Searching for explanations of black-box classifiers in the space of semantic queries

Jason Liartis *, Edmund Dervakos, Orfeas Menis-Mastromichalakis, Alexandros Chortaras and Giorgos Stamou

Artificial Intelligence and Learning Systems Laboratory, School of Electrical and Computer Engineering, National Technical University of Athens, Greece
E-mails: jliartis@ails.ece.ntua.gr, eddiedervakos@islab.ntua.gr, menorf@ails.ece.ntua.gr, achort@cs.ntua.gr, gstam@cs.ntua.gr

Abstract. Deep learning models have achieved impressive performance in various tasks, but they are usually opaque with regards to their inner complex operation, obfuscating the reasons for which they make decisions. This opacity raises ethical and legal concerns regarding the real-life use of such models, especially in critical domains such as in medicine, and has led to the emergence of the eXplainable Artificial Intelligence (XAI) field of research, which aims to make the operation of opaque AI systems more comprehensible to humans. The problem of explaining a black-box classifier is often approached by feeding it data and observing its behaviour. In this work, we feed the classifier with data that are part of a knowledge graph, and describe the behaviour with rules that are expressed in the terminology of the knowledge graph, that is understandable by humans. We first theoretically investigate the problem to provide guarantees for the extracted rules and then we investigate the relation of "explanation rules for a specific class" with "semantic queries collecting from the knowledge graph the instances classified by the black-box classifier to this specific class". Thus we approach the problem of extracting explanation rules as a semantic query reverse engineering problem. We develop algorithms for solving this inverse problem as a heuristic search in the space of semantic queries and we evaluate the proposed algorithms on four simulated use-cases and discuss the results.

Keywords: Explainable AI (XAI), opaque machine learning classifiers, knowledge graphs, description logics, semantic query answering, reverse query answering, post-hoc explainability, explanation rules

1. Introduction

The opacity of deep learning models raises ethical and legal [1] concerns regarding the real-life use of such models, especially in critical domains such as medicine and judicial, and has led to the emergence of the eXplainable Artificial Intelligence (XAI) field of research, which aims to make the operation of opaque AI systems more comprehensible to humans [2, 3]. While many traditional machine learning models, such as decision trees, are interpretable by design, they typically perform worse than deep learning approaches for various tasks. Thus, in order to not sacrifice performance for the sake of transparency, a lot of research is focused on *post hoc* explainability, in which the model to be explained is treated as a black-box. Approaches to *post hoc* explainability vary with regard to data domain (images, text, tabular), form of explanations (rule-based, counterfactual, feature importance etc.), scope (global, local) [4] and application domain [5]. In this work we focus on global explanations, which attempt to explain the general function of a black-box regardless of data, as opposed to local explanations which attempt to explain

*Corresponding author. E-mail: jliartis@ails.ece.ntua.gr.

1570-0844/$35.00 © 0 – IOS Press and the authors. All rights reserved

the prediction of a classifier on a particular data sample. Specifically, we attempt to extract *rules* which simulate the behaviour of a black-box by considering semantic descriptions of samples, in addition to external knowledge. For example such a rule might be "Images depicting animals and house-hold items are classified as domestic animals", where the semantic description of an image might be "This image depicts an elephant next to a chair" and external knowledge might contain information such as "elephants are animals" and "chairs are household items". We do this by utilizing terminological, human-understandable knowledge expressed in the form of ontologies and knowledge graphs, taking advantage of the reasoning capabilities of the underpinning description logics [6]. Such extracted rules might be very useful for an end user to understand the reasons behind why an opaque model is making its decisions, especially when combined with other forms of explanations, such as local contrastive explanations [7].

Our approach to global rule-based explanations assumes that we are given a set of data samples with semantic descriptions linked to an external knowledge. The explanations will be presented in the terminology provided by the external knowledge. We call such a set of samples an *explanation dataset*. For example, a semantic description for an image might refer to the objects it depicts and relationships between them, such as scene graphs from visual genome [8], or COCO [9]. The external knowledge utilized could be an existing knowledge base, such as WordNet [10], ConceptNet [11], DBpedia [12], and even domain specific knowledge such as SNOMED-CT [13] for the medical domain, or it could be a knowledge base curated specifically for that explanation dataset. Given such a set of semantically enriched data, we then compute global rule-based explanations characterizing the output of a black-box classifier. We approach the problem as a query reverse engineering (QRE) problem by utilizing the close relationship between definite rules and conjunctive queries.

Utilizing external knowledge to boost transparency of opaque AI is an active research area which has produced important results in recent years [14–20]. Specifically, knowledge graphs (KG) [21] as a scalable common understandable structured representation of a domain based on the way humans mentally perceive the world, have emerged as a promising complement or extension to machine learning approaches for achieving explainability. A particular aspect which might be improved by utilizing knowledge graphs, especially for generalized global explanations, is the form of the produced explanations. When the feature space of the classifier is sub-symbolic raw data, then providing explanations in terms of features might lead to unintuitive, or even misleading results [22–24]. On the other hand, if there is underlying knowledge of the data, then explanations can be provided by using the terminology of the knowledge. For example, if a black-box classified every image depicting wild animals in the class *zoo*, a rule of the form "If an image depicts a giraffe or a zebra or ... then it is classified as a *zoo*", might be more intuitive than for example sets of pixel importances. Furthermore, by exploiting external knowledge, the form can be further condensed and lead to simpler explanations which are semantically identical, such as "If an image depicts a *wild animal* then it is classified as a *zoo*". This approach to explainability mitigates some of the typical pitfalls of *post hoc* XAI, and transfers potential sources of erroneous or misleading explanations to the level of the data and the knowledge (instead of the explainability algorithm), which are easier to control in most settings. For instance, a typical pitfall of *post hoc* XAI mentioned in [24], is that explanations often do not make sense, or are incomplete (leaving out information to be understandable). In our approach, the explanations make as much sense as the terminology used in the knowledge graph, and are as informative as the explanation dataset.

There are many related rule-based explanation methods (both global and local) in recent literature. Many approaches rely on statistics in order to generate lists of IF-THEN rules which mimic the behaviour of a classifier [25–29], while others extract rules in the form of decision trees [30, 31] or First-Order Logic expressions [32]. Rules have been argued to be the preferred method of presenting explanations to humans, as they are close to the way humans reason [4, 33, 34]. There are also related works that utilize ontologies for the purpose of explainability, and we build on these ideas in our work. In [19] the authors utilize ontologies, expressed in the Description Logic $\mathcal{EL}$, to enhance explanations of black-box classifiers produced in the form of decision trees. The authors conducted a user evaluation from which they concluded that users prefer decision trees extracted with the presence of an ontology than those without. In [20] the authors present a framework quite similar to our own. They also approach the problem of extracting explanations as a query reverse engineering problem, but there is no emphasis on the data utilized from producing the explanations compared to our notion of an explanation dataset. Their work is also preliminary as they do not present any algorithms for extracting the desired queries.

Query Reverse Engineering (QRE) is the problem of reverse-engineering a query from its output [35]. This is typically presented as a way for an end-user unfamiliar with a query language such as SQL to formulate a query

by simply providing examples of what they expect to see in the query results. This is also usually approached as an interactive process; the user is continually providing more examples and marking answers of the query as desired or undesired. QRE has been extensively studied for relational databases and the language of relational algebra or SQL [36]. In this work we assume data is stored in a Knowledge Base and queries are formulated in the language of Conjunctive Queries. We also assume that we know the exact desired output of the query, without the need for a user to continually provide examples. Although conjunctive queries have the same expressivity as select-project-join relational queries using only equality predicates, adapting existing works to fit our framework is not easy or even desirable. Some common obstacles include i) the assumption of a dynamic labeling of individuals as positive/negative by an end-user [37–39], ii) the (often implicit) exclusion of the self-join operation, which would limit the expressivity of queries in ways undesirable to us [35, 37–39], and iii) foreign/primary key constraints [40].

In the setting of conjunctive queries over Knowledge Bases, referred to as Semantic Query Answering or Ontology Mediated Querying [41–45], QRE has been under-explored. Recent work focuses on establishing theoretical results, such as the existence of solutions and the complexity of computing them [46–50]. The authors in [51] present a few algorithms alongside complexity results but the algorithms are limited to tree-shaped queries. One recent article [52], along with some theoretical results, presents two algorithms computing minimally complete and maximally sound solutions to the closely related Query Separability problem. We consider these algorithms unsuitable in the context of explainability, since the solutions computed essentially amount to one large disjunction of positive examples. In this work we focus on practical QRE over KBs and we present several algorithms which either have polynomial complexity or have worst-case exponential complexity but can be efficient in practice. We employ heuristics to limit the number of computations, we take care to limit the size of queries, since their intended use is that of explanations presented to users, and we provide (approximate) solutions even when a exact solution does not exist.

Our contributions:

- Following our previous work in the area [46], we here present a framework for extracting global rule-based explanations of black-box classifiers, using exemplar items, external terminology and underlying knowledge stored in a knowledge graph and defining the problem of explanation rule extraction as a semantic query reverse engineering problem over the knowledge graph (see section 3).
- We propose practical algorithms which approximate the semantic query reverse engineering problem by using heuristics, which we then use to generate explanations in the context of the proposed framework (see section 4).
- We implement the proposed algorithms and show results from experiments explaining image classifiers on CLEVR-Hans3, Places365 and MNIST. We also compare our work with existing post-hoc, rule-based explanation methods on baseline tabular data employing the Mushroom dataset (see section 5).

2. Background

2.1. Description Logics

Let $\mathcal{V} = \langle \mathsf{CN}, \mathsf{RN}, \mathsf{IN} \rangle$ be a *vocabulary*, where CN, RN, IN are mutually disjoint finite sets of *concept names* (atomic concepts), *role names* (atomic roles) and *individual names*, respectively. Atomic concepts and roles can be combined into complex concepts or roles using appropriate *constructors*. The set of available constructors for constructing complex concepts and roles, along with their syntax and semantics defines a Description Logics (DL) *dialect*. A list of common constructors is provided in Table 1, which are the constructors used by the $\mathcal{SHROIQ}$ dialect, which underpins the Web Ontology Language OWL 2. Note that the table defines also the two special concept names $\top$ and $\bot$ that may belong to CN, representing the universal and empty concept, respectively.

Concepts and roles (either atomic or complex) are used to form terminological and assertional axioms, as shown in Table 2. Then, a *DL knowledge base*, simply a *knowledge base*, (KB) over a DL dialect $\mathcal{L}$ and a vocabulary $\mathcal{V}$ is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a set of terminological axioms (also called TBox), and $\mathcal{A}$ a set of assertional axioms (also called ABox) using the constructors of $\mathcal{L}$ and elements of $\mathcal{V}$. We will call an ABox containing only

Table 1
DL syntax and semantics.

Name	Syntax	Semantics
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Existential quantification	$\exists r.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
Value restriction	$\forall r.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.\ (a,b) \in r^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
At-least number restriction	$\geqslant nr.C$	$\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \geqslant n\}$
At-most number restriction	$\leqslant nr.C$	$\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \leqslant n\}$
Nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
Inverse role	r^-	$\{(b,a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a,b) \in r^{\mathcal{I}}\}$
Role composition	$r \circ s$	$r^{\mathcal{I}} \circ s^{\mathcal{I}}$

Table 2
Terminological and assertional axioms.

Name	Syntax	Semantics
Concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Concept equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
Role inclusion	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
Role equivalence	$r \equiv s$	$r^{\mathcal{I}} = s^{\mathcal{I}}$
Concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
Role assertion	$r(a,b)$	$(a,b)^{\mathcal{I}} \in r^{\mathcal{I}}$
Individual equality	$a = b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
Individual inequality	$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

assertions of the form $C(a)$ and $r(a,b)$, where $C \in \mathsf{CN}$, $r \in \mathsf{RN}$, $a, b \in \mathsf{IN}$, i.e. not involving complex expressions and (in)equality assertions, an *atomic* ABox.

The semantics of DL KBs are defined in the standard model-theoretic way using interpretations. In particular, given a non-empty domain Δ, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ assigns a set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each atomic concept $C \in \mathsf{CN}$, a set of pairs $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each atomic role $r \in \mathsf{RN}$, and an element $a^{\mathcal{I}} \in \Delta$ to each individual $a \in \mathsf{IN}$. Table 1 shows also the semantics of the listed constructors. An interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K}$ iff it satisfies all axioms in $\mathcal{T}$ and all assertions in $\mathcal{A}$, i.e. iff it satisfies the expression in the last column of Table 2 for the respective axioms.

The underlying DL dialect $\mathcal{L}$ determines the expressivity of a $\mathcal{K}$. Most classical DL dialects can be seen as fragments of first-order logic (fol) by viewing atomic concepts and roles as unary and binary predicates respectively. In this paper we refer only to DL dialects that are fragments of first-order logic, and hence can be translated to first-order logic theories. We will denote the translation of such a $\mathcal{K}$ to the respective first order logic theory by $\mathsf{fol}(\mathcal{K})$. E.g. the fol translation of the DL concept $\exists r.C$ is an expression of the form $\exists x.(r(y,x) \wedge \phi_C(x))$, where $\phi_C(x)$ is the fol translation of concept C. For more details see [6].

2.2. Conjunctive Queries

Given a vocabulary $\mathcal{V}$, a *conjunctive query* (simply, a *query*) q is an expression of the form $\{\, \langle x_1, \ldots, x_k \rangle \mid \exists y_1 \ldots \exists y_l.(c_1 \wedge \ldots \wedge c_n)\,\}$, where $k, l \geqslant 0$, $n \geqslant 1$, x_i, y_i are variable names, the c_is are distinct, each c_i is an atom $C(u)$ or $r(u,v)$, where $C \in \mathsf{CN}$, $r \in \mathsf{RN}$, $u, v \in \mathsf{IN} \cup \{x_1, \ldots, x_k\} \cup \{y_1, \ldots, x_l\}$ and all x_i, y_i appear in at least one atom. The vector $\langle x_1, \ldots, x_k \rangle$ is the *head* of q, its elements are the *answer variables*, and $\{c_1, \ldots, c_n\}$ is the *body* of q ($\mathsf{body}(q)$). The set $\mathsf{var}(q)$ is the set of all variables appearing in q.

On conjunctive queries one can apply substitutions, which map variables to other variables or individual names. More formally, a *substitution* θ is a finite set of mappings of the form $\{z_1 \mapsto u_1, \ldots, z_n \mapsto u_n\}$ where z_i is a variable name, u_i is either an individual name or a variable name distinct from all z_is, and the variables z_i are distinct. A substitution may contain at most one mapping for a variable. The conjunctive query $q\theta$ resulting from applying θ on q is obtained from q by replacing all occurrences of z in q by u for each $z \mapsto u \in \theta$. Any variables not mapped by θ remain unchanged.

Since in this paper we are interested in classifying single items, we focus on queries having a *single* answer variable. For convenience, we will also assume that all arguments of all c_i in a query are variables, and we call such queries *single answer variable queries*, or *sav queries* for short. For simplicity, in the rest of the paper, by saying query we will mean sav query, and we will write a (sav) query q as $\{c_1, \ldots, c_n\}$, considering always x as the answer variable, essentially identifying the query by its body. This will allow us to treat queries as sets, and write e.g. $q_1 \cup q_2$; this query represents the query that has the same head as q_1 and q_2 (i.e. $\langle x \rangle$), and body the union of the bodies of q_1 and q_2. Similarly, we will write e.g. $q_1 \subseteq q_2$, meaning $\mathsf{body}(q_1) \subseteq \mathsf{body}(q_2)$, and $c \in q$ meaning $c \in \mathsf{body}(q)$. We will also assume that the DL dialect in use includes the top concept ($\top$), a concept to which all individuals are assumed to belong to, and that every sav query always includes the atom $\top(x)$ in its body, although we will usually not write it. Thus, the sav query $q = \{\}$ is a shorthand for $\{ \langle x \rangle \mid \top(x)\}$, and similarly for any $q = \{c_1, \ldots, c_n\}$. Following the above assumptions, all definitions that follow will be stated only for sav queries, although more general formulations for general conjunctive queries might exist.

Given a KB $\mathcal{K}$, a sav query q and an interpretation $\mathcal{I}$ of $\mathcal{K}$, a *match* for q is a mapping $\pi : \mathsf{var}(q) \to \Delta^{\mathcal{I}}$ such that $\pi(u) \in C^{\mathcal{I}}$ for all $C(u) \in q$, and $(\pi(u), \pi(v)) \in r^{\mathcal{I}}$ for all $r(u, v) \in q$. Then, $a \in \mathsf{IN}$ is an *answer* for q in $\mathcal{I}$ if there is a match π for q such that $\pi(x) = a^{\mathcal{I}}$. Finally, $a \in \mathsf{IN}$ is a *certain answer* for q over $\mathcal{K}$ if in every model $\mathcal{I}$ of $\mathcal{K}$ there is a match π for q such that $\pi(x) = a^{\mathcal{I}}$. The set of certain answers to q is denoted by $\mathsf{cert}(q, \mathcal{K})$. Because obtaining the set of certain answers of a query over some KB typically requires reasoning with the axioms of the underlying TBox, such queries are characterized in the literature as *semantic queries*.

Let $\mathcal{K}$ be a knowledge base over a vocabulary $\mathcal{V}$, and $\mathcal{Q}$ the (possibly infinite) set of all sav queries over $\mathcal{V}$. We can partially order $\mathcal{Q}$ using *query subsumption*: A sav query q_2 *subsumes* a sav query q_1 (we write $q_1 \leqslant_S q_2$) iff there is a substitution θ s.t. $q_2\theta \subseteq q_1$ and θ leaves variable x unchanged. If q_1, q_2 are mutually subsumed, they are *syntactically equivalent* ($q_1 \equiv_S q_2$). $q \leqslant_S q'$ implies $\mathsf{cert}(q, \mathcal{K}) \subseteq \mathsf{cert}(q', \mathcal{K})$, since a match π for the variables of q can be composed with θ to produce a match $\pi\theta$ for the variables of q'. Let q, q' be queries s.t. $q' \subseteq q$. If q' is a minimal subset of q s.t. $q' \equiv_S q$, then q' is a *condensation* of q. If that minimal q' is the same as q, then q is *condensed* [53]. Intuitively, syntactically equivalent queries always have the same certain answers, and a condensation of some syntactically equivalent queries is the most compact query (not containing redundant atoms) that is syntactically equivalent to the rest.

Given the queries $q_1, q_2, \ldots, q_n$, a query least common subsumer $\mathsf{QLCS}(q_1, q_2, \ldots, q_n)$ of them is defined as a query q for which $q_1, q_2, \ldots, q_n \leqslant_S q$, and for all q' s.t. $q_1, q_2, \ldots, q_n \leqslant_S q'$ we have $q \leqslant_S q'$. The query least common subsumer can be seen as the most specific generalization of $q_1, q_2, \ldots, q_n$, and it is unique up to syntactical equivalence. We should note that this notion of query least common subsumer is different from the usual notion of least common subsumer of concepts which has been extensively studied for various description logic expressivities [54–57].

2.3. Graphs

A sav query q can be viewed as the directed labeled graph $\langle V, E, \ell_V, \ell_E \rangle$ (a *query graph*), where $V = \mathsf{var}(q)$ is the set of nodes, $E = \{\langle u, v \rangle \mid r(u, v) \in q\} \subseteq V \times V$ is the set of edges, $\ell_V : V \to 2^{\mathsf{CN}}$ with $\ell_V(u) = \{C \mid C(u) \in q\}$ is the node labeling function, and $\ell_E : E \to 2^{\mathsf{RN}}$ with $\ell_E(\langle u, v \rangle) = \{r \mid r(u, v) \in q\}$ is the edge labeling function. The answer variable is not explicitly identified since it is assumed to be always x. A sav query is *connected*, if the corresponding query graph is connected.

In the following, it will be useful to also treat atomic ABoxes as graphs. Similarly to the case of sav queries, an atomic ABox $\mathcal{A}$ can be represented as the graph $\langle V, E, \ell_V, \ell_E \rangle$ (an *ABox graph*), where $V \subseteq \mathsf{IN}$ is the set of nodes, in particular the subset of all individual names occurring in $\mathcal{A}$, $E = \{\langle a, b \rangle \mid r(a, b) \in \mathcal{A}\} \subseteq V \times V$ is the set of

labeled edges, $\ell_V : V \to 2^{\mathsf{CN}}$ with $\ell_V(a) = \{C \mid C(a) \in \mathcal{A}\}$ is the node labeling function, and $\ell_E : E \to 2^{\mathsf{RN}}$ with $\ell_E(\langle a,b\rangle) = \{r \mid r(a,b) \in \mathcal{A}\}$ is the edge labeling function.

Given two graphs $G_1 = \langle V_1, E_1, \ell_{V_1}, \ell_{E_1}\rangle$, $G_2 = \langle V_2, E_2, \ell_{V_2}, \ell_{E_2}\rangle$, a homomorphism $h : G_1 \to G_2$ is defined as a function from V_1 to V_2 that preserves edges and labels. More specifically it is such that: i) if $\langle a,b\rangle \in E_1$ then $\langle h(a), h(b)\rangle \in E_2$, ii) $\ell_{V_1}(a) \subseteq \ell_{V_2}(h(a))$, and iii) $\ell_{E_1}(\langle a,b\rangle) \subseteq \ell_{E_2}(\langle h(a), h(b)\rangle)$. If there exists a homomorphism from G_1 to G_2, we will write for simplicity $G_1 \to G_2$. When G_1 and G_2 are query graphs we will make the additional assumption that h preserves the answer variable, i.e. $h(x) = x$. If h is a bijection whose inverse is also a homomorphism, then h is an isomorphism. It is easy to see that the query graph of q_1 is homomorphic to the query graph of q_2 iff $q_2 \leqslant_S q_1$.

2.4. Rules

A (definite) *rule* is a fol expression of the form $\forall x_1 \ldots \forall x_n\,(c_1, \ldots, c_m \Rightarrow c_0)$, where c_i are atoms and $x_1, \ldots, x_n$ are all the variables appearing in the several c_i. The atoms $c_1, \ldots, c_m$ are the *body* of the rule, and c_0 its *head*. A rule over a vocabulary $\mathcal{V} = \langle \mathsf{CN}, \mathsf{RN}, \mathsf{IN}\rangle$ is a rule where each c_i is either $C(u)$, where $C \in \mathsf{CN}$, or $r(u,v)$, where $r \in \mathsf{RN}$. Assuming that c_0 is of the form $D(x)$, where $D \in \mathsf{CN}$, and that x appears in the body of such a rule, we will say that the rule is *connected*, if its body, seen as a sav query is connected. A rule is usually written as $c_1, \ldots, c_m \to c_0$.

2.5. Classifiers

A classifier is viewed as a function $F : \mathcal{D} \to \mathcal{C}$, where $\mathcal{D}$ is the classifier's domain (ex. images, audio, text), and $\mathcal{C}$ is a set of class names (ex. "Dog","Cat").

