# Few-Shot Knowledge Validation using Rules

Michael Loster
michael.loster@hpi.de
Hasso Plattner Institute

Davide Mottin
davide@cs.au.dk
Aarhus University

Paolo Papotti
paolo.papotti@eurecom.fr
EURECOM

Jan Ehmüller
jan.ehmueller@student.hpi.de
Hasso Plattner Institute

Benjamin Feldmann
benjamin.feldmann@student.hpi.de
Hasso Plattner Institute

Felix Naumann
felix.naumann@hpi.de
Hasso Plattner Institute

## ABSTRACT

Knowledge graphs (KGs) form the basis of modern intelligent search systems – their network structure helps with the semantic reasoning and interpretation of complex tasks. A KG is a highly dynamic structure in which facts are continuously updated, added, and removed. A typical approach to ensure data quality in the presence of continuous changes is to apply logic rules. These rules are automatically mined from the data using frequency-based approaches. As a result, these approaches depend on the data quality of the KG and are susceptible to errors and incompleteness.

To address these issues, we propose Colt, a few-shot rule-based knowledge validation framework that enables the interactive quality assessment of logic rules. It evaluates the quality of any rule by asking a user to validate only a few facts entailed by such rule on the KG. We formalize the problem as learning a validation function over the rule's outcomes and study the theoretical connections to the generalized maximum coverage problem. Our model obtains (i) an accurate estimate of the quality of a rule with fewer than 20 user interactions and (ii) 75% quality ($F_1$) with 5% annotations in the task of validating facts entailed by any rule.

**ACM Reference Format:**
Michael Loster, Davide Mottin, Paolo Papotti, Jan Ehmüller, Benjamin Feldmann, and Felix Naumann. 2021. Few-Shot Knowledge Validation using Rules. In *TheWebConf '21: 30th The Web Conference, April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

In recent years, systems such as DeepDive [33] and Knowledge Vault [11] have made it increasingly easy to automatically construct vast knowledge graphs (KGs) from structured and unstructured data. These systems often consist of many different components designed to extract and integrate facts from multiple sources. Many of these components are based on machine learning techniques, which are rarely error-free. Thus, despite dramatic improvements using deep learning techniques, it is not possible to ensure the correct extraction, linkage, and discovery of relationships between textual entities, which form the facts of a knowledge graph. Knowledge graphs suffer not only from incorrect but also from missing facts, which in their sum negatively affect all downstream applications. Correcting these errors by adding missing or removing incorrect facts represents a key task that can be approached from two different directions.

The first , often used to add missing facts, leverages *knowledge graph embeddings* (KGE) [37], such as TransE [5] or HolE [25], in training link prediction models. These models are widely studied and can be used to predict new facts; however, their quality depends on the correctness and completeness of the underlying KG. While correctly capturing large portions of the graph, they fail to model noisy or underrepresented entities and relationships [28]. Additionally, KGEs are difficult to interpret as they cannot *explain* why a fact should be included in the KG.

The second approach is applying *logic rules*, which can be derived by rule learning systems, such as AMIE [13, 14] and RuDiK [27]. By executing such rules on a KG, missing facts can be added and incorrect facts removed. For example, the rule

$$\texttt{isMarriedTo}(\texttt{a}, \texttt{c}) \land \texttt{livesIn}(\texttt{c}, \texttt{b}) \Rightarrow \texttt{livesIn}(\texttt{a}, \texttt{b})$$

states that if a person `a` is married to another person `c`, they most likely live in the same place (`b`). By applying this rule, i.e., when the left-hand-side of the rule is satisfied, a new `livesIn` fact between a person and a place can be added to the KG. While rules are easier to interpret than embeddings, their discovery also suffers from noisy and missing KG data. Moreover, rules are rarely completely correct or completely incorrect. Commonly, a rule applies only to a certain percentage of KG facts [13, 27], without providing exact and complete information in which cases it can be safely applied.

Ideally, a rule should only be applied if it contributes to the quality improvement of a KG. Although rule learning systems come with a statistical measure for the support of a discovered rule, this measure is derived from the KG facts and assumes their correctness. In practice, this dependence often leads to incorrect estimates of a rule's confidence value. Most systems refer to the confidence of a rule as a value between 0 and 1, describing how likely the rule in question implies valid facts, which in turn defines the validity of the rule itself.

Addressing such data quality issues of a KG requires *external information* to make necessary corrections. The natural way to verify facts is to consult experts on the topics covered by the KG at hand. As KGs often consist of millions of facts, a comprehensive manual verification becomes infeasible. Thus, we aim to radically reduce the manual effort by efficiently utilizing existing domain knowledge.


*TheWebConf '21, April 19–23, 2021, Ljubljana, Slovenia*

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

To retain the understandability of rule-based approaches, we focus on a solution that (i) computes reliable rule confidence measures and (ii) complements the rules by a probabilistic model that validates their resulting facts.

To address the data quality problems of KGs, we train a classifier for each rule that balances the exploration of new facts and the exploitation of already learned knowledge. This allows us to effectively reduce the number of annotations to a few interactions also referred to as a *few shots*. Unlike logic rules, the classifiers contain additional knowledge in the form of (i) information captured by the KGEs and (ii) user feedback, enabling the classifiers to make better predictions than the original rule. The learned classifiers allow the conditional application of each rule, according to the predicted validity of its facts. Compared to a data-driven estimation approach, these predictions enable us to calculate a rule's confidence with far greater accuracy. Although the learned classifier can be used as a replacement for the learned rule, it reduces the role of the rule only to a limited extent. As the classifier itself is quite complex and therefore difficult to interpret, the rule serves as an explanation of the classifier, making its behavior more comprehensible. Beyond its use in curating data quality problems, the creation of precise rule confidence estimates also plays a fundamental role in reasoning systems, such as those used in fact checking [1]. In these applications, the quality of the conclusion depends on the precise measurement of the rule quality. However, verifying the correctness of a large number of facts to precisely assess rule quality is cost-intensive. We therefore reduce the amount of work by utilizing a sampling strategy. This strategy limits the effort for manually verifying facts to an adjustable budget, while simultaneously maximizing the benefit from the verified facts.

Our core contribution constitutes the development of a model that theoretically and empirically aims to validate the confidence and precision of logical rules by posing as few questions as possible to a human oracle. In doing so, we answer the key question of whether there is a model capable of achieving maximum precision, provided that 100% of the annotated data is available. By reducing the problem to maximum coverage, we show that such a model exists, but has its drawbacks. As active learning methods guarantee no perfect precision, which is confirmed by our experiments, we turn to Gaussian processes (GP), that theoretically ensure maximum precision as the number of interactions approaches infinity. Our proposed approach learns the characteristics of a particular rule from humans who annotate a small number of its generated facts and addresses the limitations of GPs (e.g., instability of matrix inversion) by means of deep kernel learning.

We summarize our *contributions* as follows:

(1) We propose the Colt framework capable of assessing the quality and confidence of a particular rule by using KG embeddings and expert-validated facts (Section 3).
(2) Using a classifier, we enable the conditional application of declarative rules and compute their confidence (Section 4).
(3) We establish a connection between our problem, the weighted coverage problem (Section 4.1), and quality-preserving Gaussian processes (Section 4.2).
(4) We show how our interactive learning approach effectively exploits embeddings to improve the classifier with minimal human effort. Using only 20 user interactions, we halve the error in confidence obtained with rule learning systems (Section 6).
(5) We publish our dataset consisting of 26 rules with more than 23 000 annotated facts.

