Rule-Based Explanations for Retrieval-Augmented LLM Systems

Joel Rorseth[1*], Parke Godfrey[2], Lukasz Golab[1],
Divesh Srivastava[3], Jarek Szlichta[2]

[1*] University of Waterloo, Waterloo, Ontario, Canada.
[2] York University, Toronto, Ontario, Canada.
[3] AT&T Chief Data Office, Bedminster, New Jersey, USA.

*Corresponding author(s). E-mail(s): jerorset@uwaterloo.ca;
Contributing authors: godfrey@yorku.ca; lgolab@uwaterloo.ca;
divesh@research.att.com; szlichta@yorku.ca;

Abstract

If-then rules are widely used to explain machine learning models; e.g., "if employed = no, then loan application = rejected." We present the first proposal to apply rules to explain the emerging class of large language models (LLMs) with retrieval-augmented generation (RAG). Since RAG enables LLM systems to incorporate retrieved information sources at inference time, rules linking the presence or absence of sources can explain output provenance; e.g., "if a Times Higher Education ranking article is retrieved, then the LLM ranks Oxford first." To generate such rules, a brute force approach would probe the LLM with all source combinations and check if the presence or absence of any sources leads to the same output. We propose optimizations to speed up rule generation, inspired by Apriori-like pruning from frequent itemset mining but redefined within the scope of our novel problem. We conclude with qualitative and quantitative experiments demonstrating our solutions' value and efficiency.

Keywords: Explainability, Rule-Based Explanations, Large Language Models, Retrieval-Augmented Generation

1 Introduction

Advances in large language models (LLMs) have yielded increasingly sophisticated artificial intelligence (AI) systems capable of accomplishing a wide variety of tasks. These advances are enabled not only by the increasing size and architectural complexity of the models, but also by innovations in how LLMs incorporate knowledge. One such prominent innovation is *retrieval-augmented generation* (RAG), by which the static, pre-trained knowledge of an LLM is augmented with dynamic, arbitrary knowledge retrieved at inference time. Although RAG enables more control over an LLM's data, the provenance of information in the LLM's response is further complicated as a result.

The emerging field of *explainable artificial intelligence* (XAI) has been addressing issues related to understanding the behavior of complex models, including LLMs. Many XAI methods focus on the simpler problem of *local* explanation, which attempts to explain specific model predictions rather than general model behavior. One popular local XAI methodology is *feature attribution*, which explains a model output by scoring the relative importance of the input features. For RAG systems, context attribution can assign scores to each piece of information retrieved by an LLM system at inference time (Cohen-Wang et al. 2024). However, feature attribution is not actionable (e.g., to achieve recourse).

Counterfactual explanations address this problem, explaining a model's output by citing a minimal input perturbation that would induce an alternative output (Wachter et al. 2017). In RAG, a counterfactual explanation might reveal that the LLM would have generated a different answer if some piece of information were not provided or retrieved at runtime (Rorseth et al. 2024). However, there may be many minimal counterfactuals for a given prediction, which may overwhelm end users. Explanations using if-then rules mitigate these problems by summarizing patterns observed across a model's predictions (Ribeiro et al. 2018; Macha et al. 2022; Guidotti et al. 2019), but have not yet been considered in the context of LLMs with RAG.

We fill this research gap and present a rule framework for explaining the output of black-box RAG systems using rules that formulate input predicates based on the *presence* of RAG sources provided to the LLM. Let us illustrate using a healthcare example. Suppose that a clinic is using a RAG system connected to a medical database, and a medical research assistant working at the clinic asks the LLM, "What treatment is most effective for Long COVID fatigue?" Here, a rule could reveal that "Whenever 'Document 123' is included in the RAG sources, the LLM always recommends an unsafe Long COVID treatment." (A rule's antecedent could likewise be when a given document is *omitted from* the RAG sources.) This rule helps curators of the medical database flag documents such as Document 123 that, unbeknownst to them, advocate for unsafe treatments. In this case, Document 123 may advocate for a treatment that was thought to be effective until recently, while newer studies have shown it is both ineffective and unsafe for certain demographics. Our framework empowers developers, auditors, policy makers, and other end users to seek rules for *any* identifiable output condition, such as responses that are incorrect, contain misinformation, or reflect negative sentiment (Han et al. 2024).

Our main contributions are as follows.

1. We propose a **rule-based explanation** formulation for RAG, the first of its kind. We offer two variants that predicate on the basis of *retaining* or *omitting* features (RAG sources). A rule explains the output returned by an LLM by identifying conditions that consistently hold in practice whenever certain features are retained or omitted.
2. We design **efficient search algorithms** to mine rule-based explanations, exploiting intuitions of rule validity to prune the search space and enable early termination of the search.
3. We conduct an **experimental evaluation** of our explanation search algorithms, presenting case studies showing the value of rule-based explanations of RAG systems for question answering, and highlighting the efficiency improvements due to our pruning techniques.

The rest of the paper is structured as follows: Section 2 reviews related work; Section 3 introduces our rule formulation; Section 4 illustrates our rule mining algorithms; Section 5 presents an experimental evaluation; and Section 6 concludes with future research directions.

2 Related Work

In the XAI literature, many methods have been proposed to explain individual model predictions, collectively known as *local* explanations. Many local methods adopt an approach known as feature attribution, exemplified by methods such as `LIME` (Ribeiro et al. 2016) and `SHAP` (Lundberg and Lee 2017), which explain a model by scoring input features according to their relative influence on the model's output. Though intuitive, these explanations are not directly actionable, and have been shown to be potentially unreliable, misleading, and manipulable (Kindermans et al. 2019; Slack et al. 2020; Hooshyar and Yang 2024). In contrast, counterfactual explanations explain a model by perturbing the input features in a minimal way to change the original output (Wachter et al. 2017; Verma et al. 2024); e.g., a declined loan application would have been approved if the applicant's income were 20 percent higher. Counterfactual perturbations aim to produce actionable insights that facilitate recourse, but many valid counterfactuals may exist, potentially contradicting each other and overwhelming users. Some methods therefore seek to sample diverse counterfactual perturbations or prioritize those deemed more feasible (Mothilal et al. 2020; Tsirtsis and Gomez Rodriguez 2020).

At the cost of increased computational complexity, rules mitigate the above problems by summarizing patterns observed across a model's predictions (Ribeiro et al. 2018; Macha et al. 2022; Guidotti et al. 2019; Rudin and Shaposhnik 2023); e.g., "if employed = no, then loan application = rejected," regardless of the values of the other features. Recent works have established that rules essentially subsume counterfactuals, forming a duality, since both make assertions over a model's input features (Geng et al. 2022; Guidotti et al. 2024). Rule-based explanations have been defined as conjunctions of *predicates* over the features of an input, for which any matching input always leads to the model producing the same output (Rudin and Shaposhnik 2023). Several methods, such as `Anchors` (Ribeiro et al. 2018), have applied similar definitions for the purposes

of XAI. In contrast to our work, their rules are approximate (i.e., hold with high confidence), not exact, and do not provide an interface to instantiate their rules' input predicates for modern LLMs or RAG systems.

These rule formulations build implicitly upon *association rule* formulations established in the rule mining literature (Agrawal et al. 1993; Agrawal and Srikant 1994; Bayardo and Agrawal 1999). These formulations gave rise to many influential *rule mining* algorithms that search for frequent itemsets and valid association rules, most notably the Apriori algorithm (Agrawal and Srikant 1994). The rule mining methods we present in this work take algorithmic inspiration from Apriori, but have several key modifications for a fundamentally different problem: explaining AI model predictions. While Apriori iterates over observed *itemsets*, we instead iterate over the powerset of possible model inputs (features), eliminating any concept of *support* (see Definition 2) in rule validity. This enables us to seek sets that satisfy arbitrary predicates over model outputs, as opposed to those observed with some minimum *frequency*, as in the original Apriori algorithm. Our algorithms also leverage the concept of *monotonicity* (i.e., the Apriori property) to prune the search space, but we propose a new definition that operates on model output predicates rather than itemset support.

Naturally, many recent XAI works have focused on explaining LLMs, RAG, and other related phenomena, adapting general XAI methodologies as needed. *Global explanation* disciplines such as mechanistic interpretability attempt to identify low-level circuits that form among LLM neurons (Conmy et al. 2023; Gupta et al. 2024; Wang et al. 2024). However, these insights are intended primarily for scientific (i.e., expert) understanding. One intuitive explanation format for LLMs and RAG systems are *citations*, which can be generated at inference time using specialized training and fine-tuning methods (Khalifa et al. 2024; Aly et al. 2024) or post-hoc using various citation recommendation strategies (Färber and Jatowt 2020; Maheshwari et al. 2025; Li et al. 2024). However, these methods are subject to the critical limitation of hallucination (Agrawal et al. 2024), as their predictive approach to generating citations can be highly inaccurate, even with state-of-the-art models (Press et al. 2024; Cao and Wang 2024; Byun et al. 2024; Day 2023; Gao et al. 2023). Meanwhile, recent works have adapted feature attribution (Cohen-Wang et al. 2024) and counterfactual methods (Rorseth et al. 2024) to explain source importance in RAG, assigning importance weights to the retrieved sources or generating counterfactuals by removing some sources to change the LLM's output, respectively. In contrast to our rule-based approach, these recent methods are prone to the same limitations as their general methodologies, and may fail to produce concise and actionable insights that are robust and guaranteed.

3 Rule-Based Explanation Formulation

In this section, we introduce our novel rule-based explanation formulation. We start by reviewing the design space. We then provide a mathematical formalization of our rule-based explanations, clarifying how the hierarchical nature of our rules' validity leads to complementary variants that either *retain* or *omit* inputs. Finally, we explain how rule validity can be defined recursively, and interpreted using a lattice data structure.