3. Framework

3.1. A motivating example

Integration of artificial intelligence methods and tools with biomedical and health informatics is a promising area, in which technologies for explaining machine learning classifiers will play a major role. In the context of the COVID-19 pandemic for example, black-box machine learning audio classifiers have been developed, which, given audio of a patient's cough, predict whether the person should seek medical advice or not [58]. In order to develop trust and use these classifiers in practice, it is important to explain their decisions, i.e. to provide convincing answers to the question "Why does the machine learning classifier suggest to *seek medical advice*?". There are post hoc explanation methods (both global and local) that try to answer this question in terms of the input of the black-box classifier, which in this case is audio signals. Although this information could be useful for AI engineers and developers, it is not understandable to most medical experts and end users, since audio signals themselves are obscure sub-symbolic data. Thus, it is difficult to convincingly meet the explainability requirements and develop the necessary trust to utilize the black-box classifier in practice, unless explanations are expressed in the terminology used by the medical experts (using terms like "sore throat", "dry cough" etc).

In the above context, suppose we have a dataset of audio signals of coughs which have been characterized by medical professionals by using standardized clinical terms, such as "Loose Cough", "Dry Cough", "Dyspnoea", in addition to a knowledge base in which these terms and relationships between them are defined, such as SNOMED-CT [13]. For example, consider such a dataset with coughs from five patients $\mathsf{p1}, \mathsf{p2}, \mathsf{p3}, \mathsf{p4}, \mathsf{p5}$ (obviously in practice we may need a much more extended set of patients) with characterizations from the medical experts: "$\mathsf{p1}$ has a sore throat", "$\mathsf{p2}$ has dyspnoea", "$\mathsf{p3}$ has a sore throat and dyspnoea", "$\mathsf{p4}$ has a sore throat and a dry cough", "$\mathsf{p5}$ has a sore throat and a loose cough". We also have available relationships between these terms as defined in SNOMED-CT like "Loose Cough is Cough", "Dry Cough is Cough", "Cough is Lung Finding", and "Dyspnoea is Lung Finding".

Now assume that a black-box classifier predicts that $\mathsf{p3}, \mathsf{p4}$, and $\mathsf{p5}$ should seek medical advice, while $\mathsf{p1}$ and $\mathsf{p2}$ should not. Then we can say that: on this dataset, if a patient has a sore throat and a lung finding then it is classified positively by the specific classifier, i.e. the classifier suggests "seek medical advice". Depending on the

characteristics of the dataset itself and its ability to cover interesting examples, such an extracted rule could help the medical professional understand why the black-box is making decisions in order to build necessary trust, but also it could help an AI engineer improve the model's performance by indicating potential biases. Importantly, even though we cannot know that the black-box classifier is actually using the semantics that are used to describe the data, this approach gives a user control over what information they want to examine the classifier for. For example, a system developer and a medical professional would use different semantic annotations because explanations are useful to them at different levels of abstraction.

The utilization of such datasets and knowledge is main strength of the proposed framework, but it can also be its main weakness. Specifically, erroneous labeling during the construction of the dataset could lead to misleading explanations, as could biases in the data. For example, if the length of the recordings of the coughs were also available to us, and every instance of "Sore Throat" also had a short length recording, then we would not be able to distinguish between these two characteristics, and the resulting explanations would not be useful.

3.2. Explaining opaque machine learning classifiers

Explanation of opaque machine learning classifiers is based on a dataset that we call *Explanation Dataset* (see Fig. 1), containing *exemplar patterns*, that are actually characteristic examples from the set of elements that the unknown classifier takes as input. Machine learning classifiers usually take as input element features (like the cough audio signal mentioned in the motivating example). The explanation dataset additionally contains a semantic description of the exemplars in terms that are understandable by humans (like "dry cough" mentioned in the motivating example). Taking the output of the unknown classifier (the classification class) for all the exemplars, we construct the *Exemplar Data Classification* information (see Fig. 1) thus we know the exact set of exemplars that are classified by the unknown classifier to a specific class (like the "seek medical advice" class mentioned in the motivating example). Using the knowledge represented in the *Exemplar Data Semantic Description* (see Fig. 1), we define the following *reverse semantic query answering problem*: "given a set of exemplars and their semantic description find *intuitive and understandable* semantic queries that have as certain answers *exactly* this set out of all the exemplars of the explanation dataset". The specific problem is interesting, with certain difficulties and computationally very demanding [43, 45, 46, 51, 59]. Here, by extending a method presented in [60] we present an *Explainer* (see Fig. 1) that tries to solve this problem, following a *Semantic Query Heuristic Search* method, that searches in the *Semantic Query Space* (the set containing all queries that have non-empty certain answer set) for queries that are solutions to the above problem.

Introductory definitions and interesting theoretical results concerning the above approach are presented in [60]. Here, we reproduce some of them and introduce others, in order to develop the necessary framework for presenting the proposed method.

A defining aspect is that the rule explanations are provided in terms of a specific vocabulary. To do this in practice, we require a set of items (exemplar data) which can: a) be fed to a black-box classifier and b) have a semantic description using the desired terminology. As mentioned before, here we consider that: a) the exemplar data has for its items all the information that the unknown classifier needs in order to classify them (the necessary *features*), and b) the semantic data descriptions are expressed as Description Logics knowledge bases (see Fig. 1).

Definition 1 ([46]). *Let $\mathcal{D}$ be a domain of item feature data, $\mathcal{C}$ a set of classes, and $\mathcal{V} = \langle \mathsf{IN}, \mathsf{CN}, \mathsf{RN} \rangle$ a vocabulary such that $\mathcal{C} \cup \{\mathsf{Exemplar}\} \subseteq \mathsf{CN}$. Let also $\mathsf{EN} \subseteq \mathsf{IN}$ be a set of* exemplars. *An* explanation dataset *$\mathcal{E}$ in terms of $\mathcal{D}$, $\mathcal{C}$ and $\mathcal{V}$ is a tuple $\mathcal{E} = \langle \mathcal{M}, \mathcal{S} \rangle$, where $\mathcal{M} : \mathsf{EN} \rightarrow \mathcal{D}$ is a mapping from the exemplars to the item feature data, and $\mathcal{S} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a DL knowledge base over $\mathcal{V}$ such that $\mathsf{Exemplar}(a) \in \mathcal{A}$ iff $a \in \mathsf{EN}$, the elements of $\mathcal{C}$ do not appear in $\mathcal{S}$, and* Exemplar *and the elements of* EN *do not appear in $\mathcal{T}$.*

Intuitively, an explanation dataset contains a set of exemplar data (i.e. characteristic items in $\mathcal{D}$ which can be fed to the unknown classifier) semantically described in terms of a specific vocabulary $\mathcal{V}$; the semantic descriptions are in the knowledge base $\mathcal{S}$. Each exemplar data item is represented in $\mathcal{S}$ by an individual name; these individual names make up the set of exemplars EN, and each one of them is mapped to the corresponding exemplar data item by $\mathcal{M}$. Because the knowledge encoded in $\mathcal{S}$ may involve also other individuals, the Exemplar concept exists to identify exactly those individuals that correspond to exemplar data within the knowledge base. Given a classifier

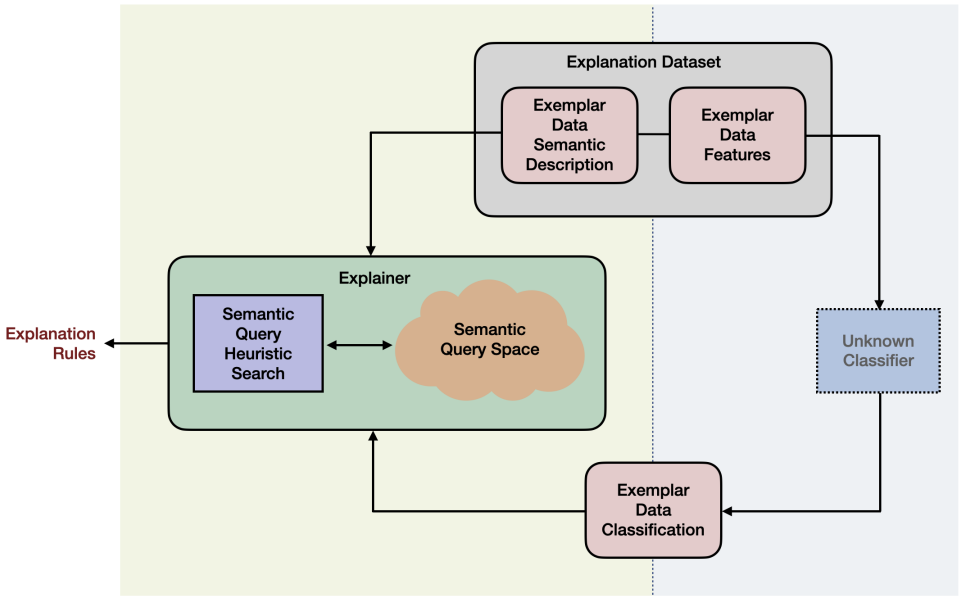


Fig. 1. A framework for explaining opaque machine learning classifiers

$F : \mathcal{D} \to \mathcal{C}$ and a set of individuals $\mathcal{I} \subseteq \mathsf{EN}$, the positive set (pos-set) of F on $\mathcal{I}$ for class $C \in \mathcal{C}$ is $\mathsf{pos}(F, \mathcal{I}, C) = \{a \in \mathcal{I} : F(\mathcal{M}(a)) = C\}$. Based on the classifier's prediction on the exemplar data for a class (i.e. the pos-set) we can produce explanation rules by grouping them according to their semantic description in the explanation dataset.

Definition 2 ([46]). *Let $F : \mathcal{D} \to \mathcal{C}$ be a classifier, $\mathcal{E} = \langle \mathcal{M}, \mathcal{S} \rangle$ an explanation dataset in terms of a domain $\mathcal{D}$ and a set of classes $\mathcal{C}$, and an appropriate vocabulary $\mathcal{V} = \langle \mathsf{CN}, \mathsf{RN}, \mathsf{IN} \rangle$ (with $\mathcal{C} \subseteq \mathsf{CN}$). Given a concept $C \in \mathcal{C}$, a connected rule of the form*

$$\mathsf{Exemplar}(x), c_1, c_2, \ldots, c_n \to C(x)$$

where c_i is an atom $D(u)$ or $r(u, v)$, where $D \in \mathsf{CN}$, $r \in \mathsf{RN}$, and u, v are variables, is an explanation rule *of F for class C over $\mathcal{E}$. We denote the above rule by $\rho(F, \mathcal{E}, C)$, or simply by ρ whenever the context is clear. We may also omit $\mathsf{Exemplar}(x)$ from the body, since it is a conjunct of any explanation rule. The explanation rule ρ is* correct *if and only if*

$$\mathsf{fol}(\mathcal{S} \cup \{\mathsf{Exemplar} \sqsubseteq \bigsqcup_{a \in \mathsf{EN}} \{a\}\} \cup \{C(a) \mid a \in \mathsf{pos}(F, \mathsf{EN}, C)\}) \models \rho$$

where $\mathsf{fol}(\mathcal{K})$ is the first-order logic translation of the DL knowledge base $\mathcal{K}$.

Note that in the above, the argument of $\mathsf{fol}(\cdot)$ is a KB, i.e. a set of DL axioms, in particular all axioms originally in S plus the technical axiom $\mathsf{Exemplar} \sqsubseteq \bigsqcup_{a \in \mathsf{EN}} \{a\}$, to force the interpretation of $\mathsf{Exemplar}$ to be contained in the interpretation of the elements of EN, and the assertions $C(a)$. (Note that because EN is a set of individuals, the expression $\bigsqcup_{a \in \mathsf{EN}} \{a\}$ makes use of the union and nominal DL constructors (one-of constructor) to form a concept by enumerating its members.)

Explanation rules describe sufficient conditions (the body of the rule) for an item to be classified in the class indicated at the head of the rule by the classifier under investigation. The correctness of a rule indicates that the rule covers every $a \in \mathsf{EN}$, meaning that for each exemplar of $\mathcal{S}$, either the body of the rule is not true, or both the body and the head of the rule are true. Of course we cannot know if the classifier is using the particular semantics, however, depending on the explanation dataset, explanation rules can provide useful information that can indicate potential biases of the classifier, that are expressed in the desired terminology.

Example 1. *Suppose we have the problem described in the motivating example of Section 3.1 with black-box classifiers predicting whether a person should seek medical advice based on audios of their cough, and that we are creating an explanation dataset in order to explain the respective classifiers. The vocabulary used should be designed so that it will produce meaningful explanations to the end-user, which in our case would probably be a doctor or another professional of the medical domain. In this case, it should contain concepts for the different medical terms like the findings (cough, sore throat, etc.), and according to the definition of the* explanation dataset*, the class categories (seek medical advice, or not) and the concept* Exemplar *as concept names (*CN*). Additionally, a role linking patients to the respective findings should exist in the role names (*RN*), and the patients as well as the findings themselves would be the individual names (*IN*). Following this, we create the vocabulary ($\mathcal{V}$) as shown below:*

$$\mathsf{IN} = \{\mathsf{p1}, \mathsf{p2}, \mathsf{p3}, \mathsf{p4}, \mathsf{p5}, \mathsf{s1}, \mathsf{s2}, \mathsf{s3}, \mathsf{s4}, \mathsf{s5}, \mathsf{s6}, \mathsf{s7}, \mathsf{s8}\}$$

$$\mathsf{CN} = \{\mathsf{DryCough}, \mathsf{LooseCough}, \mathsf{Cough}, \mathsf{SoreThroat}, \mathsf{LungFinding}, \mathsf{Finding}, \mathsf{Dyspnoea}, \mathsf{MedicalAdvice}, \mathsf{NoMedicalAdvice}, \mathsf{Exemplar}\}$$

$$\mathsf{RN} = \{\mathsf{hasFinding}\}$$

*Having the domain $\mathcal{D}$ (audio signals), the set of classes $\mathcal{C}$ (*MedicalAdvice, NoMedicalAdvice*), and the vocabulary $\mathcal{V}$ we can now define the explanation dataset $\mathcal{E} = \langle \mathcal{M}, \mathcal{S} \rangle$. The set of exemplars (*EN*) in our case contains the patient individuals of* IN*, so* $\mathsf{EN} = \{\mathsf{p1}, \mathsf{p2}, \mathsf{p3}, \mathsf{p4}, \mathsf{p5}\}$*. The mapping $\mathcal{M}$ of the explanation dataset links these exemplars to the respective audio of each patient. The only thing that is missing from our explanation dataset is the knowledge base $\mathcal{S}$ consisting of an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$. The ABox contains information regarding the patient audio characterizations from the medical professionals ("*p1 *has a sore throat", "*p2 *has dyspnoea", "*p3 *has a sore throat and dyspnoea", "*p4 *has a sore throat and a dry cough", "*p5 *has a sore throat and a loose cough") as well as the assertions regarding the exemplar status of individuals, while the TBox contains relationships between the medical terms as defined in SNOMED-CT, as shown below:*

$$\begin{aligned}\mathcal{A} = \{&\mathsf{Exemplar}(\mathsf{p1}), \mathsf{Exemplar}(\mathsf{p2}), \mathsf{Exemplar}(\mathsf{p3}), \mathsf{Exemplar}(\mathsf{p4}), \mathsf{Exemplar}(\mathsf{p5}), \mathsf{hasFinding}(\mathsf{p1}, \mathsf{s1}),\\ &\mathsf{hasFinding}(\mathsf{p2}, \mathsf{s2}), \mathsf{hasFinding}(\mathsf{p3}, \mathsf{s3}), \mathsf{hasFinding}(\mathsf{p3}, \mathsf{s4}), \mathsf{hasFinding}(\mathsf{p4}, \mathsf{s5}), \mathsf{hasFinding}(\mathsf{p4}, \mathsf{s6}),\\ &\mathsf{hasFinding}(\mathsf{p5}, \mathsf{s7}), \mathsf{hasFinding}(\mathsf{p5}, \mathsf{s8}), \mathsf{SoreThroat}(\mathsf{s1}), \mathsf{Dyspnoea}(\mathsf{s2}), \mathsf{SoreThroat}(\mathsf{s3}),\\ &\mathsf{Dyspnoea}(\mathsf{s4}), \mathsf{SoreThroat}(\mathsf{s5}), \mathsf{DryCough}(\mathsf{s6}), \mathsf{SoreThroat}(\mathsf{s7}), \mathsf{LooseCough}(\mathsf{s8})\}\end{aligned}$$

$$\begin{aligned}\mathcal{T} = \{&\mathsf{LooseCough} \sqsubseteq \mathsf{Cough}, \mathsf{DryCough} \sqsubseteq \mathsf{Cough}, \mathsf{Cough} \sqsubseteq \mathsf{LungFinding}, \mathsf{LungFinding} \sqsubseteq \mathsf{Finding},\\ &\mathsf{Dyspnoea} \sqsubseteq \mathsf{LungFinding}, \mathsf{SoreThroat} \sqsubseteq \mathsf{Finding}\}\end{aligned}$$

Now suppose that a black-box classifier F predicts that p3, p4*, and* p5 *should seek medical advice, while* p1 *and* p2 *do not need to (same as the motivating example of Section 3.1). The explanation rule*

$$\rho_1 : \mathsf{Exemplar}(x), \mathsf{hasFinding}(x, y), \mathsf{SoreThroat}(y), \mathsf{hasFinding}(x, z), \mathsf{LungFinding}(z) \rightarrow \mathsf{MedicalAdvice}(x)$$

for that classifier based on the explanation dataset $\mathcal{E} = \langle \mathcal{M}, \mathcal{S} \rangle$ is a correct rule, as well as the explanation rule

$$\rho_2 : \mathsf{Exemplar}(x), \mathsf{hasFinding}(x, y), \mathsf{Cough}(y) \rightarrow \mathsf{MedicalAdvice}(x),$$

while the rules

$$\rho_3 : \mathsf{Exemplar}(x), \mathsf{hasFinding}(x, y), \mathsf{SoreThroat}(y) \rightarrow \mathsf{MedicalAdvice}(x) \quad \textit{and}$$

$$\rho_4 : \mathsf{Exemplar}(x), \mathsf{hasFinding}(x, y), \mathsf{Dyspnea}(y) \rightarrow \mathsf{MedicalAdvice}(x)$$

are not correct, since $\mathsf{fol}(\mathcal{S}') \models \rho_1$ *and* $\mathsf{fol}(\mathcal{S}') \models \rho_2$*, but* $\mathsf{fol}(\mathcal{S}') \not\models \rho_3$ *and* $\mathsf{fol}(\mathcal{S}') \not\models \rho_4$*, where* $\mathcal{S}' = \mathcal{S} \cup \{\mathsf{Exemplar} \sqsubseteq \{a \mid a \in \mathsf{EN}\} \cup \{\mathsf{MedicalAdvice}(a) \mid a \in \mathsf{pos}(F, \mathsf{EN}, \mathsf{MedicalAdvice})\}$.

As mentioned in Section 2, a sav query is an expression of the form $\{c_1, \ldots, c_n\}$, an expression that resembles the body of explanation rules. By treating the bodies of explanation rules as queries, the problem of computing explanations can be solved as a query reverse engineering problem [46].

Definition 3 ([46]). *Let $F : \mathcal{D} \to \mathcal{C}$ be a classifier, $\mathcal{E} = \langle \mathcal{M}, \mathcal{S} \rangle$ an explanation dataset in terms of $\mathcal{D}$, $\mathcal{C}$ and an appropriate vocabulary $\mathcal{V}$, and $\rho(F, \mathcal{E}, C)$: $\mathsf{Exemplar}(x), c_1, c_2, \ldots, c_n \to C(x)$ an explanation rule. The sav query*

$$q_\rho \doteq \{\mathsf{Exemplar}(x), c_1, c_2, \ldots, c_n\}$$

is the explanation rule query *of explanation rule ρ.*

This query reverse engineering problem follows the *Query by Example* paradigm or QbE. This term refers to reverse engineering queries that contain some positive examples in their answer set but not any negative examples and is widely used in the related literature [20, 36, 47–50, 61]. We give a short formal definition of QbE adapted from [48].

Definition 4 (Query by Example). *Given a DL knowledge base $\mathcal{S} = \langle T, A \rangle$, two sets $S^+, S^- \subseteq \mathsf{IN}$ and a query language $\mathcal{Q}$. Is there a query $q \in \mathcal{Q}$ such that $S^+ \subseteq \mathsf{cert}(q, \mathcal{S})$ and $S^- \cap \mathsf{cert}(q, \mathcal{S}) = \varnothing$?*

In our case, S^+ would be $\mathsf{pos}(F, \mathsf{EN}, C)$ and $S^- = \mathsf{EN} \setminus \mathsf{pos}(F, \mathsf{EN}, C)$, while $\mathcal{Q}$, the language of conjunctive queries.

The relationship between the properties of explanation rules and the respective queries allows us to detect and compute correct rules based on the certain answers of the respective explanation rule queries, as shown in Theorem 1.

Theorem 1 ([46]). *Let $F : \mathcal{D} \to \mathcal{C}$ be a classifier, $\mathcal{E} = \langle \mathcal{M}, \mathcal{S} \rangle$ an explanation dataset in terms of $\mathcal{D}$, $\mathcal{C}$ and an appropriate vocabulary $\mathcal{V}$, $\rho(F, \mathcal{E}, C)$: $\mathsf{Exemplar}(x), c_1, c_2, \ldots, c_n \to C(x)$ an explanation rule, and q_ρ the explanation rule query of ρ. The explanation rule ρ is correct if and only if*

$$\mathsf{cert}(q_\rho, \mathcal{S}) \subseteq \mathsf{pos}(F, \mathsf{EN}, C)$$

Proof. Let $\mathcal{S} = \langle \mathcal{T}, \mathcal{A} \rangle$. Because by definition $\mathsf{Exemplar}(a) \in \mathcal{A}$ iff $a \in \mathsf{EN}$ and $\mathsf{Exemplar}$ does not appear anywhere in $\mathcal{T}$, we have

$$\begin{aligned}
\mathsf{cert}(q_\rho, \mathcal{S}) &= \mathsf{cert}(\{\mathsf{Exemplar}, c_1, \ldots, c_n\}, \langle \mathcal{T}, \mathcal{A} \rangle) \\
&= \mathsf{EN} \cap \mathsf{cert}(\{c_1, \ldots, c_n\}, \langle \mathcal{T}, \mathcal{A} \rangle) \\
&= \mathsf{cert}\left(\{\mathsf{Exemplar}\}, \left\langle \left\{ \mathsf{Exemplar} \sqsubseteq \bigsqcup_{a \in \mathsf{EN}} \{a\} \right\}, \mathcal{A} \right\rangle\right) \cap \mathsf{cert}(\{c_1, \ldots, c_n\}, \langle \mathcal{T}, \mathcal{A} \rangle) \\
&= \mathsf{cert}\left(\{\mathsf{Exemplar}, c_1, \ldots, c_n\}, \left\langle \mathcal{T} \cup \left\{ \mathsf{Exemplar} \sqsubseteq \bigsqcup_{a \in \mathsf{EN}} \{a\} \right\}, \mathcal{A} \right\rangle\right) \\
&= \mathsf{cert}\left(q_\rho, \left\langle \mathcal{T} \cup \left\{ \mathsf{Exemplar} \sqsubseteq \bigsqcup_{a \in \mathsf{EN}} \{a\} \right\}, \mathcal{A} \right\rangle\right)
\end{aligned}$$

Because by definition C does not appear anywhere in $\mathcal{S}$), we have also that $\mathsf{cert}(q_\rho, \mathcal{S}) = \mathsf{cert}(q_\rho, \mathcal{S}')$, where $\mathcal{S}' = \mathcal{S} \cup \{\mathsf{Exemplar} \sqsubseteq \bigsqcup_{a \in \mathsf{EN}} \{a\}\} \cup \{C(a) \mid a \in \mathsf{pos}(F, \mathsf{EN}, C)\}\}$, since the assertions $C(a)$ are not involved neither in the query nor in $\mathcal{S}$ and hence have no effect.

