0.568
	SAFRAN	0.272	0.524	0.357	0.459	0.578	0.502	0.254	0.458	0.325	0.491	0.693	0.564
	SV	0.266	0.526	0.352	0.459	0.574	0.499	0.266	0.467	0.335	-	-	-
AMIE3	MAX	0.167	0.384	0.236	0.414	0.511	0.445	0.191	0.383	0.255	0.350	0.592	0.431
	MAX+	0.178	0.394	0.247	0.419	0.514	0.450	0.198	0.395	0.263	0.395	0.622	0.473
	NO	0.209	0.430	0.284	0.377	0.513	0.424	0.190	0.390	0.257	0.345	0.615	0.439
	NO top-5	0.199	0.425	0.273	0.380	0.513	0.426	0.197	0.401	0.266	0.360	0.622	0.452
	NO top-h^*	0.217	0.439	0.292	0.419	0.514	0.450	0.199	0.407	0.269	0.401	0.625	0.479

Table 1: Results for the joint filtered MRR and Hits@X with rules from AnyBURL or AMIE

.

number of learned rules can be found in the appendix of the work.

5.2. Results

Table 1 shows performance results and Table 2 shows runtimes for the rules from AnyBURL. Despite the fact that the datasets are quite different NO top-5 performs surprisingly well and for the rules from AnyBURL it only falls short for the h@1 and MRR metrics for Yago3-10 compared to MAX+ while being faster on average and 1.6PP better on FB15k-237. In general we observe nevertheless that the best performing specification might be dataset specific, e.g., for the rules from AMIE3 NO performs best on FB15k-237, however, the results for these rulesets are significantly worse in general. A pragmatic approach is to simply learn the best value for h on the validation set which, not surprisingly, performs always as good or better as the second best configuration although the improvement is sometimes marginal.

Although SAFRAN and SV are superior on average in regard to performance they are significantly slower. For instance SAFRAN is outperformed on Codex-m by NO top-h^* while running approximately 55 times longer and it is 0.8PP better on FB15k-237 where it runs more than 100 times longer. SV performs 0.3PP better on FB15k-237 while being 180 times slower and it performs 0.9PP better on Codex-M with a running time that is 13 times slower.

To conclude we observe that the aggregation method can have significant impact on the overall performance of the mined rulsets. Furthermore, when runtimes are a consideration factor a simple approach might be the preferred choice of aggregation.

Approach	FB15k-237	Codex-M	Yago3-10
MAX	1.1m	5.5m	4.1m
MAX+	3.1m	10.4m	4.2m
Noisy-or	5.4m	25.0m	12.2m
Noisy-or top-5	1.5m	6.6m	4.3m
NO top-h^*	13.9m	1.27h	1.01h
SAFRAN	$\approx$24h	$\approx$72h	$>$72h
SV	$\approx$42h	$\approx$16.5h	-

Table 2: Runtimes in minutes (m) our hours (h) with rules from AnyBURL.

6. Conclusion

We have shown that the problem of rule aggregation for KGC can be expressed with marginal inference over a joint distribution over the rules. We provided probabilistic interpretations for previously defined aggregation functions. Subsequently we proposed a baseline that is slightly superior over previous simple methods while being efficient and we found that more advanced models are expensive to use while only providing a small boost in regard to predictive performance. Future work might build on these foundations by finding suitable ways of modelling the joint distribution over the rules. For instance, rules could be grouped according to syntactic similarity, distributions might be estimated from more advanced statistics such as pairwise confidences or marginals could be approximated more rigorously.

References

Baier, S., Ma, Y., and Tresp, V. Improving visual relationship detection using semantic modeling of scene descriptions. In *SEMWEB*, 2017.

Betz, P., Meilicke, C., and Stuckenschmidt, H. Adversarial explanations for knowledge graph embedding models. In *Proceedings of the 31th International Joint Conference on Artificial Intelligence*, pp. 2820–2826. Ijcai.org, 2022a.

Betz, P., Meilicke, C., and Stuckenschmidt, H. Supervised knowledge aggregation for knowledge graph completion. In *Extended Semantic Web Conference*, pp. 74–92. Springer, 2022b.

Broscheit, S., Ruffinelli, D., Kochsiek, A., Betz, P., and Gemulla, R. Libkge-a knowledge graph embedding library for reproducible research. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 165–174, 2020.