3.1 Rule Design

In the data mining literature, a *rule* expresses a conditional "if-then" relationship where satisfying the "if" part (the *antecedent*) implies that the "then" part (the *consequent*) is also satisfied (Bayardo and Agrawal 1999). One intuitive adaptation for rules describing a prediction $y = M(x)$ might define the antecedent in terms of the input x to the model M, and the consequent in terms of the resulting output y. This would yield rules of the form "if x then y". We adopt this general orientation due to its clear alignment with the primary goal of local XAI: understanding the effect of model inputs on model outputs. This orientation has also been adopted in recent XAI works, wherein rules explain model predictions by uncovering significant patterns in said predictions. In this XAI setting, rules are better known as *rule-based explanations*.

Definition 1 (Rule-Based Explanation) A rule-based explanation is a conjunction of predicates over the features of some model M, where for any input x^* satisfying *all* predicates, the model *always* generates some target output y^* (Geng et al. 2022; Rudin and Shaposhnik 2023).

In the XAI setting, the antecedent is an assertion over a conjunction of input predicates, and the consequent is an assertion over a model output. In the literature, rules are often expressed and mined in terms of some set of *observed* instances, which would correspond to a model's *predictions* in our setting. Given a set of observed instances, a rule may be judged by the number of observed instances that match both sides of the rule, or by the proportion of instances whose consequent is satisfied when the antecedent is satisfied. These empirical measurements, which are often used as a proxy for rule quality, are known formally as the *support* and *confidence* of a rule, respectively.

Definition 2 (Rule Support) Given a set of observed instances, the support of a rule is the proportion of instances that satisfy the rule.

Definition 3 (Rule Confidence) Given a set of observed instances, the confidence of a rule is the conditional probability that its consequent is satisfied, measured over all instances in the set that satisfy its antecedent.

In settings that observe such instances, rules are considered more informative when they have higher support (i.e., the rule is more likely to implicate an instance) and higher confidence (i.e., the rule is more likely to hold). In this work, we do not seek rules that characterize observed data, and thus do not presume a setting wherein a set of observed instances is available. Instead, we seek rules that characterize the full model input space, and therefore operate over the set of all possible antecedents. It follows that the classical notion of support does not apply in our setting, though it is nonetheless equal to the reciprocal of the number of possible antecedents. Furthermore, we require that all valid rules hold with certainty (i.e., confidence of 100%), but note

that future work could build upon existing rule methods that guarantee minimum confidence levels (Ribeiro et al. 2018).

In practice, ambiguity still remains when applying this formulation, as different models, features, problems, and user preferences necessitate different predicates. We now propose a concrete implementation for the RAG setting.

Antecedent: For our rules, the choice of antecedent input predicate(s) is critical, as it will determine their explanatory effectiveness and the computational complexity of mining them. Ideally, for maximum explanatory power, the antecedent of a rule should implicate as many inputs as possible (i.e., wide coverage). For example, the antecedent "If the LLM is given a *set* of sources that *includes* 'Document 123'" covers more of the feature space than "If the LLM is given 'Document 123'". Likewise, the chosen antecedent predicate must also be compatible with and reasonable for the modality of the input features. While categorical features may be constrained to specific values (e.g., marital status is "single"), and numeric features to a range of values (e.g., credit score is ≥ 600), the ideal definition for the *textual* inputs of language models is less clear.

Drawing inspiration from XAI literature, we contend that *feature ablation* (i.e., removing text segments) is a suitable perturbation strategy for explaining language models, and thus formalize antecedent predicates based on feature *presence*. In the XAI literature, ablation is a well-founded strategy for inferring feature importance (Hameed et al. 2022), and has seen applications in works that explain language models (Rorseth et al. 2024; Cohen-Wang et al. 2024) and models generally (Lundberg and Lee 2017). We assume that the user partitions the input x into two components $x = (\mathbf{s}, \mathbf{c})$, where $\mathbf{s}$ represents some *feature subset* of interest $\mathbf{s} = \{s_1, s_2, \ldots, s_n\}$ derived from the input (e.g., RAG document sources). In our rule formulation, the antecedent will be defined in terms of the presence (or similarly, the absence) of inputs in $\mathbf{s}$, while the remaining inputs in $\mathbf{c}$ are ignored. By separating $\mathbf{s}$ and $\mathbf{c}$ (i.e., $y = M(\mathbf{s}', \mathbf{c})$), we enable the user to focus their rules and explanations on small, interesting subsets of the input feature space, while reducing the complexity of our rule mining algorithms.

Consequent: We contend that the ideal choice of consequent is ultimately dictated by the user and their motivation for seeking rules. Therefore, the consequent of our rules asserts only that the output meets some *target condition* defined explicitly by the user. In essence, the target condition is an arbitrary output predicate that could test for any identifiable property (e.g., presence of misinformation, validity of in-text citations). We expect the user to define an *output predicate function* $O : \mathcal{Y} \to \{0, 1\}$ that returns 1 when a model output $y \in \mathcal{Y}$ meets these conditions of interest. By parameterizing the consequent, we empower the user to seek rules that implicate not just the presence of a specific output (as is common in many XAI and rule-based works), but more sophisticated output characteristics (e.g., presence of hallucination, sentiment).

To define an output predicate function O, the user must effectively implement a classifier that inspects an arbitrary LLM output and judges whether it meets the conditions of interest. A simple predicate might compare the answer for equality against some *target* answer, returning 1 if the LLM answer is equal to the target answer (and 0 otherwise). A more specific predicate might analyze the LLM answer for *correctness*; however, this would only be possible in settings where the correct answer is known upfront (e.g., benchmarking). For nuanced semantic assertions (e.g., judging citation

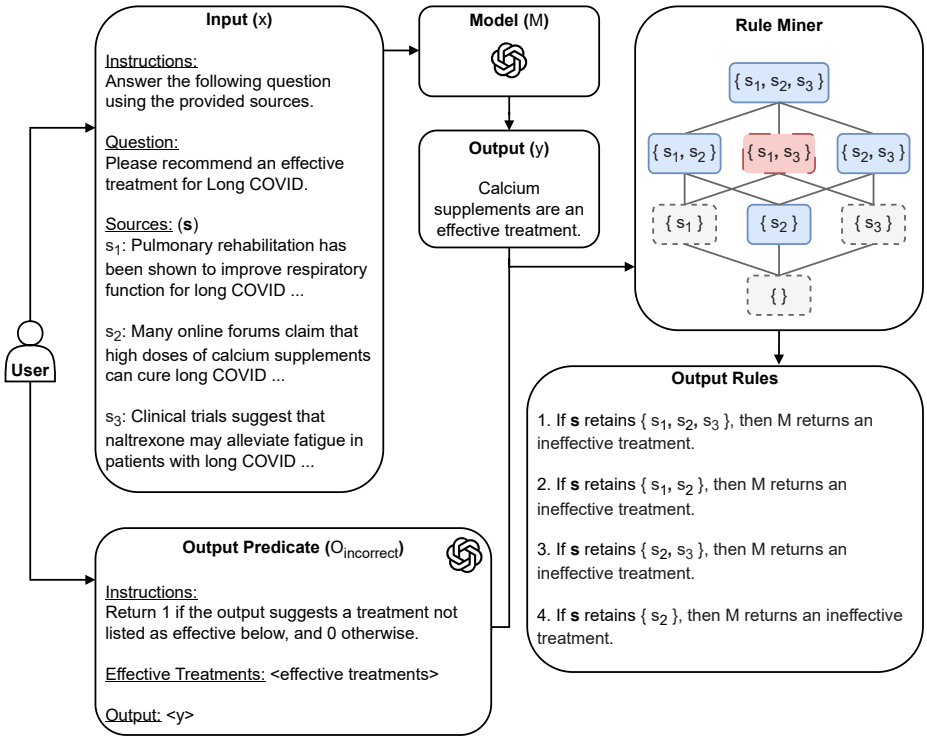


Fig. 1: This diagram illustrates an example where the user asks a RAG system to recommend an effective treatment for Long COVID, given three retrieved sources. The rules generated by our algorithm (formalized in Section 4) indicate that retaining source s_2 consistently leads the LLM to recommend treatments known to be ineffective.

validity), or those that must be robust to verbose or unstructured LLM outputs (e.g., spotting inappropriate content), a predicate function could adopt an "LLM-as-a-judge" implementation. Given detailed instructions and necessary context (e.g., ground truth information), an LLM can be a highly effective proxy for human judgment of LLM responses (Chiang and Lee 2023; Zheng et al. 2023); however, stochasticity introduced during the decoding process (e.g., token sampling) can compromise reproducibility.

Example: Consider the following running example, illustrated in Figure 1. Suppose a user wants to debug incorrect predictions made by an LLM that is being used to suggest treatments for medical illnesses. The LLM is configured to use RAG, meaning that the LLM's pre-trained knowledge is supplemented with external knowledge sources $\mathbf{s}$ (relevant medical documents) to help ground its answers. The user asks the LLM to recommend an effective treatment for Long COVID, and is investigating why an incorrect answer "calcium supplements" is consistently returned. Let x represent the corresponding prompt submitted to the LLM M by the user, and y represent its

incorrect response, as illustrated in Figure 1. All inputs in the prompt other than $\mathbf{s}$ (i.e., the question and LLM instructions) are captured in $\mathbf{c}$, and will not be part of any rules.

Suppose that the user wants to know whether any patterns emerge among the top-3 retrieved medical documents (sources) when ineffective treatments (i.e. incorrect answers) are observed. To make this intent explicit, the user defines a function $O_{incorrect}$ that returns 1 if y recommends a treatment not known to be effective for the indicated condition, and 0 otherwise. The user implements $O_{incorrect}$ by constructing a prompt to be posed to a separate LLM. It contains a list of treatments known to be clinically effective for Long COVID, and asks it to return 1 if the observed response suggests a treatment not in this list. This definition is illustrated in Figure 1, which shows how the input x and output predicate $O_{incorrect}$ (along with M and y) are subsequently provided to our rule mining algorithm. The algorithm proceeds by considering all possible combinations of the three medical documents, performing inference for certain combinations (as necessary), then finally deriving patterns across combinations that satisfied $O_{incorrect}$.