By definition of a certain answer, $e \in \mathsf{cert}(q, \mathcal{K})$ iff for every model $\mathcal{I}$ of $\mathcal{K}$ there is a match π s.t. $\pi(x) = e^{\mathcal{I}}$ and $\pi(u) \in D^{\mathcal{I}}$ for all $D(u) \in q$ and $(\pi(u), \pi(v)) \in r^{\mathcal{I}}$ for all $r(u, v) \in q$.

Assume that ρ is correct and let $e \in \mathsf{cert}(q_\rho, \mathcal{S})$. We have proved that also $e \in \mathsf{cert}(q_\rho, \mathcal{S}')$. Because ρ is correct, by Def. 3 it follows that every model $\mathcal{I}$ of $\mathcal{S}'$ is also a model of ρ. Because the body of q_ρ is the same as the body of

ρ, π makes true both the body of ρ and the head of ρ, which is $C(x)$, hence $e^{\mathcal{I}} \in C^{\mathcal{I}}$. It follows that $C(e)$ is true in $\mathcal{I}$. But the only assertions of the form $C(e)$ in $\mathcal{S}'$ are the assertions $\{C(a) \mid a \in \mathsf{pos}(F, \mathsf{EN}, C)\}$, thus $e \in \mathsf{pos}(F, \mathsf{EN}, C)$.

For the inverse, assume that $\mathsf{cert}(q_\rho, \mathcal{S}) \subseteq \mathsf{pos}(F, \mathsf{EN}, C)$, equivalently $\mathsf{cert}(q_\rho, \mathcal{S}') \subseteq \mathsf{pos}(F, \mathsf{EN}, C)$. Thus if $e \in \mathsf{cert}(q_\rho, \mathcal{S})$ then $e^{\mathcal{I}} \in C^{\mathcal{I}}$. Since this holds for every model $\mathcal{I}$ of $\mathcal{S}'$ and the body of q_ρ is the same as the body of ρ, it follows that $\mathcal{I}$ is also a model of ρ, i.e. ρ is correct.

□

Theorem 1 shows a useful property of the certain answers of the explanation rule query of a correct rule ($\mathsf{cert}(q, \mathcal{S}) \subseteq \mathsf{pos}(F, \mathsf{EN}, C)$) that can be utilized in order to identify and produce correct rules. Intuitively, an explanation rule is correct, if all of the certain answers of the respective explanation rule query are mapped by $\mathcal{M}$ to data which is classified in the class indicated at the head of the rule. However, it is obvious that according to the above we can have correct rules that barely approximate the behaviour of the classifier (e.g. an explanation rule query with only one certain answer that is in the pos-set of the classifier), while other correct rules might exactly match the output of the classifier (e.g. a query q for which $\mathsf{cert}(q, \mathcal{S}) = \mathsf{pos}(F, \mathsf{EN}, C)$). Thus, it is useful to define a *recall* metric for explanation rule queries by comparing the set of certain answers with the pos-set of a class C:

$$\mathsf{recall}(q, \mathcal{E}, C) = \frac{|\mathsf{cert}(q, \mathcal{S}) \cap \mathsf{pos}(F, \mathsf{EN}, C)|}{|\mathsf{pos}(F, \mathsf{EN}, C)|},$$

assuming that $\mathsf{pos}(F, \mathsf{EN}, C) \neq \varnothing$.

Example 2. *Continuing Example 1, we can create the explanation rule queries of the respective explanation rules of the example as follows:* $q_1(x) = \{\mathsf{Exemplar}(x), \mathsf{hasFinding}(x, y), \mathsf{SoreThroat}(y), \mathsf{hasFinding}(x, z), \mathsf{LungFinding}(z)\}$ *as the explanation rule query of* ρ_1*,* $q_2(x) = \{\mathsf{Exemplar}(x), \mathsf{hasFinding}(x, y), \mathsf{Cough}(y)\}$ *as the explanation rule query of* ρ_2*,* $q_3(x) = \{\mathsf{Exemplar}(x), \mathsf{hasFinding}(x, y), \mathsf{SoreThroat}(y)\}$ *as the explanation rule query of* ρ_3*, and* $q_4(x) = \{\mathsf{Exemplar}(x), \mathsf{hasFinding}(x, y), \mathsf{Dyspnoea}(y)\}$ *as the explanation rule query of* ρ_4*.*

For the above queries we can retrieve their certain answers over our knowledge base $\mathcal{S}$*, and get* $\mathsf{cert}(q_1, \mathcal{S}) = \{\mathsf{p3}, \mathsf{p4}, \mathsf{p5}\}$*,* $\mathsf{cert}(q_2, \mathcal{S}) = \{\mathsf{p4}, \mathsf{p5}\}$*,* $\mathsf{cert}(q_3, \mathcal{S}) = \{\mathsf{p1}, \mathsf{p3}, \mathsf{p4}, \mathsf{p5}\}$*, and* $\mathsf{cert}(q_4, \mathcal{S}) = \{\mathsf{p2}, \mathsf{p3}\}$*.*

With respect to the classifier F *of Example 1, for which* $\mathsf{pos}(F, \mathsf{EN}, \mathsf{MedicalAdvice}) = \{\mathsf{p3}, \mathsf{p4}, \mathsf{p5}\}$*, we see, as the theorem states, that for the correct rules* ρ_1 *and* ρ_2 *it holds that* $\mathsf{cert}(q_1, \mathcal{S}) \subseteq \mathsf{pos}(F, \mathsf{EN}, \mathsf{MedicalAdvice})$ *and* $\mathsf{cert}(q_2, \mathcal{S}) \subseteq \mathsf{pos}(F, \mathsf{EN}, \mathsf{MedicalAdvice})$*, while for rules* ρ_3 *and* ρ_4 *that are not correct, it holds that* $\mathsf{cert}(q_3, \mathcal{S}) \not\subseteq \mathsf{pos}(F, \mathsf{EN}, \mathsf{MedicalAdvice})$*, and* $\mathsf{cert}(q_4, \mathcal{S}) \not\subseteq \mathsf{pos}(F, \mathsf{EN}, \mathsf{MedicalAdvice})$*, respectively.*

The explanation framework described in this section provides the necessary expressivity to formulate accurate rules, using a desired terminology, even for complex problems [46]. However, some limitations of the framework, like only working with correct rules, can be a significant drawback for explanation methods built on top of that. An explanation rule query might not be correct due to the existence of individuals in the set of certain answers which are not in the pos-set. By viewing these individuals as exceptions to a rule, we are able to provide as an explanation a rule that is not correct, along with the exceptions which would make it correct if they were omitted from the explanation dataset; the exceptions could provide useful information to an end-user about the classifier under investigation. Thus, we extend the existing framework by introducing correct explanation rules with exceptions, as follows:

Definition 5. *Let* $F : \mathcal{D} \to \mathcal{C}$ *be a classifier,* $\mathcal{E} = \langle \mathcal{M}, \mathcal{S} \rangle$ *an explanation dataset in terms of* $\mathcal{D}$*,*$\mathcal{C}$ *where* $\mathcal{S}$ *is a knowledge base* $\mathcal{S} = \langle \mathcal{A}, \mathcal{T} \rangle$*,* EN *the set of exemplars of* $\mathcal{E}$*, and let* EX *be a subset of* EN*. An explanation rule* $\rho(F, \mathcal{E}, C)$ *is correct with exceptions* EX *for class* C *if the rule* $\rho(F, \mathcal{E}', C)$ *is correct for class* C*, where* $\mathcal{E}' = \langle \mathcal{M}, \mathcal{S}' \rangle$*, and* $\mathcal{S}'$ *is the knowledge* $\mathcal{S}' = \langle \mathcal{A}', \mathcal{T} \rangle$*, and* $\mathcal{A}' = \mathcal{A} \setminus \{\mathsf{Exemplar}(a) \mid a \in \mathsf{EX}\}$*.*

Since we allow exceptions to explanation rules, it is useful to define a measure of precision of the corresponding explanation rule queries as

$$\mathsf{precision}(q, \mathcal{E}, C) = \frac{|\mathsf{cert}(q, \mathcal{S}) \cap \mathsf{pos}(F, \mathsf{EN}, C)|}{|\mathsf{cert}(q, \mathcal{S})|}.$$

if $\mathsf{cert}(q, \mathcal{S}) \neq \varnothing$ and $\mathsf{precision}(q, \mathcal{E}, C) = 0$ otherwise.

Obviously, if the precision of a rule query is 1, then it represents a correct rule, otherwise it is correct with exceptions. Furthermore, we can use the Jaccard similarity between the set of certain answers of the explanation rule query and the pos-set, as a generic measure which combines recall and precision to compare the two sets of interest as:

$$\mathsf{degree}(q, \mathcal{E}, C) = \frac{|\mathsf{cert}(q, \mathcal{S}) \cap \mathsf{pos}(F, \mathsf{EN}, C)|}{|\mathsf{cert}(q, \mathcal{S}) \cup \mathsf{pos}(F, \mathsf{EN}, C)|}.$$

Example 3. *The rules ρ_3 and ρ_4 of Example 2 that are not correct; they are correct with exceptions. Table 3 shows the precision, recall, and degree metrics of the explanation rule queries of Example 2 along with the exceptions* EX *of the respective rules.*

Table 3
Metrics and Exceptions of the example Explanation Rules and the respective Explanation Rule Queries.

Rule	Query	Recall	Precision	Degree	Exceptions (EX)
ρ_1	q_1	1.0	1.0	1.0	{}
ρ_2	q_2	0.67	1.0	0.67	{}
ρ_3	q_3	1.0	0.75	0.75	{p1}
ρ_4	q_4	0.33	0.5	0.25	{p2}

It can often be useful to present a set of rules to the end user instead of a single one. This could be, for example, because the pos-set cannot be captured by a single correct rule but only a set of correct rules. This is a strategy commonly employed by other rule based systems such as those we employed in Sec. 5 to compare our algorithms to, namely RuleMatrix [26] and Skope-Rules [25]. The metrics already defined for a single query can be expanded to a set of queries $Q = \{q_1, q_2, \ldots q_n\}$, by simply expanding the definition of certain answers for a set of queries to $\mathsf{cert}(Q, \mathcal{S}) = \bigcup_{i=1}^{n} \mathsf{cert}(q_i, \mathcal{S})$.

Since, as we have previously mentioned, we do not have any formal guarantees that the semantics expressed in the explanations extracted by the reverse engineering process are linked to the ones used by the classifier, these measures provide some confidence about the existence of such a link. Assuming that the explanations produced are of high quality (high precision, low recall, short in length) and that the explanation dataset is constructed in a way that takes into account possible biases, it is reasonable to assume that there is at least some correlation between the semantics that the explanation presents with those that the classifier is using.

Unfortunately, as the semantic descriptions become richer, it is more likely that a query that perfectly separates the positive and negative individuals exists, since the space of queries that can be expressed using the vocabulary of the descriptions becomes larger. It would be helpful to have some statistical measures after the reverse engineering process that express how likely it is that the query(ies) perform well by coincidence, rather than due to a link between the semantics expressed in the explanation dataset and the ones used by the classifier. Computing such measures is far from trivial due to the complex structure of the Query Space [46]. In this work, we employ some alternatives in our experiments. Namely, using a holdout set in order to check how well the rules produced generalize (Mushrooms experiment, section 5.1), and employing other XAI methods to cross-reference the explanations produced (Visual Genome experiment, section 5.3). We consider a thorough examination of such methods to be out of the scope of this work.

4. Computation of Explanations

From Section 3, we understand that the problem that we try to solve is closely related to the query reverse engineering problem, since we need to compute queries given a set of individuals. However, since in most cases there is not a single query that fits our needs (have as certain answers the pos-set of the classifier), we need to find (out of all the semantic queries that have a specific certain answer set) a set of queries that accurately describe the

classifier under investigation. Therefore, the problem can also be seen as a heuristic search problem. The duality of rules and queries within our framework, reduces the search of correct rules (with exceptions) to the search of queries that contain elements of the pos-set in their certain answers. Reverse engineering queries for subsets of EN is challenging for the following reasons:

- The subsets of EN for which there exists a correct rule query ($\{I \mid I \subseteq \mathsf{EN}$, there exists q s.t. $\mathsf{cert}(q,\mathcal{S}) = I\}$) can potentially be exponentially many ($2^{|\mathsf{EN}|}$).
- The Query Space i.e. the set containing all queries that have non-empty certain answer set ($\{q \mid \mathsf{cert}(q,\mathcal{S}) \cap \mathsf{EN} \neq \varnothing\}$) can potentially be infinite [46].
- For any subset I of EN, the number of queries s.t. $\mathsf{cert}(q,\mathcal{S}) = I$ can be zero, positive or infinite.
- Computing the certain answers of arbitrary queries can be exponentially slow, so it is computationally prohibitive to evaluate each query under consideration while exploring the query space.

The difficulties described above are addressed in the following ways:

- In this paper, we only consider knowledge bases of which the TBox can be eliminated (such as RL [62]; see also the last paragraph of this section). This enables us to create finite queries that contain all the necessary conjuncts to fully describe each individual. We are then able to merge those queries to create descriptions of successively larger subsets of EN.
- We do not directly explore the subsets of EN for which there exists a correct rule query, and the computation of certain answers is not required for the algorithms. Instead, we explore the Query Space blindly, but heuristically. We create queries that are guaranteed to be within the Query Space and are also guaranteed to contain heuristically selected individuals in their certain answers, without knowing their exact certain answers. The heuristic we employ, aids us in selecting similar individuals to merge. Intuitively, this helps us create queries that introduce as few as possible unwanted certain answers.
- We are not concerned with the entire set of queries with non-empty sets of certain answers, but only with queries which have specific characteristics in order to be used as explanations. Specifically, the queries have to be short in length, with no redundant conjuncts, have as certain answers elements of the pos-set of the class under investigation, and as few others as possible.
- One of the proposed algorithms guarantees that given a set I, if a query q exists s.t. $\mathsf{cert}(q,\mathcal{S}) = I$, then it will find at least one such query. If there do not exist such queries then, since the computation of certain answers is not involved in the algorithms, the result of the heuristic search will be a "good guess" of queries which have similar certain answer sets to I. If there exist infinite such queries, then we do not have a guarantee that we have found the shortest, most understandable one, however the proposed algorithms take care to create queries with few variables (see Alg. 2 and section 4.3.2).

In the following we describe the proposed algorithms for computing explanations. The core algorithm, which is outlined as Alg. 1 and we call KGrules-H, takes as input an atomic ABox $\mathcal{A}$ and a set of individual names I, and produces a list of queries. It is assumed that both $\mathcal{A}$ and I are obtained from an explanation dataset $\mathcal{E}$. In particular, I is a subset of the respective EN corresponding to a pos-set of a classifier F to be explained for some class C, i.e. $\mathsf{pos}(F, \mathsf{EN}, C)$, and $\mathcal{A}$ is an atomic ABox containing all the available knowledge about the individuals in I encoded in the knowledge base of $\mathcal{E}$. The output queries are intended to serve as explanation rule queries for class C. An outline of how KGrules-H is used to generate explanations is shown in Fig. 2.

The algorithm starts by initializing an empty list of queries S, and by creating an initial description for each individual in I in the form of a most specific query (MSQ). A detailed definition of a MSQ is given in Section 4.1; intuitively, an MSQ of an individual a for an ABox $\mathcal{A}$ is a sav query q that captures the maximum possible information about a and is such that $a \in \mathsf{cert}(\mathsf{MSQ}(q,\mathcal{A}))$. Given these descriptions, one for each individual, which are added in a set L, the algorithm then tries to combine the elements of L in order to generate more general descriptions. In particular, at each iteration of the while loop, it selects two queries from L and merges them. The queries to be merged are selected based on their dissimilarity; two most similar queries are selected and merged. The intuition is that by merging relatively similar queries, the resulting queries will continue to be "as specific as possible" since they will generalize out the limited dissimilarities of the original queries. In Section 4.2 we discuss how dissimilarity is estimated, and in Section 4.3 we describe two alternative approaches for merging queries. Given

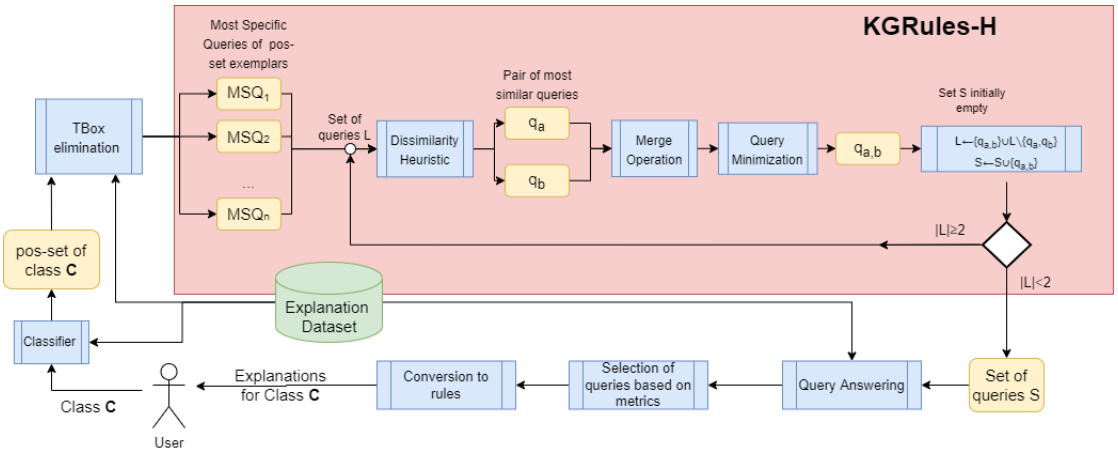


Fig. 2. Visualization of how KGrules-H is integrated into our framework.

Algorithm 1: KGRULES-H

Input: An atomic ABox $\mathcal{A}$ and a set of individual names I.
Output: A list of queries S.

$S \leftarrow [\,]$
$L \leftarrow \{\mathsf{MSQ}(a, \mathcal{A}) \mid a \in I\}$
while $|L| \geqslant 2$ **do**
  $q_A, q_B \leftarrow \arg\min_{q,q' \in L,\, q \neq q'} \mathsf{QueryDissimilarity}(q, q')$
  $q \leftarrow \mathsf{Merge}(q_A, q_B)$
  $L \leftarrow (L \setminus \{q_A, q_B\}) \cup \{q\}$
  append q to S
end
return S

two queries q_A, q_B that have been selected as most similar, the algorithm merges them by constructing a new sav query $q = \mathsf{Merge}(q_A, q_B)$ such that $\mathsf{cert}(q, \mathcal{A}) \supseteq \mathsf{cert}(q_A, \mathcal{A}) \cup \mathsf{cert}(q_B, \mathcal{A})$. The newly created query replaces the ones it was merged from in L, and is also appended to S. Once the queries to be merged have been exhausted, Alg. 1 terminates by returning the list S which will contain $|I| - 1$ sav queries, the results of each merging step in order of creation. Thus, the queries can be considered to be in S in some decreasing order of "specificness", although the actual order depends on the order the elements in L are considered.

If the queries in S have subsets of I as their certain answers, given that we have assumed that I is the pos-set of a classifier for some class C, the queries can be treated as candidate explanation rule queries for C. In this case, the queries can be interpreted as explanation rule queries, converted to the respective explanation rules, and presented as explanations. In general, however, there is no guarantee that the certain answers of a merged query will be a subset of I. In this case, which is the typical case, the computed explanation rules will be rules with exceptions. The queries produced will differ with respect to the metrics introduced in Section 3. Specifically, the queries produced early on, which are the result of few merges are expected to have low recall but high precision, with the inverse holding for queries produced in the later iterations. The importance placed on each metric can be different for each end user, therefore there is no single best query produced by Alg. 1. Instead, metrics for each query should be produced and presented to the end user to let them rank the explanations according to their preferences. This is the methodology we have employed in Sec. 5. For each experiment conducted we present a selection of queries, each performing better for a different metric. In Section 4.3 we prove some optimality results for one of the merging methods, the query least common subsumer (QLCS). In particular, if there is a correct explanation rule (without exceptions), then

we are guaranteed to find the corresponding explanation rule query, using the QLCS. This optimality property does not hold for the Greedy Matching operation defined in section 4.3.2, nor for Alg. 3.

As mentioned above, Alg. 1 works on the assumption that all available knowledge about the relevant individuals I is encoded in a (finite) atomic ABox $\mathcal{A}$. This is essential for the computation of the MSQs. Given that the knowledge base associated with an explanation dataset is in general of the form $\langle \mathcal{T}, \mathcal{A} \rangle$, this assumption means that if the original knowledge involves a non-empty TBox $\mathcal{T}$, it has to be eliminated before applying Alg. 1. In particular, *TBox elimination* is the process of expanding a given ABox by applying the axioms of a given TBox, so that the TBox in eventually not needed, in the sense that all certain answers of the original knowledge can now be obtained only from the expanded ABox. Because TBox elimination is not alwayws possible, this poses certain restrictions on the applicability of Alg. 1, namely that if the original knowledge of the explanation dataset is indeed of the form $\langle \mathcal{T}, \mathcal{A} \rangle$ with $\mathcal{T} \neq \emptyset$, it should be possible, as outlined above, to be transformed through TBox elimination to an equivalent w.r.t. query answering finite assertional-only knowledge base $\mathcal{A}'$, such that $\mathsf{cert}(q, \langle \mathcal{T}, \mathcal{A} \rangle) = \mathsf{cert}(q, \langle \emptyset, \mathcal{A}' \rangle)$ for any q. For knowledge bases that this is possible, the standard approach for TBox elimination is *materialization* and is typically performed by first encoding the axioms in $\mathcal{T}$ as a set of inference rules generating ABox assertions, and then iteratively applying them on the knowledge base until no more assertions can be generated. Materialization is possible, e.g. for Description Logic Programs and the Horn-$\mathcal{SHIQ}$ DL dialect. Finite materialization may not be possible even for low expressivity DL dialects, such as DL-Lite. [63, 64]

4.1. Most Specific Queries

As mentioned above, intuitively, a *most specific query* (MSQ) of an individual a for an atomic ABox $\mathcal{A}$ encodes the maximum possible information about a provided by $\mathcal{A}$. It is defined as a least query in terms of subsumption which has a as a certain answer; in particular, a query q is a MSQ of an individual a for an atomic ABox $\mathcal{A}$ iff $a \in \mathsf{cert}(q, \mathcal{A})$ and for all q' such that $a \in \mathsf{cert}(q', \mathcal{A})$ we have $q \leqslant_S q'$.

Following the subsumption properties, MSQs are unique up to syntactical equivalence. In Alg. 1 it is assumed that $\mathsf{MSQ}(a, \mathcal{A})$ represents one such MSQ of a for $\mathcal{A}$. Since $\mathcal{A}$ is finite, a MSQ is easy to compute. This can be done by viewing $\mathcal{A}$ as a graph, taking the connected component of that graph that contains node a, and converting it into a graph representing a query by replacing all nodes (which in the original graph represent individuals) by arbitrary, distinct variables, taking care to replace node a by variable x.

