

HARBOR: HIERARCHICAL ABDUCTION WITH BAYESIAN ORCHESTRATION FOR RELIABLE PROBABILITY INFERENCE IN LARGE LANGUAGE MODELS

Anonymous authors

Paper under double-blind review

ABSTRACT

A central challenge in large-scale decision-making under incomplete information is the estimation of reliable probabilities. Prior work has employed Large Language Models (LLMs) to generate relevant factors and provide initial, coarse-grained probability estimates. These methods typically utilize an LLM for forward abduction to generate factors, with each factor constrained to two mutually exclusive attributes. A Naïve Bayes model is then trained on combinations of these factors to provide more accurate probabilities. However, this approach often yields a sparse factor space, resulting in “unknown” predictions where the model fails to produce an output. Naïvely increasing the number of factors to densify the space not only introduces statistical noise but also violates the Naïve Bayes independence assumption, ultimately compromising the stability and reliability of the estimates. To address these limitations, we propose HARBOR, a novel inference framework that orchestrates aggregated Bayesian inference over a hierarchically structured factor space. HARBOR first constructs a dense, structured factor space through iterative generation and hierarchical clustering. It then performs context-aware mapping using retrieval and refinement operations on this hierarchy to reduce “unknown” predictions. Finally, HARBOR extends Naïve Bayes by incorporating a Causal Bayesian Network to model latent dependencies, thereby relaxing the strict independence assumption. Experiments show that HARBOR substantially reduces “unknown” predictions and yields more reliable probabilities than direct LLM baselines, achieving state-of-the-art performance with significantly reduced time and token overhead.

1 INTRODUCTION

Large language models (LLMs) are increasingly adopted in mission-critical decision-making tasks—ranging from emergency response to infrastructure planning—where accurate, interpretable, and unbiased probability estimates are essential for trustworthy outcomes (McCarthy & Hayes, 1981; Sun et al., 2024). The central challenge is to reliably estimate the conditional probability $P(O_i | C)$ for two competing hypotheses, O_1 and O_2 , given a context C . This context is typically composed of a high-level scenario, S_{cen} , and a specific downstream condition, U . Existing abductive frameworks build a factor space based on the general scenario S_{cen} , but often struggle to connect it to the specific details in U .

However, LLMs remain ill-suited for direct probability estimation for two key reasons: (1) they often produce numerical confidence scores that are both inaccurate and overconfident (Sun et al., 2024), and (2) they lack an out-of-the-box interpretable and controllable mechanism for explaining how these estimates are derived (Zhu et al., 2024). As a result, we cannot directly leverage LLMs’ decision-making capabilities in large-scale automated scenarios, motivating the need for methods to infer more reliable probabilities from LLM outputs.

Recent methods address this by combining LLM-driven abduction with Bayesian inference. A prominent approach, exemplified by BIRD (Feng et al., 2025), as shown on the left side of Figure 1, uses *forward abduction*, where an LLM first generates a factor and then its mutually exclusive

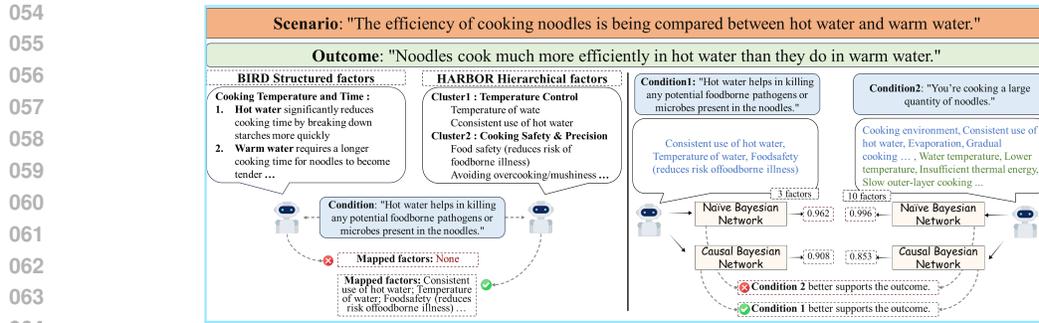


Figure 1: Illustration of current methods’ limitations in decision-making on a cooking scenario. **Left:** forward abduction yields a sparse factor space and “unknown” mappings. **Right:** when a condition activates many factors, naïve factor expansion introduces noise and violates the Naïve Bayes independence assumption; HARBOR resolves both via hierarchical factor-space construction and causal Bayesian modeling.

attributes. This method, however, often produces a sparse factor space, causing the mapping to be empty and leading to “unknown” predictions.