As shown in Figure 1, one rule found to be valid under $O_{incorrect}$ asserts that "if $\mathbf{s}$ retains all sources in $\{s_2\}$, then M returns an ineffective treatment." Other valid rules are found, but they are simply more-specific versions of this rule (i.e., implicating supersets of $\{s_2\}$). If the user were instead interested in patterns that lead to a correct answer, they could define a reciprocal predicate $O_{correct}$, potentially producing a symmetric rule such as "if $\mathbf{s}$ omits all sources in $\{s_2\}$, then M returns an effective treatment." In either case, the rule suggests that the presence of s_2 is important for the LLM to recommend an ineffective treatment.

3.2 Rule Formulation

The concepts of presence and absence have a reciprocal relationship. Suppose that we have a subset of the input set, $\mathbf{s}' \subseteq \mathbf{s}$, and that we aim to characterize the contents of $\mathbf{s}'$ w.r.t. $\mathbf{s}$. *Retaining* all elements from $\mathbf{s}$ that appear in $\mathbf{s}'$ is equivalent to *omitting* all elements from $\mathbf{s}$ that appear in the complement of $\mathbf{s}'$, $(\mathbf{s} \setminus \mathbf{s}')$. Conversely, omitting all elements from $\mathbf{s}$ that appear in $\mathbf{s}'$ is equivalent to retaining all elements in $(\mathbf{s} \setminus \mathbf{s}')$. This duality admits two variants of our rule-based explanations: *retention* and *omission* rules.

Definition 4 (Retention Rule) A retention rule for some output predicate O and implicated input subset $\mathbf{s}' \subseteq \mathbf{s}$ guarantees that $\forall \mathbf{t} \in \mathcal{P}(\mathbf{s}),\ \mathbf{s}' \subseteq \mathbf{t} \Rightarrow O(M(\mathbf{t}, \mathbf{c})) = 1$.

Definition 5 (Omission Rule) An omission rule for some output predicate O and implicated input subset $\mathbf{s}' \subseteq \mathbf{s}$ guarantees that $\forall \mathbf{t} \in \mathcal{P}(\mathbf{s}),\ (\mathbf{t} \cap \mathbf{s}' = \emptyset) \Rightarrow O(M(\mathbf{t}, \mathbf{c})) = 1$.

A retention rule identifies a subset of the input set that, when retained, always satisfies the output predicate. Similarly, an omission rules identifies a subset that, when omitted, always satisfies the output predicate. Inclusion and omission rules will often correlate with outputs that satisfy reciprocal (i.e., complementary) output predicates. Therefore, users will often use two separate, reciprocal output predicates

when explaining a single input (e.g., $O_{correct}$ and $O_{incorrect}$). Generally, both rule types identify critical inputs in $\mathbf{s}$ whose presence is necessary to ensure some condition holds.

Revisiting our RAG example from Figure 1, recall that the user can obtain rules of two symmetric types, either in terms of answer correctness or incorrectness. In this example, the retention rule formulation aligns with $O_{incorrect}$, and the omission rule formulation aligns with $O_{correct}$. That is, the aforementioned retention rule under $O_{incorrect}$ asserts that, across all combinations of sources in $\mathbf{s}$, the LLM's answer is always incorrect when the combination retains all sources in $\mathbf{s}' = \{s_2\}$. Likewise, the complementary omission rule under $O_{correct}$ would assert that the answer is correct anytime the combination omits $\mathbf{s}' = \{s_2\}$.

To guarantee these assertions for a subset $\mathbf{s}' \subseteq \mathbf{s}$, one must verify that the output predicate remains satisfied for all possible subsets of $\mathbf{s}$ wherein the inputs in $\mathbf{s}'$ are retained (or omitted, respectively). Following the sources $\mathbf{s}$ defined in Figure 1, where $|\mathbf{s}| = 3$, validating either $\{s_2\}$ rule would require evaluating the LLM output corresponding to each subset in $T = \{\{s_2\}, \{s_1, s_2\}, \{s_2, s_3\}, \{s_1, s_2, s_3\}\}$. To validate the $\{s_2\}$ retention rule against $O_{incorrect}$, one would need to ensure that $\forall \mathbf{t} \in T$, $O_{incorrect}(M(\mathbf{t}, \mathbf{c})) = 1$. The "Rule Miner" module in Figure 1 depicts these as blue nodes with a solid border, which indicates their validity. To validate the $\{s_2\}$ omission rule against the applicable predicate $O_{correct}$, one must ensure that $\forall \mathbf{t} \in T$, $O_{correct}(M(\mathbf{s} \setminus \mathbf{t}, \mathbf{c})) = 1$. In the next subsection, we formalize a definition of rule validity using recursion, which informs the design of our novel rule search algorithms.

3.3 Rule Validity

Under our definitions, the validity of a rule can be defined recursively. A retention rule implicating $\mathbf{s}'$ is valid iff the predicate O is satisfied for all supersets of $\mathbf{s}'$. Likewise, for an omission rule $\mathbf{s}'$, O must be satisfied for all subsets of its complement $(\mathbf{s} \setminus \mathbf{s}')$. Without loss of generality, the validity of a rule requires output predicate satisfaction for all *ancestors* of $\mathbf{s}'$, including $\mathbf{s}'$ itself. The set of ancestors can be defined recursively by repeatedly applying the *parent* relation, starting from $\mathbf{s}'$ and continuing until no further parents exist. Thus, valid rules have an intrinsically recursive relationship, as their validity depends on the total validity of all ancestors.

Definition 6 (Rule Validity) Let $P_{\mathbf{s}'}$ denote the set of parents of $\mathbf{s}'$, i.e., the immediate supersets of $\mathbf{s}'$ (for a retention rule) or the immediate subsets of $(\mathbf{s} \setminus \mathbf{s}')$ (for an omission rule). We can define rule validity as a recursive predicate $V(\mathbf{s}') = O(\mathbf{s}') \wedge \forall \mathbf{t} \in P_{\mathbf{s}'}, V(\mathbf{t})$.

By definition, all ancestors must be validated *before* validating a descendant. A rule mining algorithm must first validate those rules that retain or omit the *most* inputs (i.e., fully recurse) before unwinding the call stack to consider rules that retain or omit fewer inputs. Since our rules are expressed in terms of a *set* of features $\mathbf{s}'$, this relationship can be characterized as set *subsumption*. Ancestor rules are *subsumed* by descendant rules, and are therefore redundant to the end user.

Definition 7 (Rule Subsumption) A rule r_1 subsumes another rule r_2 iff every predicate in r_2 also appears in r_1.

Moreover, rules that retain or omit *fewer* inputs (i.e., descendants) are considered to have greater explanatory power, since by definition, their output predicate holds over more inputs (i.e., greater coverage). Rules that implicate smaller feature subsets $\mathbf{s}'$ are considered to be more *minimal*.

Definition 8 (Rule Minimality) A rule r_1 is minimal iff there exists no rule r_2 such that r_2 subsumes r_1 and $r_2 \neq r_1$.

In this work, we seek minimal rules, but operate under the understanding that less-minimal rules must be validated first. We acknowledge an inherent trade-off between rule minimality and algorithmic efficiency, as finding *more* minimal (higher coverage) rules ultimately requires validating *more* rules. Our algorithms (introduced in Section 4) ultimately prioritize minimality, exhausting all rule candidates and guaranteeing 100% confidence. However, to eliminate redundancy, our explanations discard any non-minimal rules before being returned to the user.

The full hierarchy of a given rule is a subset of the power set $\mathcal{P}(\mathbf{s})$. The power set, which contains *all* candidate rules that implicate $\mathbf{s}$, forms a data structure known as a *lattice* (Abiteboul et al. 1995), illustrated in Figure 2. Each of the $2^{|\mathbf{s}|}$ subsets in $\mathcal{P}(\mathbf{s})$ is a rule candidate; each node in the lattice is a candidate, and each edge delineates a subset / superset relationship between two candidates. The $\mathcal{P}(\mathbf{s})$ lattice constitutes the full search space for any rule mining algorithm under our formulation, and it contains $2^{|\mathbf{s}|}$ candidate rules. In the following subsections, we present two rule mining algorithms that efficiently navigate this lattice in search of *all* valid rules.

4 Rule Mining Algorithms

In this section, we introduce two search algorithms capable of mining our rule-based explanations, each with different advantages. First, we introduce the Mono Rule Miner, which can mine either retention or omission rules in a single pass. Second, we introduce the Dual Rule Miner, which can mine both rule types simultaneously in a single pass. Mono is more efficient in settings where a single rule type is sufficient, and can exploit greater search space pruning. For settings where patterns of both types are sought and compared, Dual is often more efficient than executing two Mono passes.

4.1 Mono Rule Miner

As discussed in Section 3.2, users will request rule-based explanations after obtaining a model output $y = M(\mathbf{s}, \mathbf{c})$. To define the desired rule, the user will specify whether they seek retention or omission rules, and will supply a definition for an output predicate O. The goal of our algorithms is to discover all valid rules that implicate subsets of $\mathbf{s}$. This necessitates a thorough (and potentially exhaustive) search through the lattice, and subsumes the simpler problem of validating a single candidate rule.

