Concept-based Explanations for Retrieval-Augmented Large Language Models

Katherine Ling

A thesis submitted to the
Faculty of Graduate Studies in partial
fulfillment of the requirements for the degree of

Master of Science

in

Graduate Program in Computer Science

York University
Toronto, Ontario

August 2026

© Katherine Ling, 2026

Abstract

Large language models (LLMs) with retrieval-augmented generation (RAG) capabilities respond differently to open-ended questions depending on the user's intent and the interpretation of retrieved sources. However, existing explainable AI techniques for these systems focus on explaining the LLM's outputs by attributing its answer to the sources retrieved using RAG rather than to these underlying signals. We present `CORAL`, a framework that organizes the answer space as a lattice of *concepts* representing different intents and interpretations. `CORAL` provides users a structured space to both generate counterfactual explanations and explore how changes in intent and interpretation shape the LLM's output. While an exhaustive method populates the entire lattice to find counterfactuals, we propose pruning strategies based on LLM-aided concept selection, entropy-guided rank aggregation, and graph-of-thoughts reasoning to reduce the number of LLM calls. Finally, we evaluate the effectiveness of our pruning strategies and the quality of our explanations through quantitative and qualitative experiments.

Dedication

To my family, whose love and sacrifice made all of this possible, and to my dog Sophie, for making me take breaks.

Acknowledgements

This thesis would not have been possible without the support of my supervisors, Dr. Jarek Szlichta and Dr. Parke Godfrey. I am deeply grateful for their guidance, their prompt feedback, and their willingness to always make time for our meetings despite their busy schedules. But above all, I am grateful that they let me explore my own ideas and to work on what I genuinely enjoyed. Their endless patience and infectious enthusiasm for research have guided both my academic and personal development.

I would like to express my gratitude to Dr. Lukasz Golab and Joel Rorseth for their invaluable feedback and contributions to this project. I would also like to thank Rafael Dolores for his assistance with debugging and running the experimental evaluation. I am thankful for the financial support I have received from York University, NSERC, 407 ETR, and the Lambert family, which gave me the freedom to focus on my research.

Finally, thank you to my family and friends for their unwavering support and cheering me on throughout this journey.

Table of Contents

Abstract ii

Dedication iii

Acknowledgements iv

Table of Contents v

List of Figures viii

List of Tables x

List of Abbreviations and Symbols xi

1 Introduction 1

1.1 Motivation 1

1.2 Contributions 4

2 Background 7

2.1 Explainable AI 7

2.2 Concept-based Explanations 9

3 System Overview **11**

3.1 Problem Statement 11

3.2 System Design Overview 12

4 Proposed Solution **15**

4.1 Lattice Structure 15

4.2 Concept Generation 17

4.2.1 Pruning Strategy 1: Concept Selection 18

4.2.2 Example 1: Criteria Selection 19

4.2.3 Example 2: Pragma Selection 20

4.3 Lattice Traversal 20

4.4 Proxy Models 21

4.5 Rank Aggregation 23

4.5.1 Pruning Strategy 2: Entropy-based Selection 27

4.6 Pruning Strategy 3: Candidate Selection 28

4.6.1 Example 3: Rank Aggregation on Criterion Lattices 29

4.6.2 Example 4: Rank Aggregation on Pragma Lattices 32

4.7 Graph-of-Thoughts (GoT) 33

4.7.1 Example 5: Graph of Thoughts on Criteria 34

5 Experimental Evaluation **38**

5.1 Quantitative Evaluation 39

5.1.1 Datasets 39

5.1.2 Search Strategies 40

5.1.3 Metrics 42

5.1.4 Ablations 42
5.1.5 Analysis 43
5.2 Qualitative Experiments 51
5.2.1 Product Comparison 51
5.2.2 Source Analysis 52

6 Conclusions **54**
6.1 Summary 54
6.2 Future Work 55

Copyright **58**

Bibliography **59**

List of Figures

3.1 The architecture of `CORAL`. 13

4.1 An example of a criterion lattice pruned using concept selection and rank aggregation. 24

4.2 An example of a pragma lattice along the temporal, geographic, and trustworthy dimensions. 25

4.3 An example of counterfactual search using GoT. 37

5.1 Distribution of minimal counterfactual layer for the size-10 lattices, by dataset and LLM. 39

5.2 Overall performance of pruning methods on accuracy, cost, and distance across the datasets using `Llama3.1-8B`. 44

5.3 Overall performance of pruning methods on accuracy, cost, and distance across the datasets using `Qwen3-32B`. 45

5.4 Overall performance of ERA's entropy variants on accuracy, cost, and distance. 46

5.5 ERA budget ablation for fixed budgets {4, 8, 16, 32, 64, 128} on accuracy, cost, and distance across the datasets using `Llama3.1-8B`. . 48

5.6 ERA budget ablation for fixed budgets {4, 8, 16, 32, 64, 128} on accuracy, cost, and distance across the datasets using `Qwen3-32B`. . . 49

List of Tables

5.1 Performance of ERA (b=4) against exhaustive BFS. 43

List of Abbreviations and Symbols

ML: ..... Machine Learning

AI: ..... Artificial Intelligence

LLM: ..... Large Language Model

RAG: ..... Retrieval-Augmented Generation

XAI: ..... Explainable Artificial Intelligence

CORAL: ..... COncept-based explanations for Retrieval Augmented LLMs

Chapter 1

Introduction

1.1 Motivation

Millions of people turn to *large language models* (LLMs) to answer their questions, talk through hard decisions, and learn new topics. However, even the most advanced LLMs remain prone to *hallucinations*, generating answers that sound reasonable but are factually wrong [1]. One approach to mitigating these hallucinations is *retrieval-augmented generation* (RAG), which grounds model outputs in external knowledge [2]. It retrieves documents relevant to the user's question from an external knowledge base and adds them to the LLM's context window. These documents, included as *sources* within the input context, condition the LLM on explicit knowledge. This reduces hallucinations and enables the incorporation of new knowledge without retraining the backbone LLM. As LLMs move into high-stakes domains such as healthcare and finance [3], both mitigating hallucinations and explaining model outputs are necessary to establish user trust.

Establishing this trust is especially difficult for open-ended questions, where mul-

tiple answers are plausible and no single one is clearly correct. We focus on this open-ended question-answering use case of LLMs with access to RAG (RAG LLMs). The appropriate answer depends on the user's *intent* with the question and on the *interpretation* of the question shaped by the retrieved sources. When answering open-ended questions, LLMs often infer the user's intent implicitly to generate and explain their output. While users can adjust constraints through follow-up prompts, current interfaces offer no structured space for exploration or recourse. As a result, users are left with the LLM's answer with no explanation of how the inferred intent was used to generate the answer, and no structured mechanism to ask what would need to change in their prompt for a different answer to emerge.

For example, the question *"What is the best treatment for a migraine?"* is inherently subjective. The answer depends on what "best" means for the user: what is the most effective for quick relief; what has the fewest side effects; or what aligns with natural medicine practices. The answer also depends on the types of retrieved sources, ranging from reputable scientific journals and health organizations (e.g., Nature Medicine and the World Health Organization) to moderated but unverified forums (e.g., Reddit). If the LLM prioritizes quick relief and recommends over-the-counter pain relievers, a user seeking other options (e.g., triptans) is left to navigate the LLM's decision boundaries through iterative prompting. Furthermore, any self-generated explanations the LLM uses to support its answer are also prone to hallucinations and cannot be relied on as explanations [4].

Addressing this challenge is the primary goal of *eXplainable Artificial Intelligence* (XAI), a field centred around forming explanations for complex systems to better understand the decision-making process. Explanations for individual model outputs

can be categorized as *factual* or *counterfactual*. Factual explanations explain model outputs by identifying the most important features in the input, while counterfactual explanations do so by identifying the minimal change to the input that would lead to an alternative answer. Existing XAI methods for RAG systems form explanations with respect to the retrieved sources. For example, a factual explanation might report that the first two sources were the most important, while a counterfactual explanation might find that removing the first source would lead the LLM to generate a different answer.

However, in open-ended question answering, there are several plausible and reasonable answers supported by the retrieved sources. The LLM's response therefore becomes a selection among candidate answers, where explaining a selection requires identifying what would have to change for a different answer to surface. Counterfactuals are commonly used to provide recourse [5], which is useful for open-ended questions where the user is not expecting a correct answer but rather one that fits their own needs. In cases where the user does not have an expected answer, counterfactuals also provide a means of exploration, where multiple explanations surface non-obvious answers and reveal more of the answer space. Yet existing explanation methods explain with respect to the sources. Knowing which source to remove does not tell the user why that source mattered, whether for its recency, its origin, or its credibility. These implicitly assumed properties, along with the intent the LLM inferred from the question, are the signals that shaped the answer. Yet no existing tool explains the answer space in terms of these signals.

1.2 Contributions

We present CORAL, *COncept-based explanations for RAG LLMs*, a novel XAI tool that lets users explore the space of answers produced by RAG LLMs [6]. CORAL structures this space as a lattice of semantic concepts: simple terms representing user intents (answer criteria) and source interpretations (source types). Each node in the lattice represents the LLM's answer when prompted with a different combination of concepts, and edges connect nodes that differ by a single concept. Lattices are data structures commonly used in data mining, particularly for data profiling and frequent itemset mining [7], [8], [9]. CORAL repurposes the lattice structure to provide users actionable and understandable insights by revealing which concepts were pivotal to generating the original answer and how to reach an alternative answer. This work has resulted in a publication at the 42nd IEEE Conference on Data Engineering (ICDE) [6].

In Section 2, we provide a background on XAI techniques and concept-based explanations. We present the architectural overview in Section 3, and evaluate the effectiveness of the system through quantitative and qualitative experiments in Section 5. Our main contributions are as follows.

1. **Conceptual Framework (Section 3).** We propose a framework to explore the answers RAG LLMs produce when responding to open-ended questions. Existing XAI work addresses low-level, "mechanical" understanding of LLMs (e.g., mapping facts to neurons) [10], [11] and the influence of RAG sources on outputs [12], [13]. We instead structure this space using simple terms, representing semantically meaningful concepts.
2. **Pruning Strategies (Section 4).** An exhaustive method that populates the entire lattice requires a separate LLM inference call per node, where the prompt is

modified to specify the corresponding intents and interpretations. CORAL employs three pruning strategies to reduce the number of these calls by leaving some lattice nodes unexplored if they are deemed unlikely to lead to a different answer.

(a) **LLM-aided Concept Selection (Section 4.2.1).** This method shrinks the search space before the search begins. It prompts the LLM to select a smaller, more relevant subset of concepts to build the lattice from. Since lattice size grows exponentially with the number of concepts, reducing the concept set is the first step.

(b) **Entropy-guided Rank Aggregation (Section 4.5.1).** This method builds a proxy model that estimates the LLM's answer for different concept combinations. To build the proxy, CORAL prompts the LLM to rank its answers for each *individual* concept, rather than just returning a single answer. Then, *rank aggregation* methods are applied [14] to approximate the answer the LLM would give for a given combination of concepts. We introduce a *budget* which allocates additional LLM calls to build this proxy model. CORAL applies *entropy* to each aggregated ranking to identify the most uncertain predictions (e.g., those resulting in ties), which are the ones worth verifying using an LLM call and consuming the budget. During the search, a node is pruned when the proxy model keeps the original answer ranked first and is confident in that prediction, indicated by low entropy.

(c) **Graph of Thoughts [15] (Section 4.7).** This method prompts the LLM in a single inference call to return its reasoning paths as a graph. The graph decomposes the user's initial question into sub-questions, identifies the concepts that support each, and maps those concepts to plausible answers. CORAL ex-

tracts the concept subsets that support each answer to populate nodes within the lattice.

3. **Evaluation (Section 5).** We evaluate the effectiveness of the pruning strategies and the resulting explanation quality through the following evaluations:

   (a) **Quantitative Evaluations (Section 5.1).** We conduct quantitative evaluations to measure how the performance of `CORAL` scales with larger search spaces. We also evaluate `CORAL` on different budget strategies to determine the optimal budget and compare the performance across different entropy variations.

   (b) **Qualitative Evaluations (Section 5.2)** We present use cases across several domains that demonstrate how users of `CORAL` can benefit from counterfactual explanations, including product comparisons and public opinion monitoring.

Chapter 2

Background

2.1 Explainable AI

Explainable AI methods can be broadly categorized into *by-design* and *post-hoc* approaches. By-design approaches create models that are inherently interpretable, such as decision trees and linear regression models, while post-hoc methods are applied after a model is trained. Many post-hoc XAI methods are *local*, seeking to explain a specific individual prediction, rather than *global*, which explains general model behaviour. Within local methods, these techniques can be further separated into *model-specific* and *model-agnostic* approaches. Model-specific methods rely on access to the model's internal architecture, such as its attention weights or gradients, which is often unavailable for state-of-the-art, closed-source LLMs. Model-agnostic methods instead treat the model as a black box, observing how changes to the input affect the output.

Local explanation methods generally fall into two categories: *factual* and *counterfactual*. Factual methods identify the most important input features in contributing

to the output. LIME and SHAP are two prominent factual methods commonly used to explain machine learning models by perturbing the input. LIME [16] perturbs the input point and fits a simple, interpretable model (e.g., linear regression or decision tree) to the model's behaviour on the perturbed points, taking the feature importance from the weights of the interpretable model. This simple, interpretable model is also referred to as a *surrogate model*. SHAP [17] estimates each feature's contribution by perturbing it and averaging the resulting change in the prediction over subsets of the other features (i.e., the Shapley value [18]). However, recent work has shown that the scores produced by these methods are often unreliable and can be easily manipulated [19], [20], [21]. On the other hand, counterfactual methods identify minimal changes to the input that would lead to an alternative answer. This provides actionable recourse by highlighting what the user could change to achieve an alternative, desired output. In both cases, existing methods typically explain a single decision in isolation without giving users a structured way to explore alternative outputs.

For RAG LLMs specifically, existing XAI methods have focused on forming explanations based on the retrieved sources. Factual explanation methods identify the key sources that contributed to generating the answer. To attribute the answers to sources, past works have used attention weights [22], [23], input gradients [24], [25], and linear surrogate models [13]. On the counterfactual side, tools like RAGE [12] identify how different combinations and permutations of sources affect the LLM's output.