## 2 RELATED WORK

This section covers the topics related to our method, namely, rule discovery and their interactive application, followed by graph embeddings and hybrid methods using both rules and embeddings.

**Rule discovery.** The derivation of logic rules from KGs has been investigated for many years [7]. Recent systems, such as AMIE [13, 14] and RuDiK [27], can derive rules from large KGs using structural information, such as frequently occurring graph patterns. The mined rules can be used to derive new facts, find errors, derive conclusions, and better understand the underlying data. However, it is difficult to automatically estimate the quality of a discovered rule. Although AMIE and RuDiK report statistical measures for every rule, such as the standard and PCA confidence, these metrics are based on pattern frequencies that are affected by data quality issues in the KG. We improve the confidence assessment of a rule by including a user as an additional source of knowledge. As such, the quality assessment of a rule does not depend solely on the KG.

**Interactive rule execution.** Different approaches have studied how to execute KG rules for a specific task with user involvement. For data repair, potential KG corrections are derived from a set of rules and suggested to the user. [2, 12]. The user validates these corrections, thus improving the underlying KG until it is transformed into a consistent state. This line of work assumes that the given rules are correct and reliable, such as those manually written by experts. Our approach handles rules that are automatically generated, thus possibly approximate or erroneous. Our solution can be seen as a preliminary step to create an understanding of the rules to be fed to the data repair step.

Interactive error detection has also been studied for relational data [17, 23]. However, such methods cannot be directly applied to graph data; while a rule can be materialized into a relational model by flattening a portion of the KG, the resulting "view" disregards other information in the graph.

**Knowledge graph embeddings.** Knowledge graph embeddings (KGEs) provide a compact representation of a KG by projecting its entities and relations into an $n$-dimensional space while retaining its structural information. These embeddings are used to improve tasks [37], such as relation extraction [42] and link prediction [21].

Besides semantic matching models [26], translational methods, such as TransE [5] and its extensions [19, 38], are widely used for the generation of embeddings. These methods maximize a score associated with each fact. Often interpreted informally as the plausibility of a fact, the score is a distance measure between the entities of the evaluated predicate. As these methods use the KG facts to produce a representation of the graph, data quality problems, such as incorrect facts, directly affect the quality of the generated embeddings [28] and thus also downstream tasks.

We address quality issues by learning classifiers that incorporate human knowledge during their training and do not solely rely on the KG data. By integrating user feedback, the classifier learns how

to counteract poor embeddings and mitigate their negative effects. This limits the impact of noisy and missing KG information.

**KGEs and rules.** Although all methods use the KG facts for computing KGEs, some approaches can consume additional information, such as textual descriptions [41], literals [15], or logical rules [16, 43]. We focus on the approaches that rely on rules during their learning process, as they are closest to our work.

One approach jointly models logical rules and fact triples in a unified framework, representing triples as atomic and rules as complex formulae, while learning takes place by minimizing a global loss function that spans both formulae [16]. Another system creates high-quality rules from noisy KGs using KGEs to control the rule generation process [18]. A third approach uses rules to generate new triples for the training of the KGEs, which are used to derive new rules [43]. These works address a problem different from ours: we train a classifier on top of existing KGEs and existing rules that counteracts poor embedding quality by incorporating user feedback into its learning process. Our goal is to interactively learn a model that can both assess the quality of a given rule and predict when the rule should be applied.

**Active learning.** Methods that actively sample data points to be labeled are called active learning methods [31]. They exploit the properties of model and data and perform well with human assessors and exploratory tasks [9]. Our work builds on the general idea of active learning but establishes important connections with weighted coverage and GP-UCB algorithm based on Gaussian processes [34] to devise quality-preserving solutions. For consistency, we evaluate the most common active learning strategies described in Section 4.4.

## 3 BACKGROUND AND PROBLEM

A *knowledge graph* (KG) is a triple $G : \langle \mathcal{E}, \mathcal{R}, \mathcal{F} \rangle$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ a set of relationships, and $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of facts. For instance, a general-purpose KG contains the entities *Max Planck* and *Germany*, connected via *isCitizenOf* relationship. A triple $(e_1, r, e_2)$ with $e_1, e_2 \in \mathcal{E}$, and $r \in \mathcal{R}$ is called a *fact*; as $r$ is a relationship in the pair $(e_1, e_2)$, each fact can be equivalently represented as an *atom* $r(e_1, e_2)$. The set of triples constitutes the knowledge graph, also known as knowledge base [8], information graph [22], or heterogeneous information network [32].

A knowledge graph might contain errors, such as *(Max Planck, isCitizenOf, China)*. The set of all true facts (knowledge) is denoted by $\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, and we assume that at least a part of the knowledge graph is correct, i.e., $\mathcal{F} \cap \mathcal{K} \neq \emptyset$. Partial correctness constitutes a reasonable assumption regarding the soundness of the KG construction process. An *atom* is the smallest logic statement composed by facts, including variables. For example, the atom *hasChild(x, John)* denotes all entities (by the variable $x$) that have a child named John.

Definition 1. *A* rule $A_1 \wedge A_2 \wedge \ldots \wedge A_n \Rightarrow C$ *is a logical implication consisting of a head and a body. The head is a single atom C, and the body is a conjunction of atoms* $A_1 \wedge \ldots \wedge A_n$.

The rule $\rho_0 : \mathit{hasChild}(a, c) \wedge \mathit{isCitizenOf}(a, b) \Rightarrow \mathit{isCitizenOf}(c, b)$ describes that if $a$ has a child $c$ and is a citizen of country $b$, their child must also be a citizen of $b$. This is a *positive* rule, which asserts the existence of specific facts in the head, given the body. We also consider *negative* rules to identify inconsistencies in the data [2], which denote sets of facts that lead to contradictions.

The *instances* $\mathcal{I}_G(\rho) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ of a rule $\rho$ in the knowledge graph $G$ are the set of facts expressed by the right-hand-side of the rule, assuming the left-hand-side is true. For example, the instances of rule $\rho_0$ are facts of the form $\mathit{isCitizenOf}(c, b)$, given that both $\mathit{hasChild}(a, c)$ and $\mathit{isCitizenOf}(a, b)$ are true.

Ideally, if we knew the true facts $\mathcal{K}$, we could validate the *confidence* of the rule on the knowledge graph $G$ by computing the instances in $\mathcal{K}$ that are correctly captured by a rule $\rho$ via the ratio $\frac{|\mathcal{I}_{\mathcal{K}}(\rho) \cap \mathcal{I}_G(\rho)|}{|\mathcal{I}_{\mathcal{K}}(\rho)|}$. As $\mathcal{K}$ is, in reality, unavailable, we model the validation as a *benefit function* $f : \mathcal{I}_G(\rho) \rightarrow [0, 1]$ over the instances $\mathcal{I}_G(\rho)$. $f$ returns a value that converges to 1 if the fact $(e_1, r, e_2) \in \mathcal{I}_G(\rho)$ is confidently correct, i.e. $(e_1, r, e_2) \in \mathcal{K}$, otherwise it converges to 0. Access to (parts of) $\mathcal{K}$ can be obtained by an external oracle, e.g., by a domain expert who verifies whether a fact is true or not. Assessing the validity of facts by consulting an oracle can be costly, which is why we limit the number of evaluations to a budget $B$, thus bounding the evaluation effort.