Chen, Y., Wang, D. Z., and Goldberg, S. Scalekb: scalable learning and inference over large knowledge bases. *The VLDB Journal*, 25(6):893–918, 2016.

De Raedt, L., Kimmig, A., and Toivonen, H. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pp. 2462–2467, 2007.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1811–1818, 2018.

Evans, R. and Grefenstette, E. Learning explanatory rules from noisy data. In *Journal of Artificial Intelligence Research*, volume 61, pp. 1–64, 2018.

Fan, W., Fu, W., Jin, R., Lu, P., and Tian, C. Discovering association rules from big graphs. *Proceedings of the VLDB Endowment*, 15(7):1479–1492, 2022.

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. ACM, 2013.

García-Durán, A. and Niepert, M. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. In Globerson, A. and Silva, R. (eds.), *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 372–381. AUAI Press, 2018.

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

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*, 2018.

Huber, M. and Marić, N. Admissible bernoulli correlations. *Journal of Statistical Distributions and Applications*, 6 (1):1–8, 2019.

Joe, H. *Multivariate models and multivariate dependence concepts*. CRC press, 1997.

Kersting, K. and De Raedt, L. Towards combining inductive logic programming with bayesian networks. In *Inductive Logic Programming: Proceedings of the 11th International Conference*, pp. 118–131. Springer, 2001.

Kuzelka, O., Wang, Y., Davis, J., and Schockaert, S. Pac-reasoning in relational domains. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, 2018.

Lajus, J., Galárraga, L., and Suchanek, F. Fast and exact rule mining with amie 3. In *Proceedings of the Extended Semantic Web Conference*, pp. 36–52. Springer, 2020.

Liu, Y., Hildebrandt, M., Joblin, M., Ringsquandl, M., Raissouni, R., and Tresp, V. Neural multi-hop reasoning with logical rules on biomedical knowledge graphs. In *European Semantic Web Conference*, pp. 375–391. Springer, 2021.

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 *Proceedings of the International Semantic Web Conference*, pp. 3–20. Springer, 2018.

Meilicke, C., Chekol, M. W., Ruffinelli, D., and Stuckenschmidt, H. Anytime bottom-up rule learning for knowledge graph completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 3137–3143. International Joint Conferences on Artificial Intelligence Organization, 2019.

Meilicke, C., Chekol, M. W., Fink, M., and Stuckenschmidt, H. Reinforced anytime bottom up rule learning for knowledge graph completion. *arXiv preprint arXiv:2004.04412*, 2020.

Meilicke, C., Betz, P., and Stuckenschmidt, H. Why a naive way to combine symbolic and latent knowledge base completion works surprisingly well. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

Meilicke, C., Chekol, M. W., Betz, P., Fink, M., and Stuckenschmidt, H. Anytime bottom-up rule learning for large scale knowledge graph completion. *The VLDB Journal—The International Journal on Very Large Data Bases*, 2023.

Minervini, P., Bošnjak, M., Rocktäschel, T., Riedel, S., and Grefenstette, E. Differentiable reasoning on large knowledge bases and natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5182–5190, 2020a.

Minervini, P., Riedel, S., Stenetorp, P., Grefenstette, E., and Rocktäschel, T. Learning reasoning strategies in end-to-end differentiable proving. In *International Conference on Machine Learning*, pp. 6938–6949. PMLR, 2020b.

Muggleton, S. and De Raedt, L. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

Muggleton et al., S. Stochastic logic programs. *Advances in inductive logic programming*, 32:254–264, 1996.

Noessner, J., Niepert, M., and Stuckenschmidt, H. Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, 2013.

Ortona, S., Meduri, V. V., and Papotti, P. Robust discovery of positive and negative rules in knowledge bases. In *34th IEEE International Conference on Data Engineering*, pp. 1168–1179. IEEE Computer Society, 2018.

Ott, S., Meilicke, C., and Samwald, M. SAFRAN: An interpretable, rule-based link prediction method outperforming embedding models. In *3rd Conference on Automated Knowledge Base Construction*, 2021. URL https://openreview.net/forum?id=jCt9S_3w_S9.

Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

Pellissier Tanon, T., Stepanova, D., Razniewski, S., Mirza, P., and Weikum, G. Completeness-aware rule learning from knowledge graphs. In *International Semantic Web Conference*, pp. 507–525. Springer, 2017.

Qu, M., Chen, J., Xhonneux, L.-P., Bengio, Y., and Tang, J. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. *Proceedings of the International Conference on Learning Representations*, 2021.