However, indiscriminately enlarging the factor space to boost coverage—as shown on the right of Figure 1—inevitably injects statistical noise and creates spurious dependencies among factors. This undermines the naïve Bayes conditional-independence assumption and distorts the resulting probability estimates (Zhang, 2004; Hand & Yu, 2001). Some works have introduced techniques such as attribute weighting, feature grouping, and structural extensions to mitigate this independence violation (Zaidi et al., 2013; Prabha et al., 2022; He et al., 2025). Other related work has explored different facets of combining LLMs and Bayesian reasoning (Reuter et al., 2025; Qiu et al., 2025; Sgouritsa et al., 2024), but the dual challenges of sparsity and dependency in abductive frameworks persist.

To overcome these limitations, we propose HARBOR (**H**ierarchical Abduction with **B**ayesian **O**rchestration for **R**eliable probability inference), a multi-stage framework that: (1) Iteratively builds a dense factor space from LLM-generated sentences using a **bottom-up abduction** strategy, and organizes these factors into a two-tier hierarchy via clustering and LLM-guided theming; (2) Implements a context-aware mapping pipeline using hierarchical retrieval; and (3) Constructs both a Naïve Bayes model and a Causal Bayesian Network whose parameters are initialized with LLM-elicited priors, capturing latent dependencies between factors. Results show that HARBOR substantially reduces “unknown” predictions and produces probability estimates that are significantly more calibrated and better aligned with human preferences than state-of-the-art baselines. In summary, our contributions are as follows:

- We design a multi-stage abduction pipeline that iteratively expands the factor space with high-quality factors, substantially reducing “unknown” predictions in downstream inference.
- We integrate a causal Bayesian network to model latent factor dependencies, enhancing probability calibration beyond the naïve Bayes assumption.
- We show that HARBOR achieves state-of-the-art performance on preference-based pairwise evaluation while substantially reducing inference time and token usage.

2 RELATED WORK

Decomposition-Based Reasoning Like many advanced reasoning frameworks, HARBOR decomposes complex problems into smaller components. This builds on a rich body of work, from explicitly breaking down questions into procedural steps (Wolfson et al., 2020; Madaan et al., 2021) or generating faithful reasoning chains (Tafjord et al., 2022; Zhou et al., 2022), to the now-widespread Chain-of-Thought (CoT) prompting (Wei et al., 2022). More recent structured approaches aim for greater cognitive plausibility (Yao et al., 2023; Lin et al., 2023), maintain an explicit belief graph (Kassner et al., 2023), or treat the textual reasoning chain itself as a latent variable (Hoffman

et al., 2023). While related, the latent variables in HARBOR’s Causal Bayesian Network represent abstract concepts rather than token sequences. The primary distinction of our method, however, is its *proactive* factor-space construction. Instead of decomposing reactively for each query, HARBOR employs “**bottom-up abduction**” to build a comprehensive, hierarchically structured factor space. This design mitigates the issue of factor sparsity, creating a persistent and reusable reasoning structure that reduces the generation of “unknown” predictions.

Probabilistic Inference and Uncertainty Estimations Many existing efforts aim to improve the uncertainty estimation of LLMs. Some approaches rely on direct model outputs, such as token probabilities (Ren et al., 2023) or verbalized confidence scores (Xiong et al., 2023), which are often poorly calibrated. Others employ sampling-based techniques (Kuhn et al., 2023) or fine-tune the model to amortize complex Bayesian inference (Hu et al., 2023). A common drawback of these methods is either unreliable or costly uncertainty estimation. In contrast, HARBOR integrates external probabilistic models for a more rigorous and principled calibration (Feng et al., 2025; Hou et al., 2023). Its unique contribution is the orchestration of two Bayesian models: The first is a classic Naïve Bayes classifier (Zhang, 2004) whose primary limitation is its well-known conditional independence assumption (Prabha et al., 2022; He et al., 2025). To address this, we introduce a Latent-Augmented Causal Bayesian Network (CBN) that explicitly models dependencies between factors. Within this framework, the LLM is not only used for parameter elicitation (Nafar et al., 2025) but is also employed as a dynamic causal discovery engine to define the CBN’s structure. By aggregating predictions from these models, HARBOR synthesizes their respective strengths to achieve a more reliable probability estimate.