Theorem 2. *Let $\mathcal{A}$ be a finite atomic ABox over a vocabulary $\langle \mathsf{CN}, \mathsf{RN}, \mathsf{IN} \rangle$, a an individual in IN, and $\mathsf{conn}(a, \mathcal{A})$ the connected component of the ABox graph of $\mathcal{A}$ containing a. Let also q be a sav query with query graph G. If there exists an isomorphism $f : G \to \mathsf{conn}(a, \mathcal{A})$ such that $f(x) = a$, then q is an MSQ of a for $\mathcal{A}$.*

Proof. Let $\mathcal{I}$ be the interpretation such that $\Delta^{\mathcal{I}} = \mathsf{IN}$, $a^{\mathcal{I}} = a$ for all $a \in \mathsf{IN}$, $C^{\mathcal{I}} = \{a \mid C(a) \in \mathcal{A}\}$, for all $C \in \mathsf{CN}$, and $r^{\mathcal{I}} = \{(a, b) \mid r(a, b) \in \mathcal{A}\}$ for all $r \in \mathsf{RN}$. Clearly, $\mathcal{I}$ is a model of $\mathcal{A}$, and it is easy to see that $\mathsf{cert}(q, \mathcal{A}) = \mathsf{ans}_{\mathcal{I}}(q, \mathcal{A})$ for any query q, where $\mathsf{ans}_{\mathcal{I}}(q, \mathcal{A})$ are the answers to q under the interpretation $\mathcal{I}$. Because of this identity, we can, without loss of generality, restrict ourselves to this $\mathcal{I}$ w.r.t. $\mathsf{cert}(q, \mathcal{A})$.

Since f exists, we can construct for q the match $\pi : \mathsf{var}(q) \to \mathsf{IN}$, such that $\pi(u) = f(u)^{\mathcal{I}}$. It follows that $\pi(x) = a$, and so $a \in \mathsf{cert}(q, \mathcal{A})$. Let q' be a query with query graph G' such that $a \in \mathsf{cert}(q', \mathcal{A})$. Since a is a certain answer of q', G' must be homomorphic to $\mathcal{A}$ by a homomorphism g such that $g(x) = a$. Since homomorphisms preserve edges and q' is connected, the image of G' under g is a subgraph of $\mathsf{conn}(a, \mathcal{A})$. Then $h = g \circ f^{-1}$ is a homomorphism from G' to G with $h(x) = x$. Thus, $q \leqslant_S q'$.

□

Since $q \leqslant_S q'$ implies that $\mathsf{cert}(q, \mathcal{A}) \subseteq \mathsf{cert}(q', \mathcal{A})$, the MSQ of a has as few as possible certain answers other than a. This is desired since it implies that the initial queries in Alg. 1 do not have any exceptions (certain answers not in the pos-set), unless this cannot be avoided. Therefore, the iterative query merging process of constructing explanations starts from an optimal standpoint.

Example 4. *Continuing the previous examples, Fig. 3 shows the graphs of the MSQs of* p3 *and* p4*, constructed by materializing the TBox, and then extracting the connected components of* p3 *and* p4 *from the new ABox.*

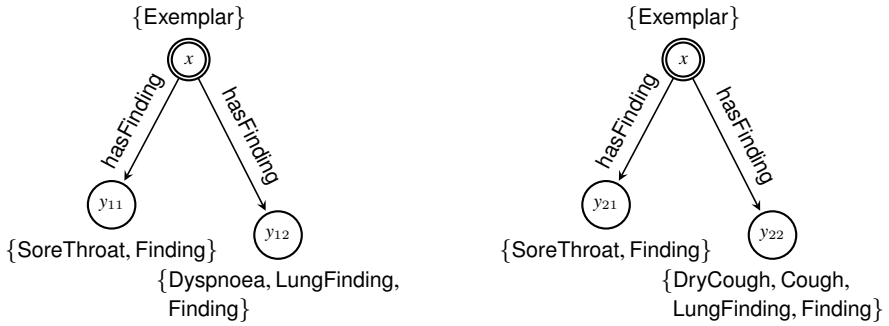


Fig. 3. The MSQs of p3 and p4 represented as graphs

For use with Alg. 1, we denote a call to a concrete function implementing the above described approach for obtaining $\mathsf{MSQ}(a, \mathcal{A})$ by $\textsc{GraphMSQ}(a, \mathcal{A})$. We should note that the result of $\textsc{GraphMSQ}(a, \mathcal{A})$ will be a MSQ, but in general it will not be condensed, and thus may contain redundant atoms and variables.

4.2. Query Dissimilarity

At each iteration, Alg. 1 selects two queries to merge in order to produce a more general description covering both queries. To make the selection, we use a heuristic which is meant to express how dissimilar two queries are, so that at each iteration the two most similar queries are selected and merged, with the purpose of generating an as least generic as possible more general description. Given two queries q_1, q_2 with respective graph representations $G_1 = (V_1, E_1, \ell_{V_1}, \ell_{E_1})$ and $G_2 = (V_2, E_2, \ell_{V_2}, \ell_{E_2})$, we define the *query dissimilarity* heuristic between q_1 and q_2 as follows:

$$\mathsf{QueryDissimilarity}(q_1, q_2) = \sum_{v_1 \in V_1} \min_{v_2 \in V_2} \mathsf{diss}_{q_1 q_2}(v_1, v_2) + \sum_{v_2 \in V_2} \min_{v_1 \in V_1} \mathsf{diss}_{q_2 q_1}(v_2, v_1)$$

where

$$\mathsf{diss}_{q_1 q_2}(v_1, v_2) = |L_1(v_1) \setminus L_2(v_2)| + \sum_{r \in R} \{\max(\mathsf{indegree}^r_{G_1}(v_1) - \mathsf{indegree}^r_{G_2}(v_2), 0) + \max(\mathsf{outdegree}^r_{G_1}(v_1) - \mathsf{outdegree}^r_{G_2}(v_2), 0)\},$$

R is the set of all role names appearing in the edge labels of the two graphs, and $\mathsf{indegree}^r_G(v)$ ($\mathsf{outdegree}^r_G(v)$) is the number of incoming (outcoming) edges e in node v of graph G with $r \in \ell(e)$.

The intuition behind the above dissimilarity measure is that the graphs of queries which are dissimilar consist of nodes with dissimilar labels connected in dissimilar ways. Intuitively, we expect such queries to have dissimilar sets of certain answers, although there is no guarantee that this will always be the case. In order to have a heuristic that can be computed efficiently, we do not examine complex ways in which the nodes may be interconnected, but only examine the local structure of the nodes by comparing their indegrees and outdegrees. The way in which we compare the nodes of two query graphs is optimistic; we compare each node with its best possible counterpart —the node of the other graph which it is the most similar to. Note that $\mathsf{diss}_{q_1 q_2}(v_1, v_2)$ does not equal $\mathsf{diss}_{q_2 q_1}(v_2, v_1)$. The first quantity expresses how many conjuncts of q_1 containing v_1 do not appear in q_2 containing v_2. This is best illustrated using a short example.

Example 5. *Let* $q_1 = \{C(x), D(x), r(x, y_{11}), r(y_{12}, x), s(x, y_{11})\}$, $q_2 = \{C(x), E(x), r(x, y_{21}), r(x, y_{23})\}$ *and assume that we are comparing variable x of* q_1 *with variable x of* q_2*. As far as concepts are concerned,* $C(x)$ *appears in both* q_1 *and* q_2*, while* $D(x)$ *only appears in* q_1*, so one concept conjunct is missing from* q_2*. Moreover, x appears*

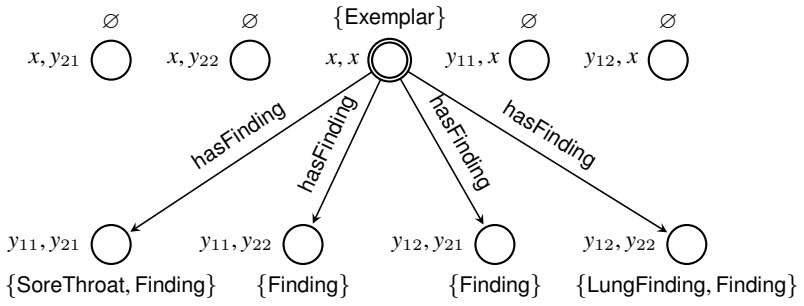


Fig. 4. The Kronecker product of the MSQs of p3 and p4

in one "outgoing" r conjunct in q_1 and in two outgoing r conjuncts in q_2, so no outgoing conjuncts are missing from q_2. Also, x appears in one "ingoing" r conjunct in q_1 but in no "ingoing" r in q_2 conjuncts, so one more conjunct is missing. Finally, the s conjunct of x is missing from q_2, giving us a total of three missing conjuncts from q_2, therefore $\mathsf{diss}q_1q_2(x_1, x_2) = 3$.

Using an efficient representation of the queries, it is easy to see that $\mathsf{QueryDissimilarity}(q_1, q_2)$ can be computed in time $O(|\mathsf{var}(q_1)| \cdot |\mathsf{var}(q_2)|)$.

4.3. Query Merging

The next step in Alg. 1 is the merging of the two most similar queries that have been selected. In particular, given a query q_A for some individuals I_A, i.e. a query such that $I_A \subseteq \mathsf{cert}(q_A, \mathcal{A})$, and a query q_B for some individuals I_B, i.e. a query such that $I_B \subseteq \mathsf{cert}(q_B, \mathcal{A})$, the purpose is to merge the two queries to produce a more general query q for $I_A \cup I_B$. The necessary condition that such query must satisfy is $I_A \cup I_B \subseteq \mathsf{cert}(q, \mathcal{A})$. In the context of Alg. 1 the queries q_A, q_B may be MSQs of some individuals, or results of previous query merges.

4.3.1. Query Least Common Subsumer

Our first approach for merging queries is using the query least common subsumer (QLCS). In the next few paragraphs we will present how to construct such a query from two sav queries which will make apparent that the QLCS of two sav queries always exists. As mentioned in Section 2, a QLCS is a most specific generalization of two queries. By choosing to use $\mathsf{QLCS}(q_A, q_B)$ (since QLCS is unique up to syntactical equivalence, we assume that $\mathsf{QLCS}(q_A, q_B)$ represents one such QLCS for q_A and q_B) in the place of $\mathsf{Merge}(q_A, q_B)$ in Alg. 1 , and assuming that the input set of individuals is $I = \{a_1 \ldots, a_n\}$, it should be obvious that the last query computed by Alg. 1 will be $q = \mathsf{QLCS}(\mathsf{MSQ}(a_1, \mathcal{A}), \mathsf{MSQ}(a_2, \mathcal{A}), \ldots, \mathsf{MSQ}(a_n, \mathcal{A}))$. For any query q' such that $\mathsf{cert}(q', \mathcal{A}) \supseteq I$, it will by definition hold that $\mathsf{MSQ}(a_i, \mathcal{A}) \leqslant_S q', i = 1, \ldots, n$. Therefore, $q \leqslant_S q'$, which implies that $\mathsf{cert}(q, \mathcal{A}) \subseteq \mathsf{cert}(q', \mathcal{A})$. This means that if there is an exact explanation rule query of I, that will be a QLCS of its MSQs, while if there is not, the computed QLCS explanation rule query will have the fewest possible exceptions. Thus, the main advantage of using the QLCS is that it guarantees "optimality" of the computed explanations.

To compute a QLCS we use an extension of the Kronecker product of graphs to labeled graphs. Given two labeled graphs $G_1 = (V_1, E_1, \ell_{V_1}, \ell_{E_1})$ and $G_2 = (V_2, E_2, \ell_{V_2}, \ell_{E_2})$, which in our case will represent queries, their Kronecker product $G = G_1 \times G_2$ is the graph $G = (V, E, \ell_V, \ell_E)$, where $V = V_1 \times V_2$, $E = \{\langle\langle u_1, v_1\rangle, \langle u_2, v_2\rangle\rangle \mid \langle u_1, u_2\rangle \in E_1, \langle v_1, v_2\rangle \in E_2,$ and $\ell_{E_1}(\langle u_1, u_2\rangle) \cap \ell_{E_2}(\langle v_1, v_2\rangle) \neq \varnothing\} \subseteq V \times V$, $\ell_V : V \to 2^{\mathsf{CN}}$ with $\ell_V(\langle v_1, v_2\rangle) = \ell_{V_1}(v_1) \cap \ell_{V_2}(v_2)$, and $\ell_E : E \to 2^{\mathsf{RN}}$ with $\ell_E(\langle\langle u_1, v_1\rangle, \langle u_2, v_2\rangle\rangle) = \ell_{E_1}(\langle u_1, u_2\rangle) \cap \ell_{E_2}(\langle v_1, v_2\rangle)$.

Example 6. *Fig. 4 shows the Kronecker product of the query graphs of Example 4.*

As with the Kronecker product of unlabeled graphs, for any graph H, we have that $H \to G_1, G_2$ if and only if $H \to G_1 \times G_2$ [65]. Since we are interested in graphs representing sav queries, we can assume that H, G_1, G_2 are connected. Assuming that G_1, G_2 represent two sav queries q_1, q_2 (with answer variable x), it will specifically hold that $H \to G_1, G_2$ if and only if $H \to \mathsf{conn}(\langle x, x\rangle, G_1 \times G_2)$, where $\mathsf{conn}(\langle x, x\rangle, G_1 \times G_2)$ is the connected component of $G_1 \times G_2$ containing the pairs of answer variables node $\langle x, x\rangle$. Let q be the query whose query graph

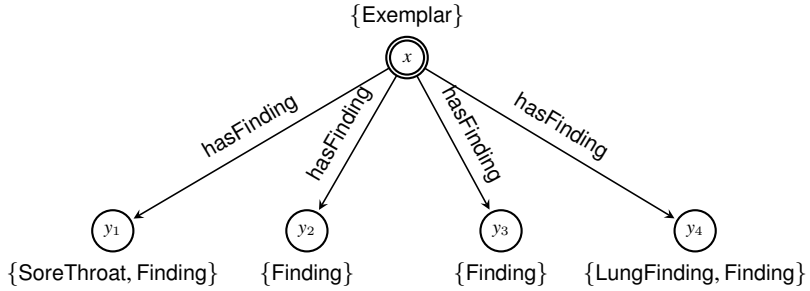


Fig. 5. The QLCS of the MSQs of p3 and p4 produced by KRONECKERQLCS represented as a graph

is the connected component of $G_1 \times G_2$ containing node $\langle x, x \rangle$, with node $\langle x, x \rangle$ replaced by x (the answer variable of the new query) and all other nodes of that connected component replaced by distinct arbitrary variable names other than x. Since all sav queries are assumed to contain the top atom the node $\langle x, x \rangle$ will always be labeled with the top atom and therefore q will always be non-empty. Because homomorphisms between query graphs correspond to subsumption relations between the respective queries, and vice versa, it will hold that $q_1, q_2 \leqslant_S q$ and for any other query q' s.t. $q_1, q_2 \leqslant_S q'$ it will hold that $G_{q'} \to G_1, G_2$, therefore $G_{q'} \to G_1 \times G_2$ and therefore $q \leqslant_S q'$. Computing $G_1 \times G_2$ is therefore a concrete way to construct the QLCS.

For use with Alg. 1, we denote a call to the concrete function implementing that approach for computing $\mathsf{QLCS}(q_1, q_2)$ by $\textsc{KroneckerQLCS}(q_1, q_2)$. The time complexity of $\textsc{KroneckerQLCS}(q_1, q_2)$ is $O(|\mathsf{var}(q_1)|^2 \cdot |\mathsf{var}(q_2)|^2)$, and the Kronecker product of the graphs G_1, G_2, contains $|V_1| \cdot |V_2|$ nodes. Therefore, the final query constructed by Alg. 1 using $\textsc{KroneckerQLCS}(q_1, q_2)$ to implement $\mathsf{Merge}(q_1, q_2)$ will have $O(m^n)$ variables, where $m = \max_{i=1,\ldots,n} |\mathsf{var}(q_i)|$ is the maximum variable count of the MSQs, and $n = |I|$ is the number of individuals in I. Constructing this query is by far the most expensive operation of the algorithm and dominates the running time with a complexity of $O(m^{2n})$. This makes the use of the above procedure for computing QLCS's prohibitive for merging queries. However, the query obtained using the Kronecker product is in general not condensed, and in general will contain many redundant atoms and variables.

Example 7. *Fig. 5 shows the QLCS of the MSQs of Example 4, extracted from the Kronecker product of Example 6. It is obvious that the nodes with label* $\{\mathsf{Finding}\}$ *are redundant due to the presence of the nodes with label* $\{\mathsf{SoreThroat}, \mathsf{Finding}\}$ *and* $\{\mathsf{LungFinding}, \mathsf{Finding}\}$.

Removing the redundant parts of a query is essential not only for reducing the running time of the algorithm. Since these queries are intended to be shown to humans as explanations, ensuring that they are compact is imperative to improve comprehensibility. As mentioned in Section 2, the operation that compacts a query by creating a syntactically equivalent query by removing all its redundant parts is condensation. However, condensing a query is coNP-complete [53]. For this reason, we utilize Alg. 2, an approximation algorithm which removes redundant conjuncts and variables without a guarantee of producing a fully condensed query.

Alg. 2 iterates through the variables of the input query, and checks if deleting one of them is equivalent to unifying it with another one. In particular, at each iteration of the main loop, the algorithm attempts to find a variable suitable for deletion. If none is found, the loop terminates. The inner loop iterates through all pairs of variables. At each iteration, variable v' is deleted if unifying it with v, by replacing all instances of v' in the query with v, produces no new conjuncts. By unifying variable v' with v, all conjuncts of the form $C(v')$ become $C(v)$ and all conjuncts of the forms $r(v', v'')$, $r(v'', v')$, $r(v', v')$ become $r(v, v'')$, $r(v'', v)$, $r(v, v)$, respectively. If these conjuncts are already present, then removing them is equivalent to unifying v' with v. Alg. 2 is correct because removing a variable from query q produces a query that subsumes q, while unifying two variables of q produces a query that is subsumed by q. Therefore, Alg. 2 produces syntactically equivalent queries.

Example 8. *We can showcase an example of Alg. 2 detecting an extraneous variable in the QLCS of* q_1 *and* q_2 *from Example 7, running the main loop for* $v = y_1$ *and* $v' = y_2$. *The condition at line 3 first checks if the label of node* v', $\{\mathsf{Finding}\}$, *is a subset of the label of node* v, $\{\mathsf{SoreThroat}, \mathsf{Finding}\}$, *which is true. Then for every incoming edge to*

Algorithm 2: APPROXQUERYMINIMIZE

Input: A query represented as a graph $q = \langle V, E, \ell_V, \ell_E \rangle$.
Output: An approximately minimized query q', also represented as a graph.

do
  $q' \leftarrow q$
  foreach pair (v, v'), $v, v' \in V$, $v \neq v'$ **do**
    if $\ell_V(v') \subseteq \ell_V(v)$ **and**
      $\langle v', v'' \rangle \in E \Rightarrow (\langle v, v'' \rangle \in E,\ \ell_E(v', v'') \subseteq \ell_E(v, v'')),\ v'' \neq v'$ **and**
      $\langle v'', v' \rangle \in E \Rightarrow (\langle v'', v \rangle \in E,\ \ell_E(v'', v') \subseteq \ell_E(v'', v)),\ v'' \neq v'$ **and**
      $\langle v', v' \rangle \in E \Rightarrow (\langle v, v \rangle \in E,\ \ell_E(v', v') \subseteq \ell_E(v, v))$ **then**
      Delete variable v' from q.
    end
  end
while $q' \neq q$
return q'

v' it checks if there is an edge incoming to v with the same label and origin. They both have one incoming edge from node x labeled with hasFinding *so the second condition will also evaluate to true. Node v' has no incoming edges, so the third condition will also evaluate to true. Therefore, Alg. 2 will detect node v' as extraneous and delete it. Running the entire algorithm for this query will result in the query represented in Fig. 6 as a graph.*

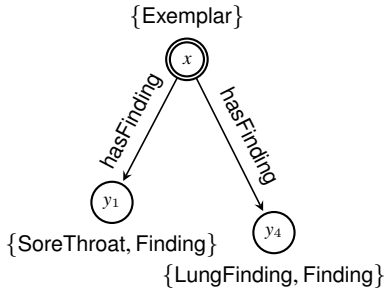


Fig. 6. The minimized graph of the QLCS of Fig. 5

The main loop of Alg. 2 is executed at most $\mathsf{var}(q)$ times, since at each loop either a variable is deleted or the loop terminates. The inner loop checks all pairs of variables ($O(|\mathsf{var}(q)|^2)$), and the if condition requires $O(\mathsf{var}(q))$ set comparisons of size at most $|\mathsf{CN}|$, and $2|\mathsf{RN}|$ comparisons of rows and columns of adjacency matrices. Treating $|\mathsf{CN}|$ and $|\mathsf{RN}|$ as constants, the complexity of Algorithm 2 is $O(|\mathsf{var}(q)|^4)$.

Given the above, our first practical implementation of Alg. 1, uses $\mathrm{GRAPHMSQ}(a, \mathcal{A})$ to implement $\mathsf{MSQ}(a, \mathcal{A})$, and $\mathrm{APPROXQUERYMINIMIZE}(\mathrm{KRONECKERQLCS}(q_A, q_B))$ to implement $\mathsf{Merge}(q_A, q_B)$.

Regardless of the approximate query minimization described above, even if full query condensation could be performed, there is no guarantee that condensing the queries computed by Alg. 1 using QLCS as the merge operation, have meaningfully smaller condensations. This led us to two different approaches for further dealing with the rapidly growing queries QLCS produces.

The simplest approach to address this problem is to reject any queries produced by $\mathrm{KRONECKERQLCS}(q_1, q_2)$ that, even after minimization, have variable counts higher than a pre-selected threshold. This strategy essentially introduces a weaker version of Alg.1, without the guarantees of optimality, but with polynomial running time, which is outlined in Alg. 3, which we call KGrules-HT.

Alg. 3 is the same as Alg. 1 except that q is not added to L and S unless its variable count is less than or equal to an input threshold, t. This simple change reduces the complexity of the algorithm to polynomial in terms of $|I|$ and

Algorithm 3: KGrules-HT

Input: An atomic ABox $\mathcal{A}$, a set of individual names I and a threshold t.
Output: A set of queries S.

$S \leftarrow \varnothing$
$L \leftarrow \{\text{GRAPHMSQ}(a, \mathcal{A}) \mid a \in I\}$
while $|L| \geqslant 2$ **do**
  $q_A, q_B \leftarrow \arg\min_{q,q' \in L, q \neq q'} \mathsf{QueryDissimilarity}(q, q')$
  $q \leftarrow \text{APPROXQUERYMINIMIZE}(\text{KRONECKERQLCS}(q_A, q_B))$
  $L \leftarrow (L \setminus \{q_A, q_B\})$
  if $|\mathsf{var}(q)| \leqslant t$ **then**
    $L \leftarrow L \cup \{q\}$
    append q to S
  end
end
return S

t. It should be noted that setting a very low threshold for t could potentially lead to all queries being rejected and the algorithm returning an empty set. The effects of t on the queries produced can vary significantly for different inputs and therefore different explanation datasets, thus t should be adjusted experimentally. It is also assumed that all MSQ's created when initializing L have less than t variables.