Richardson, M. and Domingos, P. Markov logic networks. In *Machine learning*, volume 62, pp. 107–136. Springer, 2006.

Rocktäschel, T. and Riedel, S. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 3788–3800, 2017.

Rossi, A., Barbosa, D., Firmani, D., Matinata, A., and Merialdo, P. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–49, 2021.

Ruffinelli, D., Broscheit, S., and Gemulla, R. You CAN teach an old dog new tricks! on training knowledge graph embeddings. In *8th International Conference on Learning Representations*, 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. 15321–15331, 2019.

Safavi, T. and Koutra, D. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 8328–8350. Association for Computational Linguistics, 2020.

Sato, T. and Kameya, Y. Prism: a language for symbolic-statistical modeling. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, volume 97, pp. 1330–1339, 1997.

Svatoš, M., Schockaert, S., Davis, J., and Kuželka, O. Strike: Rule-driven relational learning using stratified k-entailment. In *Proceedings of the European Conference of Artificial Intelligence*, 2020.

Tena Cucala, D. J., Cuenca Grau, B., and Motik, B. Faithful Approaches to Rule Learning. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 484–493, 8 2022.

Toutanova, K. and Chen, D. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-4007. URL https://aclanthology.org/W15-4007.

Wu, H., Wang, Z., Wang, K., and Shen, Y.-D. Learning typed rules over knowledge graphs. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, pp. 494–503, 2022.

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.

Zupanc, K. and Davis, J. Estimating rule quality for knowledge base completion with the relationship between coverage assumption. In *Proceedings of the 2018 World Wide Web Conference*, pp. 1073–1081, 2018.

A. Proofs

Proof of Proposition 4.1. We show that to perform inference $p(t|\mathcal{G})$ with the probabilistic model it is sufficient to perform marginal inference on the global distribution with respect to the rules that predicted the target candidate t.

$$p(t|\mathcal{G}) = \sum_{\tilde{\mathbf{r}}\in\{0,1\}^N} p(t|\tilde{\mathbf{r}},\mathcal{G})p(\tilde{\mathbf{r}}|\mathcal{G}) \tag{8}$$

$$= \sum_{\substack{\tilde{\mathbf{r}}\in\{0,1\}^N \\ L(\tilde{\mathbf{r}})\models_1 t}} p(\tilde{\mathbf{r}}|\mathcal{G}) \tag{9}$$

We will now split each $\tilde{\mathbf{r}}$ into vectors $\mathbf{r}^+ = (\tilde{r}_j)_{j\in\tilde{I},\, c_j\models_1 t}$, the rules that predicted the target, and $\mathbf{r}^- = (\tilde{r}_i)_{i\in\tilde{I},\, c_i\not\models_1 t}$ the rules that did not predict the target. Let $\mathbf{r}^+||\mathbf{r}^- = \tilde{\mathbf{r}}$, where $||$ denotes vector concatenation. Now we can write equation (9) as

$$\sum_{\substack{\mathbf{r}^+||\mathbf{r}^-\in\{0,1\}^N \\ L(\mathbf{r}^+||\mathbf{r}^-)\models_1 t}} p(\mathbf{r}^+,\mathbf{r}^-|\mathcal{G}) \tag{10}$$

$$= \sum_{\substack{\mathbf{r}^+\in\{0,1\}^k \\ L(\mathbf{r}^+)\models_1 t}} \sum_{\substack{\mathbf{r}^-\in\{0,1\}^{N-k} \\ L(\mathbf{r}^-)\not\models_1 t}} p(\mathbf{r}^+,\mathbf{r}^-|\mathcal{G}). \tag{11}$$

Observe that the inner sum contains all possible values of $\mathbf{r}^-$ as $L(\mathbf{r}^-) \not\models_1 t$ does not put a constraint on $\mathbf{r}^-$. Continuing from equation (11) we can therefore simply apply reverse marginalization,

$$= \sum_{\substack{\mathbf{r}^+\in\{0,1\}^k \\ L(\mathbf{r}^+)\models_1 t}} p(\mathbf{r}^+|\mathcal{G})$$

$$= \sum_{\mathbf{r}\in\{0,1\}^k} p(t|\mathbf{r},\mathcal{G})p(\mathbf{r}|\mathcal{G}). \qquad \square$$

