Explainable Benchmarking through the Lense of Concept Learning

Quannian Zhang
Faculty of Computer Science, Electrical Engineering and Mathematics
Data Science Group (DICE), Heinz Nixdorf Institute, Paderborn University
Paderborn, North Rhine-Westphalia
Germany
quannian@mail.uni-paderborn.de

Michael Röder
Faculty of Computer Science, Electrical Engineering and Mathematics
Data Science Group (DICE), Heinz Nixdorf Institute, Paderborn University
Paderborn, North Rhine-Westphalia
Germany
michael.roeder@uni-paderborn.de

Nikit Srivastava
Faculty of Computer Science, Electrical Engineering and Mathematics
Data Science Group (DICE), Heinz Nixdorf Institute, Paderborn University
Paderborn, North Rhine-Westphalia
Germany
nikit.srivastava@uni-paderborn.de

N'Dah Jean Kouagou
Faculty of Computer Science, Electrical Engineering and Mathematics
Data Science Group (DICE), Heinz Nixdorf Institute, Paderborn University
Paderborn, North Rhine-Westphalia
Germany
ndah.jean.kouagou@upb.de

Axel-Cyrille Ngonga Ngomo
Faculty of Computer Science, Electrical Engineering and Mathematics
Data Science Group (DICE), Heinz Nixdorf Institute, Paderborn University
Paderborn, North Rhine-Westphalia
Germany
axel.ngonga@upb.de

Abstract

Evaluating competing systems in a comparable way, i.e., benchmarking them, is an undeniable pillar of the scientific method. However, system performance is often summarized via a small number of metrics. The analysis of the evaluation details and the derivation of insights for further development or use remains a tedious manual task with often biased results. Thus, this paper argues for a new type of benchmarking, which is dubbed explainable benchmarking. The aim of explainable benchmarking approaches is to automatically generate explanations for the performance of systems in a benchmark. We provide a first instantiation of this paradigm for knowledge-graph-based question answering systems. We compute explanations by using a novel concept learning approach developed for large knowledge graphs called PruneCEL. Our evaluation shows that PruneCEL outperforms state-of-the-art concept learners on the task of explainable benchmarking by up to 0.55 points F1 measure. A task-driven user study with 41 participants shows that in 80% of the cases, the majority of participants can accurately predict the behavior of a system based on our explanations. Our code and data are available at https://github.com/dice-group/PruneCEL/tree/K-cap2025.

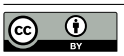
CCS Concepts

• **General and reference** → **Evaluation**; • **Information systems** → **Question answering**; • **Theory of computation** → *Description logics*.

Keywords

Benchmarks, Explainability and interpretability, Description logics

ACM Reference Format:
Quannian Zhang, Michael Röder, Nikit Srivastava, N'Dah Jean Kouagou, and Axel-Cyrille Ngonga Ngomo. 2025. Explainable Benchmarking through the Lense of Concept Learning. In *Knowledge Capture Conference 2025 (K-CAP '25), December 10–12, 2025, Dayton, OH, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3731443.3771359



K-CAP '25, Dayton, OH, USA

ACM ISBN 979-8-4007-1867-0/2025/12
https://doi.org/10.1145/3731443.3771359

1 Introduction

Comparable benchmarks are key for the improvement of solutions across disciplines with quantifiable results. This insight has led to the development of a multitude of benchmarking frameworks and online leaderboards based thereupon in recent years. Examples include the SEALS platform for link discovery [15], GERBIL QA for question answering [27], HOBBIT for big linked data applications [19], and HuggingFace's Open LLM Leaderboard[1]. However, benchmarks are of little use if the results they generate do not lead to actionable insights. Hence, some benchmarking frameworks provide insights into evaluation results [11, 13, 26], e.g., by means of correlation analyses. While these approaches give insights within

[1] https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard#/

their respective predefined dimensions, none of these approaches goes beyond that paradigm.

Explanation theory [14] suggests that actionable insights must enable the explainee[2] to perform better at system-relevant tasks, i.e., using and developing the benchmarked system in the case of benchmarking. Hence, we argue for the need for explainable benchmarking approaches that give human-understandable insights into when a system performs well and into when it is subpar. We instantiate this novel paradigm with the two following contributions:

(1) We propose an approach for generating explanations for benchmarking results. Internally, our approach builds a structured representation of the benchmark data and uses concept learning to generate an explanation that is able to describe cases in which the benchmarked system performs well, separating them from cases in which it performs subpar. We evaluate our approach with benchmark results from the area of knowledge-graph-based question answering (QA).
(2) We propose a new concept learning algorithm dubbed PruneCEL that achieves scalability by pruning the space in which it searches for concepts. Our evaluation shows that PruneCEL is able to outperform several state-of-the-art concept learning approaches on large knowledge bases.

2 Related-work

2.1 QA Benchmarking

Over the past decade, numerous QA benchmarks have been introduced, e.g., by the benchmarking series QALD [29], LcQUAD [17], and RuBQ [34].[3] These benchmarks either come with evaluation scripts or can be used in combination with benchmarking platforms like GERBIL QA [27]. However, similar to other research areas, such evaluations typically provide a summary of key performance indicators (KPIs), e.g., the F1 measure of different evaluated QA systems on each dataset. There are some attempts to give deeper insights into reasons why systems might perform good or subpar during an evaluation. For example, if provided with additional data GERBIL QA analyzes the system's performance in preprocessing steps of a typical QA pipeline, e.g., the identification of named entities or properties in the given question [27]. While these tools can help run evaluations and analyze the results, their analysis is bound to pre-defined features of the single tasks a system has to fulfill. To the best of our knowledge, we are the first to propose a generic approach for generating explanations, which avoids this limitation by transforming information available about the benchmarking process into a structured, generic representation and applying concept learning, i.e., symbolic machine learning, to learn an explanation for the benchmarking results.

While existing approaches like QED [18] utilize symbolic machine learning for post-hoc explainability in QA, they are limited to explaining individual answer choices. Our work presents a novel approach that instead explains a system's overall evaluation results on a dataset. By transforming benchmarking information into a structured representation and applying concept learning, we move beyond explaining what a system answers to explaining where its strengths and blindspots lie.