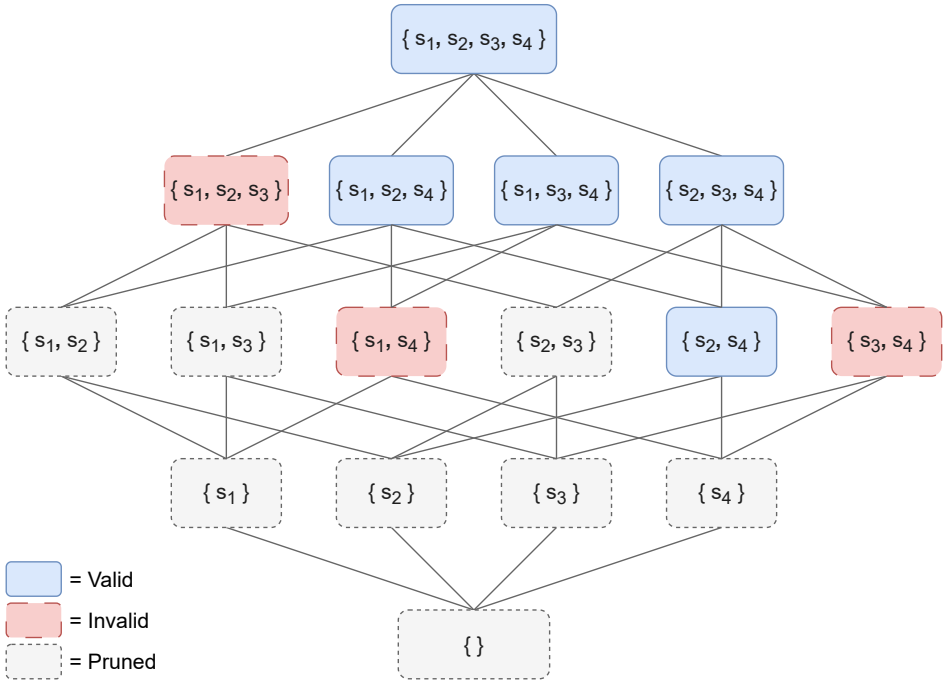


Fig. 2: A lattice representing an input set $\mathbf{s}$ where $|\mathbf{s}| = 4$. The node coloring illustrates an example run for the top-down breadth-first search performed by Mono Rule Miner (introduced in Section 4.1), which discovers valid rules (blue with solid border), invalid rules (red with long dashes), and candidates that can be pruned (grey with short dashes).

As exemplified in Figure 2, the lattice contains $|\mathbf{s}| + 1$ *levels* that group rule candidates by their size (i.e., $|\mathbf{s}'|$ for a rule based on $\mathbf{s}' \subseteq \mathbf{s}$). Since rule validity is defined recursively w.r.t. all ancestors, we must generally navigate the lattice top-down to validate rules. Due to the recursive definition, validating any single rule can be accomplished using a *depth-first search* (DFS), starting from the rule in question and recursing to all its ancestors in the lattice. Since we seek *all* valid rules, simply executing DFS for each candidate rule (top-down) would incur significant repeated computation. Recomputation can be avoided by caching rule validity during a top-down traversal, reducing runtime complexity to $\mathcal{O}(2^{|\mathbf{s}|})$, but caching would impose an expensive $\mathcal{O}(2^{|\mathbf{s}|})$ space complexity.

Instead, we propose a top-down *breadth-first search* (BFS) implementation that finds *all* valid rules of a specific type in a *single* pass. We call this our *Mono Rule Miner*. The algorithm is detailed in Algorithm 1, and its design is informed by two key observations. The first is that our recursive definition of rule validity admits a memory-efficient dynamic programming solution. The second observation is that the

recursive definition presents an opportunity for early termination when a rule is found to be invalid.

Dynamic Programming (Observation #1): To justify our dynamic programming solution, we observe that our recursive problem formulation is already a dynamic programming problem, since the validity of a rule is dependent on overlapping subproblems originating in its ancestor rules. Capitalizing on this, we can bound the number of cached subproblems (i.e., rule validity records) to a size that modestly improves over the naive $\mathcal{O}(2^{|\mathbf{s}|})$. Though the validity of a rule hinges also on the validity of all its ancestors, it is sufficient to hinge only on the validity of its immediate *parents*. This is true because the remaining ancestors must have already been validated when validating these parents. Therefore, we need only cache the validity of two levels worth of rules at any time (the current and previous level), resulting in a lower space complexity of $\mathcal{O}\binom{|\mathbf{s}|}{l^*}$ where $l^* = \lfloor |\mathbf{s}|/2 \rfloor$ (the index of the largest level).

Early Termination (Observation #2): Our second observation arises from our definition of rule validity. Since a rule's validity hinges on the validity of all ancestors, then if a rule candidate is found to be invalid, then *none* of its descendants can possibly be valid. This property is analogous to the Apriori property (Agrawal and Srikant 1994) from the data mining literature, and is key to "pruning" large subtrees of the lattice search space. Upon encountering an invalid rule, we can cache a record of its invalidity and avoid evaluating (i.e., performing model inference for) the child rules if and when we reach them during traversal. Furthermore, once a level has no valid rules, the entire search can terminate as all valid rules must have been found.

Example: To illustrate how the the Mono Rule Miner works, we will now explain how Algorithm 1 navigates the lattice in Figure 2 to find all valid rules of a specific type. Once provided with inputs $\mathbf{s}, \mathbf{c}, O$ and M, the algorithm will initialize caches for valid and invalid rule candidates (nodes corresponding to subsets of $\mathbf{s}$), then proceed with a top-down level-order traversal (line 3). Supposing that $|\mathbf{s}| = 4$ as illustrated in Figure 2, the traversal begins by gathering (line 5) and looping through (line 8) subsets of size 4 (i.e., any $\mathbf{s}' \subseteq \mathbf{s}$ where $|\mathbf{s}'| = 4$, meaning only $\mathbf{s}$). Inside this loop, inference is performed on each of these subsets (line 9), and the predicate function O is invoked with the output to determine rule validity (line 10). For this first subset $\{s_1, s_2, s_3, s_4\}$, the output is found to satisfy O (as indicated in Figure 2) and is added to the valid candidates cache.

The algorithm then proceeds to the next level containing subsets of size three. Since $\mathbf{s}$ is a valid rule and is the sole parent of all subsets on this level, then each subset *may* be a rule, and will be if it satisfies O. As reflected in Figure 2, all subsets are found to be valid rules except for $\{s_1, s_2, s_3\}$, whose descendants can now logically be ruled as invalid. When the algorithm discovers that this subset is invalid, only the subset itself is added to the invalid candidates cache (line 13), as descendants will be pruned dynamically upon navigating to subsequent levels. Proceeding to the next level with subsets of size 2, the invalid candidates cache (containing only $\{s_1, s_2, s_3\}$) is emptied and repopulated (lines 6 and 7) with the *children* of all members ($\{s_1, s_2\}$, $\{s_1, s_3\}$, $\{s_2, s_3\}$), effectively pruning nodes from the subsequent evaluation loop.

Analysis: In the worst case, when *all* $2^{|\mathbf{s}|}$ rule candidates are valid, the algorithm must make $2^{|\mathbf{s}|}$ inference calls and use $\mathcal{O}(\binom{|\mathbf{s}|}{l^*})$ space. However, in the opposite case,

Algorithm 1 Mono Rule Miner

Input: The input set $\mathbf{s}$
Input: The remaining inputs $\mathbf{c}$
Input: The output predicate O
Input: The model M
Output: The rule-based explanations

$R \leftarrow \{\}$ ▷ Initialize valid rule set
$Z \leftarrow \{\}$ ▷ Initialize invalid node set
for $l' \in [0, |\mathbf{s}|]$ **do**
  $l \leftarrow |\mathbf{s}| - l'$ ▷ Compute current level size
  $S \leftarrow \{\mathbf{s}' \subseteq \mathbf{s} \mid |\mathbf{s}'| = l\}$ ▷ Get all current level nodes
  $Z \leftarrow \{\mathbf{z} \subset \mathbf{z}' \mid \mathbf{z}' \in Z \wedge |\mathbf{z}| = l\}$ ▷ Get children of invalidated parents
  $Q \leftarrow S \setminus Z$ ▷ Prune invalid nodes in current level
  for $\mathbf{s}' \in Q$ **do**
    $y \leftarrow M(\mathbf{s}', \mathbf{c})$ ▷ Invoke the model
    if $O(y) = 1$ **then**
      $R \leftarrow R \cup \{\mathbf{s}'\}$ ▷ Record valid rule
    else
      $Z \leftarrow Z \cup \{\mathbf{s}'\}$ ▷ Record invalid node
    end if
  end for
end for
return R

when no valid rules exist, our real-time pruning short-circuits the algorithm, requiring only a single inference call. Otherwise, the complexity varies depending on the number of valid rules, their location in the lattice, and many other indirect factors. These factors are analyzed empirically in Section 5.

4.2 Dual Rule Miner

As the name implies, the limitation of the Mono Rule Miner is that only rules of a single type are mined in its sole pass. In settings where comprehensive tests are demanded (e.g., LLM red teaming), obtaining rules for a pair of reciprocal rules types enables deeper understanding of the RAG sources and their influence. With rules of two reciprocal types, a user can compare whether a set of sources $\mathbf{s}'$ was sufficient, both when retained *and* omitted, to induce an LLM output that meets a single target condition. If $\mathbf{s}'$ is implicated in both types of rules, then users can be more confident in its absolute importance. However if $\mathbf{s}'$ is implicated in only one rule type, users can infer that its influence is relative to the presence (or absence) of the other sources.

To find rules of both types using Mono, each would require a separate pass, and may utilize different output predicates. Despite potentially different predicates, the underlying BFS still navigates the same lattice, in the same order, for the same input set $\mathbf{s}$, regardless of the chosen type. Algorithmically, the key difference between the

two types stems from the *interpretation* of the "current subset;" i.e., how it will be applied. For the current subset $\mathbf{s}' \in \mathbf{s}$, a retention rule must *retain* $\mathbf{s}'$, but an omission rule must *omit* $\mathbf{s}'$.