Proof of Theorem 4.5. This proof is a generalisation of the binary case from section 4.4.1. We first show that under maximal correlation only very specific realisations of $\tilde{\mathbf{R}}$ have non-zero probability if the distribution exists. Once this is established we show the existence and uniqueness of $p(\tilde{\mathbf{R}}|\mathcal{G})$ and finally we derive the max-aggregation score from the marginal inference that at least one of the predicting rules is true. As previously we assume that out of N rules k rules predicted the query triple t and that the rule marginals are given. After we have specified $p(\tilde{\mathbf{R}}|\mathcal{G})$, by Proposition 4.1 and 4.2, we have to show that

$$1 - p(\mathbf{R} = \mathbf{0}|\mathcal{G}) = \max\left\{p(R_i = 1) \mid c_i \models_1 t \text{ and } i \in I\right\}, \tag{12}$$

where $\mathbf{R}$ is the random vector for the k rules that predicted the query triple. We assume throughout the derivations the N variables $\{\tilde{R}_1, ..., \tilde{R}_N\}$ are ordered by $\tilde{I}$ (and likewise for I) such that $\tilde{R}_1$ is the rule with the highest marginal.

First we pick two rules represented by $\tilde{R}_i$, $\tilde{R}_j$ with $p_i \geq p_j$ and $i, j \in \tilde{I}$. The correlation is defined as

$$\rho_{ij} = \frac{p_{ij} - p_i p_j}{\tilde{\sigma}_i \tilde{\sigma}_j}, \tag{13}$$

Where $p_{ij} = p(\tilde{R}_i{=}1, \tilde{R}_j{=}1|\mathcal{G})$. We now assume $\rho_{ij}{=}U(i,j)$ for every pair i, j from $\tilde{I}$. We plug in the upper bound (7) into (13) and solve for p_{ij} which leads to

$$p_{ij} = p_j, \tag{14}$$

i.e., we have the following equality

$$\sum_{\tilde{\mathbf{r}}_{-ij}\in\{0,1\}^{N-2}} p(\tilde{R}_i{=}1, \tilde{R}_j{=}1, \tilde{\mathbf{r}}_{-ij}|\mathcal{G}) =$$

$$= \sum_{\tilde{\mathbf{r}}_{-j}\in\{0,1\}^{N-1}} p(\tilde{R}_j{=}1, \tilde{\mathbf{r}}_{-j}|\mathcal{G}), \tag{15}$$

where $\tilde{\mathbf{r}}_{-j} = (\tilde{r}_i)_{i\in\tilde{I}\setminus j}$ is a vector of realisations with the j'th component dropped from $\tilde{\mathbf{r}}$ and equivalently $\tilde{\mathbf{r}}_{-ij} = (\tilde{r}_s)_{s\in\tilde{I}\setminus\{i,j\}}$. Each addend in the left hand side is contained in the right hand side, subtracting the left hand side from both sides of (15) yields a zero probability constraint:

$$0 = \sum_{\mathbf{r}_{-ij}\in\{0,1\}^{N-2}} p(\tilde{R}_i = 0, \tilde{R}_j = 1, \mathbf{r}_{-ij}|\mathcal{G}). \tag{16}$$

We are, in fact, interested in all the realisations that may be different from zero after considering the constraints imposed by all possible rule pairs. From equation (16) it follows that $p(\tilde{\mathbf{R}} = \tilde{\mathbf{r}}|\mathcal{G})$ is not affected by the zero-constraint if for $\tilde{\mathbf{r}}$

$$(\tilde{r}_s = 0) \implies (\tilde{r}_t = 0) \;\; \forall\, t > s, \tag{17}$$

for $s, t \in \tilde{I}$. Note that each assignment $\tilde{\mathbf{r}} \in \{0,1\}^N$ which satisfies (17) is associated with a unique number of components (rules) that are one.

Our goal is to specify the parameters of $p(\tilde{\mathbf{R}}|\mathcal{G})$ that are non-zero. From (17) we can observe that there are only $N+1$ of these parameters left and we will therefore introduce $N+1$ variables. Let $m \in \{0, ..., N\}$ and let z_m denote the probability for the assignment vector that has m ones and satisfies (17) which we write as $\tilde{\mathbf{r}}^{(m)}$, i.e., $z_m = p(\tilde{\mathbf{r}}^{(m)}|\mathcal{G})$. In fact, $\tilde{\mathbf{r}}^{(m)} \in \{0,1\}^N$ holds ones from the first component until the m'th component and zeros starting from the $m+1$'th component. It is easy to verify that now it holds for the N marginals with $i \in \tilde{I}$ that $p_i = \sum_{s=i}^{N} z_s$ and additionally we use the probability constraint $p_0 = \sum_{s=0}^{N} z_s = 1$. With these expressions we can set up an equation system