[2]That is, the entity receiving the insights.
[3]https://qald.aksw.org/, https://github.com/AskNowQA/LC-QuAD, and https://github.com/vladislavneon/RuBQ.

2.2 Concept Learning

The application of concept learning—i.e., the task of describing a set of positive examples, separating it from a set of negative examples based on a knowledge base using description logics—as part of our approach led to the development of a new concept learning algorithm called PruneCEL. Previous works use inductive logic programming with refinement operators [12]. These approaches start with the $\top$ concept and further refine it using a refinement operator ρ to generate new concepts. These newly created concepts are scored with respect to a scoring function, e.g., F1 measure, and the expression with the highest score that has not yet been refined before is then chosen to be refined further. This is repeated until either one of the found concepts achieves the maximum possible score or a given budget in the form of runtime or iterations has been consumed. Figure 3 shows an example of the search tree that is created. While this approach seems simple, it has been proven to be guaranteed to find a perfect solution for a given learning problem if (1) such a solution exists, and (2) the refinement operator is weakly complete, i.e., is able to generate any concept starting from $\top$ [37]. However, the search may take a very long time since the search space itself is infinite, i.e., every concept created by a weakly complete refinement operator can be further refined to create new expressions [37]. Hence, several optimizations have been proposed. For example, the latest version of CELOE achieves a reduced runtime by storing the knowledge base $\mathcal{K}$ in a triple store and collecting the counts necessary for scoring using SPARQL [31]. The refinement operator of DL-Foil [21] does not generate all refinements but a random subset, reducing the number of generated concepts. In a similar way, Rizzo et al. [10] use a refinement operator with random sampling to generate terminological decision trees and combine multiple of them, similar to a random forest. Another optimization is to calculate an upper bound of the achievable performance of the refinements of a concept and discard concepts with an upper bound lower than the quality of the best solution found so far [16]. In contrast to the previous approaches, Drill [2] does not rely on a pre-defined scoring function to choose the expression that should be further refined. Instead, it uses deep Q-learning to train an agent that makes this decision. Our approach PruneCEL shares the usage of a downward length-based refinement operator and a pre-defined scoring function with some of these approaches. However, our approach prunes the search space by avoiding the generation of concepts that lead to a low performance. This pruning allows us to achieve results even on larger knowledge bases, without any pre-training.

Not all concept learning approaches rely on a refinement operator. EvoLearner [33] uses biased random walks on the knowledge base to create a start population of concepts. After that, it uses an evolutionary algorithm to create new concepts from this population. NCES [23] tackles concept learning as a translation problem. It synthesizes a solution by using sets of positive and negative examples as input to a neural network.

3 Preliminaries

3.1 Description Logics

Description logics are a family of languages for knowledge representation [1]. Within this article, we focus on the description logic $\mathcal{ALC}$. Table 1 below defines the syntax and semantics of the $\mathcal{ALC}$ constructs.

Table 1: Syntax & semantics for $\mathcal{ALC}$ concepts [23]. $\mathcal{I}$ stands for an interpretation with domain $\Delta^{\mathcal{I}}$.

Construct	Syntax	Semantics
Atomic concept	C	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Atomic role	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Top concept	$\top$	$\Delta^{\mathcal{I}}$
Bottom concept	$\bot$	$\emptyset$
Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Existential restriction	$\exists r.C$	$\{x \mid \exists b : (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
Universal restriction	$\forall r.C$	$\{a \mid \forall b : (a,b) \in r^{\mathcal{I}} \implies b \in C^{\mathcal{I}}\}$

3.2 Refinement Operators

A quasi-ordering is a reflexive and transitive relation [36]. Let $(\mathcal{C}, \preccurlyeq)$ be a quasi-ordered space. A downward refinement operator ρ in such a space is a mapping from $\mathcal{C}$ to $2^{\mathcal{C}}$ such that $\forall C \in \mathcal{C} : D \in \rho(C) \implies D \preccurlyeq C$. D is called a specialisation of C [36].

Two quasi-orderings are often used in concept learning. Some of the earliest approaches rely on the subsumption relation $\sqsubseteq$ [12]. More recent approaches (including ours) use the length of concepts, which is defined recursively for $\mathcal{ALC}$ concepts as follows [22]:

$$l(C) = \begin{cases} 1 & \text{if } C \in N_C \cup \{\top, \bot\}, \\ 1 + l(X) & \text{if } C = \neg X, \\ 2 + l(X) & \text{if } C \in \{\exists r.X, \forall r.X\}, \\ 1 + l(X) + l(Y) & \text{if } C \in \{X \sqcap Y, X \sqcup Y\}. \end{cases} \tag{1}$$

3.3 Concept Learning

Let N_I (individuals), N_R (roles), and N_C (named concepts) be infinite, countable and pairwise disjoint sets. A knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ over a description logic $\mathcal{L}$ is a pair that consists of a T-Box $\mathcal{T}$ and an A-Box $\mathcal{A}$. The T-Box contains subsumption axioms of the form $C \sqsubseteq D$, where C and D are concepts in $\mathcal{L}$, e.g., $\mathcal{ALC}$ (see Section 3.1). The A-Box contains assertions of the form $C(a)$ or $r(a,b)$, where C is a concept in $\mathcal{L}$, $r \in N_R$ is a role, and $a, b \in N_I$ are individuals [23].

A concept learning problem over a knowledge base $\mathcal{K}$ consists of a pair $E = (E^+, E^-)$, where $E^+ \subseteq N_I$ is the set of positive examples and $E^- \subseteq N_I$ contains negative examples. The goal of concept learning is to find a concept C which satisfies [2]:

$$\forall e \in E^+ : \mathcal{K} \models C(e) \wedge \forall e \in E^- : \mathcal{K} \not\models C(e). \tag{2}$$

Finding such a concept is not always possible. Hence, most concept learners aim to maximize a quality function $q : \mathcal{C} \to [0,1]$, where $\mathcal{C}$ is the set of all concepts over N_C and N_R in $\mathcal{L}$. Quality functions, e.g., F1 measure, are commonly designed to return 1 for inputs that satisfy Equation 2 (see [24], Definition 3).