Structured Retrieval for Grounded Reasoning Recent work in Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) has moved towards structured knowledge sources, employing graphs and hierarchies to address challenges like the “knowledge gap” between local and global context (Edge et al., 2024; Huang et al., 2025; Zhang et al., 2024). These studies collectively demonstrate the benefits of hierarchical and hybrid retrieval strategies. While these advanced methods primarily focus on indexing and structuring existing document corpora, HARBOR distinguishes itself by proactively constructing its knowledge source from the ground up. Our approach first generates a comprehensive set of reasoning factors, then organizes them into an interpretable hierarchy using established clustering techniques (Simeone et al., 2023) and LLM-guided theming (Zhang et al., 2023; Azher et al., 2024). This process yields a purpose-built structure tailored specifically for reasoning, enabling a sophisticated retrieval process founded on high-quality, contextually-aware evidence, rather than relying on a general-purpose indexed document store.

3 PRELIMINARIES

3.1 PROBLEM FORMULATION

We formulate reliable decision-making as a contextual binary inference task. Given a context C and two competing hypotheses, O_1 and O_2 , the objective is to estimate the conditional probabilities $P(O_i | C)$ for $i \in \{1, 2\}$. A calibrated estimate allows a system to robustly decide which hypothesis is more plausible. Following Feng et al. (2025), we decompose C into a high-level scenario S_{cen} and a downstream condition u , enabling a neutral reasoning space based on S_{cen} that mitigates biases from u . For completeness, we treat an “unknown” prediction as abstention: letting \mathcal{F}^* denote the factor–mapping operator (see §4.2), the system yields an unknown prediction iff $|\mathcal{F}^*(u)| = 0$ or $\max_{i \in \{1, 2\}} P(O_i | C) < \tau$ for a preset $\tau \in (0, 1)$.

3.2 ABDUCTIVE-DEDUCTIVE INFERENCE FRAMEWORK

Some prior work (Feng et al., 2025) implements a two-stage process. First, in a step of *forward abduction*, it generates a set of N discrete factors $\mathcal{F} = \{F_1, \dots, F_N\}$ directly from the scenario S_{cen} . Each factor F_j has a corresponding value set \mathcal{V}_j . The Cartesian product of these value sets, $\mathcal{V} = \mathcal{V}_1 \times \dots \times \mathcal{V}_N$, forms the complete *information space*. An element $f = (f_1, \dots, f_N) \in \mathcal{V}$ represents one fully-specified state of the world.

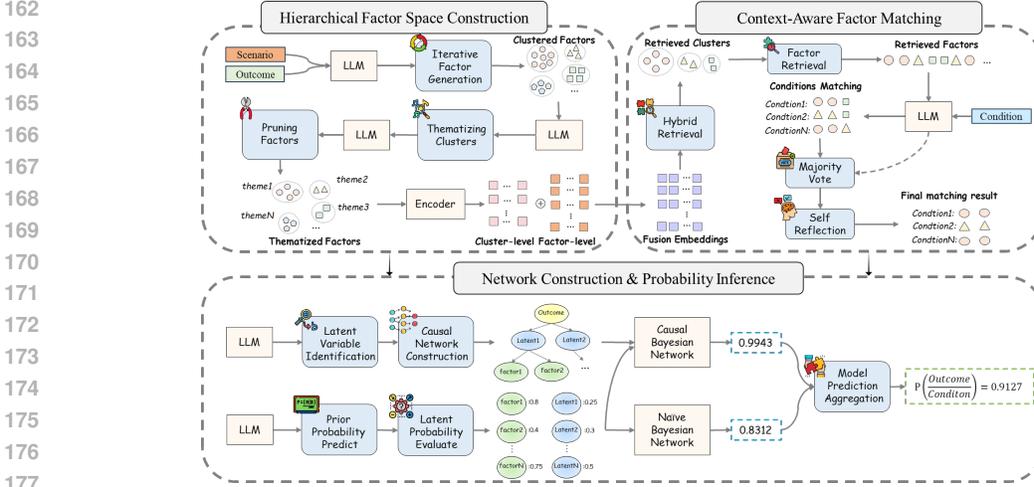


Figure 2: Overview of HARBOR: (1) *Factor-Space Construction*: iterative factor generation and hierarchical clustering generate a dense, two-level factor hierarchy; (2) *Context-Aware Mapping*: perform coarse-to-fine retrieval over the factor hierarchy, then apply self-consistent filtering and reflective refinement to select factors relevant to the condition; (3) *Inference Orchestration*: construct Naïve Bayes and Causal Bayesian networks from the mapped factors—using an LLM to identify latent variables and combine their outputs into a single calibrated probability.

Next, the framework *deductively* computes the outcome probability by marginalizing over this information space: $P(O_i | C) = \sum_{f \in \mathcal{V}} P(O_i | f) P(f | C)$.