Guided by two additional observations (to follow), we devise a strategy to navigate the lattice in search of both types of rules, simply by interpreting each subset in opposite ways. We refer to the resulting algorithm as the *Dual Rule Miner*. It proceeds with the same top-down BFS lattice navigation as the Mono Rule Miner, finding rules of both types; however, some efficiency is sacrificed. Specifically, the benefits of our early termination (to reduce runtime complexity) and dynamic programming caching (to reduce space complexity) introduced in Section 4.1 are diminished.

Mask Representation (Observation #1): Recall from Section 3.2 that presence and absence have a reciprocal relationship. Omitting $\mathbf{s}'$ is equivalent to retaining its complement w.r.t. $\mathcal{P}(\mathbf{s})$, $(\mathbf{s} \setminus \mathbf{s}')$. While we can adopt this conceptually, we require an efficient representation to enable *dual* interpretation in a *single* algorithm pass. To solve this problem, we encode each lattice node (subset) as a *mask*, where each mask element reflects one of two possible states (`True` or `False`) for the input $s \in \mathbf{s}'$ it represents. Masks permit a single canonical representation for each lattice node while simultaneously permitting two interpretations: one where `True` indicates *retention* of the corresponding input (and omission when `False`), and one where `True` indicates *omission* (and retention when `False`).

Pruning Adjustments (Observation #2): Although our Mono Rule Miner pruning strategy can still be used to prune invalid rules, we can only prune rules of the same type. Therefore, even if no further rules of one type can be found, lattice navigation will continue if rules of the other type may still be found. In short, Dual pruning opportunities can only be less common (and thus less effective overall) than Mono, since a subset must be invalid under *both* rule types before descendants can be pruned from the search. Moreover, once navigation reaches the bottom half of the lattice, the concrete subset for a given rule may have already been posed to the model M when interpreted under the opposite type. In effect, this results in the $\mathcal{O}(2^{|\mathbf{s}|})$ time complexity being multiplied by a constant factor of 2.

Example: The implementation of Dual Rule Miner is given by Algorithm 2. The algorithm requires the same inputs as Mono Rule Miner, except that two separate predicates O_{ret} and O_{omi} are expected for retention and omission rule types, respectively. Likewise, separate valid and invalid rule candidate caches are now maintained for each rule type; however, the familiar level-order traversal proceeds in a similar fashion (line 5). Each subset of the current level size is evaluated under both interpretations separately (lines 13 and 18), each applying their corresponding mask and being judged by their corresponding output predicate.

The main point where Dual diverges from Mono is the pruning step; i.e., just before looping through a level's subsets. Since a subset must be invalidated under *both* types before it can be pruned, Dual must reinitialize the respective invalid candidate caches separately (lines 8 and 9), then prune only those subsets that do not appear in *either* (lines 10 and 11). For example, suppose that all subsets of size 4 and 3 have the *same* validity as indicated in Figure 2 regardless of type (i.e., $\{s_1, s_2, s_3\}$ is invalid for retention and omission). As the search continues to subsets of size 2, Dual prunes

Algorithm 2 Dual Rule Miner

Input: The input set $\mathbf{s}$
Input: The remaining inputs $\mathbf{c}$
Input: The retention output predicate O_{ret}
Input: The omission output predicate O_{omi}
Input: The model M
Output: The rule-based explanations

1: $R_{ret} \leftarrow \{\}$ ▷ Initialize valid retention rules
2: $R_{omi} \leftarrow \{\}$ ▷ Initialize valid omission rules
3: $Z_{ret} \leftarrow \{\}$ ▷ Initialize invalid retention nodes
4: $Z_{omi} \leftarrow \{\}$ ▷ Initialize invalid omission nodes
5: **for** $l' \in [0, |\mathbf{s}|]$ **do**
6: $l \leftarrow |\mathbf{s}| - l'$ ▷ Compute current level size
7: $S \leftarrow \{\mathbf{s}' \subseteq \mathbf{s} \mid |\mathbf{s}'| = l\}$ ▷ Get all current level nodes
8: $Z_{ret} \leftarrow \{\mathbf{z} \subset \mathbf{z}' \mid \mathbf{z}' \in Z_{ret} \wedge |\mathbf{z}| = l\}$ ▷ Get children of invalidated parents
9: $Z_{omi} \leftarrow \{\mathbf{z} \subset \mathbf{z}' \mid \mathbf{z}' \in Z_{omi} \wedge |\mathbf{z}| = l\}$
10: $Q' \leftarrow \{(\mathbf{s}', [\mathbf{s}' \notin Z_{ret}], [\mathbf{s}' \notin Z_{omi}]) \mid \mathbf{s}' \in S\}$ ▷ Pair nodes with their validity
11: $Q \leftarrow Q' \setminus \{(\mathbf{s}', v_{ret}, v_{omi}) \in Q' \mid \neg v_{ret} \wedge \neg v_{omi}\}$ ▷ Prune totally invalid nodes
12: **for** $(\mathbf{s}', v_{ret}, v_{omi}) \in Q$ **do**
13: **if** $v_{ret} \wedge O_{ret}(M(\mathbf{s}', \mathbf{c})) = 1$ **then** ▷ Check for valid retention rule
14: $R_{ret} \leftarrow R_{ret} \cup \{\mathbf{s}'\}$
15: **else**
16: $Z_{ret} \leftarrow Z_{ret} \cup \{\mathbf{s}'\}$
17: **end if**
18: **if** $v_{omi} \wedge O_{omi}(M(\mathbf{s} \setminus \mathbf{s}', \mathbf{c})) = 1$ **then** ▷ Check for valid omission rule
19: $R_{omi} \leftarrow R_{omi} \cup \{\mathbf{s}'\}$
20: **else**
21: $Z_{omi} \leftarrow Z_{omi} \cup \{\mathbf{s}'\}$
22: **end if**
23: **end for**
24: **end for**
25: **return** (R_{ret}, R_{omi})

the same children as described in Section 4.1 for Mono, since these child subsets are invalid under *both* rule types.

Diverging from what is illustrated in Figure 2, suppose that $\{s_1, s_4\}$ is found to be a valid retention rule, but an invalid omission rule. Consequently, each descendant of this subset *may* be a valid retention rule, but can never be a valid omission rule. The algorithm inserts $\{s_1, s_4\}$ into the invalid omission candidates cache (line 21), resulting in the inclusion of its children $\{s_1\}$ and $\{s_4\}$ when reinitialized (line 9). Since this subset and its children were not included in *both* invalid caches, the evaluation loop will still "evaluate" these children. However, per-type validity is still tracked and associated with each evaluated subset (line 11), allowing the algorithm to "short circuit" and avoid unnecessary inference calls.

As with Mono Rule Miner, descendants of all invalid rule candidates will *continue* to propagate through the continuous reinitialization of their respective invalid candidate caches. As a result, once a descendant of a subset invalidated for one type is invalidated for the other type, all descendants of this now "totally invalidated" descendant will be pruned. Continuing our example, suppose that $\{s_1\}$ (already cached as an invalid omission rule via propagation) is found to be an invalid retention rule (via inference and judgment by O_{ret}). This subset will be added to the invalid retention rule candidates (line 16), and will therefore be present in both invalid caches, meaning that all descendants (only $\{\}$ in this case) will be pruned by the algorithm.

Analysis: The $\mathcal{O}(\binom{|\mathbf{s}|}{l^*})$ space complexity of Mono Rule Miner (established in Section 4.1) remains largely intact here, but a constant multiple of 2 is again introduced to account for each rule type requiring its own rule validity cache. In case the user wishes to avoid the aforementioned repeated inference calls, Dual Rule Miner can cache model responses, indexing them by their canonical mask. Caching responses will raise the space complexity to $\mathcal{O}(2^{|\mathbf{s}|})$. We allow the user to decide whether they would prefer repeated inference calls or higher memory use. In Section 5.2.3, we empirically study the proportion of overlapping inference calls in a real RAG setting, and analyze the effectiveness of Dual's caching in different conditions.

5 Experimental Evaluation

In this section, we conduct an experimental evaluation to analyze the quality of our rule mining algorithms. We begin with a qualitative evaluation that describes a RAG case study in the domain of healthcare. Then, we present a quantitative evaluation that studies our rule miners applied to a real-world RAG QA problem. We analyze the efficiency improvements gained by using our algorithms. Finally, in an effort to contrast these findings with theoretical boundaries, we then conduct a similar evaluation with synthetic data.

5.1 RAG Case Study

We now present a case study that considers a real world scenario in the domain of healthcare. We document several XAI challenges that arise when using RAG-enabled LLMs in this setting, and analyze how our rule-based explanations help to rectify them. Our case study follows a technical user who is debugging a medical research system used by healthcare professionals (e.g., doctors, nurses) for the purposes of real-time research. One component of this system is a chatbot that enables end users to pose domain questions and obtain answers that are grounded in medical documents. This chatbot component is implemented using a retrieval-augmented LLM connected to a medical database.