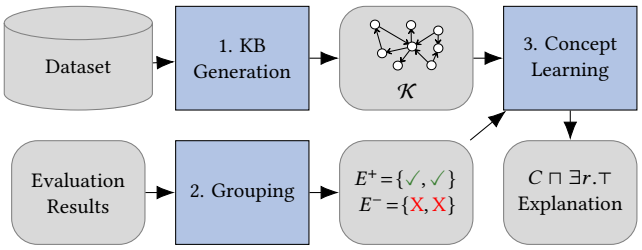


Figure 1: Overview over the three steps of our approach.

3.4 Knowledge-Graph-based Question Answering

A knowledge graph $\mathcal{G}$ is "a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities" [3]. Given a knowledge graph $\mathcal{G}$ and an input question Q in natural language, the goal of a knowledge-graph-based question answering (QA) system f is to derive the set of answers U for Q using $\mathcal{G}$ [35]. We formalize this as:

$$U = f(\mathcal{G}, Q)\,. \tag{3}$$

4 Explainable Benchmarking

Our goal is to automatically generate explanations for benchmarking results that provide users and developers with insights that allow them to better use, or improve the benchmarked system [14]. As a running example, let's assume there is a QA system that answers questions related to geography well but others subpar. The goal of our work is to give this insight into the system's performance in an automatic way based on evaluation results. Our approach goes beyond previous analysis tools that have been designed for a particular field as it does not rely on pre-defined features that are bound to a specific use case. Instead, we transform the available information about the benchmarking process into structured data and apply concept learning. Hence, our approach only has the requirement that the available data can be transformed into structured data and that the used concept learning algorithm is expressive enough to find an explanation.

4.1 Approach

Our approach to automatically generate explanations for benchmarking results consists of the three steps shown in Figure 1: (1) generate a knowledge base ($\mathcal{K}$) comprising structured information about the content of the benchmark dataset, (2) split the benchmark's tasks, e.g., the questions of a QA dataset, into correctly (E^+) and incorrectly answered tasks (E^-), and (3) use concept learning to determine an expression that separates the two groups from each other. We will explain these steps in more detail in the following.

In the first step, we generate a knowledge base comprising information from the dataset about the examples that are used during the benchmarking process (e.g., questions in the area of QA). The more information can be provided about the examples, the higher is the

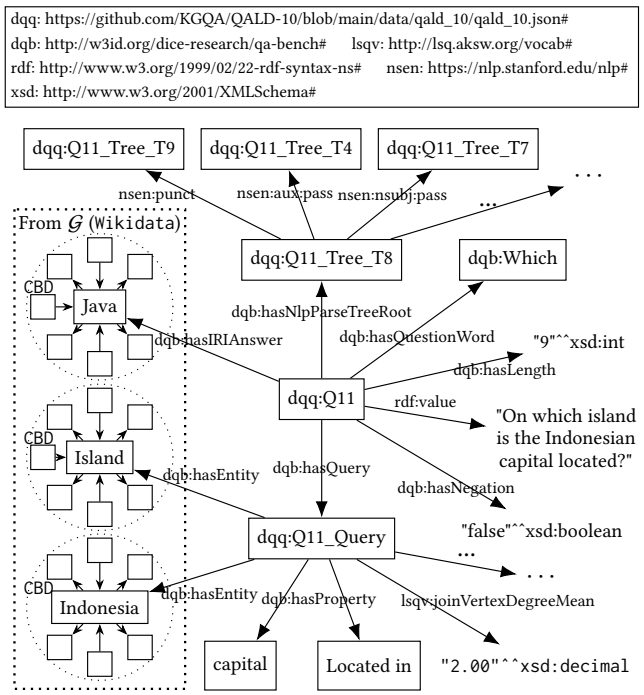


Figure 2: An excerpt of the knowledge base around the question `dqq:Q11` (question 11 from QALD-10). Wikidata entities and properties are replaced by their labels.

chance that the concept learning algorithm is going to find a good explanation in step 3. For the application of our approach on QA benchmarking results, we rely on the QALD datasets [4, 29], from which we extract features based on: (1) the natural language question (e.g., length, question word, dependency parse tree [6]), (2) the ground truth answer(s) (e.g., type of answers, concise bounded description (CBD) [38] of entities in the answer set gathered from $\mathcal{G}$), and (3) the provided SPARQL query, that returns the ground truth answer when used on $\mathcal{G}$ (e.g., LSQ [20] features, properties and CBDs of entities used the query).[4] Figure 2 shows the example question `dqq:Q11` ("On which island is the Indonesian capital located?") and an excerpt of the data in $\mathcal{K}$ connected to it.

In the second step, we separate the questions into two groups depending on the ability of the question answering system to answer the questions correctly. We determine the correctness of each question using GERBIL QA [27] and define that a question is answered correctly if its F1 score is 0.5 or higher. Conversely, questions with an F1 score below 0.5 are labeled as incorrectly answered. We use the correctly answered questions as E^+ and incorrectly answered questions as E^- as defined in Section 3. In our running example, we assume that the evaluated QA system is able to answer the question from Figure 2 and other geography-related questions correctly. Hence, they are part of E^+ while other questions that were not answered well are assigned to E^-.

In the third step, we learn a concept that characterizes the previously identified positive examples E^+ and separates them from the negative examples E^-. This concept is used as explanation and can be easily verbalized to be human-readable [8]. In our running example, a learner could return the concept ∃hasIriAnswer.Place which can be verbalized to "The system can answer questions that have places as answers" providing insight into the QA system's performance in a human-readable form. In this step, any concept learning approach can be used. However, we propose a new, scalable approach in the following.

[4] Our Github project provides a more detailed description of the information.

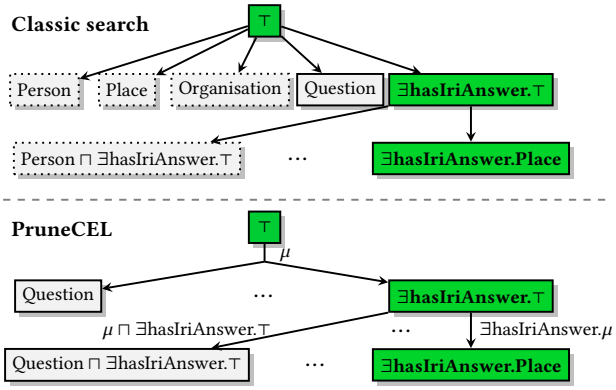


Figure 3: Examples of a classic (top) and a PruneCEL (bottom) search tree. The green expressions are further refined. Expressions with a dotted frame do not have any given example as instance. The labels on the edges are the generated templates used to derive concepts or roles from the oracle.