Here, $P(O_i | f)$ is the conditional probability table (CPT) and $P(f | C)$ is the probability of instance f given the context. To make this tractable, factors are assumed to be conditionally independent given C , simplifying the calculation to: $P(O_i | C) = \sum_{f \in \mathcal{V}} P(O_i | f) \prod_{j=1}^N P(f_j | C)$

4 METHODOLOGY

In this section, we introduce HARBOR, a three-stage framework designed to transform raw LLM outputs into well-calibrated probabilities. As illustrated in Figure 2, the process consists of factor-space construction, context-aware mapping, and probabilistic inference.

4.1 ITERATIVE ABDUCTION FOR FACTOR-SPACE EXPANSION

4.1.1 BOTTOM-UP ABDUCTION STRATEGY

We define *bottom-up abduction* as a two-stage process that inverts the traditional paradigm by decoupling factor generation from structuring. First, we iteratively generate a comprehensive set of factors, and then we cluster and theme them to build a structured factor space.

The generation stage begins with an empty set $\mathcal{F}^{(0)} = \emptyset$ and iterates until reaching a target size K or exceeding T_{\max} rounds: (a) **Contextual sentence generation**, where a few-shot prompt elicits b diverse supporting or refuting sentences per scenario, explicitly encouraging varied reasoning chains and broad aspect coverage; and (b) **Factor harvesting and validation**, where an LLM extraction prompt identifies all distinct factors in the sentences, which are parsed into a candidate set $\Delta\mathcal{F}$, validated for semantic non-redundancy and domain relevance, and merged into the main set via $\mathcal{F}^{(t+1)} = \mathcal{F}^{(t)} \cup \Delta\mathcal{F}$. Our factor-generation procedure achieves geometric convergence in recall completeness, as detailed in A.2.

4.1.2 SELF-CONSISTENCY AND STRUCTURAL ORGANIZATION

Once the factor set \mathcal{F} is sufficiently large, we impose a hierarchical structure through a four-stage pipeline: (i) encode each factor f into \mathbb{R}^d using MiniLM (Wang et al., 2020); (ii) apply UMAP to

project embeddings into a lower-dimensional space, reducing noise while preserving local semantic neighborhoods (McInnes et al., 2018); (iii) run HDBSCAN to discover cohesive factor groups without pre-specifying the cluster count (McInnes et al., 2017); and (iv) prompt an LLM to assign a concise theme (e.g., *Economic Feasibility*) to each cluster and remove redundant factors, yielding a final hierarchical structure $\tilde{\mathcal{F}}$. To derive factor attributes—supporting O_1 , supporting O_2 , or neutral—for downstream tasks, we prompt the LLM to classify each factor as “supports O_1 ”, “supports O_2 ”, or “neutral”.

4.2 CONTEXT-AWARE FACTOR MAPPING

With the hierarchical factor space $\tilde{\mathcal{F}}$ in place, mapping a downstream condition u via brute-force search is computationally intractable. To address this, we propose a multi-stage pipeline that defines a mapping operator \mathcal{F}^* : starting from a broad retrieval stage to maximize recall, then applying targeted filtering and precision-driven refinement, it maps a condition u to a compact, high-confidence factor set $\mathcal{F}^*(u)$.

4.2.1 HIERARCHICAL RETRIEVAL FOR CANDIDATE GENERATION

Our retrieval design leverages the two-level structure of the factor space $\tilde{\mathcal{F}}$ to efficiently generate a high-recall candidate set, $\mathcal{F}_{\text{cand}}(u)$. The process begins by embedding the condition u and all factor cluster prototypes into a shared vector space \mathbb{R}^d . A key design element is the cluster prototype itself, formulated as a weighted average of its thematic label and its member factor embeddings:

$$e_{C_j} = \alpha \cdot e_{\text{theme}} + (1 - \alpha) \cdot \frac{1}{|F_j|} \sum_{f \in F_j} e_f \quad (1)$$

where e denotes an embedding vector and $\alpha \in [0, 1]$ is a weighting parameter. This formulation balances high-level categorical meaning with fine-grained distributional semantics.

We then perform a **coarse-to-fine search** using K-Nearest Neighbors (KNN) (Guo et al., 2003) to rapidly zero in on relevant factors:

1. **Coarse Search (Cluster-Level):** We first identify the top- K_1 clusters whose prototypes are most semantically similar to the condition embedding e_u . This step acts as a high-level filter, dramatically narrowing the search space to the most promising regions.
2. **Fine Search (Factor-Level):** Within this curated set of clusters, we then conduct a more granular search, retrieving the top- K_2 individual factors from each selected cluster that are closest to e_u .