Unlike these methods, `CORAL` focuses not on the raw, retrieved sources but on how the perceived user intent and source interpretation affect the output. By organizing the answer space as a lattice, `CORAL` provides users with a structured space for ex-

ploration and recourse. As such, we propose a post-hoc, model-agnostic method that generates concept-based counterfactual explanations for RAG LLM outputs. Section 3 provides a system overview of our `CORAL` prototype which implements this approach.

2.2 Concept-based Explanations

Concept Bottleneck Models (CBM) are interpretable *by design* [26]. A CBM's architecture can be split into two parts: the first maps inputs to a set of pre-defined, interpretable concepts, and the second maps those concepts to the output. So, every prediction passes through this concept layer, referred to as the *bottleneck* layer, and the model is forced to rely on these concepts to form their outputs, making it interpretable. While CBM was originally introduced for image classification, recent work has adapted it to LLMs as a Concept Bottleneck Large Language Model (CB-LLM) to build inherently interpretable LLMs [27]. CB-LLM is designed for text classification, but it encounters limitations in open-ended question answering. Specifically, the CB-LLM's bottleneck layer requires a pre-defined set of concepts for each class in a dataset. In open-ended question answering, however, these concepts are not known in advance. Since a fixed bottleneck cannot accommodate new concepts, a by-design approach would require retraining the model for each new query. `CORAL` instead adopts a post-hoc approach that identifies relevant concepts at inference time and organizes them into a lattice to provide actionable recourse without needing to access model internals or to modify the model.

A second concept-based approach is TCAV (Testing with Concept Activation Vectors) [28]. TCAV represents a concept as a direction in the model's activation space. It achieves this by training a linear classifier to separate the activations of inputs

that exhibit the concept from those of random inputs, and takes the resulting boundary's normal as the concept activation vector (CAV). TCAV is a post-hoc method, but requires access to the model's internal weights. It also must train a separate linear model per concept, which can be expensive. Since `CORAL` is model-agnostic, it bypasses this obstacle.

Chapter 3

System Overview

In this section, we present CORAL, a post-hoc, model-agnostic approach to generate concept-based counterfactual explanations for RAG LLM outputs. We provide our formal problem statement in Section 3.1, and an overview of our CORAL system in Section 3.2. (In Section 4, we discuss in depth how each component of the approach is accomplished.)

3.1 Problem Statement

We focus on the open-ended question answering task for RAG-enhanced LLMs. Let q be the user's query, D be the retrieved sources, and L be the LLM we would like to explain. Depending on the type of system L, sources may be supplied by the user, retrieved using an external retrieval model, or fetched from the web in real-time.

The objective of CORAL is to explain the LLM's response to q through concept-based counterfactual explanations. Let $C = \{c_1, c_2, ..., c_n\}$ be the set of semantic *concepts* relevant to the query context and $C' \subseteq C$ be the set of *concepts of interest*.

Both are provided by oracles: $\mathscr{O}_C$ for C; and $\mathscr{O}_{C'}$ for C'. In practice, the role of these oracles is fulfilled by the user or by another LLM, as described in Section 4.2. Optionally, the user may select a pinned set $C^* \subseteq (C \setminus C')$ of concepts that are always included in the prompt. A natural language prompt p is formed by combining q, D, C^*, and C'. While p is the final input used to instruct L, we represent L's answer as $a = L(q, D, C' \cup C^*)$. For brevity, we write $L(q, D, X)$ to mean $L(q, D, X \cup C^*)$ for any concept subset X, treating C^* as implicitly included in every LLM call. A *perturbation* refers to the addition or removal of concepts from the prompt. A counterfactual is then defined as the minimal perturbation to the prompt that causes L to generate an alternative answer $a' \neq a$.

3.2 System Design Overview

The system architecture of `CORAL` is shown in Figure 3.1. Given a RAG LLM and a user's query, a set of relevant sources are retrieved and the LLM generates an initial answer. An external LLM or user then generates a set of candidate concepts, either covering criteria related to the question (Figure 3.1 (1a)) or directives related to the sources (Figure 3.1 (2a)). The user can select a subset of criteria (Figure 3.1 (1b)) or pragmas (Figure 3.1 (2b)) from this candidate list to define the counterfactual search space or allow an external LLM to choose a set of relevant candidates automatically. After the concepts are selected, `CORAL` uses rank aggregation and entropy to prioritize the evaluation of uncertain nodes and guide the counterfactual search (see Figure 3.1, 1c and 2c). Alternatively, users can draft a quick explanation using the graph of thoughts approach. The LLM is prompted to answer the question by constructing a graph outlining its reasoning paths following a guideline and one-shot example, as

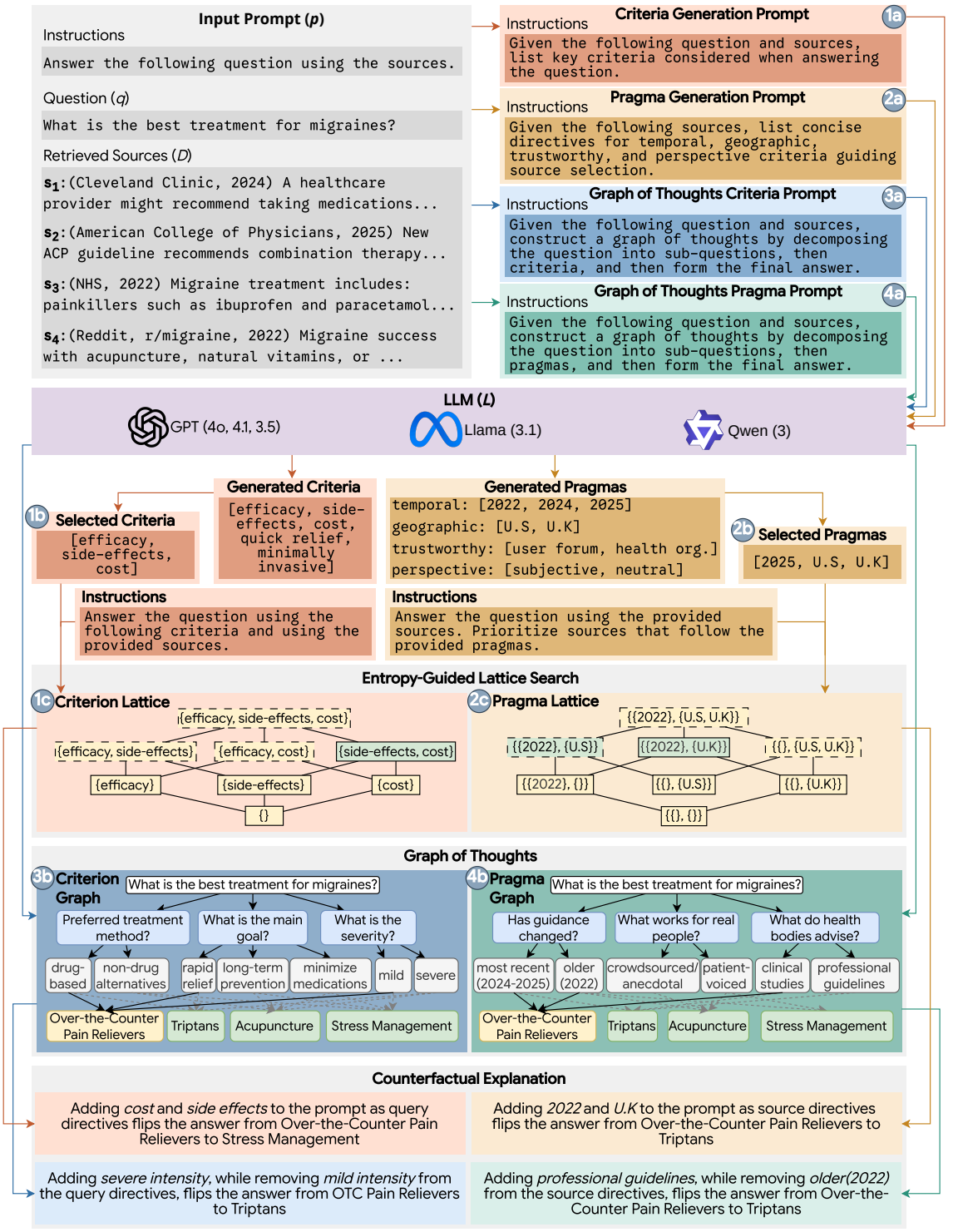


Figure 3.1: The architecture of CORAL.

seen in Figure 3.1 (3a and 4a). After the graph is generated, a projected graph is extracted to find the counterfactuals, the alternative answers closest to the initial answer (Figure 3.1, 3b and 4b).

Chapter 4

Proposed Solution

We first discuss how we structure the search space using lattices in Section 4.1. We discuss how each component of `CORAL`'s approach is realized in the subsequent sections: concept generation (Section 4.2); lattice traversal (Section 4.3); proxy-model construction using rank aggregation (Sections 4.4 and 4.5) with cut-offs (Section 4.6); and Graph of Thoughts (Section 4.7).

4.1 Lattice Structure

`CORAL` organizes the answer space as a lattice to systematically generate counterfactuals and provide users with a structured space to explore how different concepts shape the LLM's output. Each node represents the LLM's answer when prompted with a subset of concepts $X \subseteq C$. Edges connect pairs X, Y where Y extends X by one additional concept. The resulting lattice contains $2^{|C|}$ nodes, scaling exponentially with the number of concepts. Since evaluating each node requires a distinct LLM call, naive exhaustive search is costly in terms of inference costs and is further constrained

by API rate limits. To address this exponential bottleneck, `CORAL` introduces several strategies to prune and navigate the lattice efficiently.

Within the lattice structure, we identify the node representing the selected set C' as the initial node. Since perturbations to the prompt involve adding or removing concepts as directives, we define two types of counterfactuals:

- **Bottom-up** counterfactuals identify which concepts when added to the prompt cause the model's initial output to flip. The concepts in C' remain fixed, and we begin with the LLM's original answer $a = L(q, D, C')$. We search for the minimal set of concepts $X \subseteq (C \setminus C')$ such that $L(q, D, C' \cup X) \neq a$.

- **Top-down** counterfactuals identify which concepts, when removed from the prompt, cause the model's initial output to flip. Beginning from $a = L(q, D, C')$, we search for the maximal subset $X \subset C'$ (or the minimal removal) such that $L(q, D, X) \neq a$.

Since LLM APIs charge per token and are often rate limited, populating the entire lattice to generate a single counterfactual is not viable for real-time user interaction. To address this, we build a proxy model to estimate the LLM's outputs and identify nodes worth calling the LLM on. We introduce an initialization budget parameter B which defines the maximum number of LLM calls allocated for pruning the lattice. During this pre-search phase, `CORAL` calls the LLM on up to B nodes to build a proxy model of the lattice. In the search phase, this proxy model is used to prune the lattice by skipping redundant nodes and prioritizing nodes likely to be counterfactuals.

4.2 Concept Generation

`CORAL` generates counterfactuals by traversing a structured search space constructed over semantic concepts. Since the user may not explicitly state any concepts within the question, the first step is identifying a set of candidate concepts C. In our implementation, the role of the oracle described in Section 3.1 is fulfilled by an external LLM E, but a user could also supply their own concepts. The external LLM E is prompted to retrieve a set of candidate concepts based on the user's query and retrieved sources. The generation instructions ask for concepts that are semantically meaningful and distinct from one another, so that no two concepts capture the same intent or source directive.

To illustrate, consider the running example of a user asking, "What is the best university?" This is a broad question, and the resulting answer depends on how the LLM interprets and prioritizes the user's intent and the provided sources. To structure this space for exploration, we categorize concepts into two distinct dimensions.

- **Criteria (Query-side)**: These represent the user intent used to answer the query. Candidate criteria might include *teaching quality, employability, or research output.*
- **Pragmas (Document-side)**: These represent the metadata directives that guide how the LLM prioritizes information from the retrieved documents. We define four primary pragma dimensions:
  - *Temporal*: Directives regarding the source's publication date, such as retaining sources published within the past year to ensure answer recency.
  - *Geographic*: Directives regarding the origin of the sources, such as restricting to U.S. sources to reflect American audience preferences.
  - *Trustworthy*: Directives regarding the reliability of the source, such as prioritiz-

ing official ranking bodies; e.g., QS World Rankings and Times Higher Education (THE) Rankings.

– *Perspective*: Directives regarding the viewpoints represented within the source, such as emphasizing aggregated methodologies over subjective viewpoints; e.g., public forum comments such as Reddit.

Criteria and pragmas each define a separate lattice, giving users a structured space to explore how user intent or source interpretation independently shape the LLM's output. As the counterfactual search space grows exponentially with the number of concepts, we introduce pruning techniques to make this search possible. A naive approach would apply breadth-first search to traverse the lattice and exhaustively evaluate each node until the counterfactual is found. This approach guarantees that the found counterfactual is minimal, but can be constrained when the number of LLM calls is limited by a budget. We explore several strategies to reduce and navigate the search space. Note that while we separate criteria and pragmas into two search spaces, the following strategies can apply to both types of concepts.

4.2.1 Pruning Strategy 1: Concept Selection

One straightforward way to reduce the search space is to reduce the number of concepts considered, since the size of the lattice scales as $2^{|C|}$ with the candidate set C. We extract a manageable subset $C' \subseteq C$ using an oracle $\mathscr{O}$. `CORAL` supports two modes for $\mathscr{O}$:

- **Semi-automatic**: The user manually selects concepts from the candidate set C that align with their intent.
- **Fully-automatic**: The external LLM E identifies a reduced subset containing

the most relevant candidates. On the query-side, the LLM is prompted to select criteria likely to surface a different answer. On the document-side, the LLM is instructed to select the key pragma values that would result in an alternative answer.

Optionally, the user may designate a subset of concepts as pinned, forming C^*. Pinned concepts are excluded from C' and remain fixed throughout the search, allowing users to explore the effect of other concepts while preserving concepts of interest.

4.2.2 Example 1: Criteria Selection

Suppose the user is looking to explore how different criteria in the best university query lead to alternate answers. The external LLM E generates candidate criteria $C = \{$*teaching quality*, *affordability*, *employability*, *international outlook*, *prestige*, *research*, *student life*, *academic reputation*$\}$. The LLM E is further used to select a subset of relevant criteria from C; e.g., $\{$*teaching quality*, *employability*, *research*, *student life*, *academic reputation*$\}$. Suppose the user values *student life* and requires it to be fixed throughout the search. The fixed subset becomes $C^* = \{$*student life*$\}$. The user then deselects *research* as a criterion, being only interested in undergraduate studies, resulting in the subset $C' = \{$*teaching quality*, *employability*, *academic reputation*$\}$. This example illustrates how both modes can be used in sequence. The external LLM E first narrows C to a relevant subset in fully-automatic mode, after which the user makes a final adjustment in semi-automatic mode to produce C'. See the resulting reduced criterion lattice in Figure 4.1.