4.2 PruneCEL

PruneCEL is a top-down refinement-operator-based concept learning algorithm. Unlike similar approaches like CELOE or DRILL (see Section 2), PruneCEL avoids the generation of unsatisfiable concepts. Correspondingly, it is more time-efficient than similar approaches. The motivation behind our approach lies in using the monotonicity of subset inclusion, i.e., $A \subset B \implies |A| < |B|$. Hence, if ρ is a downward refinement operator, then $\forall D \in \rho(C) : C^I \cap (E^+ \cup E^-) = \emptyset \implies D^I \cap (E^+ \cup E^-) = \emptyset$. While other approaches exploit this observation by aiming not to refine concepts C with $C^I \cap (E^+ \cup E^-) = \emptyset$, the main idea of PruneCEL is to avoid generating such concepts entirely.

The upper half of Figure 3 shows a typical search tree as it would be created for our running example. The top concept is refined to a long list of concepts. The concept with the best performance ∃hasIriAnswer.⊤ is then further refined to another long list of concepts. In practice, many of the generated concepts do not include any of the given positive or negative examples as instances. For the running example, the system may generate a concept such as ∃hasIriAnswer.Person, which does not describe the positive geography-related questions but still consumes resources during its generation and evaluation.

PruneCEL avoids the generation of these concepts by relying on an oracle, which provides named concepts or roles that can be used to fill certain gaps in a template, generating concepts that have at least one of the given examples as instance.

When refining the concept ∃hasIriAnswer.⊤ in our running example, PruneCEL generates templates like ∃hasIriAnswer.μ including the concept to be refined and a marker μ. Then, it uses an oracle to get all named concepts or roles that can be used to replace μ to generate new concepts. The oracle guarantees that for each generated concept D the following holds: $D^{\mathcal{I}} \cap (E^+ \cup E^-) \neq \emptyset$. In our example, the oracle suggests to replace μ with Place in to produce a meaningful candidate concept. The lower half of Figure 3 shows the search tree for the same example as PruneCEL would create it.

This take on concept learning leads to several changes in the typical recursive workflow of the refinement operator. First, our operator has to be able to generate templates. Each of them contains exactly one marked position at which the oracle should insert named concepts or roles to create a new concept. Second, we use SPARQL queries to implement the oracle, i.e., to select named concepts and roles from $\mathcal{K}$. In the following, we provide a formal definition of our refinement operator before we provide details about PruneCEL's oracle, scoring and extensions.

4.2.1 Refinement Operator. We define a top-down length-based refinement operator $\rho : C \times 2^{N_I} \times 2^{N_I} \rightarrow 2^C$ for $\mathcal{ALC}$. ρ operates in the quasi-ordered space (C, l), where $l : C \rightarrow \mathbb{N}$ is the length of a concept as defined above. Hence, $D \in \rho(C) \implies l(D) \geq l(C)$.

Let $C^\star$ be the space of all templates that can be created when extending the $\mathcal{ALC}$ grammar defined in Section 3.1 with the symbol μ. μ is handled like a named concept but serves as a marker of the position within a template that has to be filled by the oracle. We define that each template in $C^\star$ contains μ exactly once. Further, let $m^\star : C^\star \times C^\star \rightarrow C^\star$ be a function that merges two given templates T_1 and T_2 by replacing the occurrence of μ in the template T_1 with the template T_2. Similarly, let $m : C^\star \times C \rightarrow C$ be a function that creates a new concept by replacing the marker μ in the given template with the given concept.

Based on the previous definitions, we define our top-down length-based refinement operator ρ to refine a given concept C based on the given examples E^+ and E^- as follows:

$$\begin{aligned}\rho(C, E^+, E^-) =& \rho^\star(C, \mu, E^+, E^-) \cup \\ & \left\{ (\neg D \,|\, D \in \rho^\star(C, \mu, E^+, E^-) \right\},\end{aligned} \tag{4}$$

where $\rho^\star : C \times C^\star \times 2^{N_I} \times 2^{N_I} \rightarrow 2^C$ is a function that takes a concept and a template and refines it recursively based on the given positive and negative examples. Since Equation 4 is the start of the recursion, the template only comprises the positional marker μ. It can also be seen that the refinement operator doubles the amount of expressions that $\rho^\star$ provides by creating their negations. We define the recursive function $\rho^\star$ as follows:

$$\rho^\star(C, T, E^+, E^-)$$
$$= \begin{cases} \rho^\star(X, m^\star(T, \exists r.\mu), E^+, E^-) \cup \\ \quad \{m(T, \forall r.X)\} & \text{if } C{=}\exists r.X, \\ \rho^\star(X, m^\star(T, \forall r.\mu), E^+, E^-) & \text{if } C{=}\forall r.X, \\ \rho^\star(X, m^\star(T, \neg\mu), E^+, E^-) & \text{if } C{=}\neg X \wedge X \notin N_C, \\ \rho^\star(X, m^\star(T, \mu \sqcap Y), E^+, E^-) \cup & \text{if } C{=}X \sqcap Y, \\ \quad \rho^\star(Y, m^\star(T, X \sqcap \mu), E^+, E^-) & \\ \rho^\star(X, m^\star(T, \mu \sqcup Y), E^+, E^-) \cup & \text{if } C{=}X \sqcup Y, \\ \quad \rho^\star(Y, m^\star(T, X \sqcup \mu), E^+, E^-) & \\ g(m^\star(T, C \sqcap \mu), E^+, E^-) \cup & \text{if } C \notin \{\top, \bot\}, \\ \quad g(m^\star(T, C \sqcup \mu), E^+, E^-) & \\ g(T, E^+, E^-) & \text{if } C{=}\top, \end{cases} \tag{5}$$

where $X, Y \in C$ and g is a function that generates new concepts using the oracle and the given template. For complex concepts (the first five cases in Equation 5), $\rho^\star$ calls itself recursively, focusing on one of the parts of the given concept while the other parts of the expression are added to the template using the merge function $m^\star$. An exception is the first case, in which $\exists r.X$ is also refined to $\forall r.X$. For all expressions except $\top$ or $\bot$ (case 6), our refinement tries to add a conjunction and a disjunction. The $\top$ expression is replaced with named classes or roles (case 7). The two latter cases are the base cases of the recursion and rely on the generator function g.