$$\mathbf{A}\mathbf{z} = \mathbf{p}, \tag{18}$$

where $\mathbf{z}$ is the variable vector with dimensionality $N+1$, $\mathbf{A}$ is an upper triangular coefficient matrix with all non-zero entries being one, and $\mathbf{p}$ is the vector of marginals and the probability constraint at the first entry. Given that $\mathbf{A}$ is invertible we established uniqueness and we established existence as the solution $\mathbf{z} = \mathbf{A}^{-1}\mathbf{p}$ satisfies the probability constraint $\sum_{s=0}^{N} z_s = 1$ while all z_m are between 0 and 1.

We will now derive the main result from (12). Plugging in the expressions for the marginals in the right-hand side of (12) yields

$$\max\Big\{ \sum_{s=i}^{N} z_s \;\Big|\; c_i \models_1 t \text{ and } i \in I \Big\} = \sum_{s=s^*}^{N} z_s \;, \tag{19}$$

where $s^* = \min\{i \mid c_i \models_1 t \text{ and } i \in I\}$ corresponds to the index for the rule with the highest marginal under the predicting rules. For $1 - p(\mathbf{R}{=}\mathbf{0}|\mathcal{G})$ we have to sum up all probabilities of realisations where at least one of the predicting rules is one. Clearly this includes all realisations where r_{s^*} is one which holds by construction of the z_m's for every term in the sum on the right hand side of equation (19). Now given that the remaining probabilities are zero we have that $\sum_{s=s^*}^{N} z_s = 1 - p(\mathbf{R} = \mathbf{0}|\mathcal{G})$. □

Proof of Proposition 4.7. We directly start with the result from Proposition 4.1.

$$\begin{aligned} p(t|\mathcal{G}) &= \sum_{\mathbf{r}\in\{0,1\}^k} p(t|\mathbf{r},\mathcal{G}) p(\mathbf{r}|\mathcal{G}) \\ &= 0 \cdot p(\mathbf{R} = \mathbf{0}|\mathcal{G}) + \sum_{\mathbf{r}\neq\mathbf{0}\in\{0,1\}^k} p(\mathbf{r}|\mathcal{G}) \\ &= 1 - p(\mathbf{R} = \mathbf{0}|\mathcal{G}) \\ &= 1 - \prod_{j\in I} \big(1 - p(r_j)\big) \end{aligned}$$ □

B. Evaluation Metrics

Let $\mathcal{G}_e$ be the test KG. For the target test fact $q(c_1, c_2)$ let $\mathbf{rk}(c_1|c_2, q)$ be the ranking position of the target in a filtered ranking of candidate facts for the tail query $q(c_1, ?)$ and likewise let $\mathbf{rk}(c_2|q, c_1)$ denote the filtered ranking position for the head query. MRR and Hits@X are defined as:

$$MRR = \frac{1}{2|\mathcal{G}_e|} \sum_{q(c_1,c_2)\in\mathcal{G}_e} \left(\frac{1}{\mathbf{rk}(c_1|c_2,q)} + \frac{1}{\mathbf{rk}(c_2|q,c_1)} \right),$$

$$hits@X = \frac{1}{2|\mathcal{G}_e|} \sum_{q(c_1,c_2)\in\mathcal{G}_e} \Big(\mathtt{1}\big\{\mathbf{rk}(c_1|c_2,q) \leq X\big\} \quad + \\ + \mathtt{1}\big\{\mathbf{rk}(c_2|q,c_1) \leq X\big\} \Big).$$

C. Rule Aggregation and Reasoning with Problog

The probability for a logic program $\mathcal{P}$ given a ProbLog program T is defined as

$$p(\mathcal{P}|T) = \prod_{x_i\in\mathcal{P}} p(x_i) \prod_{x_j\notin\mathcal{P}} (1 - p(x_j)), \tag{20}$$

where T is a collection of definite clauses with assigned probabilities. In the scope of this work when using the ProbLog notation we can set $T^* = \{p_i : c_i \mid i \in \tilde{I}\} \cup \{1 : t' \mid t' \in \mathcal{G}\}$ to obtain the ProbLog program representing the rules and the facts. If we, for instance, let ProbLog only perform one-step entailment we obtain the following result.