The *rule-driven knowledge validation* problem aims at finding a benefit function $f$ within a budget $B$ that maximizes the knowledge over the instances $\mathcal{I}_G(\rho)$ of rule $\rho$:

Problem 1. *Given a knowledge graph* $G : \langle \mathcal{E}, \mathcal{R}, \mathcal{F} \rangle$*, a rule* $\rho$ *with instances* $\mathcal{I}_G(\rho) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$*, and a budget B, find a set of facts L*

$$\underset{L \subseteq \mathcal{I}_G(\rho)}{\arg\max} \sum_{x \in L} f(x) \qquad \text{subject to } C(L) \leq B$$

*where C represents the sum of costs of all the facts in L.*

In many practical scenarios, the cost of a fact is 1, as the budget quantifies the number of questions to the oracle. In our setting, we make this assumption but note that the problem, even if the benefit $f$ is known, is NP-hard [36] with a non-constant cost.

## 4 THE COLT FRAMEWORK

In practice, the benefit function $f$ in Problem 1 is unknown. As such, excerpting the knowledge $f$ from an oracle makes the direct optimization in Problem 1 impossible. To circumvent this issue, Colt makes two assumptions. **First**, the benefit function can be equivalently represented by a *classifier* $\widetilde{f}$ on the facts trained on the evaluations seen so far. **Second**, since facts are related to one another, we assume the existence of *a similarity* $\kappa$:$(\mathcal{E}\times\mathcal{R}\times\mathcal{E}) \times (\mathcal{E}\times\mathcal{R}\times\mathcal{E}) \mapsto [0, 1]$ among pairs of facts. Such similarities naturally manifest in knowledge graphs by means of relationships among entities. As facts themselves are too sparse and fail to capture implicit KG relationships, it takes a different approach to capture their similarity. Section 5 provides more insights on how to compute expressive similarities.

Our Colt framework, presented in Algorithm 1, proceeds iteratively by using the information in the classifier and asking the user to evaluate facts until a specified budget $B$ is depleted. In the beginning, the classifier is initialized (Line 2) using the instances and the similarity function. At every iteration $t$, the algorithm selects a new fact $x_t$ using the classifier's knowledge (Line 5). The user then validates the fact (Line 6) providing a value $y_t = 1$ if the fact is true and $y_t = 0$ otherwise. Finally, the algorithm updates its beliefs

by incorporating the true value $y_t$ for the fact $x_t$. The classifier serves as a proxy for the unknown function $f$, which encodes the true facts $\mathcal{K}$ for the instances $\mathcal{I}_G(\rho)$ of rule $\rho$. This classifier is then used to **(i)** estimate a confidence value for the current rule and **(ii)** determine for which facts a rule is likely to be correct.

The classifier $\widetilde{f}$ is of crucial importance, as it should guarantee a high classification accuracy with few evaluated facts. Section 4.1 describes a theoretical connection to maximum set coverage and a greedy baseline that fully relies on the correctness of the similarity measure. In Section 4.2, we relax the constraint on the similarity with a model that combines neural networks and probabilistic Gaussian processes to overcome the rigidity of maximum coverage.

**Algorithm 1** The Colt framework

**Input:** knowledge graph $G$; rule instances $\mathcal{I}_G(\rho)$
**Input:** similarity $\kappa$; budget $B$
**Output:** A classifier $\widetilde{f}$

```
U ← I_G(ρ)                                  ▷ Unlabeled facts
f̃ ← Initialize(U, κ)                        ▷ Initialize the classifier
L ← ∅                                       ▷ Initialize set of labeled facts
for t = 1 ... B do
    x_t ← Select(f̃, U, κ)                   ▷ Select the next fact
    y_t ← f(x_t)                            ▷ User validates x_t
    L ← L ∪ {x_t}                           ▷ Add validated fact to L
    Update(f̃, κ, x_t, y_t)                  ▷ Update the classifier
    U ← U \ {x_t}                           ▷ Remove x_t from U
return f̃
```

## 4.1 Colt-MC: A maximum coverage baseline

We observe a connection between Problem 1 and the *maximum coverage* (MaxCover) problem [24] and devise a greedy baseline for our problem. Given some sets over a universe of elements and a fixed number $B$, the MaxCover problem finds $B$ sets that overall contain the maximum number of elements from the universe.

To appreciate the commonalities between Problem 1 and MaxCover, first note that every fact $x$ identifies a set $\kappa(x, \cdot)$ in which each of its elements has a different similarity weight. Second, the benefit $f(x)$ of each fact $x$ corresponds to the sum of similarities $f(x) = \sum_{x'} \kappa(x, x')$. As such, Problem 1 translates into finding a set of facts that *maximizes the weighted coverage* identified by each fact's similarity set $\kappa(x, \cdot)$. The variant of MaxCover in which every element's weight varies with respect to the set containing the element is called *generalized maximum coverage* (GenMaxCover) [6]. Generalized maximum coverage admits a greedy $(1-1/e)$-approximate solution. Also, the generalized version allows each element's cost to be different, which is not our case as the cost is determined by the number of facts evaluated by a user. We relax the GenMaxCover assumptions and devise a greedy solution for the problem with fixed cost. Such a greedy scheme selects at each iteration the fact that maximizes the *weighted marginal gain*,

$$\Delta(x|L) = \sum_{x' \in \mathcal{I}_G(\rho)} \kappa(x, x') - \max_{x'' \in L} \kappa(x', x'') \tag{1}$$

which quantifies the increment in similarity if the fact $x$ is added to the set $L$. As soon as the set $L$ contains $B$ facts, the algorithm stops. We refer to the greedy baseline in Algorithm 2 as Colt-MC.

The classifier $\widetilde{f}$ is a 1-nearest-neighbor (1-NN) classifier that provides a binary label for each fact. The evaluation of $\widetilde{f}$ on unlabeled facts returns the label of the most similar fact among those evaluated so far. In the end, the approximation of the benefit function $f$ provided by the classifier $\widetilde{f}$ is

$$\widetilde{f}(x) = \begin{cases} f(x) & \text{if } x \in L \\ f(\arg\max_{x' \in L} \kappa(x, x')) & \text{otherwise} \end{cases}$$

One convenient property of Colt-MC is that it ensures maximum quality if all the facts in the instances of the rule are evaluated. We observe this property also empirically in Section 6.3.

**Algorithm 2** Colt-MC

```
function Initialize(U, κ)
    f̃ ← 0                                   ▷ Initialize uniform classification
function Select(f, U, κ)
    return arg max_{x∈U} Δ(x|L)             ▷ See Eq. 1
function Update(f̃, κ, x_t, y_t)
    f̃(x_t) ← y_t
```

## 4.2 Colt-GP: A learning-based solution

The Colt-MC baseline implicitly assumes that both the similarity function $\kappa$ on the facts as well as the user's validation are correct and reliable. To relax these assumptions, we propose a method based on deep kernel learning (DKL) that combines the advantages of neural networks and Gaussian processes (GPs) (Section 4.3).