The generator function $g : C^\star \times 2^{N_I} \times 2^{N_I} \rightarrow 2^C$ takes a template and the examples as input and returns new concepts based on the named concepts and roles derived from the oracle. We define it as follows:

$$\begin{aligned} g(T, E^+, E^-) =& \{m(T, D) | D \in o_c(T, E^+, E^-)\} \cup \\ & \{m(T, \neg D) | D \in o_{\neg c}(T, E^+, E^-)\} \cup \\ & \{m(T, \exists r.\top) | r \in o_r(T, E^+, E^-)\}, \end{aligned} \tag{6}$$

where we rely on three different functions—o_c, $o_{\neg c}$, and o_r—provided by the oracle. The first oracle function $o_c : C^\star \times 2^{N_I} \times 2^{N_I} \rightarrow N_C$ takes a template and a set of positive and negative examples as input. The output is a set comprising all named concepts that when used to replace the marker μ in the template form a concept D that has at least one of the given examples as instance:

$$\begin{aligned} o_c(T, E^+, E^-) = \{D | & X = m(T, D) \wedge D \in N_C \\ & \wedge X^{\mathcal{I}} \cap (E^+ \cup E^-) \neq \emptyset\}. \end{aligned} \tag{7}$$

In the same way, we define the second function $o_{\neg c} : C^\star \times 2^{N_I} \times 2^{N_I} \rightarrow N_C$, which negates the named concepts as follows:

$$\begin{aligned} o_{\neg c}(T, E^+, E^-) = \{D | & X = m(T, \neg D) \wedge D \in N_C \\ & \wedge X^{\mathcal{I}} \cap (E^+ \cup E^-) \neq \emptyset\}. \end{aligned} \tag{8}$$

Similarly, we define the third oracle function $o_r : C^\star \times 2^{N_I} \times 2^{N_I} \rightarrow N_R$ that returns roles as follows:

$$\begin{aligned} o_r(T, E^+, E^-) = \{r | & X = m(T, \exists r.\top) \wedge r \in N_R \\ & \wedge X^{\mathcal{I}} \cap (E^+ \cup E^-) \neq \emptyset\}. \end{aligned} \tag{9}$$

According to [36], ρ is a top-down length-based refinement operator since Equation 5 guarantees that newly created concepts

have either the same length (cases 1 and 7) or are longer (case 6) than the given concept C, i.e., $D \in \rho(C) \implies l(D) \geq l(C)$. From the list of properties of a refinement operator proposed by [36], our refinement operator ρ is finite (i.e., $\rho(C)$ is finite for any concept C) and redundant (i.e., during the search, ρ may return a concept that is equivalent to a previously returned concept). However, ρ is not (weakly) complete as it is not able to generate all possible concepts since many do not select any given example. Hence, pruning the search space leads to the loss of the completeness of the operator.[5]

4.2.2 Oracle Implementation. The oracle is implemented in the form of SPARQL queries that are sent to a triple store containing $\mathcal{K}$. Since the queries used for o_c, $o_{\neg c}$, and o_r already contain the positive and negative examples, we extend these queries to derive the numbers of positive and negative examples that are instances of the newly created concepts. This further reduces the number of SPARQL queries that our approach sends to the triple store in comparison to previous approaches, which would derive these counts for all created concepts one after the other.

4.2.3 Heuristic. We score a generated class expression C by determining the number of positive p (respectively, negative n) examples that are instances of (respectively, ruled out by) this concept according to $\mathcal{K}$. Then, we compare these counts with the overall number of positive and negative examples. This can be done using any quality function q like accuracy or F1 measure. Like previous works, e.g., [33], we assume that shorter concepts are more general and, hence, preferred. So we include the length of the concept multiplied by a small constant η into our heuristic function h:

$$h(C, p, n, E^+, E^-) = q(p, n, |E^+|, |E^-|) - \eta l(C)\,. \tag{10}$$

4.2.4 Extensions. We propose two additional extensions—PruneCEL-S and PruneCEL-R. Both can be used together, which we name PruneCEL-RS.

PruneCEL-S. In this mode, a newly created concept $D \in \rho(C)$ is only considered for further refinement if (1) it received a better score than C or (2) D has been derived from C by adding a role.

PruneCEL-R. In this recursive mode, PruneCEL calls itself if it found a solution for a sub-problem. Let $E^{+\prime} \subset E^+$ be a set of positive examples with $|E^{+\prime}| \geq 2$. If our approach has found a concept D, which is an exact solution for $E^{+\prime}$, i.e., D satisfies $\forall e \in E^{+\prime} : \mathcal{K} \models D(e) \wedge \forall e \in E^- : \mathcal{K} \not\models D(e)$, PruneCEL calls itself with a smaller learning problem $(E^+ \backslash E^{+\prime}, E^-)$. It spends a limited amount of iterations on this smaller problem before it returns its best concepts. These are combined with D and introduced into the search tree as additional solutions which then can be further refined.

5 Evaluation

5.1 Experiment Setup

First, we compare the performance of different concept learners on the knowledge bases created by our approach (Experiment I). Finally, we use a survey to check whether our explanations are understood by humans (Experiment II).

[5] A similar tradeoff is encountered by other approaches, e.g., DL-Foil [21]. The interested reader is referred to [36] for the full list of properties.

Table 2: The number of learning problems (LP), their average number of positive and negative examples, and the features of $\mathcal{K}$ for the QALD-based datasets. P = Properties.

Datasets	LPs	$\lvert E^+\rvert$	$\lvert E^-\rvert$	Entities	P	Triples
QALD9+DB	3	22.7	110.3	21,518,759	918	72,737,644
QALD9+WD	2	30.5	85.5	36,565,453	826	84,345,960
QALD10	2	91.0	303.0	64,352,096	878	155,959,524

Table 3: Correctly / faulty answered questions per QA system. DB = DBpedia, WD = Wikidata.