The user is investigating a concerning trend that several doctors have identified: when posed with questions related to Long COVID, the chatbot often returns a misinformed answer that downplays the efficacy of pulmonary rehabilitation, despite clinical consensus of its benefits. To dig deeper, the technical user wishes to understand how the LLM's answer changes when seeded with different documents provided via RAG. Specifically, they wonder whether certain "bad" documents lead the LLM to

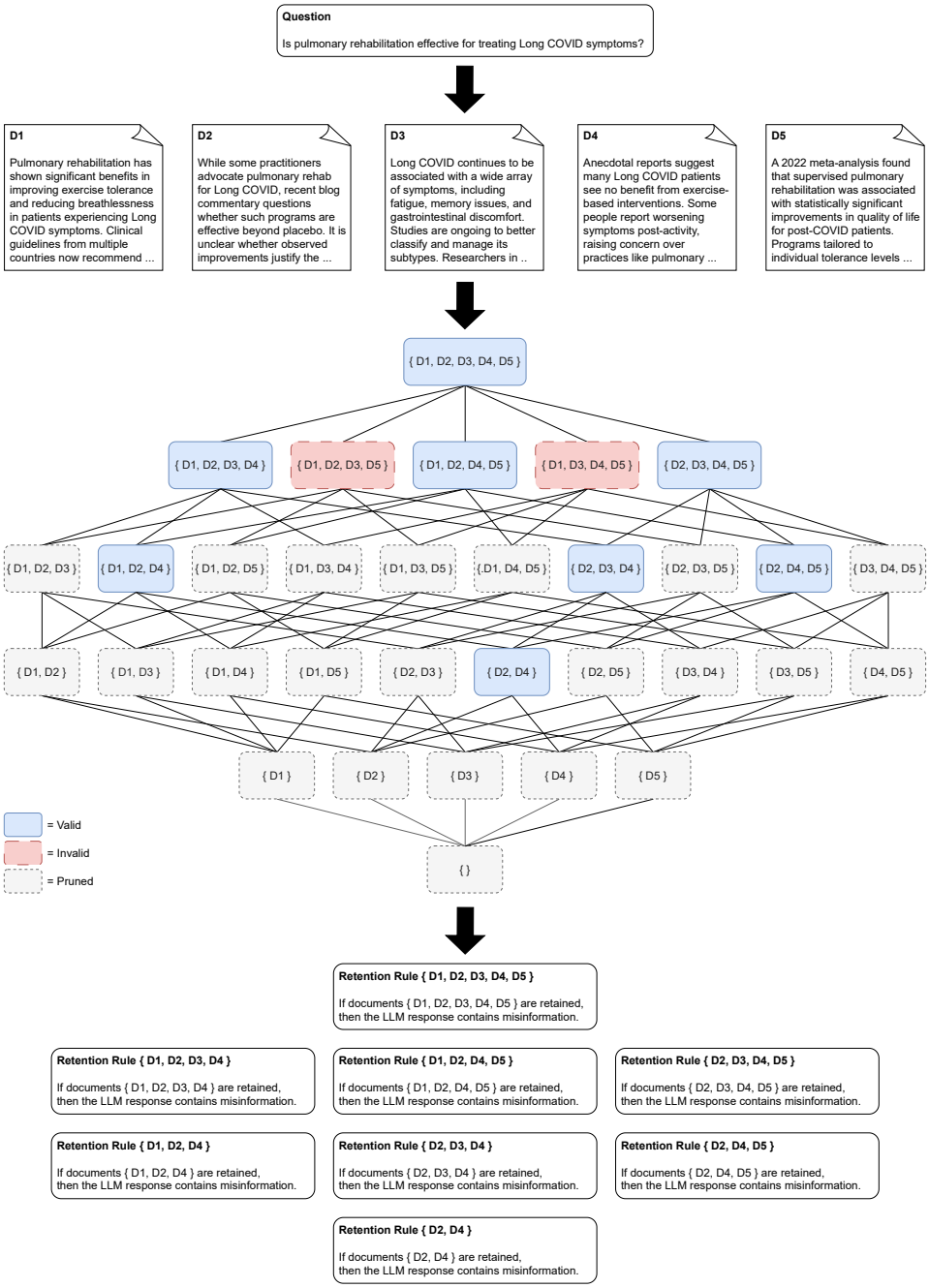


Fig. 3: A lattice corresponding to the case study presented in Section 5.1. Here, the minimal valid rule (lowest valid rule in the lattice, as per Definition 8) states that "if D2 and D4 are retained, then the LLM produces a response that contains misinformation."

produce this misinformed answer, and therefore seek rules that link specific documents to this line of misinformation.

Suppose that the user is using a commercial LLM augmented with a prompt-based RAG pipeline, which works as follows. Upon receiving a question from the end user, the pipeline first prompts the LLM to generate one or more keyword queries that can be posed to a retrieval model to obtain documents that may prove helpful in answering the question. Then, for each generated keyword query, a proprietary retriever capable of scoring document relevance obtains the top-10 documents from the medical database (i.e., top-10 deemed most relevant to the query). The system establishes a final set of document sources $\{D1, D2, D3, D4, D5\}$ by sorting all retrieved documents by their relevance scores and keeping only the 5 highest scoring. These documents are inserted into a prompt that instructs the LLM to use them to answer the end user's question.

The user decides to debug a specific question, "Is pulmonary rehabilitation effective for treating Long COVID symptoms?," focusing first on the documents returned by the system. Upon review, the user confirms that the retriever is functioning correctly, and that the retrieved documents are relevant to the query. The user suspects that, despite their perceived relevance to the question, one or more of the retrieved documents may contain misinformation, which may be repeated by the LLM in turn. Therefore, the user seeks rules that will expose problematic documents and link them to misinformed answers.

To generate the rules, the user defines an output predicate that uses a different LLM to judge whether the response contains known medical misinformation. The user employs Mono Rule Miner to find all valid retention rules. As illustrated in Figure 3, several rules are returned, all originating from a single minimal rule that states the following: whenever documents $\{D2, D4\}$ are retained, the LLM response will always contain misinformation. This suggests that, although neither D2 nor D4 are sufficient individually to induce a misinformed answer (i.e., there is no valid rule for $\{D2\}$ or $\{D4\}$), their *combined* presence can convince the LLM of its legitimacy.

With these new insights, the technical user can now flag these two documents for further review by a medical expert, and temporarily remove it from the database. After the expert revises the document to remove the problematic content, the same rule-based explanations can aid in validating a successful redeployment. By running the rule miner again, the technical user should find that no rules implicating misinformed responses exist, ensuring that this misinformation is less likely to propagate to end users.

5.2 Quantitative Evaluation: LLM Question Answering

5.2.1 Experimental Setup

To test our rule miners in a modern setting, we design an experiment that tasks a RAG system with question answering (QA), seeking rules to explain its answer to specific questions. We focus specifically on a scenario where a factual question is posed to an LLM, alongside a set of knowledge sources for reference (i.e., to help answer the question), and is prompted to return the correct answer. In an attempt to rule out the influence of pre-trained LLM knowledge, our prompt instructs the

LLM to answer the question using only the information from these sources. For this evaluation, we use OpenAI's `gpt-4o-mini-2024-07-18` LLM as our model M. To realize this experimental setting, we utilize the `HotpotQA` dataset (Yang et al. 2018), a QA benchmark for LLMs and other QA systems.

`HotpotQA` tests a model's ability to answer *multi-hop* questions, which require consideration of multiple independent pieces of information ("hops") to answer. For example, a multi-hop question might ask, "What is the chemical element named after the physicist who developed the theory of relativity?" Logically, answering this question requires one to first learn that Albert Einstein primarily developed the theory of relativity, then learn that the element named after him is Einsteinium. Each HotpotQA question is labeled with a single time-invariant correct answer, a set of "context" sources (i.e., potentially relevant documents), and a list of "supporting facts." The supporting facts are pointers to specific sentences in the context sources, and collectively represent a minimal set of facts (sentences) necessary to derive the correct answer. Naturally, each question has at least two facts.

As formalized in Section 3.2, our rule miners require a definition for the input set $\mathbf{s}$ and the output predicate(s) O. To decide on a formulation, we must motivate our search for rules. We adopt an intuitive problem definition wherein a user wishes to understand how the context sources provided to the LLM (via `HotpotQA`) affect the correctness of the LLM's answer.

Therefore, when posed with a specific `HotpotQA` question, we define $\mathbf{s}$ as the set of sources that we provide to the LLM. The question and other input components are defined under $\mathbf{c}$, since they are not the focus of our present formulation. To maintain consistent granularity across context sources and supporting facts, our definition of $\mathbf{s}$ treats each *sentence* from each context source as a unique source. When these sentences (sources) are inserted into a RAG prompt, we ensure that they appear in their original order (i.e., we partially reconstruct the context "document" sources from the sentences present). We provide different numbers of sources throughout the experiment, typically ensuring that some number of supporting sources are included.

Meanwhile, we define two complementary output predicates, $O_{correct}$ and $O_{incorrect}$, as described in Section 3.1. The predicate function first attempts to judge a predicted answer's correctness via simple regular expression matching against the ground truth answer. If there is no match, it asks an LLM (also `gpt-4o-mini-2024-07-18`) to judge correctness. Under this formulation, our algorithms will mine the following two types of rules:

1. **Retention rules:** Identify source subsets that, when *retained* within a RAG prompt, always induce a *correct* answer from the LLM.
2. **Omission rules:** Identify source subsets that, when *omitted* from a RAG prompt, always induce an *incorrect* answer from the LLM.

The goal of this quantitative evaluation is to measure the average improvement of our rule miners, which leverage novel pruning and early termination strategies, as compared to a naive method that does not. To quantify the degree of pruning, it suffices to measure the average proportion of the lattice nodes that an algorithm actually visits. In this setting, visiting a node implies executing a single LLM inference

call. Expressing this as a proportion allows us to average and consolidate results over lattices of varying sizes.

We hypothesize that the effectiveness of our pruning depends on the lattice size. In smaller lattices, there are fewer nodes to prune, so the amount of nodes visited is similar to that of a naive algorithm. However in large lattices, each node pruned invalidates a larger subtree in the lattice, leading to greater overall pruning. Intuitively, more nodes will exist in a larger lattice, meaning that more nodes (by count) could be pruned, but does the *proportion* of nodes eligible for pruning also increase? This is the hypothesis that we test in this experiment.