4.2.3 Example 2: Pragma Selection

Suppose the user is then interested in how source metadata affects the LLM's answer. The external LLM E is prompted to generate relevant pragmas based on the provided sources D, producing candidate pragmas $C = \{$*Temporal:* $\{$*2025, 2026*$\}$, *Geographic:* $\{$*Global, U.S., U.K.*$\}$, *Trustworthy:* $\{$*official ranking, public user forum*$\}$, *Perspective:* $\{$*subjective, neutral*$\}\}$. The user, interested in the latest official ranking data from a North American and European standpoint, handpicks $C' = \{$*Temporal:* $\{$*2026*$\}$, *Geographic:* $\{$*U.S., U.K.*$\}$, *Trustworthy:* $\{$*official ranking*$\}\}$. The resulting reduced lattice formed using the user selected pragmas is displayed in Figure 4.2.

4.3 Lattice Traversal

Algorithm 1 outlines the general search procedure for identifying counterfactual explanations through concept lattice traversal. Although the algorithm describes a top-down search, the logic also applies to a bottom-up search, where the search begins at $a = L(q, D, C^*)$ and incrementally adds concepts from C' to explore up the lattice. Given the inputs L, q, D, the algorithm is guided by a proxy strategy P that estimates node importance scores σ, built using the budget B. In `CORAL`, we use rank aggregation as the proxy strategy but we keep P general in the pseudocode. LLM calls made using B are saved in the cache K. The search terminates after finding n counterfactuals or after exhausting the pruned search space.

Algorithm 1 can be split into two phases: an initialization phase and a traversal phase. The initialization phase begins by identifying the set of candidate answers A from the LLM, providing q and D as input (Line 4). In Line 5, we retrieve the

baseline, initial answer $a \in A$ by prompting L to answer q with the selected set C' and provided documents D. The proxy strategy P is called in Line 6 to generate scores for each subset $S \subseteq C'$ (equivalently, every node in the lattice), using budget B to build the model. Concept subsets evaluated by the LLM using B and the resulting answers are stored in the cache K.

In the search phase, the procedure transitions into an iterative, top-down traversal of the lattice (Lines 7–23). At each level ℓ, the scores for nodes in the level, N_ℓ, are retrieved (Line 8). The pruning mechanism is enforced using the selection policy Φ in Line 9, which removes nodes whose answer is likely to remain the same as the initial answer a. In Line 10, scores are ordered to evaluate the most likely counterfactual nodes first. For each node in N_ℓ, the system either retrieves a pre-computed answer from the cache or calls the LLM on the corresponding concept subset to obtain the answer a' (Lines 11–16). When a' does not match the original answer a, a counterfactual is identified and added to the returned set of counterfactuals $\mathscr{E}$ (Lines 17–18). The search terminates after discovering n counterfactuals or exhausting the pruned search space (Lines 19–23).

In naive breadth-first search (BFS), no proxy strategy, selection policy, or budget is available. The retrieved scores σ default to 0 and the cache K is empty. As such, the prioritization of nodes via sorting scores in Line 10 would be effectively random, as expected for BFS.

4.4 Proxy Models

When BFS exhaustively traverses the lattice, nodes within a layer are evaluated in random order. A proxy model replaces this random selection by estimating the

Algorithm 1 Top-Down Counterfactual Search

Require: LLM L, user's query q, retrieved documents D, chosen concept set C', proxy model P, selection policy Φ, budget B, number of requested counterfactuals n

Ensure: Counterfactual explanations $\mathscr{E}$

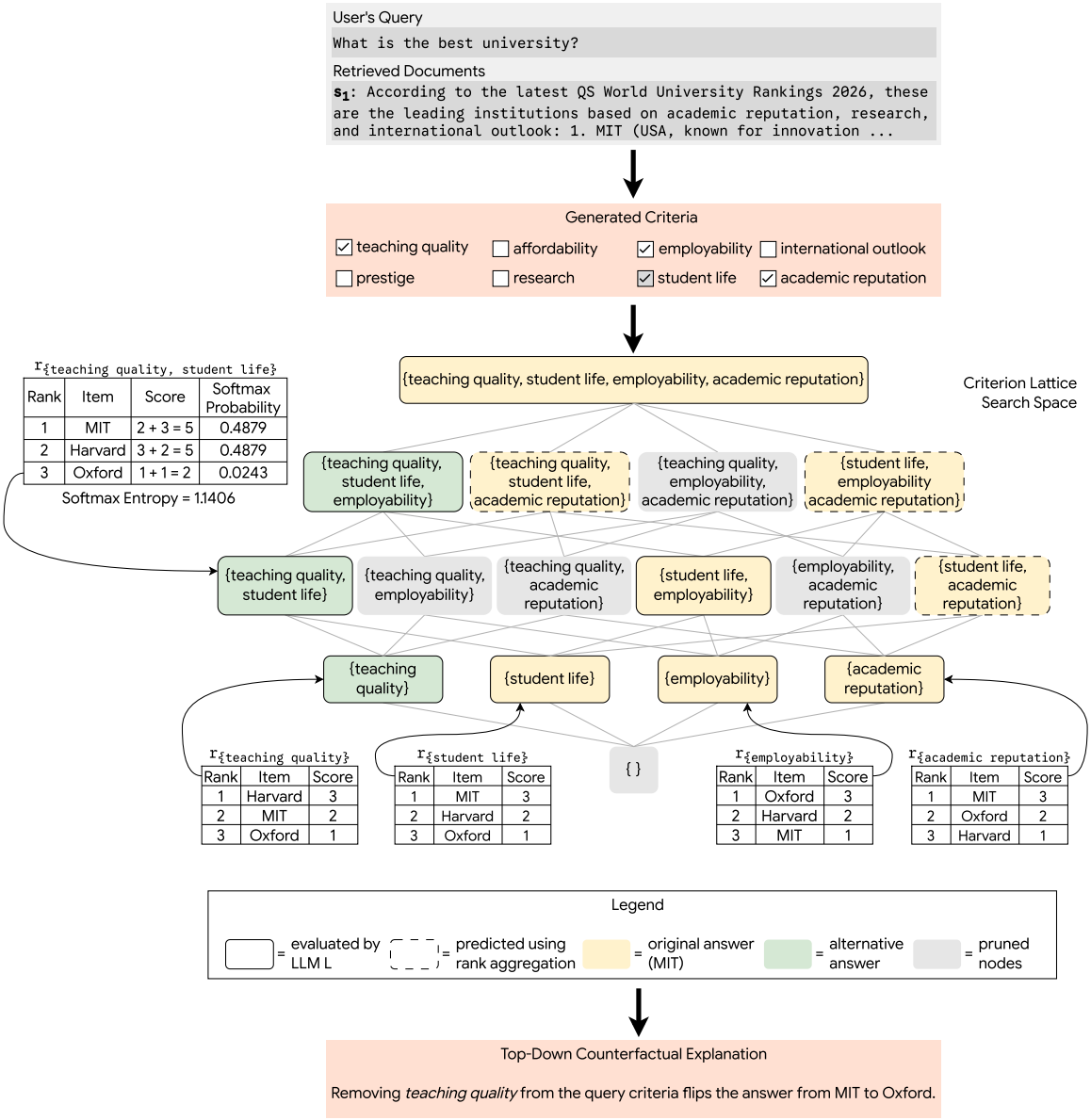
$\mathscr{E} \leftarrow \emptyset$ ▷ Initialize counterfactuals set
$K \leftarrow \emptyset$ ▷ Initialize cached LLM answers
$\sigma \leftarrow \emptyset$ ▷ Initialize scores
$A \leftarrow L(q, D)$ ▷ Get candidate answers
$a \leftarrow L(q, D, A, C')$ ▷ Get initial answer
$\sigma, K \leftarrow P(L, q, D, A, C', B)$ ▷ Score and cache subsets
for $\ell = |C'| - 1$ **to** 0 **do** ▷ Top-down lattice search
    $N_\ell \leftarrow \{S \subseteq C' \mid |S| = \ell\}$ ▷ Get size-ℓ subsets
    $N_\ell \leftarrow \Phi(N_\ell, \sigma)$ ▷ Prune undesirable nodes
    $N_\ell \leftarrow \textsc{sort}(N_\ell)$
    for $S \in N_\ell$ **do**
        if $S \in K$ **then**
            $a' \leftarrow K[S]$ ▷ Retrieve answer from cache
        else
            $a' \leftarrow L(q, D, A, S)$ ▷ Call the LLM
        end if
        if $a' \neq a$ **then** ▷ Counterfactual identified
            $\mathscr{E} \leftarrow \mathscr{E} \cup \{(S, a')\}$
            if $|\mathscr{E}| \geq n$ **then return** $\mathscr{E}$
            end if
        end if
    end for
end for
return $\mathscr{E}$

LLM's probability of generating the original answer a for any given node. We can then use these probabilities to prioritize nodes likely to surface a counterfactual and prune those that are not. To build these proxy models, we make two assumptions. First, we assume that the concepts are semantically meaningful and distinct from each other. This assumption is supported by design in the concept generation phase which prompts the LLM to generate distinct and non-overlapping concepts. For

example, *teaching quality* and *student life* capture different intents of the query and are unlikely to overlap with each other. Second, we assume that a concept affects the LLM's answer the same way regardless of which other concepts appear alongside it in the prompt. For example, we assume *teaching quality* pushes the same universities toward the top of the ranking whether it is the only criterion in the prompt or one of four. This assumption is standard among proxy-based explanation methods. For example, LIME explains a prediction by fitting a simple linear model to the black box's behaviour on perturbed inputs, giving each feature a single weight [16]. ContextCite does the same for RAG systems, randomly ablating sources from the context and fitting a linear model to predict the LLM's output, so each source receives a fixed weight [13]. In both cases, a feature's contribution is fixed and does not change based on which other features are present. Without this assumption, we would have to evaluate every combination directly to see how the concepts behave together, which is the exhaustive search we are trying to avoid. In the following section, we describe how we use rank aggregation as a proxy model.

4.5 Rank Aggregation

To reduce the number of inference calls to the backbone LLM L, we leverage rank aggregation as a proxy model P to approximate the LLM's answer for a given set of concepts. The strategy approximates the LLM's response as a consensus obtained by aggregating the ranked responses for individual concepts. This is justified by the second assumption in Section 4.4. If each concept behaves the same way alone as it does when prompted beside others, the rankings obtained for individual concepts can be combined to estimate the LLM's response for any set of concepts. In practice,



$r_{\{\text{teaching quality, student life}\}}$

Rank	Item	Score	Softmax Probability
1	MIT	2 + 3 = 5	0.4879
2	Harvard	3 + 2 = 5	0.4879
3	Oxford	1 + 1 = 2	0.0243

$r_{\{\text{teaching quality}\}}$

Rank	Item	Score
1	Harvard	3
2	MIT	2
3	Oxford	1

$r_{\{\text{student life}\}}$

Rank	Item	Score
1	MIT	3
2	Harvard	2
3	Oxford	1

$r_{\{\text{employability}\}}$

Rank	Item	Score
1	Oxford	3
2	Harvard	2
3	MIT	1

$r_{\{\text{academic reputation}\}}$

Rank	Item	Score
1	MIT	3
2	Oxford	2
3	Harvard	1

Figure 4.1: An example of a criterion lattice pruned using concept selection and rank aggregation.

we do not aggregate the individual rankings directly. Instead, we build each layer of the lattice from the layer below it. This allows rankings obtained from the LLM during the budget phase to propagate upward, as a node evaluated by L replaces

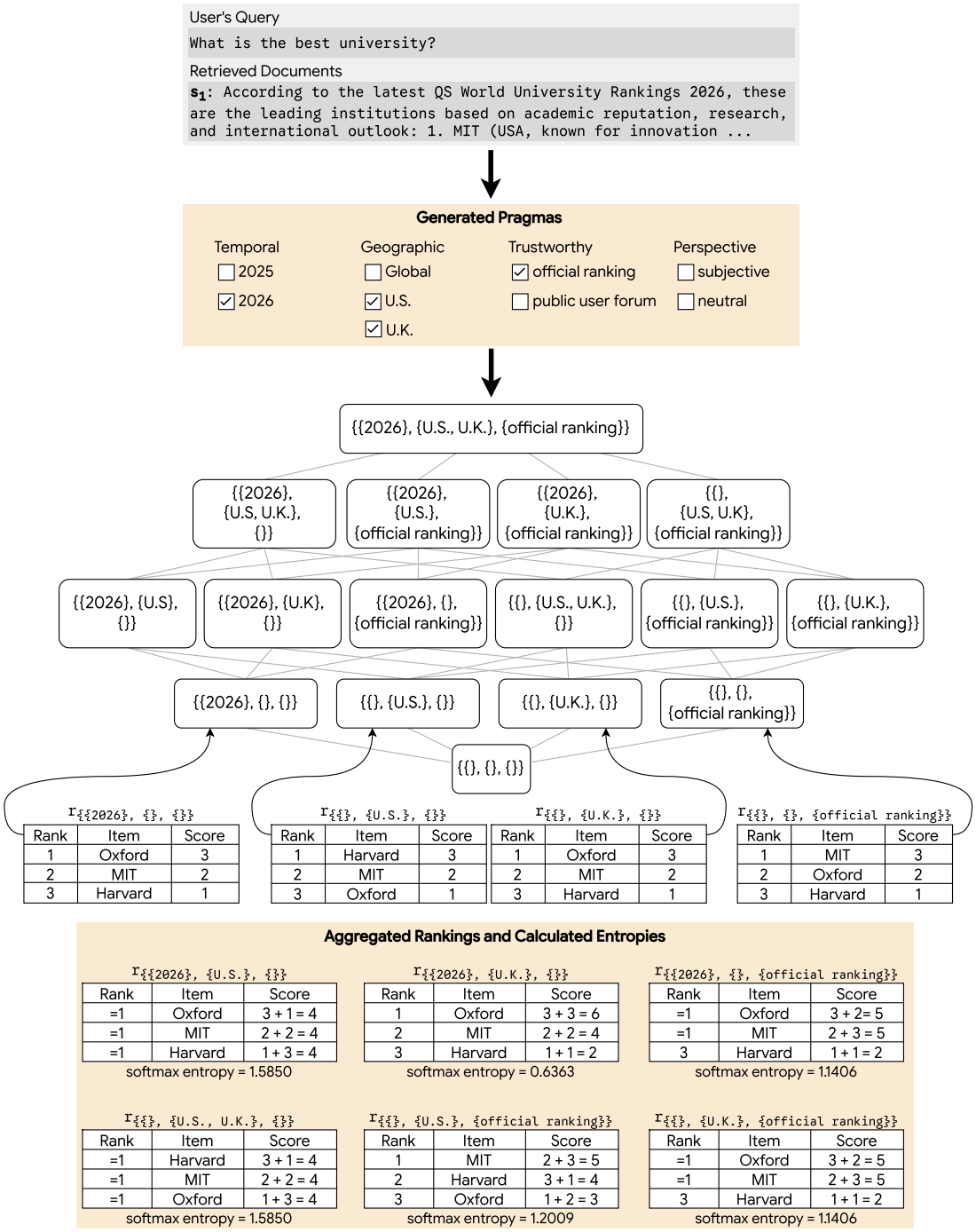


Figure 4.2: An example of a pragma lattice along the temporal, geographic, and trustworthy dimensions.