Proposition C.1. *For the probability $p(t|T^*)$ calculated with ProbLog under a one-step entailment regime it holds that $p(t|T^*) = s^{NO}(t)$.*

Proof of Proposition C.1. The query probability given the ProbLog program T^*, as defined above, is calculated as

$$p(t|T^*) = \sum_{\tilde{\mathbf{r}}\in\{0,1\}^N} p\big(t|L(\tilde{\mathbf{r}})\big) p\big(L(\tilde{\mathbf{r}})|T^*\big) \;, \tag{21}$$

where $p(t|L(\tilde{\mathbf{r}}))$ is set to equation (2) by requirement of the proposition and $p(L(\tilde{\mathbf{r}})|T)$ can be interpreted as the probability of a logic program when treating rules as logical clauses and facts as ground atoms. Note that $L(\tilde{\mathbf{r}}) = \{c_i \mid \tilde{r}_i = 1 \; and \; i \in \tilde{I}\} \cup \mathcal{G}$. Plugging in equation (20) into equation (21) and rearranging leads to:

$$
\begin{aligned}
&\sum_{\tilde{\mathbf{r}}\in\{0,1\}^N} p(t|L(\tilde{\mathbf{r}})) \prod_{x_i\in L(\tilde{\mathbf{r}})} p(x_i) \prod_{x_j\notin L(\tilde{\mathbf{r}})} (1-p(x_j)) \\
&= \sum_{\tilde{\mathbf{r}}\in\{0,1\}^N} p(t|L(\tilde{\mathbf{r}})) \prod_{c_i\in L(\tilde{\mathbf{r}})} p(\tilde{r}_i) \prod_{c_j\notin L(\tilde{\mathbf{r}})} (1-p(\tilde{r}_j)) \prod_{t\in\mathcal{G}} p(t) \\
&= \sum_{\tilde{\mathbf{r}}\in\{0,1\}^N} p(t|L(\tilde{\mathbf{r}})) \prod_{c_i\in L(\tilde{\mathbf{r}})} p(\tilde{r}_i) \prod_{c_j\notin L(\tilde{\mathbf{r}})} (1-p(\tilde{r}_j)) \cdot 1 \\
&= \sum_{\tilde{\mathbf{r}}\in\{0,1\}^N} p(t|L(\tilde{\mathbf{r}}))p(\tilde{\mathbf{r}}|\mathcal{G}) \ .
\end{aligned}
$$

The factorization of the logic program implies mutual independence and therefore applying Propositions 4.1 and 4.7 leads to the Noisy-or product over the rules that predicted t. □

The next theorem shows the behaviour when using the full ProbLog algorithm for rule aggregation.

Theorem C.2. *For the query probability $p(t|T^*)$ calculated by ProbLog it holds that $p(t|T^*) \geq s^{NO}(t)$.*

We first sketch the proof here which is straightforward when using Propositions C.1 and 4.1. The details are given below. ProbLog sums the probabilities of all programs that entail the target fact. This includes 1) the programs that entail and one-step entail the query and 2) the programs that entail but not one-step entail the query which is clearly larger or equal than only aggregating 1) as done in Noisy-or aggregation.

Proof of Theorem C.2

As before let $\tilde{\mathbf{r}} \in \{0,1\}^N$ be a vector of realisations. We will now label different assignment vectors according to their logical properties. Let $\tilde{\mathbf{r}}^{(e)} \in \{0,1\}^N$ be an assignment vector such that $L(\tilde{\mathbf{r}}^{(e)})$ entails but not one-step entails the query and let $\tilde{\mathbf{r}}^{(o)} \in \{0,1\}^N$ be the corresponding vector where $L(\tilde{\mathbf{r}}^{(o)})$ entails and one-step entails the query. Let T^* denote the ProbLog program as defined above. For the query probability under ProbLog we have

$$
p(t|T^*) = \sum_{\tilde{\mathbf{r}}\in\{0,1\}^N} \hat{p}\big(t|L(\tilde{\mathbf{r}})\big)p\big(L(\tilde{\mathbf{r}})|T^*\big), \qquad (22)
$$

where

$$
\hat{p}(t|L(\tilde{\mathbf{r}})) = \begin{cases} 1, \text{ if } L(\tilde{\mathbf{r}}) \models t \\ 0, \text{ else.} \end{cases}
$$