Systems	QALD9+DB	QALD9+WD	QALD10
DeepPavlov	– / –	26 / 90	61 / 333
gAnswer	18 / 115	– / –	– / –
MST5	28 / 105	35 / 81	121 / 273
TeBaQA	22 / 111	– / –	– / –

5.1.1 Experiment I. In the First experiment, we apply our approach to the benchmarking results of the 4 QA systems DeepPavlov [9], gAnswer [32], TeBaQA [7], and MST5 [25] on three QA datasets—QALD 9 Plus for DBpedia and Wikidata [4], and QALD 10 [29]. We remove questions that have an empty ground truth answer set from these QA datasets leading to 133, 116, and 394 questions, respectively. We generate a knowledge base $\mathcal{K}$ for each QA dataset as described in Section 4.1. We gather the answers generated by the 4 QA systems for these datasets. The DBpedia-based systems gAnswer and TeBaQA only provide answers for QALD 9 Plus DBpedia, while the Wikidata-based system DeepPavlov provides results for QALD 9 Plus Wikidata and QALD 10. MST5 [25] provides answers for all three QA datasets. We use the evaluation results from GERBIL QA [27] to identify correctly and faulty answered questions to derive E^+ and E^- for each QA system. Table 3 shows the summary of this step. Table 2 summarizes the features of the generated knowledge bases as well as the resulting concept learning datasets dubbed QALD9+DB, QALD9+WD, and QALD10.[6]

On these three concept learning dataset, we apply the four concept learners CELOE, Drill, EvoLearner, and NCES from the related work. We compare their performance to PruneCEL-RS, which showed the best performance in preliminary experiments.[7] We run all approaches with their default configuration.[8] PruneCEL-RS is executed three times using three different measures h as part of the scoring function, namely accuracy, balanced accuracy and F1 measure. In all configurations, we set $\eta = 0.01$. We set the maximum runtime of all approaches for a single learning problem to 10 minutes and compare their results using the F1-measure, the concept length and their runtime.[9]

[6] The DBpedia and Wikidata versions that we use as $\mathcal{G}$ can be found at https://downloads.dbpedia.org/2016-10/core-i18n/en/ and [28].

[7] Due to the length restriction of this publication, the results of these preliminary experiments can be found in our Github project.

[8] For Drill, we use the Keci embedding algorithm [5]. The embedding models and pre-trained Drill models can be found at doi:10.5281/zenodo.14720609 and doi:10.5281/zenodo.14720524, respectively.

[9] We provide the knowledge bases and learning problems at doi:10.5281/zenodo.14720669 and doi:10.5281/zenodo.16681824, respectively. We use an AMD EPYC 7282 with 252 GB RAM.

5.1.2 Experiment II. We conduct a survey to evaluate the quality of our explanations. We choose two concepts learned from Experiment I generated on two very different knowledge bases—QALD9+DB and QALD10—that achieve the highest F1 score when compared with a baseline that returns the concept $\top$. On both knowledge bases, these are concepts explaining the performance of MST5, which we verbalize using ChatGPT.[10] Our approach explains the performance of MST5 on QALD10 with the following concept:

$$\exists \text{hasEntityAnswer}.(\text{album} \sqcup \exists \text{copyrightStatusAsCreator}.\top \sqcup \text{profession} \sqcup \exists \text{locatedInAdministrativeTerritorialEntity}.\neg\text{country} \sqcup \exists \text{manifestationOf}.\top) \sqcup \exists \text{hasBooleanAnswer}.\top$$

which is verbalized as (naming MST5 "QAS1"):
The system "QAS1" can answer questions if:

(1) *There's an answer involving an album, a creator's copyright status, a profession, a location that's not a country, or something that has a type or form.*
(2) *Or, it can answer questions that have a simple yes/no (boolean) answer.*

For MST5 on QALD9+DB, our approach finds the following concept:

$$\exists \text{hasEntityAnswer}.((\neg\text{agent} \sqcap \exists \text{parentMountainPeak}.\top) \sqcup \text{building}) \sqcup (\exists \text{hasIRIAnswer}.(\text{astronaut} \sqcup (\neg\text{agent} \sqcap \neg\text{spatialThing})) \sqcap \exists \text{hasQuestionWord}.\top)$$

which is verbalized as (naming MST5 "QAS2"):
The system "QAS2" can answer questions if:

(1) *The answer involves either: a non-agent (not a person or entity with intent) with a parent mountain peak, or a building.*
(2) *Or, if the answer involves: an astronaut, or a non-agent, non-spatial entity (something that's neither a person nor a physical location), and if the question includes a question word (like "who," "what," or "where").*

For each chosen concept, we randomly choose 5 correctly and 5 faulty answered questions of MST5, which are classified correctly by the chosen concept. In the survey, we provide these 20 questions together with 5 or more statements from $\mathcal{K}$ that would be sufficient for a reasoner to decide whether the question with this data is an instance of the learned concept. For each question, the survey participants have to decide whether a QA system with the provided explanation would be able to answer the given question. The higher the success rate, the better do humans understand the explanation and are able to decide whether to use the QA system for it or not. We configure the survey to randomly order the questions for each participant and distribute the survey to computer scientists (e.g., via mailing lists).

5.2 Results

5.2.1 Experiment I. Table 4 summarizes the results of the four state of the art concept learners and PruneCEL-RS on the 7 learning problems. PruneCEL-RS achieves significantly better F1 scores than CELOE and Drill.[11] A deeper analysis of the behavior of CELOE and Drill shows that these approaches already need more than the provided 10 minutes to execute $\rho(\top)$, i.e., the first step at the beginning of their search. EvoLearner and NCES did not give any results on the large knowledge bases of this experiment.[12]

PruneCEL-RS provides significantly better F1-scores than CELOE and Drill for all 7 learning problems when relying on balanced accuracy or the F1 measure during the search. Consequently, it is also significantly better than a baseline that would always return the concept $\top$. The usage of accuracy leads to mixed results and seems to mislead the search when the learning problem has only a small number of positive examples.[13]

5.2.2 Experiment II. 41 people participated in our survey, answering at least 1 question. Table 5 shows the survey results. For 16 out of 20 questions (i.e., 80%), the majority of the participants was able to correctly decide whether the described QA system would be able to answer the given question. For all questions, except question 17, the answers of the volunteers are significantly different to those of a random guesser.[14] A closer look at the questions that were not classified correctly by the majority of the participants revealed two patterns of errors. First, the verbalization generated by ChatGPT was not always exactly explaining the given concept. For example, in the case of MST5 on QALD10, while one of the properties has the label "manifestation of" ChatGPT translated it into "something that has a type or form", leading to people ignoring triples with the "manifestation of" property in the provided data. Second, in the case of MST5 on QALD9+DB, ChatGPT (or the participants) relied on background knowledge that was not part of $\mathcal{K}$. This led to a misunderstanding ¬agent that was incorrectly understood as a non-person by ChatGPT. This underlines the importance of a well-defined ontology as basis for the concept learning and, hence, for our approach.