its estimated ranking with the true one and improves the estimates of every node above it. Aggregating the individual rankings directly would discard these corrections. This method also assumes that the LLM considers concepts equally, so each concept contributes equally to the aggregated ranking. This procedure is outlined in Algorithm 2. For each individual concept $c \in C$, we retrieve a ranked list r_c of these candidate answers from the LLM by modifying the prompt p to rank the candidates in A, rather than generating a single answer (Lines 4–7). Starting from the bottom up, we approximate the rankings of a node in layer ℓ by aggregating the concept subsets of size $\ell - 1$ from the layer below using a rank aggregation method $\mathscr{A}$ (Lines 9–20). For example, to predict the answer for a combination of three concepts $\{A, B, C\}$, we aggregate the previously computed rankings for the concept subsets of size 2: $\{A, B\}$, $\{A, C\}$, and $\{B, C\}$. A scoring function Ψ assigns a score to each aggregated ranking (Line 22), and a selection policy Φ is used to select a subset of nodes in layer ℓ as candidate nodes for LLM evaluation (Line 24). Candidate nodes are then sorted by score (Line 25) and evaluated by L until the budget is exhausted (Line 26–30). The cache K stores the top-ranked answer for each evaluated subset (Line 32).

Any rank aggregation method can be used in Algorithm 2. In our implementation, we use the methods provided by the PyFlagr library [29], such as Borda Count, MC4 [14], and Kemeny Young [30], as a black box to form the consensus ranking. Each aggregated list assigns a *consensus score* s_j to each candidate answer $a_j \in A$, representing its relative strength in the ranking. These consensus scores, rather than just the rankings, are stored and reused as input when aggregating rankings at higher lattice layers. We make an implicit assumption that the answer ranked first in the list corresponds to the answer generated using p. Thus, the predicted output for a

concept subset $X \subseteq C$ is defined as the top-ranked answer in the aggregated list.

4.5.1 Pruning Strategy 2: Entropy-based Selection

While rank aggregation can efficiently approximate rankings, ties for first place result in a random selection between the tied candidates. To address this, we introduce an entropy-based scorer Ψ that assigns uncertainty scores to rankings (Line 22 in Algorithm 2), using the budget B to allocate additional LLM calls to evaluate nodes with high uncertainty. We measure uncertainty using entropy. As a base, we use standard Shannon entropy [31]:

$$H_{\text{shannon}}(X) = -\sum_{i=1}^{n} p_i \log_2(p_i), \tag{4.1}$$

where the probability of the i-th item being the top-ranked item, p_i, is measured as

$$p_i = \frac{s_i}{\sum_{j=1}^{n} s_j} \tag{4.2}$$

We also consider three other variants as scorers, each modifying how Shannon entropy is computed.

The *Rényi entropy* variant is a generalization of Shannon entropy, introducing a tunable parameter α that controls the sensitivity [32]. When $\alpha < 1$, the metric becomes more sensitive to the bottom of the ranking. When $\alpha > 1$, it focuses more heavily on the top of the ranking.

$$H_\alpha(X) = \frac{1}{1-\alpha} \log_2 \left(\sum_{i=1}^{n} p_i^\alpha \right) \tag{4.3}$$

The *Weighted entropy* variant modifies Shannon entropy by applying a weight to each outcome [33]. In this work, we apply a rank-based decay weight $\frac{1}{i}$ to each term, prioritizing uncertainty at the top of the ranking over lower-ranked candidates:

$$H_{\text{weighted}}(X) = -\sum_{i=1}^{n} \frac{1}{i} \cdot p_i \log_2(p_i) \tag{4.4}$$

The *Softmax entropy* variant applies Shannon entropy but replaces the linear normalization of p_i with a softmax function, making it better suited to handling ties:

$$H_{\text{softmax}}(X) = H_{\text{shannon}}(X), \quad \text{where } p_i = \frac{e^{s_i/T}}{\sum_{j=1}^{|A|} e^{s_j/T}} \tag{4.5}$$

The temperature parameter T controls the sensitivity, where higher values of T produce a more uniform distribution, while lower values result in a sharper distribution. The main benefit of applying softmax instead of linear normalization is the handling of ties within the rankings. The Shannon entropy of a ranking with ties for first is similar to a ranking with ties for second. The exponential function differentiates between the two, assigning a higher entropy score to the ranking with ties for first.

4.6 Pruning Strategy 3: Candidate Selection

To allocate the budget B to each layer, we adopt a greedy approach that uses a selection policy to separate high-priority nodes from low-priority nodes by their scores (Line 24). In `CORAL`, we use *Max Jump*, which identifies the optimal evaluation threshold by detecting the largest gap between consecutive scores in the sorted list. Nodes with scores above the threshold are identified as candidates for LLM evaluation.

Within each lattice layer $\ell \geq 2$, nodes are organized in descending order of their scores (Line 25). The max jump cutoff i^*_ℓ is calculated as:

$$i^*_\ell = \text{argmax}_i(y_i - y_{i+1}) \tag{4.6}$$

where y_i is the entropy score of the i-th node in the sorted list. This index marks the point where the entropy drops the most significantly, separating uncertain nodes (above the cutoff) from more certain ones (below it). We prioritize the top i^*_ℓ nodes, calling the backbone LLM L to evaluate them in descending order of entropy. This process continues layer by layer, until the total budget B is exhausted or all candidate nodes have been evaluated (Lines 29–33).

The motivation behind this greedy, bottom-up approach is to leverage the information gained through spending the budget on additional LLM calls. By resolving uncertainty at lower layers early, using L to evaluate and "fix" the rankings of the most uncertain nodes, the aggregated rankings for the nodes above benefit from more reliable inputs, improving the overall quality of the proxy model.

4.6.1 Example 3: Rank Aggregation on Criterion Lattices

Continuing the university example, suppose the LLM L's initial answer based on all four criteria in C' is MIT. First, L generates a ranked list for each individual criterion. See Figure 4.1 for the generated rankings.

Suppose we would like to approximate L's response for the criteria combination $\{$*teaching quality, student life*$\}$. Using Borda Count as the aggregation method, each answer in the known ranked list receives points inversely proportional to its rank and

consensus scores are obtained by summing the points across lists. For instance, in a ranking containing three items, the item ranked first receives 3 points, the item ranked second receives 2 points, and the item ranked third receives 1 point. The resulting ranked list $r_{\{\textit{teaching quality, student life}\}}$ is obtained as:

1. MIT: 2 (teaching quality) + 3 (student life) = 5
2. Harvard: 3 (teaching quality) + 2 (student life) = 5
3. Oxford: 1 (teaching quality) + 1 (student life) = 2

To calculate the softmax entropy score, we first calculate the probabilities using the softmax function.

MIT: $\frac{e^5}{e^5+e^5+e^2} \approx 0.4879$

Harvard: $\frac{e^5}{e^5+e^5+e^2} \approx 0.4879$

Oxford: $\frac{e^2}{e^5+e^5+e^2} \approx 0.0243$

Using these probabilities, we can calculate the softmax entropy: $-0.4879 \log_2 0.4879 + -0.4879 \log_2 0.4879 + -0.0243 \log_2 0.0243 = 1.1406$.

We repeat this same process for all the criteria combinations of size two that included *student life* as a criterion. The ranked list $r_{\{\textit{student life, employability}\}}$ is:

1. MIT: 3 (student life) + 1 (employability) = 4
2. Harvard: 2 (student life) + 2 (employability) = 4
3. Oxford: 1 (student life) + 3 (employability) = 4

resulting in a softmax entropy score of 1.5850. The ranked list $r_{\{\textit{student life, academic reputation}\}}$ is:

1. MIT: 3 (student life) + 3 (academic reputation) = 6
2. Harvard: 2 (student life) + 1 (academic reputation) = 3
3. Oxford: 1 (student life) + 2 (academic reputation) = 3

resulting in a softmax entropy score of 0.5289.

Suppose we are given a budget of two. We apply the max jump function and find that the cutoff index is two; that is, the top two scoring nodes in the second layer are considered "uncertain" and would benefit from being evaluated by an LLM. See Figure 4.1 for the resulting lattice. Nodes with a solid border indicate that the node's answer has been evaluated by an LLM and a dashed border indicates that the answer has been approximated using rank aggregation. Nodes with the yellow background indicate the original answer (MIT) is ranked first, while nodes in green highlight nodes with alternative answers (counterfactuals). Recall that the user fixed *student life* as a criterion, so nodes that do not contain it are pruned.

We also observe the benefit of using softmax entropy over Shannon entropy when there are ties within rankings. Shannon entropy calculates the scores for rankings $r_{\{\textit{teaching quality, student life}\}}$, $r_{\{\textit{student life, employability}\}}$, and $r_{\{\textit{student life, academic reputation}\}}$, as 1.4834, 1.5850, and 1.5, respectively. Here, we note that Shannon entropy scores $r_{\{\textit{student life, academic reputation}\}}$ over $r_{\{\textit{teaching quality, student life}\}}$, placing the priority of rankings with two-way ties for first place below rankings containing a clear winner.

4.6.2 Example 4: Rank Aggregation on Pragma Lattices

We can repeat this process on the pragma lattice for the same university query. Suppose L's initial answer using the four pragma values selected by the user C' is MIT. Before searching for counterfactuals, we build a proxy model using rank aggregation. Similar to the previous criterion lattice example, L generates a ranked list for each individual pragma value. To estimate the rankings of the size-2 nodes, such as {{2026}, {U.S.}, {}}, we aggregate the rankings of the size-1 nodes, {{2026}, {}, {}} and {{}, {U.S.}, {}}. As a result, we are left with six aggregated rankings, one for each size-2 node. We calculate the softmax entropy as before, converting the scores into probabilities using softmax and then calculating the entropy. See Figure 4.2 for the generated rankings and softmax entropy scores. The resulting ordered list of softmax entropy scores is: {1.5850, 1.5850, 1.2009, 1.1406, 1.1406, 0.6363}. We can then apply the max jump heuristic to identify the uncertain nodes. The largest difference between scores is 0.5043 which occurs between the last two scores in the list. This effectively separates the ranking $r_{\{\{2026\},\{\text{U.K}\},\{\}\}}$ which has a clear winner (Oxford) from the other rankings that have close or exact ties for first.

If we were to instead use Shannon entropy, the resulting ordered list of entropy scores would be: {1.5850, 1.5850, 1.5546, 1.4834, 1.4834, 1.4591}. The largest jump between scores is 0.0712, which signals that the rankings $r_{\{\{2026\},\{\},\{\text{official ranking}\}\}}$, $r_{\{\{\},\{\text{U.K.}\},\{\text{official ranking}\}\}}$, and $r_{\{\{2026\},\{\text{U.K.}\},\{\}\}}$ are low entropy or "certain". However, the rankings $r_{\{\{2026\},\{\},\{\text{official ranking}\}\}}$ and $r_{\{\{\},\{\text{U.K.}\},\{\text{official ranking}\}\}}$ have a two-way tie for first place, and would benefit from being called by the LLM to verify the true ranking.

4.7 Graph-of-Thoughts (GoT)

While lattice traversal ensures counterfactual minimality, it can be computationally expensive. We introduce a GoT-based pruning strategy for scenarios where the inference budget B is highly constrained and approximate counterfactuals are sufficient. Instead of iteratively probing the lattice, `CORAL` prompts the backbone LLM L to generate a structured reasoning graph in a single inference call along with p. The GoT decomposes the reasoning process into the following logical dependencies:

Intermediate Questions: The original query q is decomposed into sub-questions. For example, "What are the global rankings?" could be a plausible intermediate question for our running university use case. These intermediate nodes are indicated in blue.

Concepts: Depending on the type of explanation, query-side or document-side, each intermediate question identifies specific criteria or pragmas that are used to answer the intermediate question. On the query-side, an intermediate question such as "What is the definition of best?" produces criteria *academic reputation* and *employability*. On the document-side, an intermediate question such as "Which university has the best student life?" produces perspective pragma values *official rankings* and *student account*. These nodes are indicated in grey.

Answers: These extracted concepts are mapped to candidate answers, where edges represent a concept's support for the answer. The original answer is highlighted in yellow, while alternative answers are highlighted in green.

The prompt used to generate the GoT includes a template for a graph-structured response and a one-shot example to make extracting the criteria (as C) straightfor-

ward. We define an approximate counterfactual as the minimal change to the set of concepts supporting the original answer, obtained by adding or removing concepts. To do this, we focus on the last two layers in the graph containing the concepts and the possible answers. These two layers form a bipartite graph. Each edge connects a concept node to an answer node when the concept is on the reasoning path leading to the answer. With the extracted bipartite graph, we project onto the answer layer, where the edge weight between two answer nodes is the symmetric difference between their supporting concept subsets. As we are only interested in counterfactuals, the symmetric difference only needs to be calculated for pairs that include the initial answer (yellow node). A promising counterfactual is identified by selecting the answer node connected to the original answer by the lowest weighted edge, since a lower weight indicates fewer concept changes are needed. An additional LLM call is made to L to verify that the updated concept subset produces a different answer.