When we plug in $\hat{p}(t|L(\tilde{\mathbf{r}}))$ then (22) becomes

$$
\begin{aligned}
p(t|T^*) &= \sum_{\substack{\tilde{\mathbf{r}}\in\{0,1\}^N \\ L(\tilde{\mathbf{r}})\models t}} p(L(\tilde{\mathbf{r}})|T^*) \\
&= \sum_{\substack{\tilde{\mathbf{r}}^{(e)}\in\{0,1\}^N \\ L(\mathbf{r}^{(e)})\models t \\ L(\mathbf{r}^{(e)})\not\models_1 t}} p(L(\tilde{\mathbf{r}}^{(e)})|T^*) + \\
&+ \sum_{\substack{\tilde{\mathbf{r}}^{(o)}\in\{0,1\}^N \\ L(\tilde{\mathbf{r}}^{(o)})\models_1 t}} p(L(\tilde{\mathbf{r}}^{(o)})|T^*) \\
&\geq \sum_{\substack{\tilde{\mathbf{r}}^{(o)}\in\{0,1\}^N \\ L(\tilde{\mathbf{r}}^{(o)})\models_1 t}} p(L(\tilde{\mathbf{r}}^{(o)})|T^*) \\
&= \sum_{\tilde{\mathbf{r}}\in\{0,1\}^N} p\big(t|L(\tilde{\mathbf{r}})\big)p\big(L(\tilde{\mathbf{r}})|T^*\big)
\end{aligned}
$$

where it follows from Proposition C.1 that the last expression is the Noisy-or product under the program factorization of ProbLog for $p\big(L(\tilde{\mathbf{r}})|T^*\big)$ □

D. Rule Aggregation and MLNs

For an MLN query answering $p(t|\mathcal{G})$ as in the main text can be performed by marginal inference given some evidence which is in our case the KG. Nevertheless we start with a simple example with unary predicates. For the MLN definitions we refer to the original publication. Consider the two formulae $smokes(X) \rightarrow cancer(X)$ and $ill(X) \rightarrow cancer(X)$, assume they have some confidence $conf_1$ and $conf_2$. Now let the evidence be $e = \{smokes(karl),\ ill(karl)\}$ and we do not have any other constant terms. Note that there are 2^3 possible worlds and we seek to calculate $p(cancer(karl){=}1|e)$, i.e., the probability that Karl has cancer. We abuse notation slightly for brevity. In particular we write $p(\{a_1, a_2\})$ for the probability of a world where a_i is a true atom and all the remaining possible atoms are false. And we write $p(a_i{=}1|e)$ for the marginal probability that an atom is true given the evidence.

For instantiating a Markov Network, as each formula has one possible grounding, we have two binary features f_1, f_2. A feature is one if in a given world the respective formula is satisfied. Assigning the confidences as weights is not useful because of the exponential formulation, therefore we set the feature weights to the log odds $w_i = \ln \frac{conf_i}{1-conf_1}$ for $i \in \{1,2\}$. Now assume $conf_1 = conf_2 = 0.9$. For $p(\{cancer(karl)\}|e)$ we have

$$
p(c(k) = 1|e) = \frac{p\big(\{c(k), i(k), sm(k)\}\big)}{p\big(sm(k) = 1, i(k) = 1\big)}
$$

Note that the denominator is a marginal that sums over all

worlds where $smokes(karl)$ and $ill(karl)$ is true, therefore

$$p(c(k)=1|e) = \frac{p\big(\{c(k),i(k),sm(k)\}\big)}{p\big(\{c(k),i(k),sm(k)\}\big)+p\big(\{i(k),sm(k)\}\big)}$$
$$= \frac{\exp{(w_1+w_2)}Z^{-1}}{\exp{(w_1+w_2)}Z^{-1}+\exp{(0)}Z^{-1}}$$
$$= \texttt{sigmoid}(w_1+w_2)$$

Plugging in the values for the weights results in 0.8581 which we also exactly recover when using the MLN solver Rockit (Noessner et al., 2013) under exact marginal inference. Now consider the rules,

$$c_1\colon married(X,Y) \leftarrow engaged(X,Y)$$
$$c_2\colon married(X,Y) \leftarrow commonChild(X,Y)$$
$$c_3\colon married(X,Y) \leftarrow inLove(X,Y)$$

Assume the hypothetical confidences $conf_1 = 0.8$, $conf_2 = 0.7$, $conf_3 = 0.5$. Further assume the evidence KG $\mathcal{G} = \{inLove(a,b), commonChild(a,b), engaged(a,b)\}$ where a, b are constants. We want to calculate $p(t=1|\mathcal{G})$ where $t = married(a,b)$. If we use ProbLog with the confidences we obtain $p(t|\mathcal{G}) = 0.97$ which is here the same as Noisy-or aggregation. Max-aggregation leads to $p(t|\mathcal{G}) = 0.8$ and using Max-group aggregation would be in between depending on the grouping. When using the log-odds for the MLN as above we obtain with Rockit $p(t|\mathcal{G}) = 0.9032$ which is again the sigmoid function applied to the the sum of the log-odds.