To estimate the uncertainty value of $\widetilde{f}$, our proxy of the benefit function, we assume that the benefit function at step $t$ is a Bernoulli-distributed random variable sampled from a logit-transformed Gaussian distribution. The Gaussian random variable is represented by a Gaussian process [4, Ch. 6], which models points in space as individual Gaussians related through a kernel function representing the similarity $\kappa$ between facts.

A GP is determined by the mean $\mu$ and covariance matrix, C in which each element $\mathrm{C}_{ij} = \kappa(x_i, x_j)$ represents the similarity between a triple pair $x_i$ and $x_j$. GPs define a distribution over functions where the *prior* of such distribution is a Gaussian $\mathcal{N}(0, \mathrm{C})$ with mean 0 and covariance $\mathrm{C}(\mathcal{I}_G(\rho), \mathcal{I}_G(\rho))$ among all instances in $\mathcal{I}_G(\rho)$. In practice, the prior is never computed. Instead, we are interested in the *posterior* probability $P(x \in \mathcal{K}|L, x)$ of a fact $x$ belonging to the true facts $\mathcal{K}$ knowing the labeled facts $L$. The posterior of a GP is Gaussian; however, in order to classify the facts, the function sampled from the posterior needs to be transformed to return a value between 0 and 1, with high probability. This transformation is made by a *logistic sigmoid* function $s(x) = \frac{1}{1+e^{-x}}$. The final posterior is the expectation of the predictions over the sample of logistic-transformed Gaussian functions:

$$P(x \in \mathcal{K}|L, x) = \int s(f_*) P(f_*|L, x)\, df_* \tag{2}$$

The integral in Eq. 2 is analytically intractable but can be approximated with sampling methods or analytical approximations, such as the Stochastic Variational Inference (SVI) used in Section 4.3.

The choice of GPs for classification has two significant implications. First, a GP is a Bayesian model that allows sampling using the predictions from the posterior. As the choice of the sampling strategy is critical for such models, we analyze multiple choices

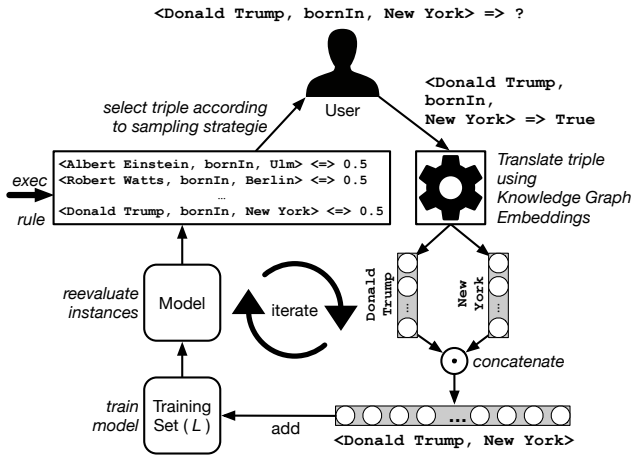


**Figure 1: Overview of our interactive model-building.**

for different sampling strategies in Section 4.4. Second, GPs are non-parametric models; thus, they can easily update their beliefs by using Bayes' theorem on the posterior computation.

We are now ready to describe Colt-GP (Algorithm 3), represented in Figure 1. For the sake of completeness, Algorithm 3 reports on the prior initialization of the random variable (Line 2), even though it is never explicitly computed. The model parameters $(\mu, \mathrm{C})$ are updated using the posterior inference in Eq. 2. Each iteration step samples an unlabeled fact using the Select function, which retrieves an element according to one of the strategies in Section 4.4. The sampled fact is then presented to a user who evaluates its correctness. For example, given the supposed fact ⟨Jon Landau, produced, Avatar⟩, a user verifies whether or not Jon Landau produced the movie Avatar. The labeled triple is then added to the training set $L$.

The posterior calculates the probability of a fact being true, which is estimated by Eq. 2. To convert this probability into a label of 0 or 1 for each fact $x$, $\widetilde{f}(x)$[1] returns 1 if the probability $\geq 0.5$, otherwise 0.

$$\widetilde{f}(x) = \begin{cases} 1 & \text{if } P(x \in \mathcal{K}|L, x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Based on the investigations in Section 5.2, this choice of probability threshold led to an overall good performance.

**Algorithm 3** Colt-GP

1: **function** Initialize($\widetilde{f}$, $U$, $\kappa$)
2: $\widetilde{f} \sim s(\mathcal{N}(0, \mathrm{C}))$ ▷ Gaussian prior, not explicitly computed
3: **function** Select($f$, $U$, $\kappa$)
4: **return** $x_t$ sampled with one strategy in Sec. 4.4
5: **function** Update($\widetilde{f}$, $\kappa$, $x_t$, $y_t$)
6: $\widetilde{f} \leftarrow$ Update the posterior using SVI [40] for Eq. 2

## 4.3 Learning with deep kernels

The inference step for GPs requires the inversion of their covariance matrix, which grows with the number of evaluated facts. The numerical instability of this inversion coupled with modest performance on high dimensional data [10] is detrimental to the final classification quality. To circumvent the GPs' deficiencies, we leverage deep kernel learning (DKL) [39], which scales linearly with the number of evaluated facts and combines the strengths of both neural networks and GPs in one unified model. We use DKL to learn a flexible similarity function, while at the same time incorporating the user's evaluations of the facts. To learn the model parameters and perform posterior inference, we use stochastic variational inference (SVI) [40], which enables the use of non-Gaussian likelihoods.

Equation 3 defines the general model architecture as a composition of functions. It consists of two main components: a set of fully connected network layers ($FC_1$-$FC_4$) and one GP layer applied to the network's output vector. This combination results in a deep probabilistic neural network that meets our requirements.

$$\phi(\mathrm{x}) = GP(FC_3(FC_2(FC_1(\mathrm{x})))) \tag{3}$$

The neural network part of the model ($\phi$) consists of three fully connected layers $FC_n(\mathrm{x}) = \sigma(\boldsymbol{W}\boldsymbol{x} + b)$, where $W$ and $b$ correspond to the weight and bias terms, and $\sigma$ denotes the activation function, which in our case is set to ReLU. The network part of the model effectively performs a dimensionality reduction, by transforming $n$-dimensional input vectors $\mathrm{x} = (x_1 \ldots x_n)$ into a low-dimensional vector representation, which is then passed to the GP layer.

The GP part of the model is connected to the output layer of the network and consists of $j$ Gaussian Processes $g_1 \ldots g_j$, each having their own corresponding kernel $k_1 \ldots k_j$ operating on a subset of the vector generated by the network. When selecting the number of GPs in this layer, we follow Wilson et al. [40] and use one GP for each dimension in the output vector ($j = 4$). To obtain the final result, the individual GPs are first additively combined and then transformed by an observation model, which in our case is a Bernoulli likelihood, to receive the final result $Y$.

## 4.4 Sampling strategies

As stated in Section 4.2, the sampling strategy is at the heart of our Colt-GP algorithm, as it is responsible for the incremental selection of new facts. A good sampling strategy aims to select training examples of high information density that are likely to improve classifier performance when used for training. In the following, we present the sampling strategies considered in our experiments.