4.7.1 Example 5: Graph of Thoughts on Criteria

Returning to the university example, the LLM L is prompted to generate a GoT for the query using the provided documents. The resulting GoT is shown in Figure 4.3a. From the GoT, a bipartite graph is extracted to isolate the relationship between the criteria and the answers, as shown in Figure 4.3b.

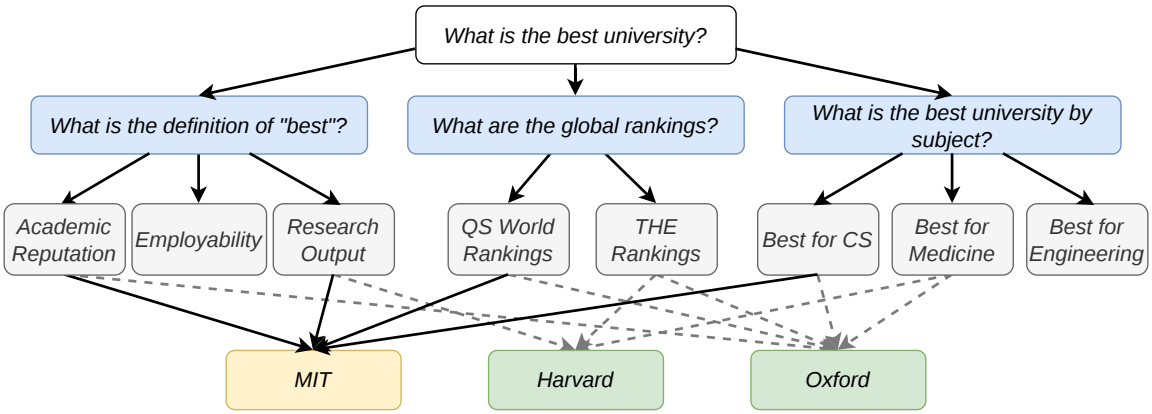
Assume the LLM's initial answer MIT is supported by the criteria set A, {*academic reputation*, *research output*, *QS world rankings*, *best for CS*}. The alternative answer Oxford is supported by the criteria set B, {*academic reputation*, *QS world rankings*, *THE rankings*, *best for CS*, *best for medicine*}. The symmetric difference between the answers MIT and Oxford is calculated as 3. That is, the number of edits required to

transform concept set A to B is 3: remove *research output*, then add *THE rankings* and *best for medicine* to A. Similarly, the symmetric difference between MIT and Harvard is calculated as 5. See Figure 4.3c for the resulting projected graph. We call the LLM to answer the question based on the concept set B to verify that it surfaces an alternative answer. If not, we move on to the next candidate set of concepts with the next lowest symmetric difference.

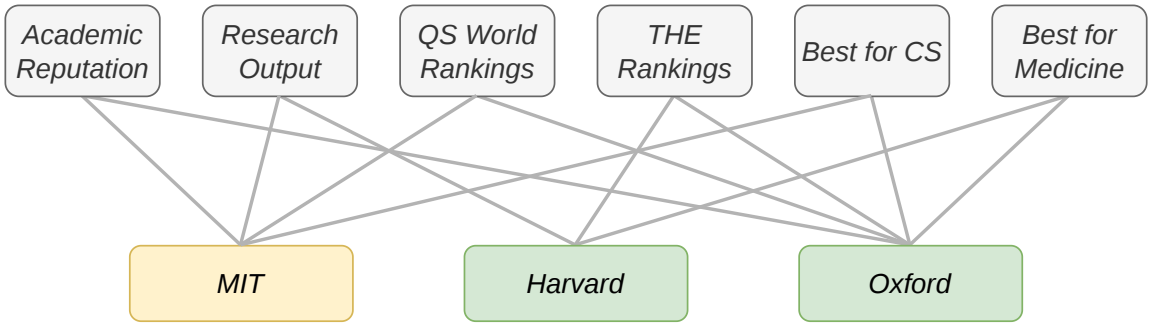
Algorithm 2 Proxy Model: Rank Aggregation

Require: LLM L, user's query q, retrieved documents D, candidate answer set A, concept set C, budget B, scorer Ψ, selection policy Φ
Ensure: Scores σ, cached results K

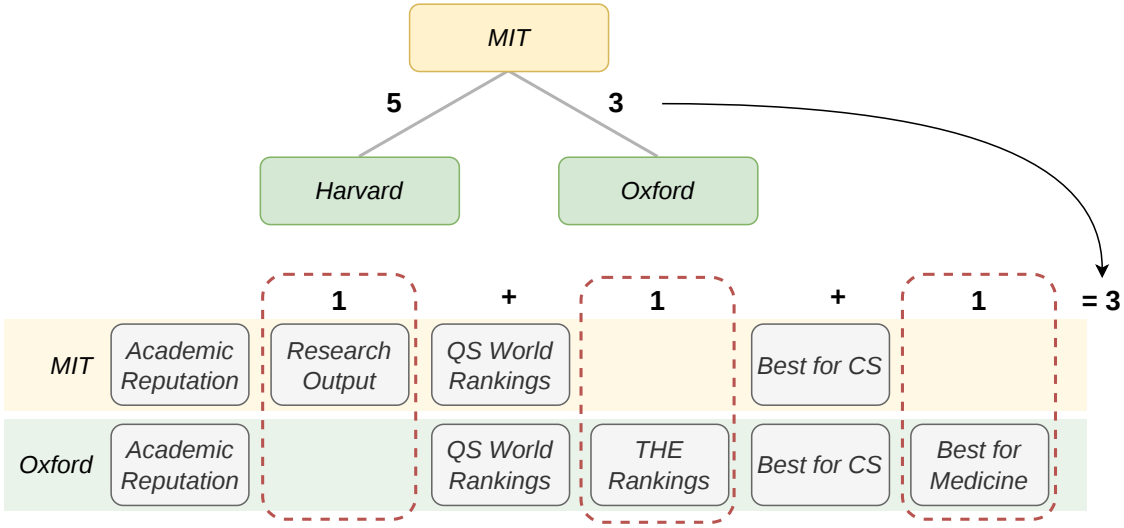
$R_{agg} \leftarrow \emptyset$ ▷ Initialize rankings obtained using $\mathscr{A}$
$R_{eval} \leftarrow \emptyset$ ▷ Initialize rankings obtained using L
$b \leftarrow 0$ ▷ Initialize budget counter
for $c \in C$ **do** ▷ Get rankings for each concept
    $r_c \leftarrow L(q, D, A, c)$ ▷ Call L for ranking
    $R_{eval}[c] \leftarrow r_c$ ▷ Store L's ranking
end for
for $\ell = 2$ **to** $|C|$ **do** ▷ Iterate through lattice layers
    $N_\ell \leftarrow \{S \subseteq C \mid |S| = \ell\}$ ▷ Get size-ℓ subsets
    for $S \in N_\ell$ **do**
        $N_S \leftarrow \{S' \subset S \mid |S'| = \ell - 1\}$
        $R_S \leftarrow \emptyset$
        for $S' \in N_S$ **do** ▷ Get rankings of size $\ell - 1$
            if $S' \in R_{eval}$ **then**
                $R_S \leftarrow R_S \cup \{R_{eval}[S']\}$
            else
                $R_S \leftarrow R_S \cup \{R_{agg}[S']\}$
            end if
        end for
        $r_S \leftarrow \mathscr{A}(R_S)$ ▷ Estimate ranking via aggregation
        $R_{agg}[S] \leftarrow r_S$ ▷ Store aggregated ranking
        $\sigma[S] \leftarrow \Psi(r_S)$ ▷ Assign score to ranking
    end for
    $N_{eval} \leftarrow \Phi(N_\ell, \sigma)$ ▷ Apply selection policy
    $N_{eval} \leftarrow \textsc{sort}(N_{eval})$
    while $b < B$ **and** $N_{eval} \neq \emptyset$ **do**
        $S^* \leftarrow \textsc{pop_highest}(N_{eval})$
        $R_{eval}[S^*] \leftarrow L(q, D, A, S^*)$ ▷ Evaluate S^* via L
        $b \leftarrow b + 1$ ▷ Increment budget counter
    end while
end for
$K \leftarrow \{(S, R_{eval}[S][0]) \mid S \in R_{eval}\}$ ▷ Top answer per
return σ, K



(a) An example of an LLM-generated GoT.



(b) An example of an extracted bigraph from a GoT.



(c) An example of a projection from a bigraph.

Figure 4.3: An example of counterfactual search using GoT.

Chapter 5

Experimental Evaluation

This section evaluates CORAL through quantitative experiments and presents use cases to display the usefulness of CORAL as a tool. Section 5.1 contains the quantitative experiments used to measure the effectiveness of the proposed pruning strategies. Dataset curation is described in Section 5.1.1, the methods evaluated for comparison are detailed in Section 5.1.2, and the metrics we use to evaluate the system in Section 5.1.3. The setup for ablations is described in Section 5.1.4, and Section 5.1.5 discusses the key insights from these quantitative evaluations. Section 5.2 describes several qualitative experiments which cover use cases that display the quality of our tool in helping explain RAG LLMs to users.

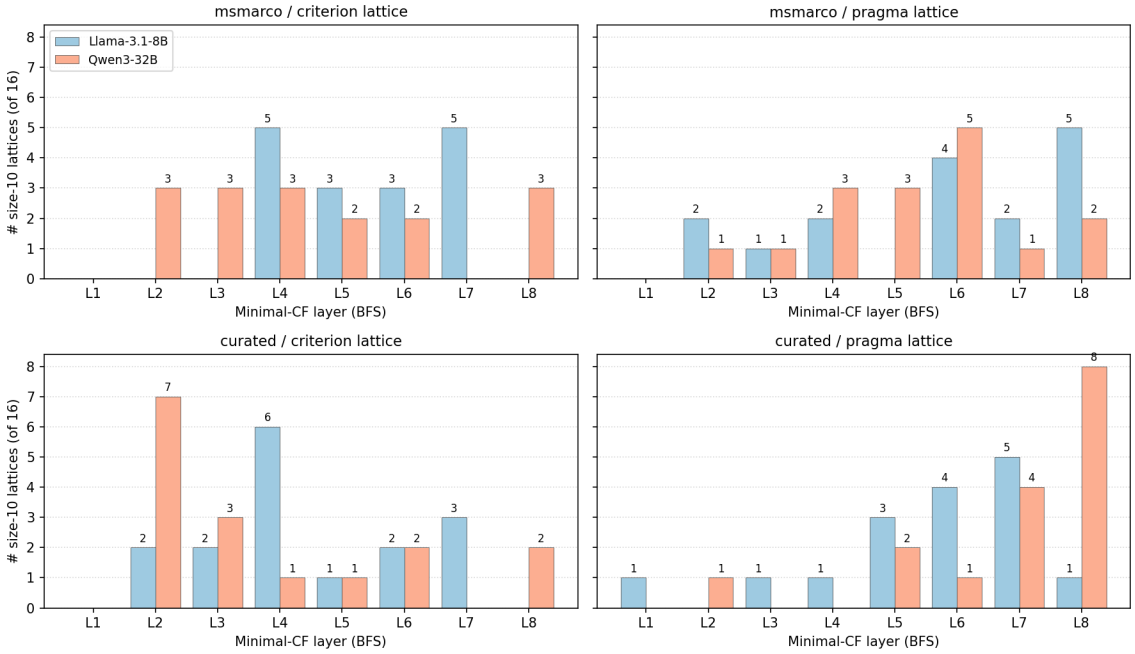


Figure 5.1: Distribution of minimal counterfactual layer for the size-10 lattices, by dataset and LLM.

5.1 Quantitative Evaluation

5.1.1 Datasets

As there are limited datasets for our open-ended question answering task, we constructed two datasets to evaluate `CORAL`. The first dataset was formed using queries and sources from MSMARCO [34], a large-scale dataset of closed-book questions and the related source snippets retrieved using the Bing search engine. We filtered the queries to find questions that could be slightly modified to be open-ended in order to retain the sources as the retrieved sources in `CORAL`. The second dataset was generated using Claude Sonnet [35], prompting it to produce plausible open-ended questions and supporting sources. In our results, we refer to the MSMARCO dataset as `msmarco`

and the dataset formed by Sonnet as `curated`.

For both datasets, we start with roughly 60 query/source pairs, each associated with concept sets of size 10 for both criteria and pragmas. We form explanations for two open-source LLMs: Meta's Llama 3.1 (`Llama-3.1-8B-Instruct`) [36] and Alibaba's Qwen 3 (`Qwen3-32B`) [37] to observe how the performance of the pruning strategies changes for larger models. For reproducibility, we set both LLMs' temperature to 0 and save the fully populated lattice containing 2^{10} nodes so that the answer space remains the same.

To evaluate our pruning methods, we focused on finding non-trivial counterfactuals where a brute-force method will need to consume several LLM calls. We filter out lattices that require fewer than two removals to find a counterfactual. The distribution of where the minimal top-down counterfactual lies is shown in Figure 5.1. From the remaining size-10 lattices, we select 16 criterion lattices and 16 pragma lattices. Smaller lattices, ranging from sizes 3 to 9, are sampled from these 16 size-10 lattices, with 20 sublattices sampled per size. As a result, we obtain 156 criterion lattices and 156 pragma lattices per LLM.

5.1.2 Search Strategies

We evaluate the following pruning and search strategies.

1. **Breadth-First Search (BFS)** ***(Baseline)***: BFS begins with the initial node and explores the lattice layer by layer. Within each layer, nodes are randomly selected for counterfactual evaluation. BFS only advances to the next layer after exhausting all possible concept combinations in the current layer.
2. **Simple Rank Aggregation (SRA)** ***(Baseline)***: SRA is the pruning strategy

introduced in the demonstration paper for `CORAL`. The strategy uses rank aggregation as a heuristic to guide search and prune nodes. The search begins by evaluating all nodes in the first layer (size-1 concept combinations) and obtaining the LLM's answer rankings. These initial rankings are combined via rank aggregation to predict the answer for a given concept combination. Nodes are prioritized within each layer by their *displacement score*, the rank of the original answer within the ranked list, and nodes with a displacement score of 0 are pruned.