Note that in these two examples the only form of reasoning in regard to the query fact is one-step entailment due to the simplicity of the examples. This can be emulated for MLN's also for more complicated examples with the common solvers by defining the rule bodies as observed predicates. Note that additionally the log odds would be weighted with each rule grounding regarding the rules that predicted the query. Nevertheless the resulting aggregation baseline would not perform well in the context of KGC as, for instance, a rule with confidence of 0.4 and many groundings would lead to a small value for the final probability. However, note that, as we mentioned in the main text, a MLN is a more general framework not based on one-step entailment and expressing the aggregation problem requires to make several non trivial decisions.

E. Experimental Details

We show dataset statistics and the overall number of learned rules in Table 3 and 4. Further experimental details are given in the following subsections.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{P}\rvert$	Num facts $\lvert\mathcal{G}\rvert$ Train	Valid	Test
FB15k-237	14 505	237	272 115	17 535	20 466
WNRR	40 559	11	86 835	3 034	3 134
Codex-M	17 050	51	185 584	10 310	10 311
Yago3-10	123 182	37	1 079 040	5 000	5 000

Table 3: Dataset summary statistics

Dataset	AMIE	AnyBURL
FB15k-237	983 546	5 084 903
WNRR	3426	97 329
Codex-M	179 898	7 409 385
Yago3-10	900 951	6 692 784

Table 4: Number of rules learned

E.1. Rule learning

For AnyBURL we use the rulesets provided in previous work (Meilicke et al., 2021) except for Yago3-10 where we learn with the default parameters for 3600 seconds with the AnyBURL-JUNO version. For AMIE3 under the default parameters only a few rules are learned so we adjusted the parameters. When including rules with constants and longer rules the approach does not terminate within a day therefore we use one execution with rules of length one including constants and one execution with longer rules without constants and subsequently the rulsets are merged. We show below the program execution with constants and the second execution with longer rules.

```
java -jar amie.jar train.txt -const
  -bias default -minhc 0.0 -minc
  0.001 -minpca 0.001 -maxad 2

java -jar amie.jar train.txt -bias
  default -minhc 0.0 -minc 0.001
  -minpca 0.001 -maxad 4 -pm
  support -mins 2
```

E.2. Implementation and Evaluation Details

Rule-based KGC uses commonly a parameter $topX$ that denotes how many candidate facts $q(c_1, c^*)$ for the query $q(c_1, ?)$ (likewise in head direction) should be predicted and ranked. For all our experiments we set $topX$=200 but we did not notice significant differences to using 100. When two candidates are assigned to the same probability we use random tie handling. The aggregation functions are implemented under the AnyBURL-JUNO codebase. For MAX+ we use the existing implementation. For Noisy-or top-h we start predicting with the rule with the highest marginal continuing with the second highest rule and we stop when for $topX$ candidates each one was predicted by at least h rules. For Noisy-or we set $h = k$ and for MAX we set $h = 1$.

E.3. Experiment Details and Execution

Our experiments are run on a CPU server with 768 GB RAM and two Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz cores with 40 logical cores each. For the standard experiments we use the AnyBURL-JUNO codebase with 30 threads. For the Noisy-or top-h^* approach we spawn individual processes with 15 threads each for every value of h. The runtimes for the experiments are wall-clock times, i.e., we measure the time before and after the execution on each dataset. For SAFRAN we obtained runtime estimates from the authors for FB15k-237 and Codex-m and we run the approach (architecture as above) on Yago3-10 where it did not terminate within 3 days. Runtimes for SV are also obtained from the authors. Results for SAFRAN are obtained from the authors where on FB15k-237 SAFRAN is run in accordance to a newer AnyBURL version that does not exploit the characteristic that connected entities in the training set cannot form a fact in the test set. See Meilicke et al. (2020) for a discussion. The results for SV are from the respective publication.