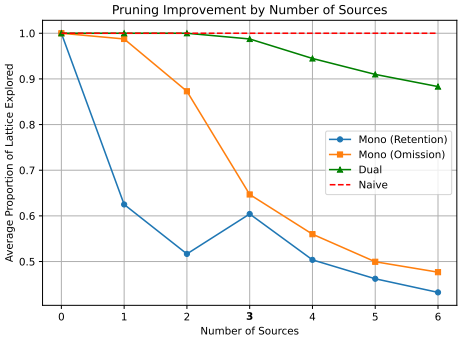
To test this hypothesis, we proceed as follows. For each mining algorithm under test (Mono and Dual), we measure the proportion of nodes pruned over increasing numbers of sources in $\mathbf{s}$ (i.e., increasing lattice sizes), averaged over 120 `HotpotQA` examples (questions). One potential confounder is the true number of supporting facts required to piece together the correct answer. We hypothesize also that this number affects the number of valid rules in our RAG setting, and that in turn, it affects the number of nodes that can be pruned. To verify this hypothesis, we pose only those `HotpotQA` questions that have exactly three supporting facts, then test input source sets $\mathbf{s}$ where $0 \leq |\mathbf{s}| \leq 6$. When $|\mathbf{s}| \leq 3$, we ensure that $\mathbf{s}$ consists only of supporting facts. Thus, our results will illustrate the pruning improvements when the LLM has too few supporting sources, all supporting sources, and extra sources in addition to all supporting.

5.2.2 Experimental Results: Pruning Improvements

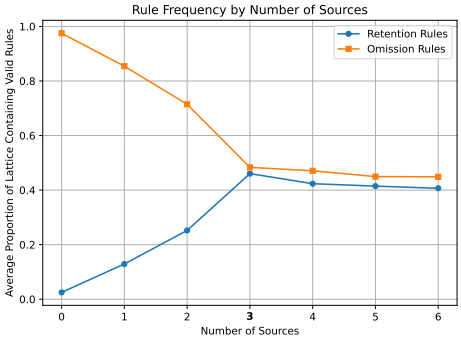
As illustrated in Figure 4a, the results support our hypothesis. For both Mono and Dual rule miners, the degree of pruning (i.e., the inverse of "Average Proportion of Lattice Explored") is positively correlated with increasing lattice size ("Number of Sources"). As hypothesized, smaller lattices (i.e., few supporting sources) offer less opportunity for pruning; however, Mono prunes at an increasing rate as more supporting sources are included. We observe that Mono prunes more retention rule candidates than omission candidates for small lattice sizes; however, both progressively converge at the point where $\mathbf{s}$ includes all three supporting sources. After this point (when more sources are added), the proportion of pruning continues to increase at a slowing rate.

This trend is consistent with and supported by the patterns in the adjoining Figure 4b. As more of the supporting sources are considered by the LLM, the number of omission rules decreases rapidly, but the number of retention rules increases rapidly. Once all supporting sources are included, then even when additional non-supporting sources are added, roughly half the nodes in the lattice are valid retention rules, and half the nodes are valid omission rules (these are not necessarily mutually exclusive). In short, Figure 4b indicates that there are generally fewer retention rules than omission rules, and Figure 4a illustrates how our algorithm therefore prunes more retention candidates.

Dual's pruning proportion also increases as more supporting sources are considered, but until all three sources are included, pruning opportunities are negligible. This suggests that, so long as some supporting sources are not yet considered by the LLM, nearly all source subsets are promising candidates for at least one rule type. After three sources are added, Dual's pruning proportion continues to increase at a linear rate.



(a) The average degree of pruning achieved by different rule miners, as measured by the average proportion of lattice nodes evaluated. These measurements are plotted over increasing lattice sizes, showing that our Mono and Dual Rule Miners prune greater proportions of a lattice as its size increases.



(b) The average frequency of each rule type, measured as an average proportion of the lattice. These measurements are also plotted over increasing lattice sizes, showing that once all supporting sources are given to the LLM, valid retention and omission rules emerge in nearly half of the lattice nodes.

Fig. 4: As more `HotpotQA` supporting sources are provided to a RAG system, our rule miners can prune larger portions of the lattice.

5.2.3 Experimental Results: Mono vs. Dual

A truly naive Dual algorithm would make two naive Mono passes (one for each rule type), but since Figure 4a measures only the proportion of lattice nodes visited (i.e., *at least once*), it only expresses that Dual usually visits most nodes at least *once*. For this reason, we seek to compare the number of source subsets evaluated by a single Dual run to that of two Mono runs (one for each rule type). Figure 5 illustrates the results of this comparison, which are derived from the same `HotpotQA` RAG experiment. The results are quantified in terms of the proportion of evaluated subsets across *two* lattices, as Mono could evaluate each subset twice (once for each rule type). With caching enabled, our Dual algorithm will avoid evaluating any subset more than once, thereby preventing unnecessary LLM inference.

Figure 5 highlights how Dual evaluates far fewer total subsets than two Mono runs. For fewer sources, running Mono twice results in most subsets being needlessly evaluated twice. As more supporting sources are considered by the LLM, the total number of subsets evaluated over both Mono runs drops at a sublinear rate, which follows from the rule count trends in Figure 4b. Meanwhile, the number of subsets evaluated by Dual decreases at a very slow rate, with nearly all subsets being evaluated until all three supporting sources are considered. Upon reaching six sources, the number of subsets evaluated under these separate strategies converges, suggesting that the degree of overlap in subsets implicated by opposing rule types generally decreases as more sources are considered.

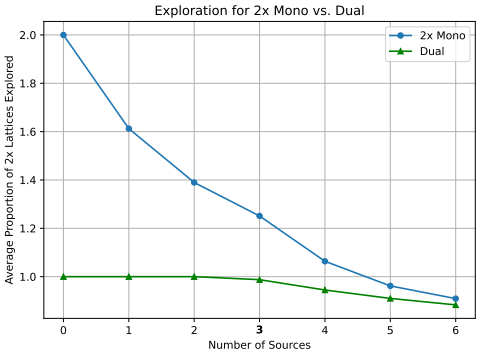


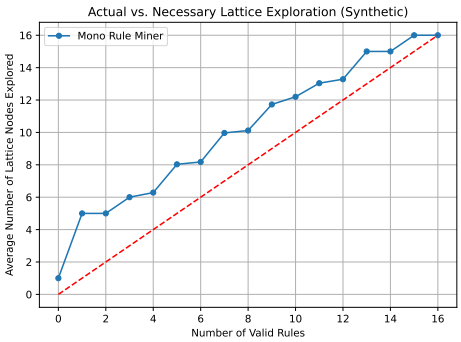
Fig. 5: The average proportion of input subsets evaluated (as measured over *two* lattices) by Dual vs. two Mono runs (one for each rule type). Two Mono runs results in many overlapping evaluations (identical LLM inference calls); however, Dual's caching mechanism allows it to avoid evaluating any subset more than once.

5.3 Quantitative Evaluation: Synthetic Data

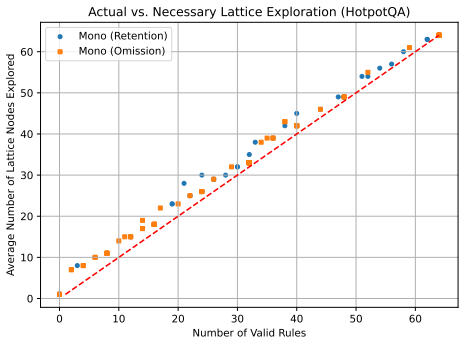
Finally, we repeat a similar experiment on a synthetic dataset. By doing so, we control for model nondeterminism and illustrate the limits of our algorithms' pruning. In this experiment, we study a lattice corresponding to four elements (i.e., $|\mathbf{s}| = 4$). We iterate every possible assignment of nodes' validity, computing the number of valid rules that exist in each. For each of these validity assignments, we measure the number of lattice nodes that our Mono Rule Miner explores to find them. In short, we measure the degree of lattice exploration necessary to find every possible rule in every possible lattice of size 2^4.

We first group these raw measurements by the number of valid rules, then average the number of visited nodes in each group. We plot the results in Figure 6a. The red dashed line delineates the number of valid rules in the lattice; in order to find every rule, our algorithms cannot visit fewer nodes than this. The results indicate that, on average, our Mono rule miner terminates relatively soon after finding all valid rules. For the sizes tested, Mono visits only one or two additional nodes in most cases. We contend that this is reasonable in practice, given that a search needs to encounter some invalid rules in order to realize it can terminate early.

In Figure 6b, we visualize similar statistics from the telemetry of our `HotpotQA` experiment. We focus only on LLM predictions that were made with six sources (i.e., $|\mathbf{s}| = 6$), wherein there may exist anywhere from zero to 2^6 valid rules of each type. Since there are relatively few data points, we present these data as a scatter plot. The plot shows that more valid omission rules are present than retention rules, consistent with Figure 4b. More importantly, Figure 6b, supports the conclusions evident from the synthetic data in Figure 6a.



(a) Efficiency of Mono Rule Miner in a synthetic setting, seeking rules for each possible validity configuration (lattice) for an input set of size 4 (as described in Section 5.3).



(b) Efficiency of Mono Rule Miner in the `HotpotQA` setting, as measured over examples where six sources were provided to the LLM (as described in Section 5.2).

Fig. 6: In both synthetic and real settings, the actual proportion of lattice nodes evaluated by our rule miners is very close to the theoretical minimum necessary to find all valid rules.

6 Conclusions

We proposed the first formulation of rule-based explanations for retrieval-augmented LLM systems. We presented two complementary variants that leverage the concept of feature ablation to link certain input features to the satisfaction of arbitrary output predicates. We implemented efficient algorithms for mining these rules, which make use of pruning and early termination. By way of quantitative and qualitative evaluations, we demonstrated the efficiency of our algorithms and the utility of our explanations, as applied in real-world RAG scenarios.