3. **Linear Surrogate Model (LSM)** ***(Baseline)***: This strategy trains a sparse linear surrogate model using LASSO [38] by randomly ablating parts of the input prompt and measuring the LLM's probability of generating the original answer. We introduce LSM as a baseline comparison against our proposed method because it also builds a proxy model using a budget. The weights of the linear model can be interpreted as attribution scores, indicating the most important parts of the input. We use the surrogate model as a predictor to estimate the logit-scaled probability that a node's concept set will result in generating the original answer. The logit-scaled probability spans from $(-\infty, \infty)$. Nodes are prioritized for evaluation in ascending order of probability, with nodes with logit-scaled probabilities over 0 pruned. We evaluate LSM on three fixed budgets: 4, 8, and 16.
4. **Entropy-guided Rank Aggregation (ERA)**: This is the strategy we introduced in Section 4.5.1. For the reported results, we apply softmax to the aggregated ranking scores before calculating entropy. To compare directly against LSM, we evaluate ERA on budgets 4, 8, and 16.

5.1.3 Metrics

To evaluate the effectiveness of our pruning strategies, we use the following metrics:

- **Efficiency.** Since the primary cost of lattice exploration stems from invoking LLM calls, we measure efficiency as the number of LLM calls used to find the counterfactual.

- **Accuracy.** Counterfactuals found using pruning methods are not guaranteed to be minimal. To measure how well the pruning methods preserve counterfactual quality, we evaluate the proportion of test cases where the pruning strategy successfully discovers the minimal counterfactual found using BFS over all test cases.

- **Distance.** We also measure the counterfactual distance, the difference in concept set *size* between the counterfactual found by the pruning strategy and the minimal counterfactual found by BFS, to evaluate how close the pruning strategy gets when it fails to find the exact minimal counterfactual. For distance, we only report the cases where the strategy fails to find the minimal counterfactual.

5.1.4 Ablations

We evaluate the effect of the selected budget on the performance of ERA by ablating the budget. See the results in Figures 5.5 and 5.6. We also examine the usefulness of applying the softmax normalization function before calculating entropy. As explained in Section 4.5.1, the benefit of using softmax comes from its ability to distinguish and prioritize ties for first place. From the datasets described in Section 5.1.1, we

Table 5.1: Performance of ERA (b=4) against exhaustive BFS.

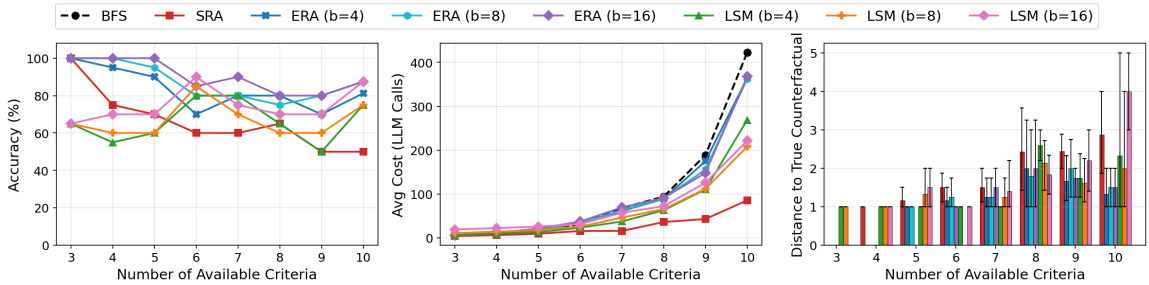
Model	Dataset	BFS	ERA	Saved	Reduction	ERA Acc. (%)
`Llama3.1-8B`	msmarco criteria	96.71	75.86	20.85	21.6%	97.44 ± 2.48
	curated criteria	139.96	64.97	74.98	53.6%	98.08 ± 2.16
	msmarco pragmas	95.96	45.44	50.51	52.6%	92.95 ± 4.02
	curated pragmas	82.73	34.73	48.00	58.0%	90.67 ± 4.66
`Qwen3-32B`	msmarco criteria	151.81	59.61	92.21	60.7%	97.44 ± 2.48
	curated criteria	160.27	74.08	86.19	53.8%	98.72 ± 1.77
	msmarco pragmas	102.92	49.24	53.68	52.2%	96.15 ± 3.02
	curated pragmas	58.91	29.95	28.96	49.2%	90.38 ± 4.63

retain lattices that contain ties for first in the aggregated rankings and measure the performance of the entropy variants. The results are shown in Figure 5.4.

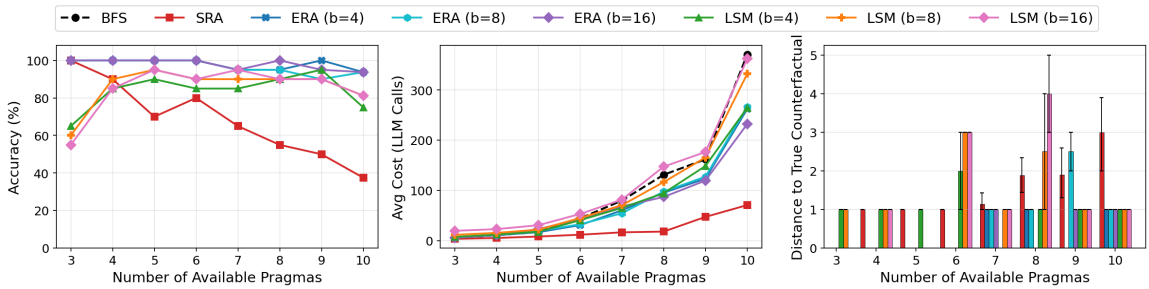
5.1.5 Analysis

This section discusses the insights from the experimental results. Figures 5.2 and 5.3 display the overall performance of the search strategies on accuracy, cost, and distance. Table 5.1 reports the cost and accuracy of ERA with a fixed budget of 4 against the exhaustive BFS baseline.

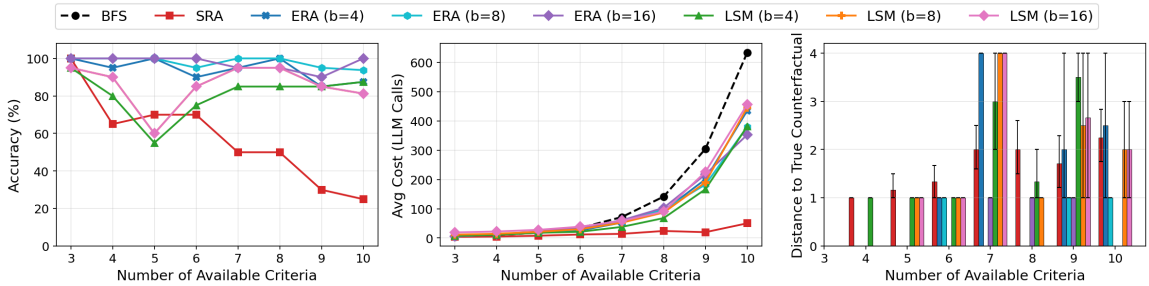
ERA halves cost while counterfactuals remain near-minimal. As seen in Table 5.1, on seven of the eight settings, ERA with a budget of four reduces between 49% and 61% of BFS's calls while recovering around 90% of the minimal counterfactuals. The sole exception is on `Llama3.1-8B`'s lattices on the msmarco criteria dataset, achieving a reduction of 21.6%. Compared to the number of calls needed by the `Qwen3-32B` model, this outcome is expected as BFS is already inexpensive (96.7 calls on average). Given the number of LLM calls avoided using ERA's pruning



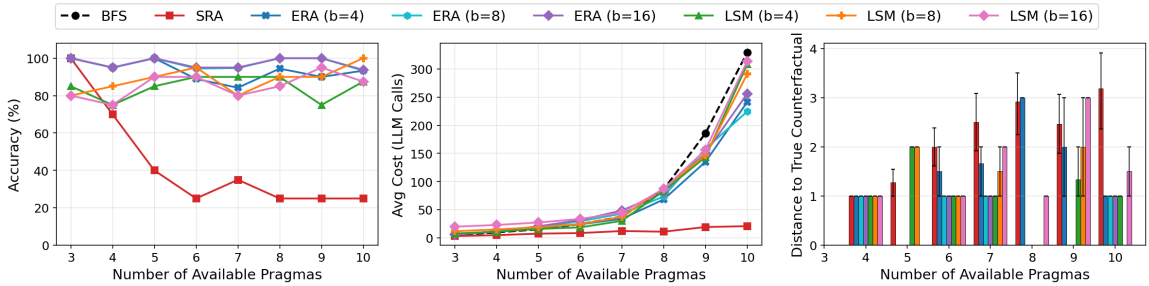
(a) Performance of pruning methods on criterion lattices from the MSMARCO dataset.



(b) Performance of pruning methods on pragma lattices from the MSMARCO dataset.

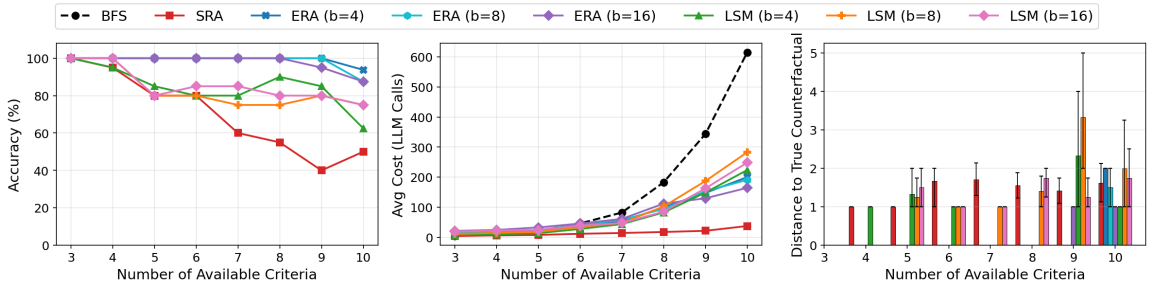


(c) Performance of pruning methods on criterion lattices from the curated dataset.

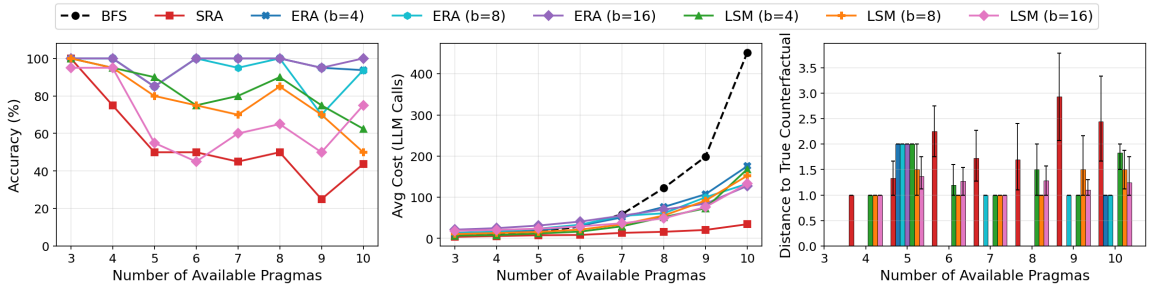


(d) Performance of pruning methods on pragma lattices from the curated dataset.

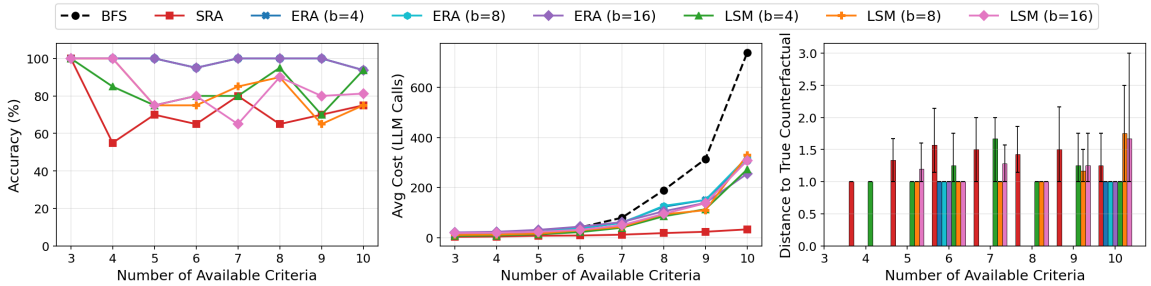
Figure 5.2: Overall performance of pruning methods on accuracy, cost, and distance across the datasets using `Llama3.1-8B`.



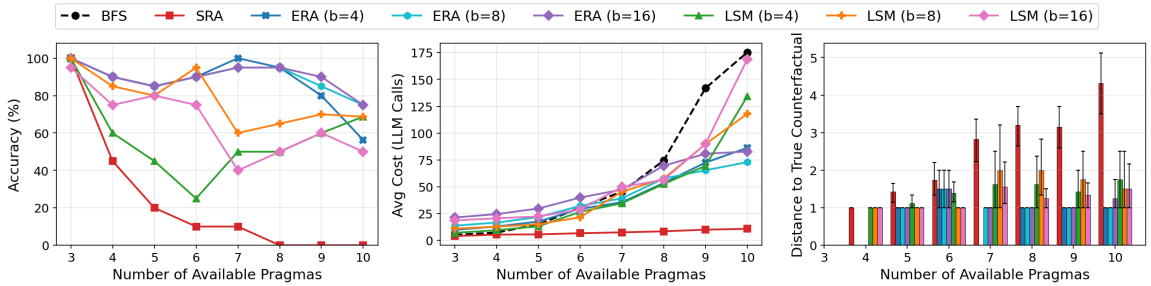
(a) Performance of pruning methods on criterion lattices from the MSMARCO dataset.



(b) Performance of pruning methods on pragma lattices from the MSMARCO dataset.



(c) Performance of pruning methods on criterion lattices from the curated dataset.



(d) Performance of pruning methods on pragma lattices from the curated dataset.

Figure 5.3: Overall performance of pruning methods on accuracy, cost, and distance across the datasets using `Qwen3-32B`.

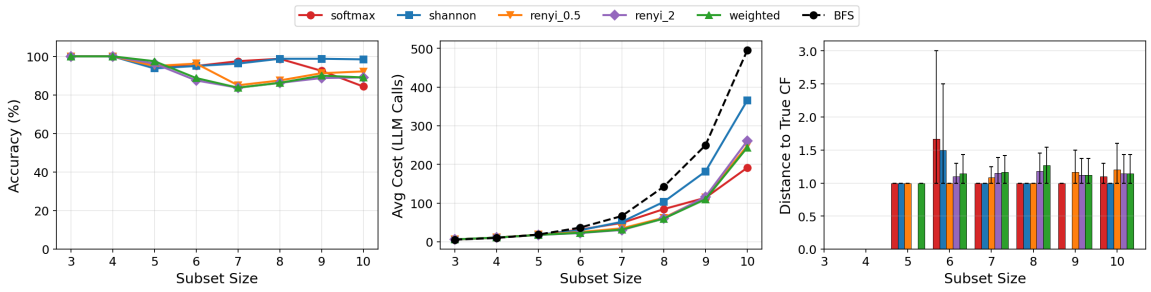


Figure 5.4: Overall performance of ERA's entropy variants on accuracy, cost, and distance.

mechanism, the near-counterfactuals provide a favourable trade-off between minimality and cost.