**Random.** Random sampling is the simplest sampling technique: it randomly selects training examples from the data and adds them to the training dataset, ignoring any prior knowledge.

**Maximum uncertainty.** Maximum uncertainty sampling (MAXU) uses the model itself to assess how valuable the labeling of each data point is in terms of confidence gain [20]. The sampling uses the model's uncertainty on unlabeled data points: the data points for which the classifier is most uncertain are those closest to its decision boundary. The incremental labeling of the data points with maximum uncertainty leads to the refinement of the decision boundary. In binary classification, the binary entropy measure determines the classifier's uncertainty with respect to its classification where p determines the probability of X = 1 and 1-p the probability of X = 0:

$$H(X) = -p \log_2(p) - (1-p) \log_2(1-p)$$

[1] With a little abuse of notation $\widetilde{f}$ indicates both the posterior and the classifier.

$$\text{with } X = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}$$

As mentioned in [29], a disadvantage of this strategy can be its sensitivity to outliers, which is tackled by GP-UCB.

**GP-UCB.** The Gaussian process upper confidence bound algorithm (GP-UCB) by Srinivas et al. [34] defines a sampling criterion that aims at finding a trade-off between exploring the function space and exploiting maximal function areas. To this end, the authors propose the following sampling strategy:

$$x_t = \arg\max_{x \in U} \mu_{t-1}(x) + \sqrt{\beta_t}\sigma_{t-1}(x)$$

where $U$ denotes the unlabeled facts as defined in Algorithm 1. The exclusive selection of elements $x \in U$ where the model has either a high variance ($\sigma_{t-1}$) or a high expected reward ($\mu_{t-1}$) cannot be regarded as the optimal strategy, as useful information is neglected and the model's ability to generalize is compromised. To overcome this issue, the proposed GP-UCB sampling strategy strikes a balance between *exploration* by selecting elements where the model has a certain degree of uncertainty (large $\sigma_{t-1}$), and *exploitation* by choosing elements with high belief (large $\mu_{t-1}$). By simultaneously optimizing both criteria, the strategy seeks to achieve an equilibrium between exploration and exploitation.

The parameter $\beta_t$ trades-off exploration and exploitation. A large $\beta_t$ corresponds to more exploration, while a small $\beta_t$ leads to more exploitation. The GP-UCB strategy ensures a logarithmic growth of the regret, the deviation in quality with respect to the optimal sampling, by setting $\beta_t = 2\log(|\mathcal{I}_G(\rho)|t^2\pi^2/6\delta)$ with $\delta \in (0,1)$. In other words, the regret on the evaluations tends to 0 when the number of validated facts tends to infinity.

According to the above formula, $\beta_t$ starts at a relatively small value, which encourages it to be more exploitative. Along with the increasing number of sampling steps $t$, $\beta_t$ also increases, making it more exploratory. We use this policy to update $\beta_t$ during our experiments.

## 5 COMPUTING SIMILARITIES

The choice of an adequate similarity measure $\kappa$ depends on the specifics of the data and the application domain. We avoid the need to design an ad-hoc similarity measure by proposing a flexible one based on *knowledge graph embeddings* [37] to represent a graph in a low-dimensional space. These methods implicitly capture structural and semantic information and are agnostic to the application domain. Although errors and missing data in the knowledge graph affect the quality of the generated embeddings [28], they constitute a solid starting point. The design of such embedding methods is outside of the scope of this work, thus we refer to [37] for a comprehensive survey on these methods.

For our purposes, we employ HypER [3], an expressive and fast embedding method. For each fact $\langle s, r, o\rangle$ with subject $s$, object $o$, and relationship $r$, HypER computes 200-dimensional vectors $\mathrm{s}$, $\mathrm{o}$, and $\mathrm{r}$. HypER maximizes the likelihood of the existence of a fact in the knowledge graph. Additionally, the method aggregates information from the subject and the relationship employing convolution operators. The object $\mathrm{o}$ in the vector space is then computed as a non-linear transformation of the subject and relationship vector.

These embeddings are calculated for all subjects, relationships, and objects in a knowledge graph before any rule is processed. The final embedding $\mathrm{x}$ of a fact $x = \langle s, r, o\rangle$ is the 400-dimensional vector concatenation $\mathrm{x} = \mathrm{s} \circ \mathrm{o}$ of the subject and object vectors. Since we consider one rule at a time, the relationship $r$ in the rule body always remains the same and can be omitted.

Colt builds on HypER embeddings, but any choice of superior embedding technique would only enhance Colt's performance.

**Colt-MC:** The similarity score between fact $x = \langle s, r, o\rangle$ and fact $x' = \langle s', r', o'\rangle$ is the cosine similarity among the fact vectors:

$$k_{cos}(x, x') = \frac{\mathrm{x}\cdot\mathrm{x}'}{\|\mathrm{x}\|\|\mathrm{x}'\|} = \frac{(\mathrm{s}\circ\mathrm{o})\cdot(\mathrm{s}'\circ\mathrm{o}')}{\|\mathrm{s}\circ\mathrm{o}\|\|\mathrm{s}'\circ\mathrm{o}'\|}$$

**Colt-GP:** The Gaussian processes included in Colt-GP are kernel methods, which naturally incorporate similarity scores. The default kernel for Gaussian processes is the RBF kernel:

$$k_{RBF}(\mathrm{x}, \mathrm{x}') = \exp\left(-\frac{||\mathrm{x} - \mathrm{x}'||}{2\sigma^2}\right)$$

The RBF kernel provides smoothness as the similarity exponentially decreases with an increase of the Euclidean distance. The parameter $\sigma$ is a learned parameter in the optimization. GPs with RBF kernels implicitly project the points into an infinite-dimensional space so that the GP layer represents a hidden layer with an infinite number of neurons, thereby significantly increasing the model's flexibility. The transformation induced by deep kernel learning (DKL) in Section 4.3 further modifies the embedding space through a parametric non-linear function $g$. The final kernel is:

$$k_{DKL}(\mathrm{x}, \mathrm{x}') = \exp\left(-\frac{||g(\mathrm{x}) - g(\mathrm{x}')||}{2\sigma^2}\right)$$

## 6 EXPERIMENTAL EVALUATION

In the experimental evaluation, we answer the following questions. **Q1**: Is Colt-GP able to learn the truth value for each rule (Sec. 6.1)? **Q2**: How do different sampling strategies affect the training process (Sec. 6.2)? **Q3**: How does Colt-GP fare in comparison to our strong baseline Colt-MC and active learning (Sec. 6.3)? **Q4**: What is the relationship between user interactions, real confidence, and rule support (Sec. 6.4)?

**Datasets.** We use YAGO [35], an open-source knowledge base, to derive logical rules and graph embeddings. At the time of writing, YAGO2 [2] comprises 948 358 triples, 36 relationship types, and 470 485 entities. We ran RuDiK [27] and AMIE [13] with default parameters on YAGO to obtain positive and negative rules. Out of 1 517 mined rules, 928 produced more than 5 500 triples. Each rule can be thought of as a query asking for triples in the KG that satisfy the rule's left-hand side.