Our novel formulations and algorithms invite promising directions for future work. For instance, a relaxed version of our formulation could model rules that hold *approximately* (i.e., with some minimum confidence) rather than *exactly* (i.e., with certainty), which could help model the nuances of LLM non-determinism. Furthermore, with our rule mining algorithms, `HotpotQA` and similar benchmarks present an opportunity for novel benchmarking studies. Since expected rules can be derived from `HotpotQA`'s ground truth answers and supporting sources, a study could empirically compare LLMs on the basis of whether rules expected to hold actually do. Similarly, this ground truth data could be used to assess how often an LLM hallucinates answers not present in retrieved sources, or how often it deviates from RAG prompt instructions.

References

Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley Publishing Company, Reading, MA (1995)

Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. SIGMOD Rec. **22**(2), 207–216 (1993) https://doi.org/10.1145/170036.170072

Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: PVLDB. VLDB '94, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1994)

Agrawal, A., Suzgun, M., Mackey, L., Kalai, A.: Do language models know when they're hallucinating references? In: Graham, Y., Purver, M. (eds.) Findings of the Association for Computational Linguistics: EACL 2024, pp. 912–928. Association for Computational Linguistics, St. Julian's, Malta (2024)

Aly, R., Tang, Z., Tan, S., Karypis, G.: Learning to generate answers with citations via factual consistency models. In: Ku, L.-W., Martins, A., Srikumar, V. (eds.) Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 11876–11896. Association for Computational Linguistics, Bangkok, Thailand (2024). https://doi.org/10.18653/v1/2024.acl-long.641

Bayardo, R.J., Agrawal, R.: Mining the most interesting rules. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '99, pp. 145–154. Association for Computing Machinery, New York, NY, USA (1999). https://doi.org/10.1145/312129.312219

Byun, C., Vasicek, P., Seppi, K.: This reference does not exist: An exploration of LLM citation accuracy and relevance. In: Blodgett, S.L., Cercas Curry, A., Dev, S., Madaio, M., Nenkova, A., Yang, D., Xiao, Z. (eds.) Proceedings of the Third Workshop on Bridging Human–Computer Interaction and Natural Language Processing, pp. 28–39. Association for Computational Linguistics, Mexico City, Mexico (2024). https://doi.org/10.18653/v1/2024.hcinlp-1.3

Chiang, C.-H., Lee, H.-y.: Can large language models be an alternative to human evaluations? In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 15607–15631. Association for Computational Linguistics, Toronto, Canada (2023). https://doi.org/10.18653/v1/2023.acl-long.870

Conmy, A., Mavor-Parker, A., Lynch, A., Heimersheim, S., Garriga-Alonso, A.: Towards automated circuit discovery for mechanistic interpretability. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems, vol. 36, pp. 16318–16352. Curran Associates Inc., Red Hook, NY, USA (2023)

Cao, S., Wang, L.: Verifiable generation with subsentence-level fine-grained citations. In: Ku, L.-W., Martins, A., Srikumar, V. (eds.) Findings of the Association for

Computational Linguistics: ACL 2024, pp. 15584–15596. Association for Computational Linguistics, Bangkok, Thailand (2024). https://doi.org/10.18653/v1/2024.findings-acl.920

Cohen-Wang, B., Shah, H., Georgiev, K., Mądry, A.: Contextcite: Attributing model generation to context. In: Advances in Neural Information Processing Systems, vol. 37, pp. 95764–95807. Curran Associates Inc., Red Hook, NY, USA (2024)

Day, T.: A preliminary investigation of fake peer-reviewed citations and references generated by chatgpt. The Professional Geographer **75**(6), 1024–1027 (2023) https://doi.org/10.1080/00330124.2023.2190373

Färber, M., Jatowt, A.: Citation recommendation: approaches and datasets. International Journal on Digital Libraries **21**(4), 375–405 (2020) https://doi.org/10.1007/s00799-020-00288-2

Gupta, R., Arcuschin, I., Kwa, T., Garriga-Alonso, A.: Interpbench: Semi-synthetic transformers for evaluating mechanistic interpretability techniques. In: Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., Zhang, C. (eds.) Advances in Neural Information Processing Systems, vol. 37, pp. 92922–92951. Curran Associates Inc., Red Hook, NY, USA (2024)

Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., Turini, F.: Factual and counterfactual explanations for black box decision making. IEEE Intelligent Systems **34**(6), 14–23 (2019) https://doi.org/10.1109/MIS.2019.2957223

Guidotti, R., Monreale, A., Ruggieri, S., Naretto, F., Turini, F., Pedreschi, D., Giannotti, F.: Stable and actionable explanations of black-box models through factual and counterfactual rules. Data Mining and Knowledge Discovery **38**(5), 2825–2862 (2024) https://doi.org/10.1007/s10618-022-00878-5

Geng, Z., Schleich, M., Suciu, D.: Computing rule-based explanations by leveraging counterfactuals. PVLDB **16**(3), 420–432 (2022) https://doi.org/10.14778/3570690.3570693

Gao, T., Yen, H., Yu, J., Chen, D.: Enabling large language models to generate text with citations. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 6465–6488. Association for Computational Linguistics, Singapore (2023). https://doi.org/10.18653/v1/2023.emnlp-main.398

Han, T., Nebelung, S., Khader, F., Wang, T., Müller-Franzes, G., Kuhl, C., Försch, S., Kleesiek, J., Haarburger, C., Bressem, K.K., *et al.*: Medical large language models are susceptible to targeted misinformation attacks. NPJ digital medicine **7**(1), 288 (2024) https://doi.org/10.1038/s41746-024-01282-7

Hameed, I., Sharpe, S., Barcklow, D., Au-Yeung, J., Verma, S., Huang, J., Barr,

B., Bruss, C.B.: BASED-XAI: Breaking Ablation Studies Down for Explainable Artificial Intelligence. In: Workshop on Machine Learning in Finance (2022)

Hooshyar, D., Yang, Y.: Problems with shap and lime in interpretable ai for education: A comparative study of post-hoc explanations and neural-symbolic rule extraction. IEEE Access **12**, 137472–137490 (2024) https://doi.org/10.1109/ACCESS.2024.3463948

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K.T., Dähne, S., Erhan, D., Kim, B.: In: Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K.-R. (eds.) The (Un)reliability of Saliency Methods, pp. 267–280. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-28954-6_14

Khalifa, M., Wadden, D., Strubell, E., Lee, H., Wang, L., Beltagy, I., Peng, H.: Source-aware training enables knowledge attribution in language models. In: First Conference on Language Modeling (2024)

Lundberg, S.M., Lee, S.-I.: A unified approach to interpreting model predictions. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 4765–4774. Curran Associates Inc., Red Hook, NY, USA (2017)

Li, W., Li, J., Ma, W., Liu, Y.: Citation-enhanced generation for LLM-based chatbots. In: Ku, L.-W., Martins, A., Srikumar, V. (eds.) Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1451–1466. Association for Computational Linguistics, Bangkok, Thailand (2024). https://doi.org/10.18653/v1/2024.acl-long.79

Macha, D., Kozielski, M., Wróbel, Sikora, M.: Rulexai—a package for rule-based explanations of machine learning model. SoftwareX **20**, 101209 (2022) https://doi.org/10.1016/j.softx.2022.101209

Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. FAT* '20, pp. 607–617. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3351095.3372850

Maheshwari, H., Tenneti, S., Nakkiran, A.: CiteFix: Enhancing RAG accuracy through post-processing citation correction. In: Rehm, G., Li, Y. (eds.) Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), pp. 310–317. Association for Computational Linguistics, Vienna, Austria (2025). https://doi.org/10.18653/v1/2025.acl-industry.23

Press, O., Hochlehnert, A., Prabhu, A., Udandarao, V., Press, O., Bethge, M.: Citeme: Can language models accurately cite scientific claims? In: Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., Zhang, C. (eds.) Advances in

Neural Information Processing Systems, vol. 37, pp. 7847–7877. Curran Associates Inc., Red Hook, NY, USA (2024)

Rorseth, J., Godfrey, P., Golab, L., Srivastava, D., Szlichta, J.: Rage against the machine: Retrieval-augmented llm explanations. In: 2024 IEEE 40th International Conference on Data Engineering (ICDE), pp. 5469–5472 (2024). https://doi.org/10.1109/ICDE60146.2024.00430

Rudin, C., Shaposhnik, Y.: Globally-consistent rule-based summary-explanations for machine learning models: Application to credit-risk evaluation. Journal of Machine Learning Research **24**(16), 1–44 (2023)

Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16, pp. 1135–1144. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939778

Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: high-precision model-agnostic explanations. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI'18/IAAI'18/EAAI'18. AAAI Press, Washington, DC, USA (2018)

Slack, D., Hilgard, S., Jia, E., Singh, S., Lakkaraju, H.: Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. AIES '20, pp. 180–186. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3375627.3375830

Tsirtsis, S., Gomez Rodriguez, M.: Decisions, counterfactual explanations and strategic behavior. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 16749–16760. Curran Associates Inc., Red Hook, NY, USA (2020)

Verma, S., Boonsanong, V., Hoang, M., Hines, K., Dickerson, J., Shah, C.: Counterfactual explanations and algorithmic recourses for machine learning: A review. ACM Comput. Surv. **56**(12) (2024) https://doi.org/10.1145/3677119

Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the gdpr. Harv. JL & Tech. **31**, 841 (2017)

Wang, B., Yue, X., Su, Y., Sun, H.: Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. In: Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., Zhang, C. (eds.) Advances in Neural Information Processing Systems, vol. 37, pp. 95238–95265. Curran Associates Inc., Red Hook, NY, USA (2024)

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., Manning, C.D.: HotpotQA: A dataset for diverse, explainable multi-hop question answering. In: Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J. (eds.) Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 2369–2380. Association for Computational Linguistics, Brussels, Belgium (2018). https://doi.org/10.18653/v1/D18-1259

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E.P., Zhang, H., Gonzalez, J.E., Stoica, I.: Judging llm-as-a-judge with mt-bench and chatbot arena. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA (2023)