If we let $n = |I|$, calculating the query dissimilarity for all pairs of queries, costs $O(n^2t^2)$ operations. Using memoization limits the cost of calculating the dissimilarity through all iterations to $O(n^2t^2)$. Each iteration of the main loop involves $O(n^2)$ operations for selecting q_A, q_B and $O(t^4)$ operations for calculating their QLCS. Thus, in total, Alg. 3 has a running time of $O(n^3 + n^2t^2 + nt^4)$.

Our second approach to overcome the problem posed by the rapidly growing size of the queries produced by QLCS was to consider a different merge operation, which is described in the following section.

4.3.2. Greedy Matching

As already mentioned, the above described procedure for merging queries by computing a QLCS, often introduces too many variables that Alg. 2 can't minimize effectively. A QLCS of q_1, q_2, if condensed, is optimal at producing a query that combines their certain answers while introducing few as possible new variables. However, the new variables that are introduced may still be too many. In practice, we may be able to afford some leniency in terms of certain answers and the loss of optimality that the QLCS provides, so we can explore alternative methods of unifying queries without the cost of increased variables.

One such method, is finding a *common subquery*, i.e. finding the common conjuncts of two queries q_1, q_2. Assuming that $|\mathsf{var}(q_1)| \geqslant |\mathsf{var}(q_2)|$ and that the variables of q_1, q_2 are renamed apart except for the answer variable x, a *matching* θ is a 1-1 mapping from the variables of q_2 to the variables of q_1. The *common subquery* q is formed by renaming the variables of q_2 according to the matching, and keeping the conjuncts also found in q_1, i.e. $q = q_1 \cap q_2\theta$.

Example 9. *Continuing the previous examples, we will find common subqueries of the MSQs of* p3 *and* p4*. We can match the lung finding of* p4 *with the lung finding of* p3 *as well as their sore throat findings. This would be done by renaming* y_{21} *to* y_{11} *and* y_{22} *to* y_{12} *and then keeping the common conjuncts resulting in the query* $q(x) = \{\mathsf{Exemplar}(x), \mathsf{hasFinding}(x, y_{11}), \mathsf{SoreThroat}(y_{11}), \mathsf{Finding}(y_{11}), \mathsf{hasFinding}(x, y_{12}), \mathsf{LungFinding}(y_{12}), \mathsf{Finding}(y_{12})\}$*. In our approach we evaluate the quality of matchings based on the number of conjuncts in the query they induce, which in this case is 7.*

Finding the maximum common subquery belongs to a larger family of problems of finding maximal common substructures in graphs, such as the mapping problem for processors [66], structural resemblance in proteins [67] and the maximal common substructure match (MCSSM) [68]. Our problem can be expressed as a weighted version of most of these problems, since they only seek to maximize the number of nodes in the common substructure, which, in our case, corresponds to the number of variables in the resulting subquery. Since we want to maximize the

number of common conjuncts, we could assign weights to variable matchings (y corresponds to z) due to concept conjuncts, and to pairs of variable matchings (y corresponds to z *and* y' corresponds to z') due to role conjuncts. This is an instance of the general graph matching problem. These problems are NP-hard and are therefore usually solved with approximation algorithms [69, 70]. With Alg. 4, we introduce our own method for approximately solving this problem since we need to impose the additional restriction that the resulting query must be connected. Any conjuncts in the resulting query that aren't connected to the answer variable do not influence the query's certain answers, so we consider them extraneous.

Algorithm 4: GREEDYCOMMONCONJUNCTS

Input: Two queries q_1, q_2, with query graphs $G_{q_1} = \langle V_1, E_1, \ell_{V_1}, \ell_{E_1} \rangle, G_{q_2} = \langle V_2, E_2, \ell_{V_2}, \ell_{E_2} \rangle$.
Output: A query consisting of common conjuncts of q_1 and q_2.

if $|\mathsf{var}(q_1)| < |\mathsf{var}(q_2)|$ **then**
  Swap q_1, q_2
end
$\theta \leftarrow \{\,\}$
$U \leftarrow \mathsf{var}(q_1) \setminus \{x\}$
$V \leftarrow \mathsf{var}(q_2) \setminus \{x\}$
while true do
  $S \leftarrow \{z \mapsto y \mid z \in V,\ y \in U,\ \langle z, z' \rangle \in E_1,\ \langle y, y' \rangle \in E_2,\ \ell_{E_1}(\langle z, z' \rangle) \cap \ell_{E_2}(\langle y, y' \rangle) \neq \varnothing,\ z' \mapsto y' \in \theta\}$
    $\cup \{z \mapsto y \mid z \in V,\ y \in U,\ \langle z', z \rangle \in E_1,\ \langle y', y \rangle \in E_2,\ \ell_{E_1}(\langle z', z \rangle) \cap \ell_{E_2}(\langle y', y \rangle) \neq \varnothing,\ z' \mapsto y' \in \theta\}$
  if $S \neq \varnothing$ **then**
    $\hat{z} \mapsto \hat{y} \leftarrow \arg\max_{z \mapsto y \in S} \{|q_1 \cap q_2(\theta \cup \{z \mapsto y\})| - |q_1 \cap q_2\theta|\}$
    $\theta \leftarrow \theta \cup \{\hat{z} \mapsto \hat{y}\}$
    $V \leftarrow V \setminus \{\hat{z}\}$
    $U \leftarrow U \setminus \{\hat{y}\}$
  else
    break
  end
end
$q \leftarrow q_1 \cap q_2\theta$
return q

Alg. 4 initializes an empty variable renaming θ. V and U contain the variables that have not been matched yet. At each iteration of the main loop, the algorithm attempts to match one of the variables of q_2 with one of the variables of q_1. Only matches that conserve the connectedness of the induced query are considered (set S). If there are no such matches the main loop terminates. Otherwise, the match that adds the largest number of conjuncts to the induced query is selected. This match is added to θ and the corresponding variables are removed from V and U. Finally, the query q which consists of the common conjuncts of q_1 and q_2 is constructed, by renaming the variables of q_2 according to θ and keeping the conjuncts that also appear in q_1.

An efficient implementation of Alg. 4 will use a max-priority queue to select at each iteration the match that adds the largest number of conjuncts to the induced query. The priority queue should contain an element for each pair of variables that can be matched, with the priority of the element being either 0 if the pair is not in S, and otherwise equal to the number of conjuncts it would add to the query. At each iteration some priorities may need to be updated. The time complexity of priority queues varies depending on their implementation. Implementations such as Strict Fibonacci Heaps have lower time complexity but perform worse in practice than simpler implementations with higher time complexities such as Binary Heaps. A Strict Fibonacci Heap implementation would have a time complexity of $O(n^2m^2)$, where $n = |\mathsf{var}(q_1)|$, $m = |\mathsf{var}(q_2)|$, while a Binary Heap implementation $O(n^2m^2 \log(nm))$.

5. Experiments

We evaluated the proposed algorithms and framework by generating explanations for various classifiers, computing the metrics presented in Section 3, comparing with other rule-based approaches where possible, and discussing the quality and usefulness of the results. Specifically, we explored four use-cases: a) we are given a tabular dataset which serves as both the features of the classifier and the explanation dataset, b) we are given raw data along with curated semantic descriptions (ABox) but without a TBox, c) we are given curated semantic descriptions, along with a TBox, and d) we are only given raw data, so semantic descriptions need to be constructed automatically. The first use-case facilitated comparison of the proposed KGRules-H algorithm with other rule-based XAI approaches from literature, the second and third use-cases were more suitable for validating the usefulness of the proposed framework in its intended setting, while the fourth explored how KGRules-H could be utilized in a scenario in which semantic data descriptions are not available. For the first use-case we experimented on the Mushroom[1] dataset which involves only categorical features. For the second use-case we used the CLEVR-Hans3 dataset [71] which consists of images of 3D geometric shapes in addition to rich metadata which we use as semantic data descriptions. For the third we constructed an explanation dataset from a subset of the Visual Genome [8] to explain a classifier trained on Places365 [72] for scene classification, while for the fourth we used MNIST [73] which contains images of handwritten digits, and no metadata .

The components needed for each experiment, were: a) a black-box classifier for which we provide rule-based explanations, b) an explanation dataset with semantic descriptions of exemplar data using the appropriate terminology and c) reasoning capabilities for semantic query answering, in order to evaluate the generated rules. As classifiers we chose widely used neural networks, which are provided as default models in most deep learning frameworks.

For constructing explanation datasets in practice, we identified two general approaches: a) the curated approach, and b) the automated information extraction approach. For the manual curated approach, the semantic descriptions of exemplar data were provided. In an ideal scenario, curated explanation datasets are created by domain experts, providing semantic descriptions which are meaningful for the task, and by using the appropriate terminology, lead to meaningful rules as explanations. In our experiments, we simulated such a manually curated dataset by using Visual Genome, which provides semantic descriptions for each image present in the dataset, using terminology linked to WordNet [10]. Of course, using human labor for the creation of explanation datasets in real-world applications would be expensive, thus we also experimented with automatically generating semantic descriptions for exemplar data. Specifically, for the automated information extraction approach we used domain specific, robust, feature extraction techniques and then provided semantic descriptions in terms of the extracted features. In these experiments, we automatically generated semantic descriptions for images in MNIST, by using ridge detection, and then describing each image as a set of intersecting lines.

For acquiring certain answers of semantic queries, which requires reasoning, we set up repositories on GraphDB.[2] For the case of the Mushrooms dataset we used the certain answers to measure fidelity, number of rules and average rule length and compared our approach with RuleMatrix [26] which implements scalable bayesian rule lists [25], Skope-Rules[3] and the closely related KGrules[46]. For CLEVR-Hans3, we mainly used the certain answers to explore whether our explainability framework can detect the foreknown bias of the classifier, while for Visual Genome we attempt to discover biases, by observing the best generated rule-queries with respect to precision, recall, and degree as defined in Section 3. Finally, for MNIST we observed quality and usability related properties of generated rule queries and their exceptions.

5.1. Mushrooms

The purpose of the Mushroom experiment was to compare our results with other rule-based approaches from the literature. Since other approaches mostly provide explanation rules in terms of the feature space, the explanation dataset was created containing only this information. We should note that this was only done for comparison's sake,

[1] https://archive.ics.uci.edu/ml/datasets/mushroom
[2] https://graphdb.ontotext.com/
[3] https://github.com/scikit-learn-contrib/Skope-Rules

and is not the intended use-case for the proposed framework, in which there would exist semantic descriptions which cannot be necessarily represented in tabular form, and possibly a TBox.

5.1.1. Explanation Dataset

The Mushroom dataset contains 8124 rows, each with 22 categorical features of 2 to 9 possible values. We randomly chose subsets of up to 4000 rows to serve as the exemplars of explanation datasets. The vocabulary $\langle \mathsf{CN}, \mathsf{RN}, \mathsf{IN} \rangle$ used by the explanation datasets consisted of: an individual name for each index of a row included in the explanation dataset (in IN), and a concept name for each combination of categorical feature and value (in CN), giving us a total of $|\mathsf{CN}| = 123$. In this case, the set of exemplars (EN) coincided with the set of individual names. Then, for each feature and for each row, we added an assertion to the ABox, representing the value of the feature for the corresponding row. Thus, the knowledge base of an explanation dataset and its vocabulary had the form:

$$\mathsf{IN} = \{\mathsf{row}_1, \mathsf{row}_2 \dots \mathsf{row}_n\}$$

$$\mathsf{CN} = \{\mathsf{CapShapeBell}, \mathsf{CapShapeConical}, \mathsf{CapShapeConvex}, \dots, \mathsf{HabitatGrass}, \mathsf{HabitatLeaves}, \dots\}$$

$$\mathsf{RN} = \varnothing$$

$$\mathcal{A} = \{\mathsf{CapShapeFlat}(\mathsf{row}_1), \mathsf{CapSurfaceFibrous}(\mathsf{row}_1), \dots, \mathsf{HabitatPaths}(\mathsf{row}_n)\}$$

5.1.2. Setting

For this set of experiments, we used a simple multi-layer perceptron with one hidden layer as the black-box classifier. The classifier achieved 100% accuracy on train and test set in all experiments. After training, we generated explanations using Alg. 1 (in the case of tabular data, both merge operations outlined in section 4.3 are reduced to a simple intersection of the concept assertions of the merged individuals), along with three other rule-based approaches from related literature, and compared them.

We split the dataset into four parts: 1) a training set, used to train the classifier, 2) a test set, used to evaluate the classifier, 3) an explanation-train set, used to generate explanation rules, and 4) an explanation-test set, used to evaluate the generated explanation rules. When running KGrules-H and KGrules, the explanation dataset was constructed from the explanation-train set. We experimented by changing the size of this dataset, from 100 to 4000 rows, and observed the effect it had on the explanation rules. On the explanation-test set, we measured the fidelity of the generated rules which is defined as the proportion of items for which classifier and explainer agree. We also measured the number of generated rules and average rule length. We used the proposed KGrules-H algorithm to generate explanations, and we also generated rules (on the same data and classifier) with RuleMatrix, Skope-rules and the closely related KGrules.

To compare with the other methods, which return a set of rules at their output, in this experiment we only considered correct rules we generated. To choose which rules to consider (what would be shown to a user), from the set of all correct rule-queries generated, we greedily chose queries starting with the one that had the highest count of certain answers on the explanation-train set, and then iteratively adding queries that provided the highest count of certain answers, not provided by any of the previously chosen queries. This is not necessarily the optimal strategy of rule selection for showing to a user (it never considers rules with exceptions), and we plan to explore alternatives in future work.

Finally, for all methods except for the related KGrules we measured running-time using same runtime on Google Colab [4], by using the "`%%timeit`" [5] magic command with default parameters. KGrules was not compatible with this benchmarking test, since it is implemented in Java as opposed to Python which is the implementation language provided for the other methods. In addition, KGrules implements an exhaustive exponential algorithm, so it is expected to be much slower than all other methods.

[4] https://colab.research.google.com/

[5] https://ipython.readthedocs.io/en/stable/interactive/magics.html#magic-timeit

5.1.3. Results

The results of the comparative study are shown in Table 4. A first observation is that for small explanation datasets, the proposed approach did not perform as well as the other methods regarding fidelity, while for large ones it even outperformed them. This could be because for small explanation datasets, when the exemplars are chosen randomly, there are not enough individuals, and variety in the MSQs of these individuals, for the algorithm to generalize by merging their MSQs. This is hinted also from the average rule length, which is longer for both KGrules-H and KGrules for explanation dataset sizes 100 and 200, which indicates less general queries. Conversely, for explanation datasets of 600 exemplars and larger, the proposed approach performs similarly, in terms of fidelity, with related methods. Regarding running time, KGrules-H is the fastest except for the case of the largest explanation dataset, for which Skope-rules is faster. However, in this case Skope-rules's performance suffers with respect to fidelity, whereas RuleMatrix and KGrules-H achieve perfect results. Furthermore, our proposed approach seems to generate longer rules than all other methods which on the one hand means that they are more informative, though on the other hand they are potentially less understandable by a user. This highlights the disadvantages constructing queries using the MSQs as a starting point. This was validated upon closer investigation, where we saw that the rules generated from the small explanation datasets were more specific than needed, as the low value of fidelity was due only to false negatives, and there were no false positives. Detecting redundant conjuncts might be done more efficiently for rules concerning tabular data, but for general rules, on which our approach is based, this task is very computationally demanding.

Size	Method	Fidelity	Nr. of Rules	Avg. Length	Time (seconds)
100	KGrules-H	92.70%	4	14	0.15
	KGrules	**97.56%**	11	5	-
	RuleMatrix	94.53%	3	2	3.65
	Skope-rules	97.01%	3	2	1.29
200	KGrules-H	93.40%	8	14	0.18
	KGrules	98.37%	11	5	-
	RuleMatrix	97.78%	4	2	3.65
	Skope-rules	**98.49%**	4	2	1.47
600	KGrules-H	**99.60%**	10	13.3	0.61
	KGrules	99.41%	13	4	-
	RuleMatrix	99.43%	6	1	7.69
	Skope-rules	98.52%	4	2	1.58
1000	KGrules-H	**100%**	9	14.2	1.67
	KGrules	99.58%	14	6.57	-
	RuleMatrix	99.90%	6	1.33	11.1
	Skope-rules	98.50%	4	2.25	2.95
4000	KGrules-H	**100%**	10	14.2	34.30
	KGrules	99.72%	16	5.81	-
	RuleMatrix	**100%**	7	1.43	43.00
	Skope-rules	96.85%	2	2	3.01

Table 4

Performance on the Mushroom dataset.

5.2. CLEVR-Hans3

The second set of experiments involved CLEVR-Hans3 [71], which is an image classification dataset designed to evaluate algorithms that detect and fix biases of classifiers. This dataset provides sufficient and reliable information

to create an explanation dataset, while the given train-test split contains intentional biases, which was ideal as a grounds for experimentation, as we could observe the extent to which the proposed explanation rules can detect them.

5.2.1. Explanation Dataset

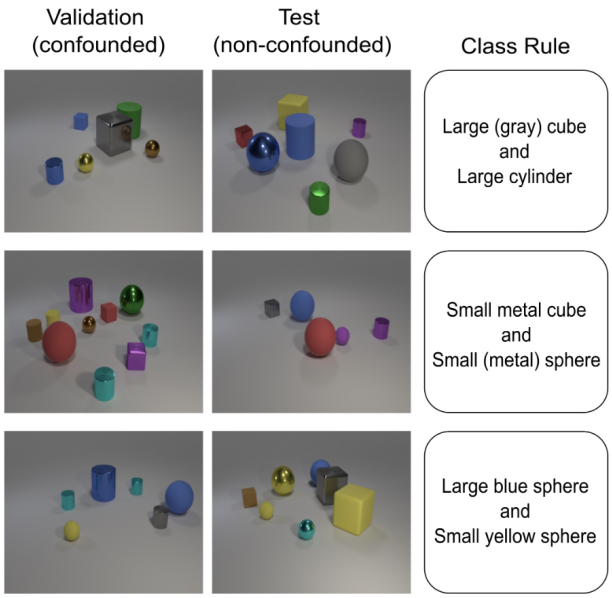


Fig. 7. Image samples of the three classes of CLEVR-Hans3 along with the class rules and the confounding factors in parentheses.

We created an explanation dataset $\mathcal{E}$ using the test set of CLEVR-Hans3, consisting of 750 images for each class. By utilizing the descriptions of images provided as metadata, we defined a vocabulary $\langle \mathsf{CN}, \mathsf{RN}, \mathsf{IN} \rangle$, in which there was an individual name for each image and for each depicted object (in IN), a concept name for each size, color, shape and material in addition to three concept names (Class1, Class2, Class3) corresponding to the set of classes $\mathcal{C}$ and two indicative concept names Image and Object, as concept names (in CN), and the role name (in RN) contains which was used to link images to objects it depicts. We used the (unique) names of the image files of the test set as names for the exemplars (EN) of our explanation dataset, so the mapping $\mathcal{M}$ was straightforward (mapping the file name to the respective image). We then created the knowledge base $\mathcal{S}$ over this vocabulary, with the ABox containing the semantic description of all images and the respective objects, and the TBox containing certain rather trivial inclusion axioms. The sets IN, CN, RN and the TBox of our knowledge base and the respective vocabulary were the following:

$$\mathsf{IN} = \{\mathsf{image1.png}, \mathsf{object1}_1, \mathsf{object2}_1, \ldots, \mathsf{objectN}_1, \mathsf{image2.png}, \mathsf{object1}_2, \mathsf{object2}_2, \ldots, \mathsf{image2250.png}, \mathsf{object1}_{2250}, \ldots, \mathsf{objectM}_{2250}\}$$

$$\mathsf{CN} = \{\mathsf{Image}, \mathsf{Object}, \mathsf{Cube}, \mathsf{Cylinder}, \mathsf{Sphere}, \mathsf{Metal}, \mathsf{Rubber}, \mathsf{Blue}, \mathsf{Brown}, \mathsf{Cyan}, \mathsf{Gray}, \mathsf{Green}, \mathsf{Purple}, \mathsf{Red}, \mathsf{Yellow}, \mathsf{Large}, \mathsf{Small}, \mathsf{Class1}, \mathsf{Class2}, \mathsf{Class3}\}$$

$$\mathsf{RN} = \{\mathsf{contains}\}$$

$$\mathcal{T} = \{C \sqsubseteq \mathsf{Object} \mid C \notin \{\mathsf{Image}, \mathsf{Object}, \mathsf{Class1}, \mathsf{Class2}, \mathsf{Class3}\}\}.$$

5.2.2. Setting

For the experiments on CLEVR-Hans3 we used the same ResNet34 [74] classifier and training procedure as those used by the creators of the dataset in [71]. The performance of the classifier is shown in Table 5 and so is the

confusion matrix which summarizes the predictions of the classifier indicating the differences between the actual and the predicted classes of the test samples. As expected, the classifier has lower values on some metrics regarding the first two classes, and this is attributed to the confounding factors and not the quality of the classifier, since it achieved 99.4% accuracy in the validation set. After training the classifier, we acquired its predictions on the test set and generated explanations for each class with Alg. 1 with QLCS as the merge operation. Afterwards, utilizing the reasoning capabilities of GraphDB, we loaded the knowledge base $\mathcal{S}$ of our explanation dataset, and obtained the certain answers of the corresponding explanation rule queries in order to evaluate the produced explanation rules. The evaluation was based on the metrics mentioned in Section 3, essentially comparing the certain answers of the explanation rule queries, with the corresponding exemplars of the pos-set of the classifier.

Table 5
Performance of the ResNet34 model on CLEVR-Hans3.

	Test set metrics			Confusion matrix		
True label	Precision	Recall	F1-score	Class 1	Class 2	Class 3
Class 1	0.94	0.16	0.27	118	511	121
Class 2	0.59	0.98	0.54	5	736	9
Class 3	0.85	1.00	0.92	2	0	748

5.2.3. Results

The explanation rules generated for the ResNet34 classifier using KGrules-H and QLCS as the merge operation, as outlined in section 4.3.1, are shown in Table 6, where we show the rule, the value of each metric and the numbers of positive individuals. The term positive individuals refers to the certain answers of the respective explanation rule query that are also elements of the pos-set (they are classified in the respective class).

In our representation of explanation rule queries in Tables 6,7 we have omitted the answer variable x, along with all conjuncts of the form x contains y and conjuncts of the form $\mathsf{Object}(y)$, for brevity. In addition, the rules in the Table are not formally written, to make more visually clear the characteristics of the objects involved. For example, the rule of the first row (Best precision for class 1) would formally be written $\mathsf{Exemplar}(x), \mathsf{contains}(x, y_1), \mathsf{contains}(x, y_2), \mathsf{contains}(x, y_3), \mathsf{Large}(y_1), \mathsf{Cube}(y_1), \mathsf{Gray}(y_1), \mathsf{Large}(y_2), \mathsf{Cylinder}(y_2), \mathsf{Large}(y_3), \mathsf{Metal}(y_3) \rightarrow \mathsf{Class1}(x)$.