We *manually annotated* the instances of 26 rules (22 positive; 4 negative) out of the 589 rules that yielded fewer than 5 500 instances. The decisive criteria for the selection of these rules included the number of triples that had to be annotated as well as the variety of predicates involved. Also, we selected a majority of rules with three atoms in their body to focus on use cases where analyzing and validating is harder for users. For negative rules, where the

[2] http://resources.mpi-inf.mpg.de/yago-naga/amie/data/yago2/yago2core_facts.clean.notypes.tsv.7z

| ID | Rule | #KG | #correct | #total | type |
|---|---|---|---|---|---|
| 1 | dealsWith(a, $v_0$) ∧ isLeaderOf($v_1$, b) ∧ wasBornIn($v_1$, $v_0$) ⇒ dealsWith(a, b) | 9 | 24 | 99 | pos |
| 2 | dealsWith($v_0$, b) ∧ isCitizenOf($v_1$, a) ∧ isLeaderOf($v_1$, $v_0$) ⇒ dealsWith(a, b) | 42 | 42 | 42 | pos |
| 3 | dealsWith($v_0$, b) ∧ isCitizenOf($v_1$, a) ∧ isLocatedIn($v_1$, $v_0$) ⇒ dealsWith(a, b) | 108 | 134 | 134 | pos |
| 4 | isCitizenOf($v_0$, b) ∧ livesIn($v_0$, a) ⇒ dealsWith(a, b) | 18 | 664 | 734 | pos |
| 5 | diedIn(a, $v_0$) ∧ isKnownFor($v_1$, $v_0$) ∧ livesIn($v_1$, b) ⇒ diedIn(a, b) | 808 | 815 | 838 | pos |
| 6 | diedIn(a, $v_0$) ∧ isLeaderOf($v_1$, $v_0$) ∧ livesIn($v_1$, b) ⇒ diedIn(a, b) | 2,143 | 3,135 | 4,292 | pos |
| 7 | actedIn(a, b) ∧ created(a, b) ⇒ directed(a, b) | 384 | 388 | 1,003 | pos |
| 8 | actedIn($v_0$, b) ∧ created(a, b) ∧ directed($v_0$, b) ⇒ directed(a, b) | 421 | 414 | 804 | pos |
| 9 | hasWonPrize(a, $v_0$) ∧ hasWonPrize(b, $v_0$) ∧ hasAcademicAdvisor(b, a) ⇒ hasAcademicAdvisor(a, b) | 85 | 85 | 85 | neg |
| 10 | influences(a, b) ⇒ hasAcademicAdvisor(a, b) | 1,000 | 1,000 | 1,000 | neg |
| 11 | isCitizenOf($v_0$, a) ∧ livesIn($v_0$, b) ⇒ hasCapital(a, b) | 43 | 50 | 734 | pos |
| 12 | dealsWith(a, $v_0$) ∧ hasCurrency($v_0$, b) ⇒ hasCurrency(a, b) | 7 | 10 | 293 | pos |
| 13 | hasCapital($v_0$, $v_1$) ∧ hasCurrency($v_0$, b) ∧ isLocatedIn($v_1$, a) ⇒ hasCurrency(a, b) | 18 | 46 | 49 | pos |
| 14 | dealsWith($v_0$, a) ∧ hasOfficialLanguage($v_0$, b) ⇒ hasOfficialLanguage(a, b) | 52 | 72 | 668 | pos |
| 15 | hasOfficialLanguage($v_0$, b) ∧ isLocatedIn($v_0$, a) ⇒ hasOfficialLanguage(a, b) | 18 | 62 | 126 | pos |
| 16 | hasOfficialLanguage($v_0$, b) ∧ isLocatedIn($v_1$, a) ∧ livesIn($v_1$, $v_0$) ⇒ hasOfficialLanguage(a, b) | 20 | 52 | 65 | pos |
| 17 | hasOfficialLanguage($v_0$, b) ∧ isLocatedIn($v_1$, $v_0$) ∧ livesIn($v_1$, a) ⇒ hasOfficialLanguage(a, b) | 20 | 16 | 40 | pos |
| 18 | influences(a, $v_0$) ∧ isCitizenOf($v_0$, b) ⇒ isCitizenOf(a, b) | 146 | 547 | 1,382 | pos |
| 19 | hasChild(a, b) ⇒ isMarriedTo(a, b) | 991 | 990 | 1,000 | neg |
| 20 | isLocatedIn($v_0$, b) ∧ livesIn(a, $v_0$) ⇒ isPoliticianOf(a, b) | 130 | 2,257 | 5,269 | pos |
| 21 | isCitizenOf($v_0$, b) ∧ isLeaderOf($v_0$, $v_1$) ∧ livesIn(a, $v_1$) ⇒ livesIn(a, b) | 416 | 863 | 882 | pos |
| 22 | isMarriedTo(a, $v_0$) ∧ livesIn($v_0$, b) ⇒ livesIn(a, b) | 181 | 442 | 537 | pos |
| 23 | actedIn(a, b) ∧ created(a, b) ⇒ produced(a, b) | 207 | 261 | 1,003 | pos |
| 23 | actedIn($v_0$, b) ∧ directed(a, b) ∧ produced($v_0$, b) ⇒ produced(a, b) | 246 | 281 | 702 | pos |
| 25 | hasAcademicAdvisor($v_0$, a) ∧ worksAt($v_0$, b) ⇒ worksAt(a, b) | 47 | 163 | 543 | pos |
| 26 | actedIn(a, b) ⇒ wroteMusicFor(a, b) | 1000 | 998 | 1,000 | neg |

**Table 1: Generated rules and their statistics: number of KG facts that satisfy the rule, number of correct facts, total number of facts, type of the rule (colors are for ease of reading).**

execution in most cases yielded a very high number of triples, we randomly selected 1,000 triples for annotation. To cover a range of different output sizes, we annotated rules with a small output, as few as 40 triples, up to rules with 5 269 triples. Similarly, we annotated rules that are mostly right, rules that are nearly always wrong, and rules with mixed confidence. For the listed 26 rules, *we manually annotated a total of 23 324 triples*, of which 5 524 corrected errors and missing facts in the underlying KG. For reviewing purposes, we make the annotated dataset available under an anonymized and untracked link (http://e.pc.cd/weoctalK). To our knowledge, this is the largest available dataset of annotated rules.

**Algorithms.** In addition to Colt-MC (Section 4.1), we evaluate the Colt-GP algorithm (Section 4.2) against two active learning baselines. Both baselines follow the basic active learning procedure as described in [31]. While the first baseline employs a linear regression model (AL-LogReg), the second uses a more expressive SVM with RBF kernel (AL-SVM) [30]. Following the results of Section 6.2, we choose maximum uncertainty as the selection strategy for both active learning approaches.

**Training details.** To train the model presented in Section 4.3, we use stochastic variational inference (SVI) and gradient descent. We employ the ADAM optimizer with default parameters $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$. We jointly train the weights of the neural network and the parameters of the GPs with batch size 32 until convergence or a maximum of 1000 epochs.

**Runtime.** Recall that this work focuses on the study of methods for quality assessment and confidence estimation of logic rules using a few human-validated rule instances. While a thorough evaluation of the runtime deviates from the main focus of this work, we observe that our methods run in real-time in less than 10 seconds on 20–100 evaluated instances. This result, obtained on a commodity machine, endorses the use of the Colt framework and its most expressive Colt-GP method on production systems.