The union of factors retrieved from this two-step process forms the candidate set $\mathcal{F}_{\text{cand}}(u)$, which is intentionally broad to ensure no relevant factors are prematurely discarded.

4.2.2 SELF-CONSISTENT FILTERING FOR ROBUST SELECTION

The high-recall nature of our retrieval strategy may introduce noise. To address this, we perform a robust filtering step to identify the most pertinent factors from the candidate set. Emulating the principles of self-consistency (Wang et al., 2022), we query an LLM R times with the same prompt, asking it to select the subset of $\mathcal{F}_{\text{cand}}(u)$ that is directly supported by the condition u . We aggregate the results by tallying votes for each factor using the formula $v_f(u) = \sum_{r=1}^R \mathbf{1}[f \in m^{(r)}(u)]$, where $m^{(r)}(u)$ is the subset returned in the r -th query. We then retain only those factors whose vote count meets a configurable threshold γ (e.g., $\gamma = \lceil \text{vote_ratio} \cdot R \rceil$). This majority-vote mechanism yields a filtered set, $\mathcal{F}_{\text{vote}}(u)$, which effectively prunes spurious or weakly supported factors.

4.2.3 REFLECTIVE REFINEMENT FOR FINAL PRECISION

As a final quality control step, the filtered set $\mathcal{F}_{\text{vote}}(u)$ is submitted to a specialized reflection prompt. This prompt instructs the LLM to review and prune any remaining factors that, upon second inspection, lack clear and direct relevance to the condition u . This structured self-critique has been shown to reduce hallucinations and improve factual precision (Ji et al., 2023), yielding the final, high-precision factor set $\mathcal{F}^*(u)$.

4.3 PROBABILISTIC INFERENCE WITH ELICITED PARAMETERS

The final stage of our framework transforms the mapped factor set, $\mathcal{F}^*(u)$, into a calibrated probability for the competing hypotheses, O_1 and O_2 . To achieve this, we first define the structure of two probabilistic models—a Naïve Bayes (NB) model and a Causal Bayesian Network (CBN). We then describe the process of eliciting the necessary parameters from an LLM and, finally, how these parameters are used to perform inference in each model.

Naïve Bayes Model. The Naïve Bayes model is structured on the assumption that all factors are conditionally independent given the hypothesis. It consists of a root *Outcome* node (representing O_1 and O_2) with directed edges to each child factor node $f_j \in \mathcal{F}^*(u)$.

Latent-Augmented Causal Bayesian Network. To capture dependencies between factors, we construct a CBN whose structure is learned dynamically for each scenario. We prompt an LLM to act as a causal discovery engine: given the list of relevant factors, it identifies a set of latent variables $\mathcal{L} = \{L_1, \dots, L_k\}$ and partitions the factors among them. In this model, the binary states of each latent variable, $L_i = 1$ and $L_i = 0$, serve as proxies for the final hypotheses O_1 and O_2 define the network’s edges. Specifically, there is a directed edge from each latent variable to the factors it groups, represented as $L_i \rightarrow f_j$, and a directed edge from the Outcome node to each latent variable, represented as Outcome $\rightarrow L_i$. Outcomes serve as roots and the Latents as their shared child, relaxing NB’s independence assumption by making factors independent only when conditioned on their latent parent.

4.3.1 PARAMETERIZATION AND INFERENCE

Inspired by prior work validating the use of LLMs to elicit informative priors (Gouk & Gao, 2024; Thambawita & Hammer, 2024; Nafar et al., 2025), our framework efficiently parameterizes its Bayesian networks. We directly query the LLM to obtain the necessary conditional probabilities—both at the factor and latent levels—thus bypassing the need for costly data sampling.

Factor-Level Parameters: For each factor f , we elicit its posterior probability given hypothesis O_1 , denoted as $\phi_f = P(O_1 | f)$. This captures the evidential strength of a single factor in favor of O_1 . For the Naïve Bayes (NB) model, which is parameterized by likelihoods $P(f | O_k)$, we adopt a simple binary, symmetric-prior approximation and set $\theta_f = P(f | O_1) \approx \phi_f, P(f | O_2) \approx 1 - \phi_f$.

Latent-Level Parameters: Exclusively for the Causal Bayesian Network (CBN), we elicit the conditional probability of each latent variable L_i given a hypothesis, such as $P(L_i = 1 | O_1)$ and $P(L_i = 0 | O_2)$. To ensure numerical stability, all elicited probabilities are smoothed. These parameters are then used to build and perform inference in our two probabilistic models.