The algorithm found a correct rule (precision = 1) for each class, in addition to a rule query with recall = 1, whose certain answers are a superset of the positive set. The best degree was achieved for class 3, which lacks a confounding factor, meaning the classifier is not expected to be biased. Correct rule queries are of particular interest since they can be translated into guaranteed IF-THEN rules which the classifier follows on the particular dataset. For instance the highest recall correct rule query for class 1 is translated into the rule “If the image contains a Large Gray Cube, a Large Cylinder and a Large Metal Object then it is classified to class 1.". This rule clearly shows the bias of the classifier, since it is the description of the class with the added confounding factor (the Large Cube is Gray). Similarly the (not correct) rule query with recall = 1 for the same class can be translated into the rule “If the image does not contain a Large Cube then it is not classified to class 1", since the set of certain answers is a super set of the positive set. We observed that correct rule queries tend to be more specific than others, with the most general rules with exceptions being those with recall = 1. Other rules which were correct with exceptions, tended to lie somewhere in the middle with respect to how general or specific they are, but they were the ones which lead to the highest values of degree. By observing these results, we concluded that in practice, a set of rules, both correct and with exceptions, can give us a very clear picture of what the black-box classifier is doing. However, in order to not overwhelm an end-user with a large number of rules, we should develop a strategy to select which rules to show to the user. Here, as opposed to the Mushroom experiment (5.1) the strategy we used was to show the highest recall, highest precision and highest degree rules, along with their exceptions if any, but as mentioned, we plan to explore additional strategies in the future, such as showing disjunctions of correct rules.

It is interesting to note that the rule query with recall = 1 produced for class 1 contained a Large Cube but not a Large Cylinder, which is also in the description of the class. This shows that in the training process the classifier

Table 6
Optimal explanations with regard to the three metrics on CLEVR-Hans3.

Metric	Explanation Rules	Precision	Recall	Degree	Positives
		Class 1			
Best Precision	y1 is Large, Cube, Gray. y2 is Large, Cylinder. y3 is Large, Metal.	1.00	0.66	0.66	83
Best Recall	y1 is Large, Cube.	0.09	1.00	0.09	125
Best Degree	y1 is Large, Cube, Gray. y2 is Large, Cylinder. y3 is Large, Metal.	1.00	0.66	0.66	83
		Class 2			
Best Precision	y1 is Small, Sphere. y2 is Large, Rubber. y3 is Small, Metal, Cube. y4 is Small, Brown. y5 is Small, Rubber, Cylinder.	1.00	0.09	0.09	116
Best Recall	y1 is Cube.	0.63	1.00	0.63	1247
Best Degree	y1 is Metal, Cube. y2 is Small, Metal.	0.78	0.8	0.65	1005
		Class 3			
Best Precision	y1 is Metal, Blue. y2 is Large, Blue, Sphere. y3 is Yellow, Small, Sphere. y4 is Small, Rubber. y5 is Metal, Sphere.	1.00	0.42	0.42	365
Best Recall	y1 is Large. y2 is Sphere.	0.42	1.00	0.42	878
Best Degree	y1 is Yellow, Small, Sphere. y2 is Large, Blue, Sphere.	0.99	0.85	0.85	748

learned to pay more attention to the presence of cubes rather than the presence of cylinders. The elements of the highest recall correct rule that differ from the true description of class 1 can be a great starting point for a closer inspection of the classifier. We expected the presence of a Gray Cube from the confounding factor introduced in the training and validation sets, but in a real world scenario similar insights can be reached by inspecting the queries. In our case, we further inquired the role that the Gray Cube and the Large Metal Object play in the correct rule by removing either of them from the query and examining its behavior. In Table 7 we can see that the gray color was essential for the correct rule while the Large Metal Object was not, and in fact its removal improved the rule and returned almost the entire class.

Another result that piqued our attention was the highest degree explanation for class 3 which is the actual rule that describes this class. This explanation was not a correct rule, since it had two exceptions, which we can also see in the confusion matrix of the classifier and we were interested to examine what sets these two individuals apart. We

found that both of these individuals are answers to the query "y1 is Large, Gray, Cube". This showed us once again the great effect the confounding factor of class 1 had on the classifier.

Our overall results show that the classifier tended to emphasize low level information such as color and shape and ignored higher level information such as texture and the combined presence of multiple objects. This was the reason why the confounding factor of class 1 had an important effect to the way images were classified, while the confounding factor of class 2 seemed to have had a much smaller one. Furthermore, the added bias made the classifier reject class 1 images, which however had to be classified to one of the other two classes (no class was not an option). Therefore one of the other classes had to be "polluted" by samples which were not confidently classified to a class. This motivates us to expand the framework in the future to work with more informative sets than the pos-set, such as elements which were classified with high confidence, and false and true, negatives and positives.

Table 7
Two modified versions of the class 1 correct rule produced by removing conjuncts.

Query	Positives	Negatives
y1 is Large, Cube. y2 is Large, Cylinder. y3 is Large, Metal.	108	547
y1 is Large, Cube, Gray. y2 is Large, Cylinder.	93	0

5.3. Visual Genome and Places

For the third set of experiments, we produced explanations of an image classifier trained on the Places365 dataset for scene classification. To do this, we constructed an explanation dataset from a subset of the visual genome, which includes a *scene-graph* for each image. Nodes of scene-graphs represent depicted objects, and edges represent relationships between them, while both edges and nodes are labeled with WordNet synsets. This allows us to encode the provided scene-graphs as an ABox, and to use the hyponym-hypernym hierarchy of WordNet as a TBox.

5.3.1. Explanation Dataset

To construct the explanation dataset, we first selected the two most confused classes based on the confusion matrix of the classifier on the Places365 test set. These were "Desert Sand" and "Desert Road". Then, we acquired the predictions of the classifier on the entirety of the visual genome dataset, and kept as expemplars the images for which the top prediction was one of the aformentioned classes. There were 273 such images. We defined a vocabulary $\langle \mathsf{CN}, \mathsf{RN}, \mathsf{IN} \rangle$, where the set of individual names IN contained a unique name for each image and for each depicted object, based on their visual genome IDs, while the mapping function was a dictionary linking image IDs (EN) to the corresponding image files. The size of IN was 6462. We initilized the set of concept names CN with two concepts: Image and Object and then added to CN all WordNet synsets that appear as node labels, in addition to all of their hypernyms. The size of CN was 1155. Similarly, the set of role names (RN) was initialized with the role $\mathsf{depicts}$, which links images to their depicted objects, and then extended with the set of WordNet synsets that appear as edge labels in the selected images' scene graphs, in addition to all of their hypernyms, totaling 226 role names. Using this vocabulary we constructed a knowledge base $\langle \mathcal{T}, \mathcal{A} \rangle$. The ABox $\mathcal{A}$ was initialized with assertions of the form $\mathsf{Image}(\mathsf{image}), \mathsf{depicts}(\mathsf{image}, \mathsf{object})$, encoding information about which images depict which objects. Then, based on the labels of the nodes of the scene-graphs of images we introduced concept assertions of the form $SomeWordNetConcept(\mathsf{object})$ and based on the labels of the edges of scene-graphs we introduced role assertions of the form $SomeWordNetRole(\mathsf{objectA}, \mathsf{objectB})$. Finally, the TBox $\mathcal{T}$ contained hierarchies of concepts and roles, based on the hypernym-hyponym relation from WordNet, encoded as $hyponym \sqsubseteq hypernym$.

$$\mathsf{IN} = \{\mathsf{image1.png}, \mathsf{object1}_1, \mathsf{object2}_1, \dots, \mathsf{objectN}_1, \mathsf{image2.png}, \mathsf{object1}_2, \mathsf{object2}_2, \dots, \mathsf{image2250.png}, \mathsf{object1}_{2250}, \dots, \mathsf{objectM}_{2250}\}$$

$$\mathsf{CN} = \{\mathsf{Image}, \mathsf{Object}, \mathsf{motorcycle.n.01}, \mathsf{car.n.01}, \mathsf{motor_vehicle.n.01}, \dots\}$$

$\mathsf{RN} = \{\mathsf{depicts}, \mathsf{drink.v.01}, \mathsf{eat.v.02}, \mathsf{consume.v.02} \dots\}$

$\mathcal{T} = \{\mathsf{motorcycle.n.01} \sqsubseteq \mathsf{motor_vehicle.n.01}, \mathsf{car.n.01} \sqsubseteq \mathsf{motor_vehicle.n.01} \dots, \mathsf{drink.v.02} \sqsubseteq \mathsf{consume.v.02},$
$\mathsf{eat.v.02} \sqsubseteq \mathsf{consume.v.02} \dots\}.$

5.3.2. Setting

As a black-box classifier in this set of experiments we used the *ResNet50* classifier[6] provided by the official GitHub repository[7] for models pretrained on Places365, which classifies images to 365 different classes. [8] The top1 error is 44.82% and the top5 error is 14.71%. As mentioned, we used the confusion matrix to select the two most confused classes to generate explanations for, which were "Desert Sand" and "Desert Road". As a merge operation for Alg. 1 we used GreedyCommonConjuncts (Alg. 4), as the QLCS produced queries which could not be sufficiently minimized. Similarly with before, we loaded the knowledge of the explanation dataset in GraphDB, and after acquiring the certain answers of queries generated by Alg. 1, we validated their quality with respect to the precision, recall and degree metrics. After observing the best generated queries, we sought to further verify their accuracy for describing the black-box classifier, by looking into the black box to find out which parts of the image contributed to the prediction. We used a technique to generate heatmaps of pixel importance called *Score-CAM*[75]. Score-CAM is a white-box method for producing heatmaps that works by selectively masking areas of the input image and determining their importance to the prediction of the classifier.

5.3.3. Results

The best rule queries for each metric for each of the two classes is shown in Table 8. For simplicity of presentation, we have ommited conjuncts of the form $\mathsf{Exemplar}(\mathsf{x})$, $\mathsf{Image}(\mathsf{x})$, $\mathsf{depicts}(\mathsf{x}, \mathsf{y})$ and $\mathsf{Object}(\mathsf{y})$, where x is the answer variable. For both classes the generated queries have some unexpected conjuncts despite decent performance with respect to the three metrics. For example the conjunct $\mathsf{giraffe.n.01}(\mathsf{y3})$ appearing in the best correct rule for "Desert Road", and the best degree rule for the same class being simply $\mathsf{Image}(\mathsf{x}), \mathsf{depicts}(\mathsf{x}, \mathsf{y1}), \mathsf{animal.n.01}(\mathsf{y1})$. Furthermore, for "Desert Sand", the concept road.n.01 stands out. The concept communicaton.n.02 has the hyopnyms sign.n.02 and written_communication.n.01 in the WordNet hierarchy which could refer to license plates and traffic signs. The second best query in terms of precision was $\mathsf{Image}(\mathsf{x}), \mathsf{depicts}(\mathsf{x}, \mathsf{y1}), \mathsf{depicts}(\mathsf{x}, \mathsf{y2}), \mathsf{depicts}(\mathsf{x}, \mathsf{y3}).\mathsf{motor_vehicle.n.01}(\mathsf{y1}), \mathsf{sky.n.01}(\mathsf{y2}), \mathsf{road.n.01}(\mathsf{y3}),$
$\mathsf{along.r.01}(\mathsf{y1}, \mathsf{y3})$. Again the concept motor_vehicle.n.01 stands out.

In Fig. 8 we present the saliency maps of three images from Visual Genome classified as "Desert Road" on the left and three classified as "Desert Sand" on the right. The saliency maps support the associations the queries highlighted, namely that images containing animals, especially giraffes, get classified as "Desert Road", while images with roads and vehicles get classified as "Desert Sand". Not all images exhibited such strong association. In Fig. 9 the classifier seems to be mostly influenced by the ground which is covered in dry grass. Given that the second highest value in the confusion matrix of this classifier was for the pair "Desert Sand", "Desert Vegetation" we conjencture that some mistakes were made during the training of this classifier or several images in the training set of the Places365 dataset are mislabeled. The classifier may have been fed images that should be described as "Desert Road" but with the target label being "Desert Sand" and images that should be described as "Desert Vegetation" but with the target label being "Desert Road". This would explain the weird associations exhibited and the conjuncts appearing in the explanations. The association of giraffes with the label "Desert Road" can be explained with mislabeled images depicting desert vegetation. This is still a wrongful association since giraffes can also populate grasslands such as in the top left image of Fig. 8.

We would like to point out that our method of producing explanations discovered these erroneous associations without asking from the end-user to manually inspect the heatmaps of a large number of images, rather than to read the best performing query for each one of our metrics. We believe that this provides a better user experience since it requires less effort for the initial discovery of suspect associations, which can then be thoroughly tested

[6] PyTorch model: http://places2.csail.mit.edu/models_places365/resnet50_places365.pth.tar
[7] https://github.com/CSAILVision/places365
[8] https://github.com/zhoubolei/places_devkit/blob/master/categories_places365.txt

Table 8
Optimal explanations with regard to the three metrics using the VG explanation dataset

Metric	Explanation Rules	Precision	Recall	Degree	Positives
	Desert Road				
Best Precision	y1 is field.n.01. y2 is natural_object.n.01. y3 is giraffe.n.01. y4 is body_part.n.01. y5 is woody_plant.n.01.	1.00	0.12	0.12	16
Best Recall	y1 is organism.n.01	0.54	1.00	0.54	139
Best Degree	y1 is animal.n.01	0.72	0.84	0.64	118
	Desert Sand				
Best Precision	y1 is instrumentality.n.03 y2 is road.n.01 y3 is communication.n.02 y4 is sky.n.01 y5 is tree.n.01	1.00	0.12	0.12	16
Best Recall	y1 is physical_entity.n.01	0.49	1.00	0.49	134
Best Degree	y1 is instrumentality.n.01 y2 is road.n.01	0.92	0.56	0.56	76

by examining other explanations such as heatmaps. Our method is also model-agnostic, while methods such as Score-CAM require access to the layers within a convolutional neural network.

5.4. MNIST

For the fourth and final set of experiments we used MNIST, which is a dataset containing grayscale images of handwritten digits [73]. It is a very popular dataset for machine learning research, and even though classification on MNIST is essentially a solved problem, many recent explainability approaches experiment on this dataset (for example [76]). For us, MNIST was ideal for experimenting with automatic explanation dataset generation by using traditional feature extraction from computer vision. An extension to this approach would be using more complex information extraction from images, such as object detection or scene graph generation, for applying the explainability framework to explain generic image classifiers. This however is left for future work.

5.4.1. Explanation Dataset

For creating the explanation dataset for MNIST, we manually selected a combination of 250 images from the test set, including both typical and unusual exemplars for each digit. The unusual exemplars were chosen following the mushroom experiment (5.1), in which we saw that small explanation datasets do not facilitate good explanation rules when the exemplars are chosen randomly, so we aimed for variety of semantic descriptions. In addition, the unusual exemplars tended to be misclassified, and we wanted to see how their presence would impact the explanations.

Since there was no semantic information available that could be used to construct an explanation dataset, we automatically extracted descriptions of the images, by using feature extraction methods. Specifically, the images were described as a collection of intersecting lines, varying in angle, length and location within the image. These lines were detected using the technique of ridge detection [77]. The angles of the lines were quantized to 0, 45, 90

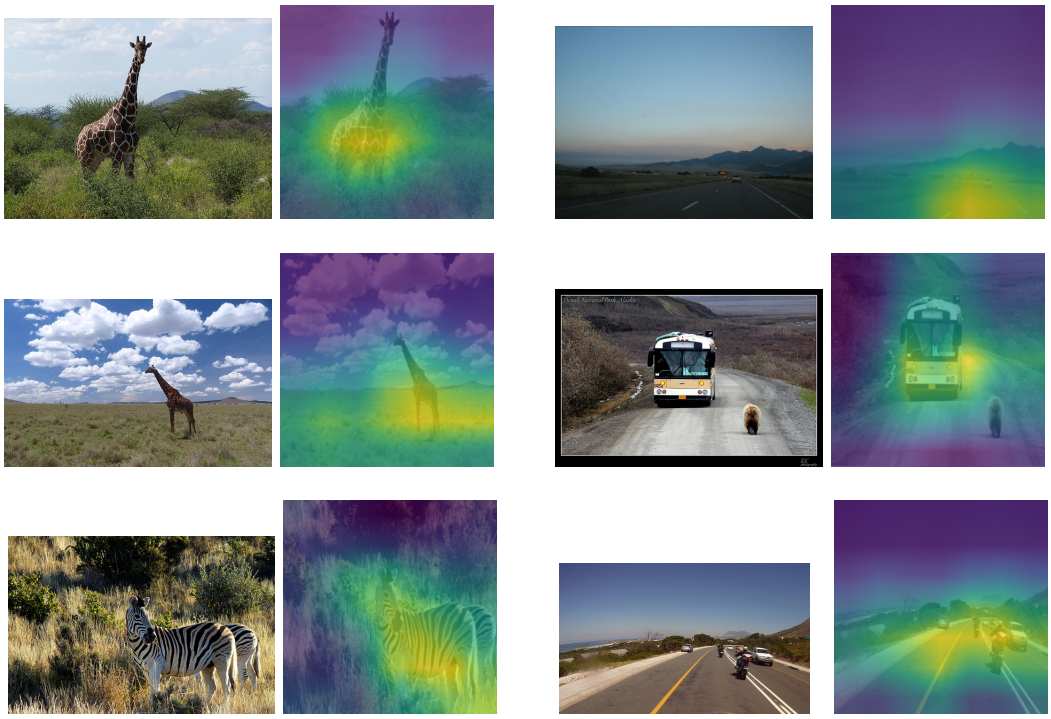
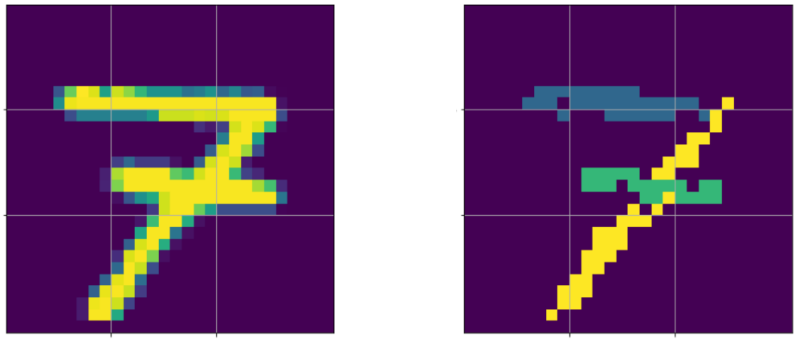


Fig. 8. Images from the Visual Genome dataset and their saliency maps produced by Score-CAM. Images on the left were classified as "Desert Road" and images on the right as "Desert Sand".

Fig. 9. An image classified as "Desert Road" exhibiting weaker association to the presence of a giraffe.

or 135 degrees, and the images were split into 3 horizontal (top, middle, bottom) and 3 vertical (left, center, right) zones which define 9 areas (top left, top center, ... , bottom right). For each line we noted the areas it passes through. In Fig. 10 we show an example of an MNIST image, along with the results of the aforementioned information extraction procedure using ridge detection.

Based on the selected images and the extracted information, we created our explanation dataset $\mathcal{E}$. We constructed a vocabulary $\langle \mathsf{CN}, \mathsf{RN}, \mathsf{IN} \rangle$, with an individual for each image and each line therein as individual names (IN), the concepts defining the angle, location and length of each line, two indicative concepts Image and Line, as well as ten concepts (one for each digit) corresponding to the set of classes $\mathcal{C}$, as concept names (CN), and the roles contains with domain Image, and range Line, indicating the existence of a line in a specific image, and the symmetric intersects with both domain and range Line, indicating that two lines intersect each other, as the role names (RN). To define the mapping $\mathcal{M}$ of the explanation dataset, after extracting the images and separating them into digits, we numbered them and named the Exemplars (and the corresponding file names) according to their digit and index. This also makes the images easy to retrieve. We then created the knowledge base $\mathcal{S}$ over this vocabulary, with the ABox containing the semantic description of all exemplar images and the respective lines, and the TBox containing certain



```
image contains 3 lines
line 0 is: Line0deg TopLeft TopCenter TopRight MidLeft MidCenter MidRight Long
line 1 is: Line0deg MidCenter MidRight Long
line 2 is: Line45deg TopRight MidCenter MidRight BotLeft BotCenter Long
line 0 intersects line 2
line 1 intersects line 2
```

Fig. 10. An example of a digit, the results of ridge detection, and the corresponding description.

rather trivial inclusion axioms. The vocabulary IN, CN, RN and the Tbox of our knowledge base are the following:

$$\mathsf{IN} = \{\mathsf{test_zero1}, \mathsf{test_zero1_line0}, \ldots, \mathsf{test_zero1_line7}, \mathsf{test_zero6}, \mathsf{test_zero6_line0}, \ldots, \mathsf{test_nine979}, \mathsf{test_nine979_line7}, \ldots, \mathsf{lineM}_{250}\}$$

$$\mathsf{CN} = \{\mathsf{Image}, \mathsf{Line}, \mathsf{Line0deg}, \mathsf{Line45deg}, \mathsf{Line90deg}, \mathsf{Line135deg}, \mathsf{TopLeft}, \mathsf{TopCenter}, \mathsf{TopRight}, \mathsf{MidLeft}, \mathsf{MidCenter}, \mathsf{MidRight}, \mathsf{BotLeft}, \mathsf{BotCenter}, \mathsf{BotRight}, \mathsf{Short}, \mathsf{Medium}, \mathsf{Long}\}$$

$$\mathsf{RN} = \{\mathsf{contains}, \mathsf{intersects}\}.$$

$$\mathcal{T} = \{C \sqsubseteq \mathsf{Line} \mid C \notin \{\mathsf{Image}, \mathsf{Line}\}\}.$$

5.4.2. *Setting*

For MNIST we used the example neural network provided by PyTorch [9] as the classifier to be explained. The classifier achieved 99.8% accuracy on the training set and 99.2% on the test set. On the explanation dataset, the accuracy is 73%, and a confusion matrix for the classifier on the explanation dataset is shown in Table 11. The performance of the classifier is poor when compared to the whole test set, due to the fact that we have included several manually selected exemplars, which are unusually drawn, but still valid as digits (based on our own judgement). Similarly to CLEVR-Hans3, we generated explanations for the predictions of the classifier on the exemplars using KGrules-HT (Alg. 3) and KGrules-H (Alg. 1) with Alg. 4 as the merge operation, and loaded the explanation dataset in GraphDB for acquiring certain answers. We also experimented with the QLCS as the merge operation, but the resulting queries mostly contained a large number of variables which could not be effectively minimized, which could be due to the complex connections between variables with a symmetric role.

5.4.3. *Results*

For MNIST there does not exist a ground truth semantic description for each class, as was the case for CLEVR-Hans3, nor is there a pre-determined bias of the classifier, thus we could not easily measure our framework's usefulness in this regard. Instead, since the explanation dataset was constructed automatically, we explored quality related features of the generated explanations.