## 6.1 Q1: Learning rule characteristics

Here we aim to show that Colt-GP is able to learn the characteristics of a rule from the instances it generates. We divide the annotated instances into training and test sets, using a ratio of 70 to 30. We train Colt-GP on the training and test the ability to predict whether each rule's instance holds for the unseen instances in the test set. This scenario primarily aims to test the learned model's generalization capabilities and, at the same time, represents the most challenging classification scenario.

Figure 2 summarizes the results by showing the ROC curves for the annotated rules as well as the mean and variance of each of the rule's ROC curves. The ROC curves of the individual rules and, hence, also the average ROC curve are all significantly above the dashed line representing the behavior of a random classifier. Colt-GP attains 89.78% average AUC, which testifies the correctness of the classifier in predicting unknown facts. Even the lower end of

the variance line attains 79.38% AUC, which is significantly above the performance of a random classifier. We conclude that Colt-GP can learn a meaningful model by using individual facts.

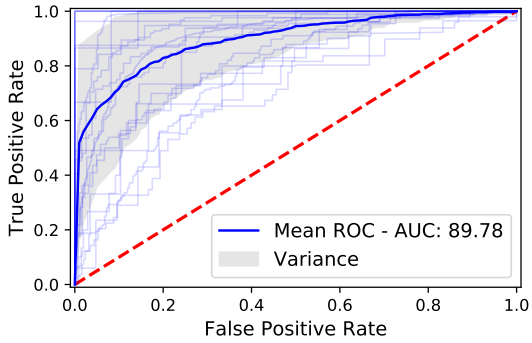


**Figure 2: ROC curve for classifying relations**

## 6.2 Q2: Selecting the sampling strategy

The sampling strategies are responsible for collecting the data used for training. The purpose of this experiment is to investigate how the performance of our approach depends on the sampling strategy.

**Experimental setup.** With $\mathcal{I}_G(\rho)$ being the set of all instances created by the rule $\rho$, we follow Algorithm 1, as specified in Section 4. To test the trained models, we make a prediction on all instances in $\mathcal{I}_G(\rho)$, which includes the instances added to the training set $L$. This creates a realistic evaluation scenario where instances used for training also remain in the KG. Without limiting the budget $B$, $L$ eventually contains the same elements as $\mathcal{I}_G(\rho)$. This setup also helps assess the information capacity of the proposed model. If a model cannot achieve an F1-score of 100% when training and predicting on identical data, it is likely that its capacity is not high enough to perfectly separate the classification space. We examine this scenario in more detail in Section 6.3.

**Evaluation.** Figure 3 shows how the different sampling strategies affect precision, recall, and F-measure. As expected, the random selection of training instances proves to be the worst strategy. Although its precision initially increases to 78%, it then decreases and remains, on average, 3.93% and 6.22% below the precision value of GP-UCB and max-uncertainty (MAXU), respectively. In terms of F-measure, the random sampling strategy lags 4.42% and 6.72% behind the GP-UCB and MAXU strategies, regardless of the percentage of training instances. GP-UCB takes up to 6% of training instances to achieve a competitive precision value of 77.44% and remains below the precision value of MAXU throughout the experiment. Overall, the MAXU strategy provides the best results in F-measure, outperforming the other two strategies at all times.

GP-UCB's inability to achieve a better performance than MAXU can be attributed to the $\beta_t$ parameter discussed in Section 4.4. While the formula in [34] for selecting $\beta_t$ provides a theoretical bound on prediction quality, it appears to be too conservative for our problem. As a result, GP-UCB makes too conservative choices and is unable to outperform MAXU due to an unfavorable trade-off between exploration and exploitation. As the MAXU strategy achieves 75% F1 with only 5% training instances, we employ this strategy in all subsequent experiments.

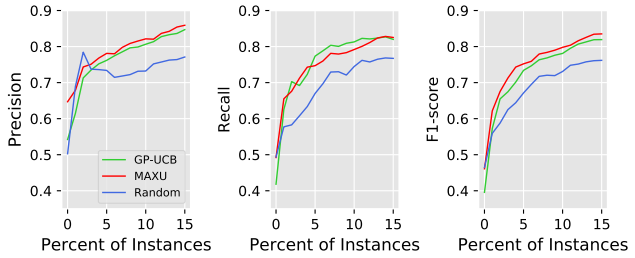


**Figure 3: Impact of sampling strategies on predictions.**

## 6.3 Q3: Model performance

This experiment analyzes the behavior of Colt-GP w.r.t. the amount of training data and compares the performance to our Colt-MC and two active learning baselines. We aim to answer question (**Q3**) on the necessity of our learning strategy as opposed to active learning methods and direct use of similarities between KG embeddings in Colt-MC.

Figure 4 shows precision, recall, and F1-score curves of all algorithms for an increasing amount of training data. AL-LogReg and AL-SVM rapidly fall behind the Colt algorithms. Although AL-SVM, leveraging its nonlinear RBF kernel, provides better results than AL-LogReg, neither model possesses sufficient capacity to meet the complexity of the problem. This is reflected in the performance metrics of Figure 4, where after using 15% (AL-SVM) and 20% (AL-LogReg) of the training data, improvements begin to stagnate for both models, so that their respective F-measures fluctuate around 84% and 80%. In contrast, Colt-GP and Colt-MC outperform the active learning approaches and continue to improve as training data increases.

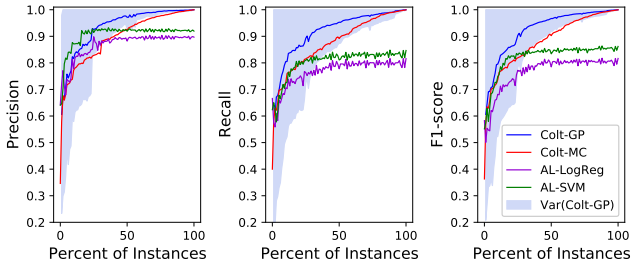


**Figure 4: Colt compared to active learning.**

While Colt-GP exceeds Colt-MC in all plots, it exhibits a faster increase than Colt-MC. Regarding the F1-score, Colt-GP grows on average 1.54% faster than Colt-MC between 5-8% of used training instances and slows down to 0.85% on 22-25% range. In contrast, Colt-MC increases almost linearly between 12% and 100% F1-score.

Variance in prediction for Colt-GP decreases with the amount of training data, as the model becomes increasingly confident. In terms of F-measure, Colt-GP achieves 100% when all training data is used, proving that the model is expressive enough to gain the necessary knowledge. Using only 5% and 10% of the training data, Colt-GP achieves 75% and 80% F-measure. We measured the biggest improvement over Colt-MC when using 31% of the training data, resulting in an F-measure of 92.8% compared to 80.2% produced by Colt-MC. This represents an improvement of 12.6% in F-measure.

Looking at the evaluation of all four approaches, we conclude that it is neither sufficient to solely rely on the similarities of knowledge graph embedding nor to apply simple active learning techniques to solve the problem adequately.