The NB model is parameterized by using the elicited posterior ϕ_f as a proxy for the likelihood, defining the CPT for each factor f as $P(f | O_1) = \theta_f \approx \phi_f, P(f | O_2) \approx 1 - \phi_f$. Given an evidence set \mathcal{E} and assuming a uniform prior over outcomes, the posterior probability for hypothesis O_1 is calculated as:

$$P(O_1 | \mathcal{E}) = \frac{\prod_{f \in \mathcal{E}} P(f | O_1)}{\prod_{f \in \mathcal{E}} P(f | O_1) + \prod_{f \in \mathcal{E}} P(f | O_2)}. \quad (2)$$

The CBN’s CPTs are parameterized as follows. **Latent Nodes** (L_i), as root nodes, are given a uniform prior, so $P(L_i = 1) = 0.5$. The CPT for a **Factor Node** (f_j), conditioned on its latent parent L_i , uses the same NB-style likelihood parameters, specifically $P(f_j | L_i = 1) = \theta_{f_j} \approx \phi_{f_j}, P(f_j | L_i = 0) \approx 1 - \phi_{f_j}$. The CPT for the **Outcome Node**, $P(\text{Outcome} | L_1, \dots, L_k)$, is derived from the elicited latent-level parameters ($P(L_i | O_k)$) via Bayes’ rule.

Given the evidence \mathcal{E} , we perform exact inference using the variable elimination algorithm (Zhang & Poole, 1994) to compute the posterior $P(\text{Outcome} = O_1 | \mathcal{E})$.

4.3.2 MODEL AGGREGATION

To synthesize the predictions from our NB and CBN models into a single, more reliable estimate, we employ the Linear Opinion Pool (LOP) aggregation strategies: The LOP is a simple and powerful method that forms a weighted average of individual model predictions (Stone, 1961; Neyman &

Roughgarden, 2023; Stratigakos et al., 2024). We select it as our primary strategy for its computational efficiency and proven robustness. The aggregated probability is a convex combination: $P_{\text{LOF}}(O_1 | \mathcal{E}) = \sum_{M \in \{\text{NB}, \text{CBN}\}} w_M \cdot P(O_1 | \mathcal{E}, M)$, where the weights w_M are fixed and sum to 1.

5 EXPERIMENT

5.1 EXPERIMENTAL SETUPS

Datasets We evaluate our model on three reasoning and planning benchmarks from Feng et al. (2025), each formatted as a decision-support tuple (S_{cen}, U, O) : **Today** (1,000 instances), **Plasma** (279 scenarios, 1,395 instances) and **Common2Sense** (216 scenarios, 3,822 instances), the Common2Sense’s test split being manually expanded under an unbiased protocol to boost diversity. To assess decision-making beyond planning, we further sample 100 examples from each of the four fact-checking datasets—**XSum** (Tang et al., 2024), **CNN** (Tang et al., 2024), **ExpertQA** (Malaviya et al., 2023) and **COVID** (Saakyan et al., 2021)—treating each document as condition U and each claim as outcome O , with S_{cen} left empty. (dataset details refer to A.5).

Settings We conduct all experiments with four LLMs: Qwen2.5-32b, Qwen2.5-72b (Bai et al., 2023), DeepSeekV3-671b (Liu et al., 2024), and GPT-4 (Achiam et al., 2023). The Qwen models were deployed locally, while DeepSeekV3-671b was accessed through DeepSeek’s official API. We set $K_1 = 3$ and $K_2 = 5$ in the KNN search; more hyperparameters and settings are provided in Appendix A.6. All experiments ran on 4× NVIDIA RTX 4090 (24GB) GPUs.

Preference-based Pairwise Evaluation To begin, we evaluate our model on Common2Sense using the *preference-based pairwise evaluation* framework introduced by Feng et al. (2025). The experimental setup is as follows. Given a scenario S_{cen} and two potential outcomes, O_1 and O_2 , we select two distinct additional conditions, U_1 and U_2 . Crucially, both conditions are constructed to support the same outcome, O_1 , over O_2 . These are then used to form two slightly different contexts: $C_1 = S_{\text{cen}} + U_1$ and $C_2 = S_{\text{cen}} + U_2$. The model’s task is to assign conditional probabilities $P(O_i|C_1)$ and $P(O_i|C_2)$ for each outcome $i \in \{1, 2\}$. A successful evaluation requires the model to correctly discern the subtle difference in the degree of support between the two contexts. For instance, if human judgment determines that C_1 provides stronger support for O_1 than C_2 does, we expect the model’s probability assignments to satisfy the following relationship: $P(O_1|C_1) > P(O_1|C_2) > P(O_2|C_2) > P(O_2|C_1)$, we set the abstention threshold $\tau = 0$, so the system yields an unknown prediction only when no factors are matched, i.e., $|\mathcal{F}^*(u)| = 0$.