ERA outperforms both SRA and LSM. SRA achieves the lowest cost of any method, but it also has the lowest accuracy and greatest distance away from the minimal counterfactual. SRA prunes a node whenever the original answer stays ranked first in its aggregated ranking. This indicates that SRA's pruning is too aggressive, causing it to frequently miss the minimal counterfactuals. ERA adds a second condition, pruning only when the ranking's entropy is also low. ERA therefore retains and evaluates high-entropy nodes where SRA would have pruned, which is where SRA tends to miss counterfactuals.

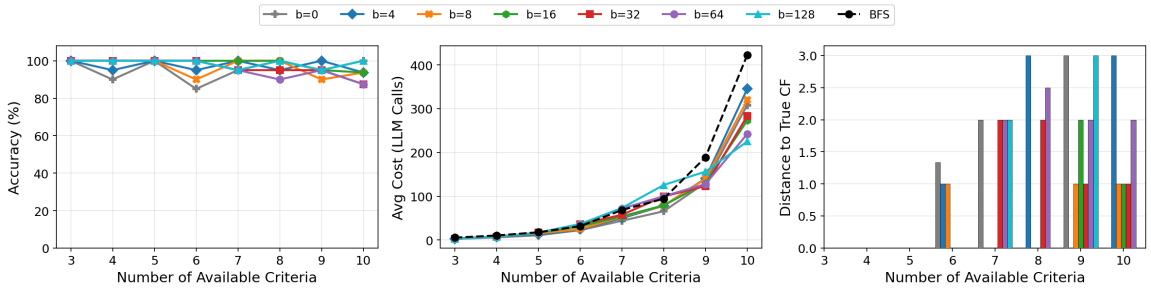
Unsurprisingly, LSM and ERA, the two proxy models with access to a budget, performed better than SRA. The gap in cost between BFS and the proxy models is larger on the lattices generated using `Qwen3-32B` than on `Llama3.1-8B`. Since Qwen is the larger model, its LLM calls are more expensive to run, making this saving especially valuable. The non-minimal counterfactuals found by ERA are also close to minimal, averaging around one removal away on `Qwen3-32B`. This means that ERA

found the counterfactual one layer below the true minimal counterfactual. Given the LLM calls saved by using these pruning strategies, a near minimal counterfactual is arguably worth the cost reduction.

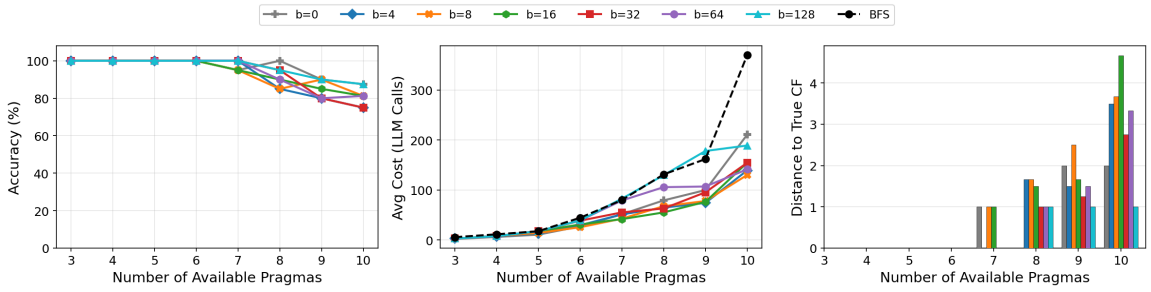
We also observe that ERA's performance on accuracy is typically better than LSM, especially for smaller lattices. Recall that LSM spends its budget on random concept subsets to calculate the logit-scaled probability. Unlike ERA, these ablations results are not cached, giving ERA a slight advantage.

Finding counterfactuals in pragma lattices is challenging. The query-side and document-side lattices are imbalanced across both LLMs, as shown in Figure 5.1. Counterfactuals in pragma lattices typically require removing only a few pragma values, yielding fewer LLM calls on average than their criterion counterparts. This explains SRA's rapid decline in accuracy on pragma lattices. SRA's aggressive pruning strategy misses counterfactuals reachable by removing just a few concepts. The distance plots confirm this, with SRA showing the largest distance to the true counterfactual across strategies. On Qwen pragma lattices (Figure 5.3), LSM's accuracy also drops sharply with no corresponding cost reduction. On the curated pragma dataset, its cost and average distance to the counterfactual both exceeds ERA's.

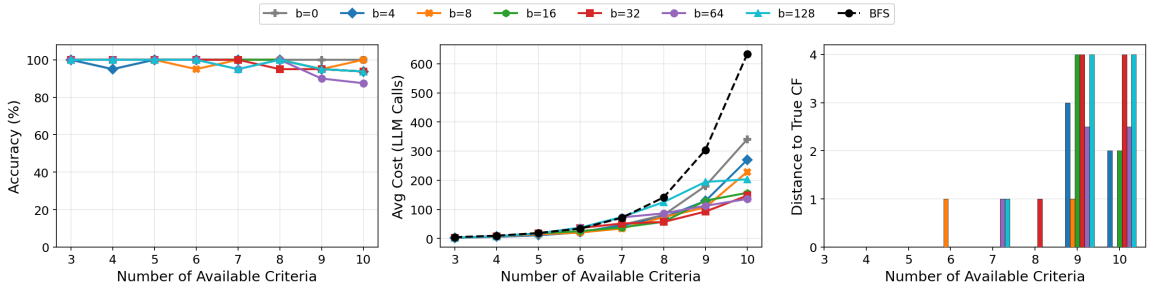
Budget behaviour. We observe that ERA and LSM behave differently as budget increases. On the smallest tested budget of 4, ERA performs well across both accuracy and cost as seen in Figures 5.2 and 5.3. In cases where the counterfactuals identified by ERA miss the true counterfactual by a large margin, like in Figures 5.2c and 5.2d, increasing the budget largely reduces the distance issue. For ERA, increas-



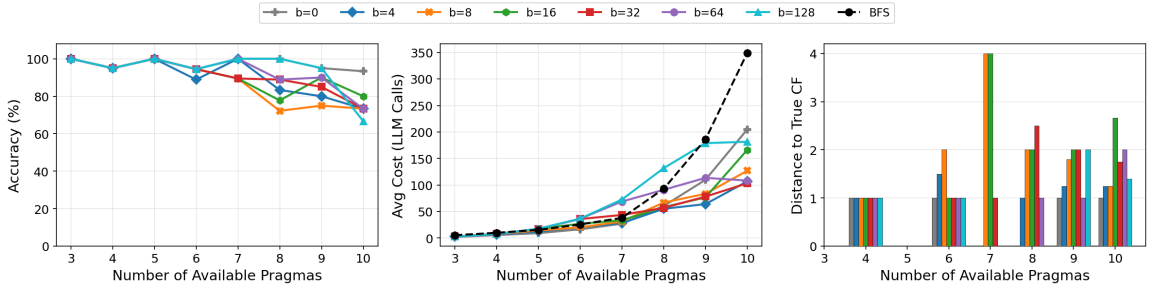
(a) ERA budget ablation on MSMARCO criterion lattices.



(b) ERA budget ablation on MSMARCO pragma lattices.

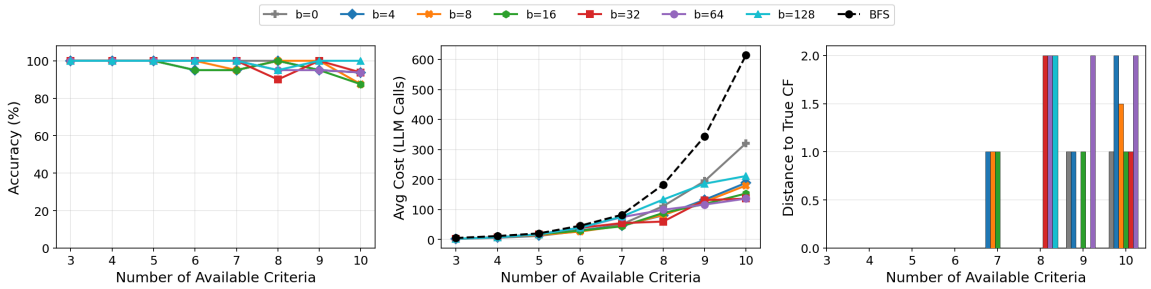


(c) ERA budget ablation on curated criterion lattices.

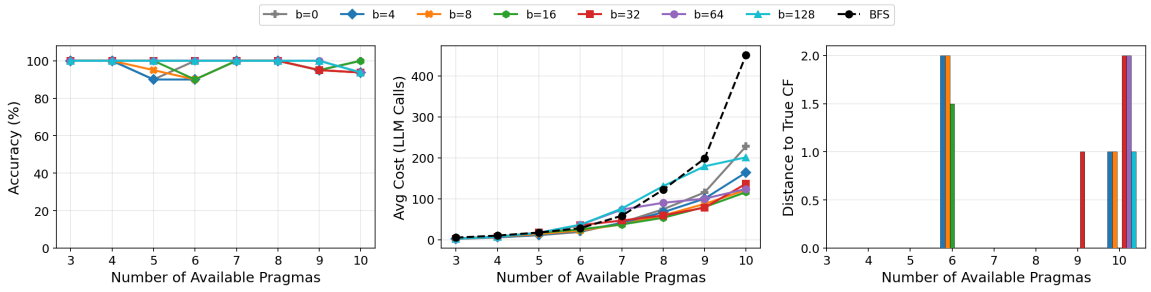


(d) ERA budget ablation on curated pragma lattices.

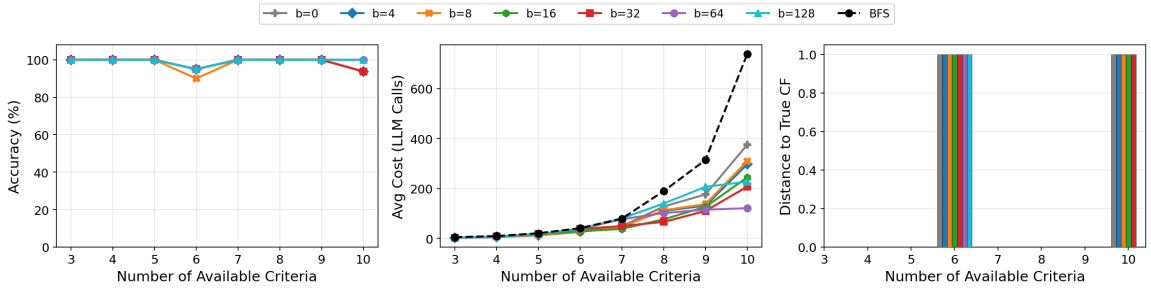
Figure 5.5: ERA budget ablation for fixed budgets {4, 8, 16, 32, 64, 128} on accuracy, cost, and distance across the datasets using `Llama3.1-8B`.



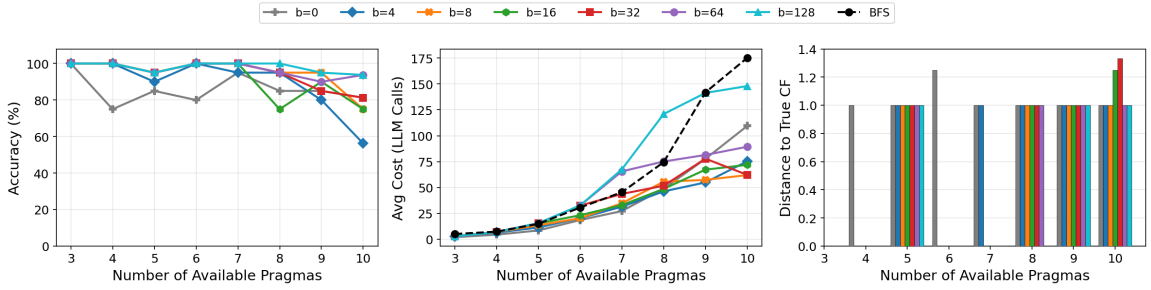
(a) ERA budget ablation on MSMARCO criterion lattices.



(b) ERA budget ablation on MSMARCO pragma lattices.



(c) ERA budget ablation on curated criterion lattices.



(d) ERA budget ablation on curated pragma lattices.

Figure 5.6: ERA budget ablation for fixed budgets {4, 8, 16, 32, 64, 128} on accuracy, cost, and distance across the datasets using `Qwen3-32B`.

ing the budget generally results in better average accuracy and decrease in distance to the true counterfactual. In some cases, allocating a larger budget (e.g., 16) for ERA results in a reduction in average cost compared to smaller budgets (e.g., 4). However, increasing the budget on LSM produces mixed results. Supplying LSM with a higher budget to train the surrogate model does not necessarily produce better accuracy. This is best seen in Figures 5.3b and 5.3d where LSM under different budgets swings back and forth between accuracy. We also note that there is also randomness involved in training LSM's proxy model. Recall that LSM trains a linear model using concept ablations and the associated logit-scaled probability. The concept ablations themselves are randomly sampled, with each concept having a 50% chance of being selected for the ablation.

Figures 5.5 and 5.6 show ERA's behaviour on different fixed budgets. We first observe that the accuracy is already quite high at low budget. ERA is already achieving over 95% accuracy when ERA uses a fixed budget of 4 in almost every setting. The gains from allocating more calls to the budget are diminished because there is little to improve upon. Additionally, higher budgets tend to overspend. Since ERA allocates calls to high-entropy nodes first, once the uncertain nodes are resolved, the remaining budget is spent on progressively less uncertain nodes. As the budget is spent up front to build the proxy before the search begins, a large budget can even exceed BFS's cost. See ERA's cost when budget is set to 128 in Figure 5.5. While BFS can terminate early after finding a counterfactual, the process to build the proxy model already partially traversed the lattice from the bottom-up.