For all digits the algorithm produced at least one correct rule (precision = 1) and a rule with exceptions with recall = 1. The highest degree of rule queries for each digit are shown as a bar-plot in Fig. 12a. In general, the values of the metric seem low, with the exception of digit 0, which would indicate that the algorithms did not find a single

[9] https://github.com/pytorch/examples/tree/master/mnist

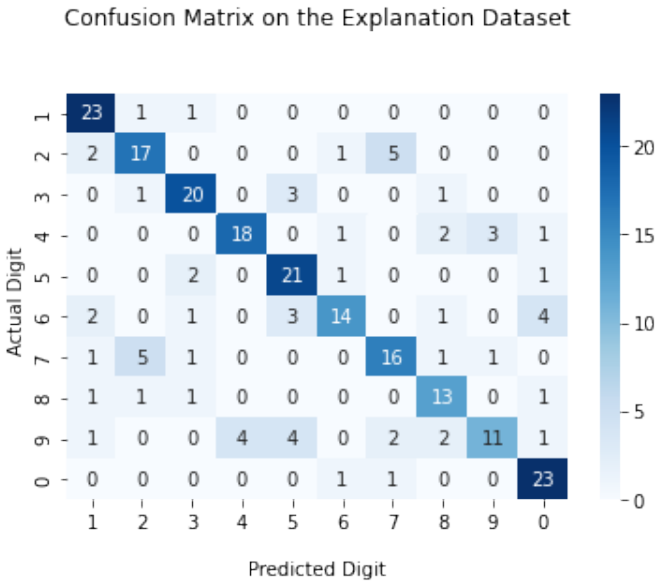
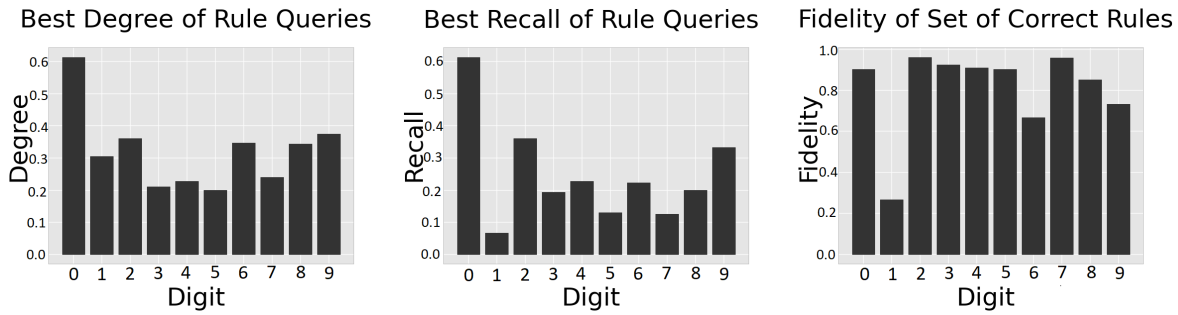


Fig. 11. Confusion matrix of the classifier on the explanation dataset for MNIST

rule which approximates the pos-set to a high degree. For some of the digits, including 0, the highest degree rule is also a correct rule. For closer inspection, we show the best degree rule query for digit 0 which is the highest, and for digit 5 which is the lowest.

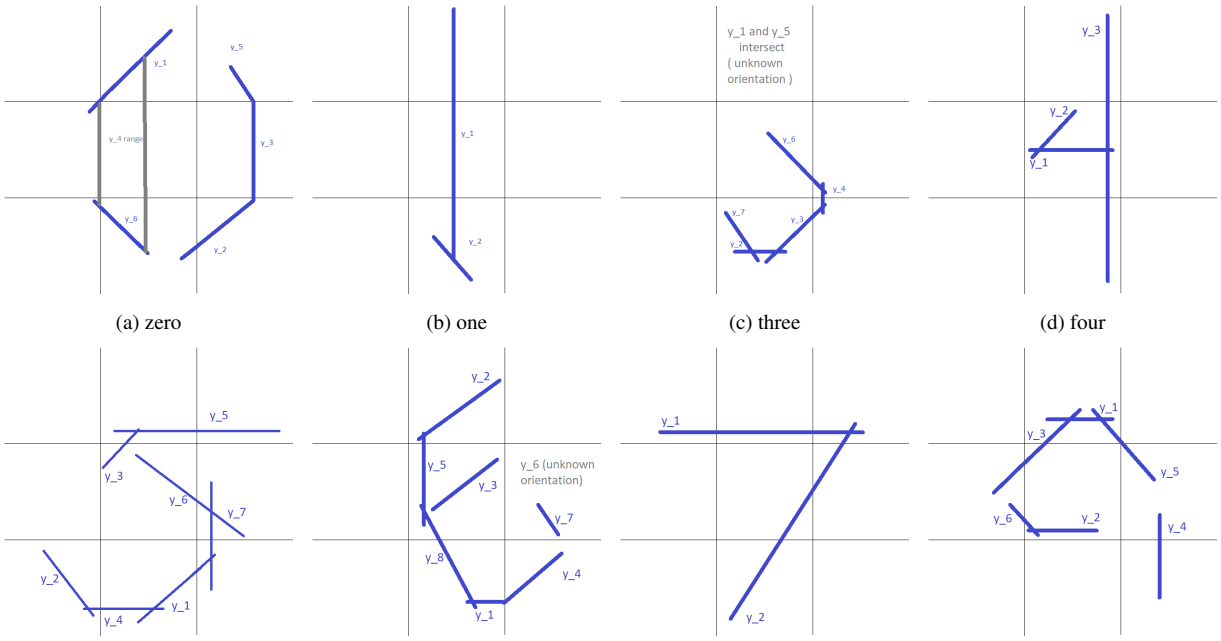


(a) Best degree for each digit (b) Best recall of correct rule queries (c) Fidelity of set of correct rules

Fig. 12. Metrics of generated rule queries for MNIST

The explanation rule for digit 0 involved six lines, as indicated by the conjuncts $\mathsf{contains}(x, y_1)$, $\mathsf{contains}(x, y_2)$, $\mathsf{contains}(x, y_3)$, $\mathsf{contains}(x, y_4)$, $\mathsf{contains}(x, y_5)$, $\mathsf{contains}(x, y_6)$. For five of the six lines, the explanation rule query included their location in the image, indicated by the conjuncts $\mathsf{TopCenter}(y_1)$, $\mathsf{BotRight}(y_2)$, $\mathsf{BotCenter}(y_2)$, $\mathsf{MidRight}(y_3)$, $\mathsf{TopRight}(y_5)$, $\mathsf{BotCenter}(y_6)$. For all six lines the explanation rule included information about their orientation, indicated by the conjuncts $\mathsf{Line45deg}(y_1)$, $\mathsf{Line45deg}(y_2)$, $\mathsf{Line90deg}(y_3)$, $\mathsf{Line90deg}(y_4)$, $\mathsf{Line135deg}(y_5)$, $\mathsf{Line135deg}(y_6)$. Finally, the rule-query included the following conjuncts which show which lines intersect each other $\mathsf{intersects}(y_1, y_4)$, $\mathsf{intersects}(y_2, y_3)$, $\mathsf{intersects}(y_3, y_5)$, $\mathsf{intersects}(y_4, y_6)$. A rule query with so many conjuncts could potentially be difficult for a user to decipher, so in this case we found it useful to visualize the rules. The above rule is visualized in Fig. 13a. The visualization is clear and intuitive as an explanation for digits classified as zeroes, however visualization of rules will not be possible in all applications. This shows the importance of taking under consideration understandability when designing explaination pipelines, which in our case depends mostly on the vocabulary and expressivity of the underlying explanation dataset. In this case, the vocabulary used

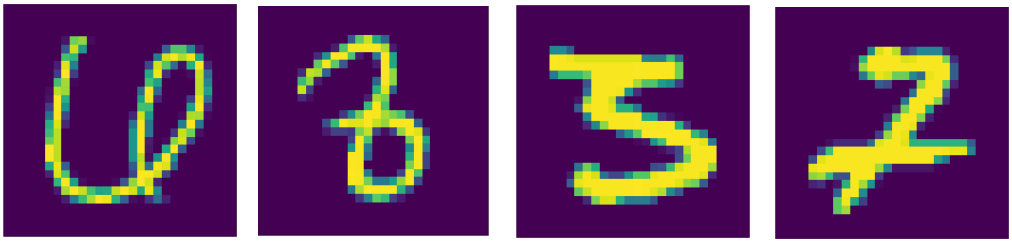
was itself somewhat obscure for users (sets of intersecting lines are not easy to understand by reading a rule), which could have been mitigated if the explanation dataset had been curated by humans and not created automatically. In this particular use-case it was not a problem however, since the visualization of rules was easy in most cases.



(a) zero (b) one (c) three (d) four

(e) five (f) six (g) seven (h) nine

Fig. 13. Visualizations of best recall correct rules for digits



(a) 6 misclassified as 0 (b) 8 misclassified as 3 (c) 3 misclassified as 5 (d) 7 misclassified as 2

Fig. 14. Misclassified digits that follow the best recall correct rules.

The highest degree explanation rule for digit 5, which was the lowest out of the best of all digits, again involved six lines indicated by the conjuncts contains(x, y_1), contains(x, y_2), contains(x, y_3), contains(x, y_4), contains(x, y_5), contains(x, y_6). This time however, only three lines had information about their location in the image, indicated by the conjuncts BotCenter(y_2), BotCenter(y_4), MidCenter(y_5), and five lines had information about their orientation Line0deg(y_1), Line0deg(y_2), Line45deg(y_3), Line45deg(y_4) Line135deg(y_6). Furthermore, this rule-query included information about the size of two lines, indicated by the conjuncts Medium(y_4),Short(y_6). Finally, as with before, we get a set of conjuncts showing which lines intersect each other: intersects(y_1, y_3), intersects(y_2, y_4), intersects(y_3, y_5). This rule query is not easy to understand and it is even difficult to visualize, since there is not enough information about the location of each line, thus it is not actually usable. This was expected to an extent, due to the low value of the degree metric, but again highlights the importance of taking usability under consideration when choosing which rule-queries to show to a user.

Regarding correct rules, the algorithm produced several for each digit. Since the sets of certain answers of correct rule queries are subsets of the pos-set of each class, we measured the per class fidelity of the disjunction of all correct rules, as if giving a user a rule-set, similarly to the Mushroom experiment (5.1). In Fig. 12c we show as a bar-plot the fidelity for each class. With the exception of digit one, the pos-sets of all digits were sufficiently covered by the set of correct rules. The failure for digit 1 was expected, since the descriptions of the exemplars classified as 1 contain few lines (for example consisting of a single large line in the middle) which tend to be part of descriptions of other digits as well (all digits could be drawn in a way in which there is a single line in the middle). This is a drawback of the open world assumption of DLs since we cannot guarantee the non-existence of lines that are not provided in the descriptions. The open world assumption is still desirable since it allows for incomplete descriptions of exemplars. In cases such as the medical motivating example used throughout this paper, a missing finding such as "Dyspnoea" does not always imply that the patient does not suffer from dyspnoea. It could also be a symptom that has not been detected or has been overlooked.

The highest recall of a single correct rule for each digit is shown as a bar-plot in Fig. 12b. Since correct rules are easily translated into IF-THEN rules, we expected them to be more informative than the highest degree ones, which requires looking at the exceptions to gain a clearer understanding of the rule. We investigate closer by analyzing the best correct rule for each digit.

For digit 0, the best correct rule rule was the same as the highest degree rule presented previously. In Fig. 14a we provide an example of a six misclassified as a 0, which follows this correct rule. Comparing the misclassified 6 with the visualizations for rules of the digits 0 (Fig. 13a) and 6 (Fig. 13f) we can see that this 6 might have been misclassified as a 0 because the closed loop part of the digit reaches the top of the image. According to the correct rule for 0, an image that contains two vertical semicircles in the left and right sides of the image is classified as a 0, and because of this peculiarity in the drawing of the misclassified six, the image (Fig. 13a) obeys this rule.

The best correct rule for digit 1 had the lowest recall out of all correct rules for other digits, which means that it returned a small subset of the positives. Specifically, this rule returned only two of the 30 individuals classified as the digit 1. It was still usable as an explanation however, as it only involved two lines $\mathsf{contains}(x, y_1)$, $\mathsf{contains}(x, y_2)$, both of which were thoroughly described with regard to their location $\mathsf{BotCenter}(y_1)$, $\mathsf{MidCenter}(y_1)$, $\mathsf{TopCenter}(y_1)$, $\mathsf{BotCenter}(y_2)$, their orientation $\mathsf{Line90deg}(y_1)$, $\mathsf{Line135deg}(y_2)$, their length $\mathsf{Long}(y_1)$, $\mathsf{Short}(y_2)$ and the fact that they intersect $\mathsf{intersects}(y_1, y_2)$. This rule is visualized in Fig. 13b.

For digit 2, the best correct rule query returned nine out of the 25 positives, three of which were missclassified by the classifier. This rule involved three lines, of which two had conjuncts indicating their location $\mathsf{BotRight}(y_1)$, $\mathsf{BotLeft}(y_3)$, $\mathsf{BotCenter}(y_3)$, $\mathsf{MidCenter}(y_3)$, only one line contained information about its orientation and size $\mathsf{Line45deg}(y_3)$, $\mathsf{Long}(y_3)$, and the only other information was that $\mathsf{intersects}(y_2, y_3)$. Note that for y_2 the only available information was that it intersects y_3. This query is difficult to visualize, due to the missing information about two of the three lines, however it is still useful as an explanation. Specifically, y_3 represents a long diagonal line from the bottom left to the middle of the image, which is a characteristic of only the digits 2 and 7. Additionally, there is a line of any orientation in the bottom right of the image, which would differentiate it from a typical 7, and another line which intersects the long diagonal at any position. As is apparent also in the confusion matrix of the classifier on the explanation dataset (Table 11), the black-box often mixed up sevens with twos, and this explanation rule returns one of the sevens which is misclassified as a two, shown in Fig. 14d. This digit is not typically drawn, and from the rule query the information we get is that it might have been misclassified because of the existence of a line at the bottom right.

To investigate closer, the next correct rule which we analyze is that of highest recall for digit 7. This query returned only three of the 24 images which were classified as sevens, and all three were correct predictions by the classifier. The rule involved two intersecting lines ($\mathsf{intersects}(\mathsf{y}_1, \mathsf{y}_2)$), of which the first is described as $\mathsf{Line0deg}(y_1)$, $\mathsf{Long}(y_1)$ $\mathsf{TopLeft}(y_1)$, $\mathsf{TopCenter}(y_1)$, $\mathsf{TopRight}(y_1)$ which is clearly the characteristic top part of the digit, while the second line is described as $\mathsf{Line45deg}(\mathsf{y}_2)$, $\mathsf{BotCenter}(\mathsf{y}_2)$, $\mathsf{MidCenter}(\mathsf{y}_2)$, $\mathsf{MidRight}(\mathsf{y}_2)$, $\mathsf{Long}(\mathsf{y}_2)$, which is the diagonal part of the digit. The description of the diagonal has a different description than the diagonal line which was part of the rule for digit 2. Instead of $\mathsf{BotLeft}(\mathsf{y})$, the rule contains a conjunct $\mathsf{MidRight}(\mathsf{y})$. This could be another hint as to why the 7 shown in Fig. 14d was not classified as correctly, as the digit appears to be leaning slightly to the right, which makes the diagonal pass through BotLeft, which is in the description of the diagonal for a digit 2 instead of

MidRight which is for digit 7. To conclude if this is the case however would require investigating more rules, since the one presented covers only a subset of exemplars classified as digit 7.

For digit 3, the best correct explanation rule returned five of the 26 individuals which were classified as 3, including one misclassified 8. This rule involved seven different lines, thus it was not expected to be understandable by a user at a glance. However, there was plentiful information for each line in the rule, which made it possible to visualize. Specifically, regarding the location of the lines, the rule query contained the conjuncts TopCenter(y_1), BotCenter(y_2), BotCenter(y_3), BotRight(y_4), MidRight(y_4), TopCenter(y_5), MidCenter(y_6), BotCenter(y_7). Regarding orientation, five of the seven lines had relevant conjuncts: Line0deg(y_2), Line45deg(y_3), Line90deg(y_4), Line135deg(y_6), Line135deg(y_7). Additionally, three lines had information about their size Short(y_4), Short(y_5), Medium(y_6). Finally, the explanation rule contained conjuncts showing which lines intersect each other: intersects(y_1, y_5), intersects(y_2, y_3), intersects(y_2, y_7), intersects(y_3, y_4), intersects(y_4, y_6). This rule-query is visualized in Fig. 13c. An interesting aspect of this explanation, is that the two lines which are at the top center (y_1 and y_5) do not have a specified orientation, while the other four lines are involved in more conjuncts in the explanation rule query and are described in more detail. This could be an indication of the importance of these lines for a digit to be classified as a 3. However, these lines could also be a part of other digits such as 5, which is the next digit which we analyze.

For the digit 5, the best correct rule query returned four of the 31 positives for the class of which one is a misclassified 3. It is a very specific query involving seven lines, all of which are described regarding their orientation Line45deg(y_1), Line135deg(y_2), Line45deg(y_3), Line0deg(y_4), Line0deg(y_5), Line135deg(y_6), Line90deg(y_7), and their location BotCenter(y_1), BotLeft(y_2), BotCenter(y_2), MidCenter(y_3), BotCenter(y_4), TopRight(y_5), TopCenter(y_5), MidRight(y_6), MidCenter(y_6), MidRight(y_7), BotRight(y_7). There was no information about lines' sizes, and there are five line intersections intersects(y_1, y_4), intersects(y_1, y_7), intersects(y_2, y_4), intersects(y_3, y_5), intersects(y_6, y_7). This query is visualized in Fig. 13e. An interesting aspect of this rule-query is the fact that it contains a misclassified 3 in its set of certain answers, specifically the three shown in Fig. 14c. In the context of the proposed framework, the 3 is misclassified because it obeys the correct rule for the digit 5. From comparing the visualization of the correct rule for the digit 5, and the misclassified 3, we can see that the digit obeys the rule because the top part of the three is vertically compressed, making it less distinguishable from a 5. This clearly shows us a potential flaw of the classifier.

For the digit 4, the best correct rule query returned five of the 22 positives all of which were correct predictions. The query involved three lines which were all well described regarding their orientation Line0deg(y_1), Line45deg(y_2), Line90deg(y_3), and their location MidCenter(y_1), MidCenter(y_2), BotCenter(y_3), MidCenter(y_3), TopCenter(y_3). Two lines were also described with respect to their size Medium(y_1), Long(y_3), and there were two intersections of lines intersects(y_1, y_2), intersects(y_1, y_3). This rule is visualized in Fig. 13d. This is a straightforward description of the digit four, and as expected it covers only true positives.

For the digit 6, the best resulting correct rule involved the most variables (each representing a line) out of all correct rules. It returned four of the 18 positives for the class all of which were correct predictions. Of the eight lines described in the query, seven had information about their orientation Line0deg(y_1), Line45deg(y_2), Line45deg(y_3), Line45deg(y_4), Line90deg(y_5), Line135deg(y_7), Line135deg(y_8). All lines were desribed with respect to their position BotCenter(y_1), TopCenter(y_2), MidCenter(y_3), BotRight(y_4), MidCenter(y_5), MidRight(y_6), MidRight(y_7), BotCenter(y_8), MidCenter(y_8). Additionally four lines had a determined size Short(y_1), Short(y_6), Short(y_7), Medium(y_8), and there were five intersections of lines intersects(y_1, y_4), intersects(y_1, y_8), intersects(y_2, y_5), intersects(y_5, y_8), intersects(y_6, y_7). This rule is visualized in Fig. 13f, it is a straight-forward description of a digit 6.

For digit 8, the best correct rule query returned four of the 20 positives, all of which are classified correctly. It involved seven lines, of which five were described with respect to their orientation Line45deg(y_2), Line90deg(y_3), Line135deg(y_5), Line45deg(y_6), Line0deg(y_7), six regarding their location MidCenter(y_1), BotCenter(y_2), BotCenter(y_3), MidCenter(y_3), BotCenter(y_4), TopCenter(y_6), TopCenter(y_7), and five regarding their size: Medium(y_1), Short(y_3), Short(y_4), Short(y_5), Short(y_7). Finally, the rule query involved four intersections of lines intersects(y_1, y_3), intersects(y_2, y_4), intersects(y_5, y_7), intersects(y_6, y_7). This rule is difficult to visualize due to the missing information (only four of the seven lines have information about both their location and orientation), and thus is not really useful as an explanation.

Finally, the best correct rule query for digit 9 returned five of the 15 positives, of which all were correct predictions by the classifier. It involved six lines which were all thoroughly described regarding their orientation Line0deg(y_1), Line0deg(y_2), Line45deg(y_3), Line90deg(y_4), Line135deg(y_5), Line135deg(y_6), and their location TopCenter(y_1), MidCenter(y_2), TopCenter(y_3), MidCenter(y_3), MidLeft(y_3), MidRight(y_4), BotRight(y_4), TopCenter(y_5), MidRight(y_5), MidLeft(y_6). Additionally, two lines were described regarding their size Medium(y_3), Short(y_6). The query also contained three conjuncts which indicated intersections of lines intersects(y_1, y_3), intersects(y_1, y_5), intersects(y_2, y_6). This query is visualized in Fig. 13h, it is a straight-forward description of a digit 9.

5.5. Discussion

The proposed approach performed similarly with the state-of-the-art on tabular data, was able to detect biases in the CLEVR-Hans case, detect flaws of the classifier in the Visual Genome case, and provided meaningful explanations even in the MNIST case. In this framework, the resulting explanations depend (almost exclusively) on the properties of the explanation dataset. In an ideal scenario, end-users trust the explanation dataset, the information it provides about the exemplars and the terminology it uses. It is like a quiz, or an exam, for the black-box, which is carefully curated by domain experts. This scenario was simulated in the CLEVR-Hans3 and Visual Genome use-cases in which the set of rules produced by the proposed algorithms clearly showed in which cases the black-box classifies items in specific classes, highlighting potential biases acquired during training. The framework is also useful when the explanation dataset is created automatically by leveraging traditional feature extraction, as is shown in the MNIST use-case. In this case, we found the resulting queries to be less understandable than before, which stems mainly from the vocabulary used, since sets of intersecting lines are not easily understandable unless they are visualized. They are also subjectively less trustworthy, since there are usually flaws with most automatic information extraction procedures. However, since sets of correct rules sufficiently covered the sets of individuals, and rules with exceptions achieved decent performance regarding precision, recall and degree, if an end-user invested time and effort to analyze the resulting rules, they could get a more clear picture about what the black-box is doing.

We also found interesting the comparison of correct rules with those with exceptions. Correct rules are, in general, more specific than others, as they always have a subset of the pos-set as certain answers. This means that, even though they might be more informative, they tend to involve more conjuncts than rules with exceptions, which in extreme cases could impact understandability. On the other hand, rules with exceptions can be more general, with fewer conjuncts, which could positively impact understandability. Although, utilizing these rules should involve examining the actual exceptions, which could be a lot of work for an end-user, we believe that these exceptions themselves can be very useful in order to detect biases and investigate the behaviour of the model on outliers. In order to eliminate the complexity involved in the investigation of these exceptions, in the future we plan to further study the area and propose a systematic way of exploiting them. These conclusions were apparent in the explanations generated for the class 3 of CLEVR-Hans3 (Table 6), where the best correct rule was very specific, involved five objects and had a relatively low recall (0.42), while the best rule with exceptions was exactly the ground truth class description and had very high precision (0.99). So in this case a user would probably gain more information about the classifier if they examined the rule with exceptions along with the few false positives, instead of examining the best correct rule, or a set of correct rules.