Decision Making Evaluation We evaluate HARBOR’s decision-making on reasoning and planning datasets (Plasma, Today) and four fact-checking datasets for generalization (ExpertQA, XSum, COVID, CNN). In each task, given a context C and two outcomes O_1 and O_2 , the model must select the more plausible one. A decision is considered correct if the chosen outcome has a higher conditional probability, e.g., satisfying $P(O_1|C) > P(O_2|C)$ when O_1 is the ground truth.

Baseline Methods We compare our approach against several baselines for probability estimation from LLMs in our experiments: (1) **BIRD**: Builds a structured factor space via single-pass abductive reasoning and uses a Naïve Bayes model for inference (Feng et al., 2025). (2) **Vanilla**: Prompts the model to verbalize its estimated probability (Wang et al., 2022). (3) **Logits**: Converts the normalized token probability of the decision token into a probability score. (4) **CoT**: Elicits a chain-of-thought reasoning process before asking for the final probability (Wei et al., 2022). (5) **Compare**: Selects which of two conditions better supports the outcome without producing individual probabilities, an unfair setting for methods relying on those estimates (Feng et al., 2025). (6) **Factor-based**: Averages probabilities over five factors generated with knowledge of the gold outcome, making direct comparison invalid.

5.2 RESULTS

The results are listed in Table 1. Our framework, HARBOR, exhibits a superior alignment with human preference, achieving a top F1 of 59.6%, more than 14% higher than DeepSeek-V3-671b. This demonstrates that HARBOR produces reliable probability estimations.

Method	Model	Context1	Context2	Same	Average	Coverage
Random	Guess	0.333	0.333	0.333	0.333	–
CoT	Qwen2.5-32b	0.394	0.370	0.118	0.319	–
	Qwen2.5-72b	0.370	0.382	0.159	0.311	–
	DeepSeek-V3-671b	0.414	0.395	0.146	0.338	–
Vanilla	Qwen2.5-32b	0.528	0.528	0.211	0.462	–
	Qwen2.5-72b	0.526	0.528	0.183	0.470	–
	DeepSeek-V3-671b	0.489	0.498	0.137	0.425	–
Logits	Qwen2.5-32b	0.504	0.491	0.150	0.445	–
	Qwen2.5-72b	0.523	0.500	0.156	0.447	–
	DeepSeek-V3-671b	0.516	0.513	0.162	0.453	–
Compare	Qwen2.5-32b	0.583	0.480	0.221	0.477	–
	Qwen2.5-72b	0.566	0.570	0.267	0.534	–
	DeepSeek-V3-671b	0.560	0.578	0.286	0.542	–
	GPT-4	0.587	0.548	<u>0.302</u>	0.556	–
Factor-based	Qwen2.5-32b	0.556	0.564	0.054	0.525	–
	Qwen2.5-72b	0.509	0.509	0.250	0.496	–
	DeepSeek-V3-671b	0.553	0.546	0.165	0.506	–
BIRD	Qwen2.5-72b [†]	0.497	0.412	0.271	0.421	87.77%
	Qwen2.5-72b [‡]	0.567	0.520	0.307	0.513	99.99%
HARBOR (ours)	Qwen2.5-32b	<u>0.612</u>	0.587	0.152	<u>0.572</u>	99.79%
	Qwen2.5-72b	0.626	0.627	0.281	0.596	99.95%

Table 1: Preference-based pairwise evaluation using F1 score on the Common2sense dataset. Context1 is the F1 score for the class where the first context better supports the outcome than the second, and Context2 is for the reverse. The same is the F1 score for cases where both contexts offer equal support. Avg is the global micro-averaged F1 score. BIRD[†] uses the original, unmodified mapping output of the BIRD method. BIRD[‡] employs a relaxed mapping condition, forcing the model to match each condition with at least one factor from the factor space. Unlike BIRD, which manually filtered “unknown” cases, we remove this intervention for fairness.

Some baselines, such as Compare, perform joint context processing, giving them an unfair advantage. We also introduce Factor-based, another unfair baseline that assumes the model can construct a perfect factor space and always match the gold-aligned factor. Despite these advantages, HARBOR still outperforms both baselines—achieving up to 5% in F1. Even our smaller Qwen2.5-32b model outperforms the baselines built on the larger Qwen2.5-72b. In terms of coverage, HARBOR achieves near-complete coverage with its default mapping—no manual intervention required—while BIRD attains only 87.8% under its original mapping and must relax matching to reach comparable coverage.