6 Discussion

The results of Experiment I show that due to PruneCEL's scalability, it outperforms the other four approaches on knowledge bases containing tens of millions of triples. The importance of the scalability is further underlined when taking into account the sizes of real-world knowledge graphs like DBpedia [30] or Wikidata [39].

In Experiment I, PruneCEL achieves F1 scores that are higher than the performance of the concept $\top$ would be. This suggests that our approach is able to provide meaningful explanations. This is supported by the results of Experiment II which suggest that humans can accurately predict the behavior of the benchmarked system based on these explanations. However, future work will have to take the two identified sources of errors into account.

7 Conclusion

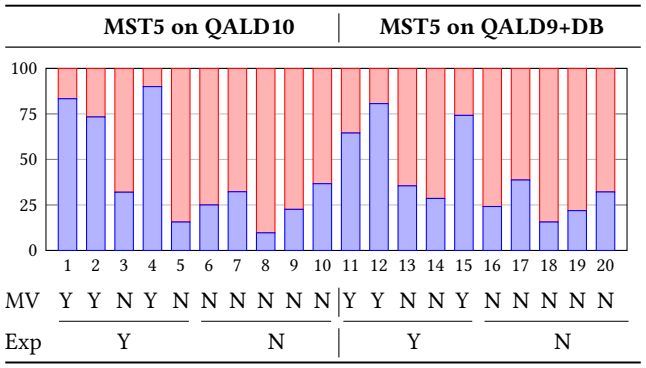
We proposed an approach for generating explainations for benchmarking results. Our approach relies on the transformation of the benchmarking dataset into a knowledge base and on concept learning to find a concept that separates cases in which the benchmarked system performed good from those in which it performed subpar. We also proposed a new, scalable concept learning algorithm

[10] The prompt for the verbalization can be found in our Github project.
[11] We use a Wilcoxon signed-rank test with $\alpha = 0.05$.
[12] We worked together with the authors of EvoLearner and NCES but couldn't find a solution before the submission deadline.
[13] The full set of results of PruneCEL-RS can be found in our Github project.
[14] Significant = a random guesser has a probability below 5% to create the answer set.

Table 4: F1 score, length of the generated concepts ($l(C)$), and runtime (RT, in seconds) for the learning problems (LP) of Experiment II. For PruneCEL-RS, we report the quality measure (q, A = accuracy, B = balanced accuracy, F = F1 measure) leading to the best F1 score. The complete set of results for PruneCEL-RS can be found in the appendix.

Dataset		Drill			Evolearner			CELOE			NCES			PruneCEL-RS			
$\mathcal{K}$	LP	F1	$l(C)$	RT	F1	$l(C)$	RT	F1	$l(C)$	RT	F1	$l(C)$	RT	F1	$l(C)$	RT	q
QALD9+DB	gAnswer	0.24	1	2299.2	–	–	–	0.26	3	10306.2	–	–	–	**0.35**	19	600.1	A
	MST5	0.35	1	2694.5	–	–	–	0.35	1	10392.3	–	–	–	**0.57**	24	600.1	B
	TeBaQA	0.30	3	2267.4	–	–	–	0.30	3	10356.7	–	–	–	**0.44**	23	650.5	F
QALD9+WD	DeepPavlov	0.37	1	3793.6	–	–	–	0.41	3	12416.7	–	–	–	**0.96**	107	600.9	A
	MST5	0.46	1	3779.8	–	–	–	0.50	3	12369.0	–	–	–	**0.84**	28	600.9	B
QALD10	DeepPavlov	0.27	1	3495.3	–	–	–	0.27	1	2326.2	–	–	–	**0.34**	10	600.8	F
	MST5	0.47	1	3468.1	–	–	–	0.47	1	2271.3	–	–	–	**0.56**	20	602.0	B

Table 5: Survey results per question for the two learning problems (in %, Yes (Y), No (N)). The MV row shows the summary of the answers as a majority vote and the Exp row the expected results.



named PruneCEL, which uses the monotonicity of subset inclusion to prune its search tree. Our evaluation used benchmark datasets and real-world benchmarking results from the knowledge-graph-based Question Answering domain. Our evaluation results show that PruneCEL significantly outperforms state-of-the-art concept learners on the knowledge bases created by our approach due to its scalability. A survey including the answers of 41 participants showed that in 80% of the cases the majority of participants were able to understand the explanations our approach generates for the evaluation results of QA systems.

Our future work is threefold. First, we plan to apply our generic approach to other application areas. Second, we want to further improve existing concept learners. Third, a large scale experiment is needed to ensure that the generated explanations cannot only be understood be the explainee, but also support them in improving the benchmarked system over time.

Acknowledgments

This work has been supported by the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW) within the projects SAIL (NW21-059D) and WHALE (LFN 1-04, under the Lamarr Fellow Network programme), and the European Union's Horizon Europe research and innovation programme in the project ENEXA (No. 101070305).