5.2 Qualitative Experiments

In this section, we describe two use cases on product comparison and source analysis to show how users can benefit from the explanations produced by `CORAL`.

5.2.1 Product Comparison

Consider a Canadian user who wants to explore different electric vehicle (EV) options from China following the recent drop in EV tariffs. They pose the following query to the RAG LLM: "What is the best Chinese EV?" The question is broad, lacks specific criteria, and has multiple answers that could be considered correct. Given sources containing opinions from Reddit and car blogs, the LLM identifies the NIO ET7 as the top choice. However, the user remains skeptical and wonders why the LLM recommended the NIO ET7 over more globally dominant models like the BYD Seal or Geely EX5, or the viral Xiaomi SU7.

The user decides to generate query-side explanations first to understand how different criteria can sway the answer and personalize their query to their own needs. The user first selects the GoT mode to produce a quick explanation. NIO ET7 is supported by the criteria: *interior quality*, *comfort*, *charging tech*, and *range*. The closest counterfactual is found to be the Xiaomi SU7, which surfaces when *charging tech* is removed and *performance* is added as criteria for answering the query.

To gain more control over criteria selection, the user switches to lattice mode. `CORAL` retrieves the following six criteria from the sources using a cheaper, external LLM: *driving range*, *performance*, *technology*, *comfort*, *charging*, and *value*. Based on the provided criteria, the user deselects *performance* and *driving range*, as they are more interested in an affordable, comfortable car to drive in the city. With these

four criteria, the RAG LLM determines that the Xiaomi SU7 is the best car. A top-down counterfactual reveals that the NIO ET7 emerges as the answer once *value* is removed as a criterion. Upon further investigation, the user finds that the price range for ET7 is significantly higher than that of the other three alternatives. Further counterfactual search reveals that the cheapest car of the four, BYD Seal, emerges as the answer when comfort and technology are removed from the criteria list. `CORAL` allows users to explore trade-offs among their options, personalize their search, and make informed decisions.

5.2.2 Source Analysis

Consider a user interested in learning how public support for the Trump administration has shifted ahead of the 2026 midterm elections. The user asks the LLM: "Does a majority of the American public approve of Trump's performance on jobs and the economy?" Based on information from polling platforms like CNN's *Poll of Polls: Trump approval rating* and The New York Times' *President Trump's Approval Rating: Latest Polls*, the LLM responds with "disapprove".

The user is aware that the economy was one of the major pillars of Trump's platform during his presidential campaign. Curious about how his approval rating has decreased over time and across different states, the user uses `CORAL` to identify source-sided counterfactual explanations. `CORAL` generates pragma values along the temporal and geographic axes. On the temporal side, `CORAL` identifies key events from the sources:

- March 4, 2025. Trump announces a 25% U.S. tariff on most goods from Canada and Mexico.

- April 2, 2025 (Liberation Day). Trump announces a sweeping reciprocal tariff on all countries.
- July 5, 2025 (One Big Beautiful Bill Act). Trump signed a major tax-and-spending package that included tax cuts, reduced corporate taxes, and spending cuts to social programs.
- November 2025 (Rising Unemployment). After a year of stagnation, unemployment rose to 4.4% (highest since September 2021), despite reported job growth.
- February 20, 2026 (Supreme Court Ruling). SCOTUS ruled the Trump administration's tariffs illegal.
- March 6, 2026. The BLS reported a net loss of 92,000 jobs in February.

On the geographic axis, `CORAL` identifies the following regions of interest: *California*, *Michigan*, *Ohio*, and *U.S.* The user requests a bottom-up counterfactual, which reveals that the LLM responds with "approve" when considering sources published before the initial tariff shock of March 4, 2025. Furthermore, `CORAL` reveals that in Ohio, the LLM responds with "approve" when considering sources from Ohio and published before November 2025. However, the LLM responds with "disapprove" when considering Ohio and sources published after the November 2025 unemployment surge.

Separately, the user is interested in how different sources affect the LLM's answer. When a post from the social media platform Truth Social is included, the LLM's answer shifts towards "approve". `CORAL` identifies pivotal events where answers flip, analyzes shifts in sentiment across regions, and reveals the influence of source reliability.

Chapter 6

Conclusions

6.1 Summary

In this work, we introduced CORAL, a novel framework for explaining the outputs of RAG LLMs by organizing the answer space as a lattice of concepts. While existing XAI methods for RAG systems form explanations with respect to the retrieved sources, CORAL forms explanations based on the inferred user intent and source interpretations. CORAL uses lattices to represent the answer space to provide users with counterfactual explanations that are understandable.

Populating the entire lattice to find the counterfactual requires one LLM call per node, so CORAL introduces three pruning strategies to reduce the number of calls. LLM-aided concept selection first reduces the candidate concept set before the search begins, immediately shrinking the search space. Entropy-guided rank aggregation (ERA) builds a proxy model from answer rankings based on singleton concept nodes, and applies entropy to identify suitable nodes to evaluate. Graph-of-thoughts (GoT) reasoning generates an approximate counterfactual in a single LLM call by extracting

a bipartite graph over concept and answers from the reasoning graph.

The quantitative evaluations compared ERA against three baselines, breadth-first search (BFS), simple rank aggregation (SRA) and linear surrogate models (LSM). As shown in our experimental results, ERA outperformed the three baselines in terms of cost while maintaining high accuracy. The qualitative evaluation demonstrated how users can benefit from using `CORAL` through product analysis and source analysis use cases. These use cases showed how users can use counterfactual explanations to personalize an LLM's answer and understand how answers change according to the types of sources retrieved.

6.2 Future Work

The work done in `CORAL` is an initial step to understanding RAG LLMs. This section outlines several directions for future work.

Concept Hierarchies. One of the key challenges with generated concept-based explanations is determining the set of concepts to use. An open-ended question can have many relevant concepts, and it is not always clear how many should be included for the user. In our current implementation of `CORAL`, the LLM is simply prompted to generate up to ten relevant concepts. The choice of ten concepts was arbitrary, and was intended to keep the overall search space small so we could generate our datasets. In addition, we claim that providing too many concepts makes the explanation and search space incomprehensible. Inspired by `HEX` [39], one possible direction is to organize these concepts into a hierarchy that groups similar concepts together. By introducing a concept hierarchy, we allow the user to drill down into concepts they care about or roll an explanation up to a higher-level summary. For example, an

explanation suited for a doctor may be too technical for a patient. Consider a use case where an LLM is asked to diagnose a patient and is given the patient's medical history and doctor's notes as supporting sources. Suppose the criteria *bloodwork results*, *family history*, *demographics*, and *patient symptoms* are extracted. With all four criteria considered, the LLM responds with "type 2 diabetes" as the most plausible diagnosis. A top-down counterfactual on a rolled-up lattice reveals that removing *bloodwork results* produces a different diagnosis, "prediabetes". Expanding the *bloodwork results* node reveals its sub-criteria: *HbA1c*, *fasting glucose*, and *fasting insulin*. On this drilled-down lattice, the counterfactual reveals that removing *HbA1c* and *fasting glucose* flips the answer to "prediabetes", indicating those two results drove the original diagnosis.

Scaling to larger LLMs. This work evaluated `CORAL` on `Llama3.1-8B` and `Qwen3-32B`, two small LLMs, due to computational and time constraints. Populating the full lattice containing 10 concepts requires 1,024 LLM calls per test case. Conducting further evaluation on LSM for comparison requires additional LLM calls to train the proxy model. Future work should evaluate how the pruning strategies perform on larger, open-source models.

Weighing concepts. `CORAL` currently treats all concepts as equally important, giving each concept equal weight when aggregating rankings. A user may instead want to prioritize one concept over the other. Going back to the university example, a user might weight the importance of *student life* as 50% and teaching quality as 30%. The LLM itself may also implicitly weigh concepts unequally. When asked for the best migraine treatment, the LLM may weigh and rely on scientific sources more heavily compared to forum comments.

Explaining rankings. This thesis was focused on the open-ended question answering use case where the LLM returns a single answer. ERA already prompts the LLM for a ranking of answers to build its proxy model, so a natural extension would be to explain the full ranking of answers instead of the top answer. A counterfactual could then be the minimal change to the prompt that moves an answer out or into the top k ranks.

Copyright

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of York University's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Bibliography

[1] Z. Ji et al., "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, 2023. DOI: 10.1145/3571730.

[2] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, vol. 33, Vancouver, BC, Canada, 2020, pp. 9459–9474.

[3] Z. Chen et al., *A survey on large language models for critical societal domains: Finance, healthcare, and law*, 2024. arXiv: 2405.01769 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2405.01769.

[4] M. Turpin, J. Michael, E. Perez, and S. R. Bowman, "Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, New Orleans, LA, USA, 2023, pp. 74 952–74 965.

[5] S. Wachter, B. Mittelstadt, and C. Russell, *Counterfactual explanations without opening the black box: Automated decisions and the GDPR*, 2018. arXiv: 1711.00399 [cs.AI]. [Online]. Available: https://arxiv.org/abs/1711.00399.

[6] K. Ling, J. Rorseth, P. Godfrey, L. Golab, and J. Szlichta, "CORAL: Concept-based explanations for RAG LLMs," in *Proceedings of the 42nd IEEE Interna-*

tional Conference on Data Engineering, Montreal, Canada: Institute of Electrical and Electronics Engineers, 2026, pp. 4176–4180.

[7] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995. [Online]. Available: `http://webdam.inria.fr/Alice/`.

[8] J. Szlichta, P. Godfrey, L. Golab, M. Kargar, and D. Srivastava, "Effective and complete discovery of order dependencies via set-based axiomatization," *Proceedings of the VLDB Endowment*, vol. 10, no. 7, pp. 721–732, 2017, ISSN: 2150-8097. DOI: `10.14778/3067421.3067422`. [Online]. Available: `https://doi.org/10.14778/3067421.3067422`.

[9] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[10] S. Bills et al., *Language models can explain neurons in language models*, OpenAI, 2023.

[11] E. Ameisen et al., "Circuit tracing: Revealing computational graphs in language models," *Transformer Circuits Thread*, 2025. [Online]. Available: `https://transformer-circuits.pub/2025/attribution-graphs/methods.html`.

[12] J. Rorseth, P. Godfrey, L. Golab, D. Srivastava, and J. Szlichta, "RAGE against the machine: Retrieval-augmented LLM explanations," in *Proceedings of the 42nd IEEE International Conference on Data Engineering*, 2024, pp. 5469–5472.

[13] B. Cohen-Wang, H. Shah, K. Georgiev, and A. Madry, "ContextCite: Attributing model generation to context," in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, vol. 37, Vancouver, BC, Canada, 2024, pp. 95 764–95 807.

[14] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proceedings of the 10th International Conference on World Wide Web*, Hong Kong, Hong Kong, 2001, pp. 613–622.

[15] M. Besta et al., "Graph of thoughts: Solving elaborate problems with large language models," in *Proceedings of the 38th AAAI Conference on Artificial Intelligence and 36th Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, vol. 38, 2024, pp. 17 682–17 690.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, ""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*, 2016, pp. 1135–1144.

[17] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, vol. 30, 2017, pp. 4768–4777.

[18] L. S. Shapley, "A value for n-person games," in *Classics in Game Theory*, H. W. Kuhn, Ed. Princeton, NJ: Princeton University Press, 1997, pp. 69–79. DOI: 10.1515/9781400829156-012.

[19] P.-J. Kindermans et al., "The (un)reliability of saliency methods," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer-Verlag, 2022, pp. 267–280.

[20] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 180–186. DOI: 10.1145/3375627.3375830.

[21] D. Hooshyar and Y. Yang, "Problems with SHAP and LIME in interpretable AI for education: A comparative study of post-hoc explanations and neural-symbolic rule extraction," *IEEE Access*, vol. 12, pp. 137 472–137 490, 2024. DOI: 10.1109/ACCESS.2024.3463948.

[22] S. Wiegreffe and Y. Pinter, *Attention is not not explanation*, 2019. arXiv: 1908.04626 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1908.04626.

[23] S. Abnar and W. Zuidema, *Quantifying attention flow in transformers*, 2020. arXiv: 2005.00928 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2005.00928.

[24] K. Yin and G. Neubig, *Interpreting language models with contrastive explanations*, 2022. arXiv: 2202.10419 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2202.10419.

[25] J. Enguehard, *Sequential integrated gradients: A simple but effective method for explaining language models*, 2023. arXiv: 2305.15853 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2305.15853.

[26] P. W. Koh et al., "Concept bottleneck models," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, 2020, pp. 5338–5348.

[27] C.-E. Sun, T. Oikarinen, B. Ustun, and T.-W. Weng, "Concept bottleneck large language models," in *Proceedings of the 13th International Conference on Learning Representations*, 2025, pp. 89 371–89 411.

[28] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al., "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)," in *International Conference on Machine Learning*, 2018, pp. 2668–2677.

[29] L. Akritidis, M. Alamaniotis, and P. Bozanis, "FLAGR: A flexible high-performance library for rank aggregation," *SoftwareX*, pp. 1–6, 2023.

[30] J. G. Kemeny, "Mathematics without numbers," *Daedalus*, vol. 88, no. 4, pp. 577–591, 1959.

[31] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[32] A. Rényi, "On measures of entropy and information," in *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, University of California Press, 1961, pp. 547–562.

[33] S. Guiaşu, "Weighted entropy," *Reports on Mathematical Physics*, vol. 2, no. 3, pp. 165–179, 1971.

[34] P. Bajaj et al., "MS MARCO: A human generated machine reading comprehension dataset," *arXiv preprint arXiv:1611.09268*, 2016.

[35] Anthropic, *System card: Claude Sonnet 4.6*, https://www.anthropic.com/, Feb. 2026.

[36] A. Grattafiori et al., "The Llama 3 Herd of Models," *arXiv e-prints*, arXiv:2407.21783, Jul. 2024, Art. no. arXiv:2407.21783. DOI: 10.48550/arXiv.2407.21783. arXiv: 2407.21783 [cs.AI].

[37] Q. Team, *Qwen3 technical report*, 2025. arXiv: 2505.09388 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2505.09388.

[38] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, Jan. 1996. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[39] K. Carbone, P. Godfrey, L. Golab, J. Szlichta, and R. Cohen, "HEX: OLAP-Enabled hierarchical explanations," in *Proceedings of the 42nd IEEE International Conference on Data Engineering*, Montreal, Canada: Institute of Electrical and Electronics Engineers, 2026, pp. 4193–4197.