## 6.4 Q4: Rule confidence estimation

We compare the rule confidence estimation of Colt-GP with the data-driven standard confidence measure (*SCF*) as used by AMIE [13]. It is defined as $\frac{\#KG}{\#total}$ in Table 1. We first determine the correct confidence value for each rule based on its annotated facts and use it to calculate the average estimation error over all rules. The continuous red line (SCF) in Figure 5 shows that the average *SCF* estimate is 19.6% away from the correct confidence value. To calculate the estimation error of Colt-GP, we use an increasing number of instances to train a model for each rule and report the average error of these models. For this calculation, we only use rules from Table 1, that produce at least 100 instances. The blue line (Colt-GP) in Figure 5 shows that, from the beginning, Colt-GP estimates the average rule confidence more accurately than AMIE's *SCF* estimate. As a result, 10 to 20 training instances are sufficient to achieve an average estimated confidence error of 12.9 to 10.8 percent, which corresponds to an improvement of 6.7 and 8.8 percentage points over the *SCF* estimate. As the number of training data increases, the estimation error decreases and reaches its lowest value of 4.5 percentage points when using 100 training instances. Figure 5 also shows that, although the estimated error between 50-100 training instances improves by only 1.3 percentage points, the variance decreases by 3.5%, indicating that Colt-GP gains more confidence in its predictions.

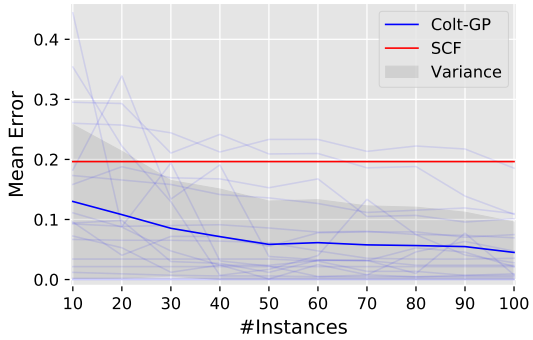


**Figure 5: Error reduction in rule confidence estimation using an increasing number of training instances. Light blue lines indicate individual rules error.**

## 7 CONCLUSION & FUTURE WORK

We introduced Colt, a few-shot rule-based knowledge validation framework for interactive quality assessment of rules. Our algorithm, Colt-GP, uses Gaussian processes and deep kernel learning to interactively learn classifiers on rules. Colt-GP benefits from knowledge graph embeddings and user feedback to remedy data quality issues in KGs. We juxtapose Colt-GP with a strong baseline based on maximum coverage. Colt-GP attains 10% error in confidence estimation with only 20 user-validated facts and 75% prediction quality in rule validation with only 5% labeled instances. We release the code and dataset of 26 rules and 23 000 labeled facts.

We plan to extend our results to efficiently learn the confidence of multiple rules simultaneously. Furthermore, the Colt framework could generate refined rules on the facts validated by the model on each rule to accomplish a fine-grained maintenance of KGs. We note that Colt's focus on rule-driven validation does not prevent its adoption to triples returned by other query mechanisms.

## REFERENCES

[1] N. Ahmadi, J. Lee, P. Papotti, and M. Saeed. Explainable fact checking with probabilistic answer set programming. In *Conference for Truth and Trust Online (TTO)*, 2019.
[2] A. Arioua and A. Bonifati. User-guided repairing of inconsistent knowledge bases. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, 2018.
[3] I. Balazevic, C. Allen, and T. M. Hospedales. Hypernetwork knowledge graph embeddings. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 553–565, 2019.
[4] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.
[5] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pages 2787–2795, 2013.
[6] R. Cohen and L. Katzir. The generalized maximum coverage problem. *Information Processing Letters*, 108(1):15–22, 2008.
[7] L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, Mar 1999.
[8] O. Deshpande, D. S. Lamba, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan. Building, maintaining, and using knowledge bases: a report from the trenches. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 1209–1220, 2013.
[9] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 517–528. ACM, 2014.
[10] J. Djolonga, A. Krause, and V. Cevher. High-dimensional Gaussian process bandits. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pages 1025–1033, 2013.
[11] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *Proceedings of the International Conference on Knowledge discovery and data mining (SIGKDD)*, pages 601–610, 2014.
[12] W. Fan, P. Lu, C. Tian, and J. Zhou. Deducing certain fixes to graphs. *PVLDB*, 12(7):752–765, 2019.
[13] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB Journal*, 24(6):707–730, 2015.
[14] L. A. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 413–422, 2013.
[15] G. A. Gesese, R. Biswas, and H. Sack. A comprehensive survey of knowledge graph embeddings with literals: Techniques and applications. In *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG)*, pages 31–40, 2019.
[16] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 192–202, 2016.
[17] A. Heidari, J. McGrath, I. F. Ilyas, and T. Rekatsinas. HoloDetect: Few-shot learning for error detection. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 829–846, 2019.
[18] V. T. Ho, D. Stepanova, M. H. Gad-Elrab, E. Kharlamov, and G. Weikum. Rule learning from knowledge graphs guided by embedding models. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 72–90, 2018.
[19] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, (ACL)*, pages 687–696, 2015.
[20] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the International Conference on Information retrieval (SIGIR)*, pages 3–12, 1994.
[21] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.
[22] M. Lissandrini, D. Mottin, D. Papadimitriou, T. Palpanas, and Y. Velegrakis. Unleashing the power of information graphs. *SIGMOD Record*, 43(4), 2014.

[23] M. Mahdavi, Z. Abedjan, R. C. Fernandez, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Raha: A configuration-free error detection system. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 865–882, 2019.

[24] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical programming*, 14(1):265–294, 1978.

[25] M. Nickel, L. Rosasco, and T. A. Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1955–1961, 2016.

[26] M. Nickel, V. Tresp, and H. Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 809–816, 2011.

[27] S. Ortona, V. V. Meduri, and P. Papotti. Robust discovery of positive and negative rules in knowledge bases. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 1168–1179, 2018.

[28] J. Pujara, E. Augustine, and L. Getoor. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1751–1756, 2017.

[29] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 441–448, 2001.

[30] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 839–846, 2000.

[31] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[32] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 29(1):17–37, 2016.

[33] J. Shin, S. Wu, F. Wang, C. D. Sa, C. Zhang, and C. Ré. Incremental knowledge base construction using DeepDive. *PVLDB*, 8(11):1310–1321, 2015.

[34] N. Srinivas, A. Krause, S. Kakade, and M. W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1015–1022, 2010.

[35] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 697–706, 2007.

[36] H. P. Vanchinathan, A. Marfurt, C.-A. Robelin, D. Kossmann, and A. Krause. Discovering valuable items from massive data. In *Proceedings of the International Conference on Knowledge discovery and data mining (SIGKDD)*, pages 1195–1204, 2015.

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

[38] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1112–1119, 2014.

[39] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 370–378, 2016.

[40] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Stochastic variational deep kernel learning. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pages 2586–2594, 2016.

[41] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 2659–2665, 2016.

[42] P. Xu and D. Barbosa. Connecting language and knowledge with heterogeneous representations for neural relation extraction. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT)*, pages 3201–3206, 2019.

[43] W. Zhang, B. Paudel, L. Wang, J. Chen, H. Zhu, W. Zhang, A. Bernstein, and H. Chen. Iteratively learning embeddings and rules for knowledge graph reasoning. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 2366–2377, 2019.