References

[1] Franz Baader. 2003. *The description logic handbook: Theory, implementation and applications.* Cambridge university press.
[2] Caglar Demir and Axel-Cyrille Ngonga Ngomo. 2023. Neuro-symbolic class expression learning. In *Proceedings of IJCAI 2023.* doi:10.24963/ijcai.2023/403
[3] Aidan Hogan et al. 2021. *Knowledge Graphs.* Number 22 in Synthesis Lectures on Data. Morgan & Claypool. 1–237 pages. doi:10.2200/S01125ED1V01Y202109DSK022
[4] Aleksandr Perevalov et al. 2022. QALD-9-plus: A Multilingual Dataset for Question Answering over DBpedia and Wikidata Translated by Native Speakers. In *2022 IEEE 16th ICSC.* IEEE, 229–234.
[5] Caglar Demir et al. 2023. Clifford Embeddings–A Generalized Approach for Embedding in Normed Algebras. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 567–582.
[6] Christopher Manning et al. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd ACL: System Demonstrations.*
[7] Daniel Vollmers et al. 2021. Knowledge Graph Question Answering Using Graph-Pattern Isomorphism. In *Studies on the Semantic Web.* IOS Press.
[8] Daniel Vollmers et al. 2024. Enhancing Answers Verbalization Using Large Language Models. In *Proceedings of the 20th International Conference on Semantic Systems (Studies on the Semantic Web).* IOSPress, 345–352. doi:10.3233/SSW240027
[9] Diliara Zharikova et al. 2023. DeepPavlov Dream: Platform for Building Generative AI Assistants. In *Proc. of the 61st ACL (Volume 3: System Demonstrations).*
[10] Giuseppe Rizzo et al. 2017. Tree-based models for inductive classification on the Web Of Data. *Journal of Web Semantics* (2017). doi:10.1016/j.websem.2017.05.001
[11] Hannah Bast et al. 2022. ELEVANT: A Fully Automatic Fine-Grained Entity Linking Evaluation and Analysis Tool. In *EMNLP 2022 Demo.* https://aclanthology.org/2022.emnlp-demos.8/
[12] Jens Lehmann et al. 2011. Class expression learning for ontology engineering. *Journal of Web Semantics* 9, 1 (2011), 71–81. doi:10.1016/j.websem.2011.01.001
[13] Jörg Waitelonis et al. 2016. Don't compare Apples to Oranges: Extending GERBIL for a fine grained NEL evaluation. In *Proceedings of the 12th SEMANTiCS.* ACM.
[14] Katharina J Rohlfing et al. 2020. Explanation as a social practice: Toward a conceptual framework for the social design of AI systems. *IEEE Transactions on Cognitive and Developmental Systems* 13, 3 (2020), 717–728.
[15] Mina Abd Nikooie Pour et al. 2021. Results of the Ontology Alignment Evaluation Initiative 2021. In *Proceedings of the 16th International Workshop on Ontology Matching co-located with the 20th ISWC (ISWC 2021).*
[16] Mohamed Ahmed Sherif et al. 2017. Wombat – A Generalization Approach for Automatic Link Discovery. In *The Semantic Web.* Springer, 103–119.
[17] Mohnish Dubey et al. 2019. LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia. In *The Semantic Web – ISWC 2019.* Springer, 69–78.
[18] Matthew Lamm et al. 2021. Qed: A framework and dataset for explanations in question answering. *Transactions of the ACL* 9 (2021), 790–806.
[19] Michael Röder et al. 2020. HOBBIT: A platform for benchmarking Big Linked Data. *Data Science* 3, 1 (2020), 15–35. doi:10.3233/DS-190021

[20] Muhammad Saleem et al. 2015. LSQ: The Linked SPARQL Queries Dataset. In *The Semantic Web - ISWC 2015*. Springer.
[21] Nicola Fanizzi et al. 2018. DLFoil: Class Expression Learning Revisited. In *Knowledge Engineering and Knowledge Management*. Springer.
[22] N'Dah Jean Kouagou et al. 2022. Learning Concept Lengths Accelerates Concept Learning in ALC. In *The Semantic Web*. Springer.
[23] N'Dah Jean Kouagou et al. 2023. Neural Class Expression Synthesis. In *The Semantic Web*. Springer.
[24] N'Dah Jean Kouagou Jean et al. 2023. Neural class expression synthesis in ALCHIQ (D). In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 196–212.
[25] Nikit Srivastava et al. 2024. MST5 – Multilingual Question Answering over Knowledge Graphs. arXiv:2407.06041 [cs.CL]
[26] Ricardo Usbeck et al. 2015. Evaluating Entity Annotators Using GERBIL. In *The Semantic Web: ESWC 2015 Satellite Events*. Springer.
[27] Ricardo Usbeck et al. 2019. Benchmarking question answering systems. *Semantic Web* 10, 2 (2019).
[28] Ricardo Usbeck et al. 2022. *QALD-10 Wikidata Dump*. doi:10.5281/zenodo.7496690
[29] Ricardo Usbeck et al. 2023. QALD-10 — The 10th Challenge on Question Answering over Linked Data. *Semantic Web Journal* 15, 6 (2023).
[30] Sören Auer et al. 2007. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*. Springer Berlin Heidelberg, 722–735.
[31] Simon Bin et al. 2016. Towards SPARQL-based induction for large-scale RDF data sets. In *Proceedings of ECAI 2016 (ECAI'16)*. IOS Press. doi:10.3233/978-1-61499-672-9-1551
[32] Sen Hu et al. 2018. Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs. *IEEE Transactions on Knowledge and Data Engineering* 30 (2018), 824–837. https://api.semanticscholar.org/CorpusID:4569766
[33] Stefan Heindorf et al. 2022. EvoLearner: Learning Description Logics with Evolutionary Algorithms. In *Proceedings of the ACM Web Conference 2022*. ACM.
[34] Vladislav Korablinov et al. 2020. RuBQ: A Russian dataset for question answering over Wikidata. In *International Semantic Web Conference*. Springer, 97–110.
[35] Yixin Ji et al. 2024. Retrieval and Reasoning on KGs: Integrate Knowledge Graphs into Large Language Models for Complex Question Answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. ACL.
[36] Jens Lehmann and Pascal Hitzler. 2007. Foundations of refinement operators for description logics. In *Proceedings of ILP 2007*. Springer.
[37] Jens Lehmann and Pascal Hitzler. 2007. A refinement operator based learning algorithm for the ALC description logic. In *Proceedings of ILP 2007*. Springer.
[38] Patrick Stickler. 2005. CBD - Concise Bounded Description. W3C Member Submission. https://www.w3.org/submissions/2005/SUBM-CBD-20050603/ Accessed: 2024-10-25.
[39] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (Sept. 2014), 78–85. doi:10.1145/2629489