Another observation we made, is the fact that some conjuncts were more understandable than others when they were part of explanation rules. For instance in MNIST, knowing a line's location and orientation was imperative for understanding the rule via visualization, while conjuncts involving line intersections and sizes seemed not that important, regardless of metrics. This is something which could be leveraged either in explanation dataset construction (for example domain experts weigh concepts and roles depending on their importance for understandability), or in algorithm design (for example a user could provide as input concepts and roles which they want to appear in explanation rules). We are considering these ideas as a main direction for future work which involves developing strategies for choosing which rules are best to show to a user.

Finally, in the first experiment (5.1), it is shown that KGrules-H can be used to generate explanations in terms of feature data similarly to other rule-based methods, even if it is not the intended use-case. An interesting comparison for a user study would be between different vocabularies (for example using the features vs using external

knowledge). We note here that the proposed approach can always be applied on categorical feature data, since their transformation to an explanation dataset is straight-forward. This would not be the case if we also had numerical continuous features, in which case we would either require more expressive knowledge to represent these features, or that the continuous features be discretized. Another result which motivates us to explore different knowledge expressivities in the future, was the failure of the algorithms to produce a good (w.r.t. the metrics) explanation for the digit 1 in the MNIST experiment (5.4). Specifically, it was difficult to find a query which only returns images of this digit, since a typical description of a "1" is general and tends to always partially describe other digits. This is something which could be mitigated if we allowed for negation in the generated rules, and this is the second direction which we plan to explore in the future.

6. Conclusions and Future Work

In this work we have further developed a framework for rule-based *post hoc* explanations of black-box classifiers. The explanations are expressed as Horn rules which use terms from a given vocabulary, instead of the classifier's features. This allows for intuitive explanations even when the feature space of the classifier is too complex to be used within understandable rules (pixels, audio signals etc). The rules are also accompanied by theoretical guarantees regarding their *correctness* for explaining the classifier, given what we call an explanation dataset. The idea of the explanation dataset is at the core of our framework, as it is the probe we use to observe the black-box, by feeding it exemplar data and finding rules which explain its output. The explanation dataset also contains the knowledge from which the semantics of the rules are derived. The problem of finding such rules given an explanation dataset was approached as a search problem in the space of semantic queries, by starting with the most specific queries describing positively classified exemplars, and then progressively merging them using heuristic criteria. The queries are then approximately condensed, converted to rules and are shown to the end-user.

There are multiple directions towards which we plan to extend the framework in the future. First of all, we are currently investigating different strategies for choosing which explanation rules are best to show to a user such that they are both informative and understandable. To do this, we also plan to extend our evaluation framework for real world applications to include user studies. Specifically, we are focusing on decision critical domains in which explainability is crucial, such as the medical domain, and in collaboration with domain experts, we are developing explanation datasets, in addition to a crowd-sourced explanation evaluation platform. There are many interesting research questions which we are exploring in this context, such as what constitutes a good explanation dataset, what is a good explanation, and how can we build the trust required for opaque AI to be utilized in such applications.

Another direction which we are currently exploring involves further investigation of our framework in order to fully exploit its capabilities. We are currently working on the systematic study of the exceptions, and we're searching for ways to utilize them efficiently within our framework. We believe that this could provide to the end-user useful information regarding biases and outliers, and could drive the way towards incorporating local explanations to our framework. Stemming from this, we want to extend the framework, both in theory and in practice, to incorporate different types of explanations. This includes local explanations which explain individual predictions of the black-box, and counterfactual or contrastive explanations which highlight how a specific input should be modified in order for the prediction of the classifier to change. This extension is being researched with the end-user in mind, and we are exploring the merits of providing a blend of explanations (global, local, counterfactual) to an end-user.

A fourth and final direction to be explored involves extending the expressivity of explanation rules, in addition to that of the underlying knowledge. Specifically, the algorithms developed in this work require that if the knowledge has a non-empty TBox, it has to be eliminated via materialization before running the algorithms. Thus, we are exploring ideas for algorithms which generate explanation rules in the case where the underlying knowledge is represented with DL dialects in which the TBox cannot be eliminated, such as DL_Lite. Finally, regarding the expressivity of explanation rules, we plan to extend the framework to allow for disjunction, which is a straight-forward extension, and for negation, which is much harder to incorporate in the framework while maintaining the theoretical guarantees, which we believe are crucial for building trust with end-users.

References

[1] B. Goodman and S.R. Flaxman, European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation", *AI Mag.* **38**(3) (2017), 50–57.

[2] M. Turek, Explainable artificial intelligence (XAI), *Defense Advanced Research Projects Agency. http://web.archive.org/web/20190728055815/https://www.darpa.mil/program/explainable-artificial-intelligence* (2018).

[3] W.J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl and B. Yu, Interpretable machine learning: definitions, methods, and applications, *CoRR* **abs/1901.04592** (2019).

[4] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti and D. Pedreschi, A Survey of Methods for Explaining Black Box Models, *ACM Comput. Surv.* **51**(5) (2019), 93:1–93:42.

[5] F. Lecue, On the role of knowledge graphs in explainable AI, *Semantic Web* **11** (2019), 1–11. doi:10.3233/SW-190374.

[6] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi and P.F. Patel-Schneider (eds), *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, 2003.

[7] J. van der Waa, E. Nieuwburg, A. Cremers and M. Neerincx, Evaluating XAI: A comparison of rule-based and example-based explanations, *Artificial Intelligence* **291** (2021), 103404.

[8] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D.A. Shamma et al., Visual genome: Connecting language and vision using crowdsourced dense image annotations, *International journal of computer vision* **123**(1) (2017), 32–73.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C.L. Zitnick, Microsoft coco: Common objects in context, in: *European conference on computer vision*, Springer, 2014, pp. 740–755.

[10] G.A. Miller, WordNet: a lexical database for English, *Communications of the ACM* **38**(11) (1995), 39–41.

[11] R. Speer, J. Chin and C. Havasi, ConceptNet 5.5: An Open Multilingual Graph of General Knowledge, in: *AAAI*, AAAI Press, 2017, pp. 4444–4451.

[12] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer et al., Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia, *Semantic web* **6**(2) (2015), 167–195.

[13] M.Q. Stearns, C. Price, K.A. Spackman and A.Y. Wang, SNOMED clinical terms: overview of the development process and project status, in: *AMIA*, AMIA, 2001.

[14] G. Futia and A. Vetrò, On the integration of knowledge graphs into deep learning models for a more comprehensible AI—Three Challenges for future research, *Information* **11**(2) (2020), 122.

[15] M.K. Sarker, N. Xie, D. Doran, M. Raymer and P. Hitzler, Explaining Trained Neural Networks with Semantic Web Technologies: First Steps, in: *NeSy*, CEUR Workshop Proceedings, Vol. 2003, CEUR-WS.org, 2017.

[16] C. Panigutti, A. Perotti and D. Pedreschi, Doctor XAI: an ontology-based approach to black-box sequential data classification explanations, in: *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 2020, pp. 629–639.

[17] Z.A. Daniels, L.D. Frank, C.J. Menart, M. Raymer and P. Hitzler, A framework for explainable deep neural models using external knowledge graphs, in: *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II*, Vol. 11413, International Society for Optics and Photonics, 2020, p. 114131C.

[18] I. Tiddi and S. Schlobach, Knowledge Graphs as tools for Explainable Machine Learning: a survey, *Artificial Intelligence* (2021), 103627.

[19] R. Confalonieri, T. Weyde, T.R. Besold and F. Moscoso del Prado Martín, Using ontologies to enhance human understandability of global post-hoc explanations of black-box models, *Artificial Intelligence* **296** (2021), 103471. doi:https://doi.org/10.1016/j.artint.2021.103471. https://www.sciencedirect.com/science/article/pii/S0004370221000229.

[20] F. Croce, G. Cima, M. Lenzerini and T. Catarci, Ontology-based explanation of classifiers, in: *EDBT/ICDT Workshops*, 2020.

[21] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutiérrez, J.E.L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.N. Ngomo, S.M. Rashid, A. Rula, L. Schmelzeisen, J.F. Sequeda, S. Staab and A. Zimmermann, Knowledge Graphs, *CoRR* **abs/2003.02320** (2020).

[22] B. Mittelstadt, C. Russell and S. Wachter, Explaining explanations in AI, in: *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 279–288.

[23] V. Praher, K. Prinz, A. Flexer and G. Widmer, On the Veracity of Local, Model-agnostic Explanations in Audio Classification: Targeted Investigations with Adversarial Examples, *arXiv preprint arXiv:2107.09045* (2021).

[24] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, *Nature Machine Intelligence* **1**(5) (2019), 206–215.

[25] H. Yang, C. Rudin and M.I. Seltzer, Scalable Bayesian Rule Lists, in: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 3921–3930. http://proceedings.mlr.press/v70/yang17h.html.

[26] Y. Ming, H. Qu and E. Bertini, RuleMatrix: Visualizing and Understanding Classifiers with Rules, *IEEE Trans. Vis. Comput. Graph.* **25**(1) (2019), 342–352. doi:10.1109/TVCG.2018.2864812.

[27] J. Lehmann, S. Bader and P. Hitzler, Extracting reduced logic programs from artificial neural networks, *Appl. Intell.* **32**(3) (2010), 249–266. doi:10.1007/s10489-008-0142-y.

[28] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini and F. Giannotti, Local rule-based explanations of black box decision systems, *arXiv preprint arXiv:1805.10820* (2018).

[29] M.T. Ribeiro, S. Singh and C. Guestrin, Anchors: High-precision model-agnostic explanations, in: *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32, 2018.

[30] M.W. Craven and J.W. Shavlik, Extracting Tree-Structured Representations of Trained Networks, in: *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*, 1995, pp. 24–30. http://papers.nips.cc/paper/1152-extracting-tree-structured-representations-of-trained-networks.

[31] Y. Zhou and G. Hooker, Interpreting models via single tree approximation, *arXiv preprint arXiv:1610.09036* (2016).

[32] G. Ciravegna, F. Giannini, M. Gori, M. Maggini and S. Melacci, Human-Driven FOL Explanations of Deep Learning, in: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 2020, pp. 2234–2240. doi:10.24963/ijcai.2020/309.

[33] D. Pedreschi, F. Giannotti, R. Guidotti, A. Monreale, S. Ruggieri and F. Turini, Meaningful Explanations of Black Box AI Decision Systems, in: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 2019, pp. 9780–9784. doi:10.1609/aaai.v33i01.33019780.

[34] A.B. Arrieta, N.D. Rodríguez, J.D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila and F. Herrera, Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, *Inf. Fusion* **58** (2020), 82–115.

[35] Q.T. Tran, C.Y. Chan and S. Parthasarathy, Query reverse engineering, *VLDB J.* **23**(5) (2014), 721–746. doi:10.1007/s00778-013-0349-3.

[36] D.M.L. Martins, Reverse engineering database queries from examples: State-of-the-art, challenges, and research opportunities, *Information Systems* **83** (2019), 89–100. doi:https://doi.org/10.1016/j.is.2019.03.002. https://www.sciencedirect.com/science/article/pii/S0306437918300978.

[37] H. Li, C.-Y. Chan and D. Maier, Query from Examples: An Iterative, Data-Driven Approach to Query Construction, *Proc. VLDB Endow.* **8**(13) (2015), 2158–2169–. doi:10.14778/2831360.2831369.

[38] A. Bonifati, R. Ciucanu and S. Staworko, Learning Join Queries from User Examples, *ACM Trans. Database Syst.* **40**(4) (2016). doi:10.1145/2818637.

[39] C. Wang, A. Cheung and R. Bodik, Synthesizing Highly Expressive SQL Queries from Input-Output Examples, *SIGPLAN Not.* **52**(6) (2017), 452–466–. doi:10.1145/3140587.3062365.

[40] M. Zhang, H. Elmeleegy, C.M. Procopiuc and D. Srivastava, Reverse Engineering Complex Join Queries, in: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 809–820–. ISBN 9781450320375. doi:10.1145/2463676.2465320.

[41] D. Calvanese, G.D. Giacomo, D. Lembo, M. Lenzerini and R. Rosati, Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family, *J. Autom. Reason.* **39**(3) (2007), 385–429.

[42] B.C. Grau, B. Motik, G. Stoilos and I. Horrocks, Computing Datalog Rewritings Beyond Horn Ontologies, in: *IJCAI*, IJCAI/AAAI, 2013, pp. 832–838.

[43] A. Chortaras, M. Giazitzoglou and G. Stamou, Inside the Query Space of DL Knowledge Bases, in: *Description Logics*, CEUR Workshop Proceedings, Vol. 2373, CEUR-WS.org, 2019.

[44] D. Trivela, G. Stoilos, A. Chortaras and G. Stamou, Resolution-based rewriting for Horn-*SHIQ* ontologies, *Knowl. Inf. Syst.* **62**(1) (2020), 107–143.

[45] A. Petrova, E.V. Kostylev, B.C. Grau and I. Horrocks, Query-Based Entity Comparison in Knowledge Graphs Revisited, in: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I.F. Cruz, A. Hogan, J. Song, M. Lefrançois and F. Gandon, eds, Lecture Notes in Computer Science, Vol. 11778, Springer, 2019, pp. 558–575. doi:10.1007/978-3-030-30793-6_32.

[46] E. Dervakos, O. Menis-Mastromichalakis, A. Chortaras and G. Stamou, Computing Rule-Based Explanations of Machine Learning Classifiers using Knowledge Graphs, 2022.

[47] V. Gutiérrez-Basulto, J.C. Jung and L. Sabellek, Reverse Engineering Queries in Ontology-Enriched Systems: The Case of Expressive Horn Description Logic Ontologies, in: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 1847–1853. doi:10.24963/ijcai.2018/255.

[48] M. Ortiz, Ontology-Mediated Queries from Examples: a Glimpse at the DL-Lite Case, in: *GCAI 2019. Proceedings of the 5th Global Conference on Artificial Intelligence*, D. Calvanese and L. Iocchi, eds, EPiC Series in Computing, Vol. 65, EasyChair, 2019, pp. 1–14. ISSN 2398-7340. doi:10.29007/jhtz. https://easychair.org/publications/paper/c3CT.

[49] P. Barceló and M. Romero, The complexity of reverse engineering problems for conjunctive queries (2016).

[50] J. Jung, C. Lutz, H. Pulcini and F. Wolter, Logical Separability of Incomplete Data under Ontologies, 2020, pp. 517–528. doi:10.24963/kr.2020/52.

[51] M. Arenas, G.I. Diaz and E.V. Kostylev, Reverse Engineering SPARQL Queries, in: *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks and B.Y. Zhao, eds, ACM, 2016, pp. 239–249. doi:10.1145/2872427.2882989.

[52] G. Cima, F. Croce and M. Lenzerini, Query Definability and Its Approximations in Ontology-Based Data Management, in: *Proceedings of the 30th ACM International Conference on Information &; Knowledge Management*, CIKM '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 271–280–. ISBN 9781450384469. doi:10.1145/3459637.3482466.

[53] G. Gottlob and C.G. Fermüller, Removing Redundancy from a Clause, *Artif. Intell.* **61**(2) (1993), 263–289.

[54] W.W. Cohen, A. Borgida and H. Hirsh, Computing Least Common Subsumers in Description Logics, in: *Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, USA, July 12-16, 1992*, 1992, pp. 754–760. http://www.aaai.org/Library/AAAI/1992/aaai92-117.php.

[55] R. Küsters and R. Molitor, Structural Subsumption and Least Common Subsumers in a Description Logic with Existential and Number Restrictions, *Stud Logica* **81**(2) (2005), 227–259. doi:10.1007/s11225-005-3705-5.

[56] F. Baader, B. Sertkaya and A. Turhan, Computing the least common subsumer w.r.t. a background terminology, *J. Appl. Log.* **5**(3) (2007), 392–420. doi:10.1016/j.jal.2006.03.002.
[57] F.M. Donini, S. Colucci, T.D. Noia and E.D. Sciascio, A Tableaux-based Method for Computing Least Common Subsumers for Expressive Description Logics, in: *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30, 2009*, 2009. http://ceur-ws.org/Vol-477/paper_22.pdf.
[58] J. Laguarta, F. Hueto and B. Subirana, COVID-19 artificial intelligence diagnosis using only cough recordings, *IEEE Open Journal of Engineering in Medicine and Biology* **1** (2020), 275–281.
[59] G. Diaz, M. Arenas and M. Benedikt, SPARQLByE: Querying RDF Data by Example, *Proc. VLDB Endow.* **9**(13) (2016), 1533–1536–. doi:10.14778/3007263.3007302.
[60] J. Liartis, E. Dervakos, O. Menis-Mastromichalakis, A. Chortaras and G. Stamou, Semantic Queries Explaining Opaque Machine Learning Classifiers, in: *Proceedings of the Workshop on Data meets Applied Ontologies in Explainable AI (DAO-XAI 2021) part of Bratislava Knowledge September (BAKS 2021), Bratislava, Slovakia, September 18th to 19th, 2021*, R. Confalonieri, O. Kutz and D. Calvanese, eds, CEUR Workshop Proceedings, Vol. 2998, CEUR-WS.org, 2021. http://ceur-ws.org/Vol-2998/paper2.pdf.
[61] M.M. Zloof, Query-by-example: the invocation and definition of tables and forms, in: *VLDB '75*, 1975.
[62] R. Kontchakov and M. Zakharyaschev, in: *Reasoning Web. Reasoning on the Web in the Big Data Era: 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings*, Springer International Publishing, Cham, 2014, pp. 195–244. ISBN 978-3-319-10587-1. doi:10.1007/978-3-319-10587-1_5.
[63] R. Kontchakov, C. Lutz, D. Toman, F. Wolter and M. Zakharyaschev, The Combined Approach to Ontology-Based Data Access, in: *IJCAI*, IJCAI/AAAI, 2011, pp. 2656–2661.
[64] B. Glimm, Y. Kazakov and T. Tran, Ontology Materialization by Abstraction Refinement in Horn SHOIF, in: *AAAI*, AAAI Press, 2017, pp. 1114–1120.
[65] B. ten Cate and V. Dalmau, The Product Homomorphism Problem and Applications, in: *18th International Conference on Database Theory (ICDT 2015)*, M. Arenas and M. Ugarte, eds, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 31, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2015, pp. 161–176. ISSN 1868-8969. ISBN 978-3-939897-79-8. doi:10.4230/LIPIcs.ICDT.2015.161. http://drops.dagstuhl.de/opus/volltexte/2015/4983.
[66] S.H. Bokhari, On the Mapping Problem **30**(3) (1981). doi:10.1109/TC.1981.1675756.
[67] H.M. Grindley, P.J. Artymiuk, D.W. Rice and P. Willett, Identification of Tertiary Structure Resemblance in Proteins Using a Maximal Common Subgraph Isomorphism Algorithm, *Journal of Molecular Biology* **229**(3) (1993), 707–721. doi:https://doi.org/10.1006/jmbi.1993.1074. https://www.sciencedirect.com/science/article/pii/S0022283683710740.
[68] J. Xu, GMA: A Generic Match Algorithm for Structural Homomorphism, Isomorphism, and Maximal Common Substructure Match and Its Applications, *Journal of Chemical Information and Computer Sciences* **36**(1) (1996), 25–34. doi:10.1021/ci950061u.
[69] A. Egozi, Y. Keller and H. Guterman, A Probabilistic Approach to Spectral Graph Matching, *IEEE Transactions on Pattern Analysis & Machine Intelligence* **35**(01) (2013), 18–27. doi:10.1109/TPAMI.2012.51.
[70] M. Leordeanu and M. Hebert, A spectral technique for correspondence problems using pairwise constraints, in: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Vol. 2, 2005, pp. 1482–1489 Vol. 2. doi:10.1109/ICCV.2005.20.
[71] W. Stammer, P. Schramowski and K. Kersting, Right for the Right Concept: Revising Neuro-Symbolic Concepts by Interacting with their Explanations, *arXiv preprint arXiv:2011.12854* (2020).
[72] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva and A. Torralba, Places: A 10 million image database for scene recognition, *IEEE transactions on pattern analysis and machine intelligence* **40**(6) (2017), 1452–1464.
[73] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, Gradient-Based Learning Applied to Document Recognition, in: *Proceedings of the IEEE, 86(11):2278-2324*, 1998.
[74] K. He, X. Zhang, S. Ren and J. Sun, Deep Residual Learning for Image Recognition, *CoRR* **abs/1512.03385** (2015). http://arxiv.org/abs/1512.03385.
[75] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel and X. Hu, Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks, arXiv, 2019. doi:10.48550/ARXIV.1910.01279. https://arxiv.org/abs/1910.01279.
[76] R. Poyiadzi, K. Sokol, R. Santos-Rodríguez, T.D. Bie and P.A. Flach, FACE: Feasible and Actionable Counterfactual Explanations, in: *AIES*, ACM, 2020, pp. 344–350.
[77] T. Lindeberg, *Scale-Space*, in: *Wiley Encyclopedia of Computer Science and Engineering*, American Cancer Society, 2008, pp. 2495–2504. ISBN 9780470050118. doi:https://doi.org/10.1002/9780470050118.ecse609.
[78] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. Li, D.A. Shamma, M.S. Bernstein and L. Fei-Fei, Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations, *Int. J. Comput. Vis.* **123**(1) (2017), 32–73.
[79] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C.L. Zitnick and R.B. Girshick, CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning, *CoRR* **abs/1612.06890** (2016). http://arxiv.org/abs/1612.06890.
[80] F. Baader, I. Horrocks, C. Lutz and U. Sattler, *An Introduction to Description Logic*, Cambridge University Press, 2017. doi:10.1017/9781139025355.
[81] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019.
[82] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, RoBERTa: A Robustly Optimized BERT Pretraining Approach, 2019.

[83] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C.H. So and J. Kang, BioBERT: a pre-trained biomedical language representation model for biomedical text mining, *Bioinformatics* (2019). doi:10.1093/bioinformatics/btz682.
[84] G.A. Miller, WordNet: A Lexical Database for English, *Commun. ACM* **38**(11) (1995), 39–41–. doi:10.1145/219717.219748.
[85] M. Alirezaie, M. Längkvist, M. Sioutis and A. Loutfi, A symbolic approach for explaining errors in image classification tasks, in: *Working Papers and Documents of the IJCAI-ECAI-2018 Workshop on*, 2018.
[86] H. Wang, F. Zhang, X. Xie and M. Guo, DKN: Deep Knowledge-Aware Network for News Recommendation, in: *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, 2018, pp. 1835–1844. doi:10.1145/3178876.3186175.
[87] Q. Ai, V. Azizi, X. Chen and Y. Zhang, Learning heterogeneous knowledge base embeddings for explainable recommendation, *Algorithms* **11**(9) (2018), 137.
[88] V.S. Silva, A. Freitas and S. Handschuh, Exploring knowledge graphs in an interpretable composite approach for text entailment, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 7023–7030.
[89] Y. Zhang, P. Tiňo, A. Leonardis and K. Tang, A Survey on Neural Network Interpretability, *IEEE Trans. Emerg. Top. Comput. Intell.* **5**(5) (2021), 726–742. doi:10.1109/TETCI.2021.3100641.