HARBOR is designed to be highly sensitive to nuanced differences between contexts, which results in its F1 being lower on the Same class compared to other categories. Given the severe class imbalance (the Same class is only 9.82% of the dataset), the Micro Avg F1 score provides a more meaningful measure of overall performance, where our method clearly excels.

For decision-making, following Feng et al. (2025), when the BIRD baseline predicts “unknown”, we default to the CoT method to ensure a decision is made. As other methods do not predict “unknown”, we omit the coverage metric. ExpertQA, XSum, CNN, and Today represent more challenging settings due to long contexts or temporal reasoning, while COVID and Plasma are relatively simpler (Feng et al., 2025). For decision-making on fact-checking datasets, we convert probabilistic outputs into support labels by applying a fixed threshold, following prior work in fact verification (Jayaweera et al., 2024). For the Plasma and Today datasets, we adhere to the evaluation of (Feng et al., 2025). As shown in Table 2, HARBOR consistently outperforms all baselines at the 72b scale and even surpasses the 671b model on the harder tasks.

Model	expertqa	xsum	covid	cnn	today	plasma
<i>DeepSeek-V3-671b</i>						
Vanilla	0.538	0.537	<u>0.752</u>	0.495	0.710	0.774
Vanilla *	0.562	0.483	0.686	0.500	0.647	0.781
CoT	0.541	0.527	0.685	0.495	0.783	<u>0.784</u>
CoT *	0.498	0.517	0.754	0.462	<u>0.803</u>	0.850
<i>Qwen2.5-72b</i>						
Vanilla	0.517	<u>0.557</u>	0.538	<u>0.604</u>	0.583	0.604
Vanilla *	0.530	0.473	0.686	0.500	0.597	0.649
CoT	<u>0.590</u>	0.531	0.721	0.589	0.593	0.565
CoT *	0.552	0.484	0.701	0.500	0.713	0.745
BIRD	0.585	0.495	0.670	0.528	0.741	0.724
HARBOR(ours)	0.605	0.567	0.730	0.629	0.827	0.762

Table 2: Balanced accuracy on fact-checking tasks and accuracy on reasoning/planning tasks. Methods marked * make direct decisions without probability outputs.

5.3 ABLATION STUDY

To validate our design, we conducted an ablation study on Common2Sense (Table 3), analyzing five key variants. In addition to ablating core components, the study compares our standard LOP with an alternative aggregation strategy: Bayesian Model Averaging (BMA), a theoretically rigorous method that weights models by their posterior probabilities (Hoeting et al., 1999; Fragoso et al., 2018).

The five variants are: (1) **w/o cbn**, which uses only the Naïve Bayes model in the aggregation step; (2) **w/o nb**, which uses only the Causal Bayesian Network model in the aggregation step; (3) **w/o cluster**, which expands factors without hierarchical clustering, equivalent to simply increasing the number of factors; (4) **w/o hierarchy**, which retains clustering but removes weighted fusion and coarse-to-fine retrieval; (5) **w/o pe-llm**, which replaces LLM-elicited parameters with frequency-based estimates (A.3); and (6) **HARBOR (BMA)**, which implements the BMA strategy (A.4).

Component	Qwen2.5-32b					Qwen2.5-72b				
	Ctx. 1	Ctx. 2	Same	Avg.	Cov.	Ctx. 1	Ctx. 2	Same	Avg.	Cov.
w/o cbn	0.592	0.579	0.152	0.558 _{↓0.014}	–	0.530	0.518	0.219	0.506 _{↓0.090}	–
w/o nb	0.518	0.514	0.197	0.479 _{↓0.093}	–	0.592	0.583	0.260	0.557 _{↓0.039}	–
w/o cluster	0.480	0.497	0.193	0.457 _{↓0.115}	98.93%	0.420	0.429	0.173	0.400 _{↓0.196}	98.82%
w/o hierarchy	0.514	0.511	0.131	0.491 _{↓0.081}	99.84%	0.515	0.512	0.127	0.491 _{↓0.105}	99.92%
w/o pe-llm	0.459	0.484	0.271	0.435 _{↓0.137}	–	0.505	0.549	0.281	0.483 _{↓0.113}	–
HARBOR (BMA)	0.615	0.592	0.152	0.575 _{↑0.003}	–	0.604	0.621	0.281	0.592 _{↓0.004}	–

Table 3: Ablation study results based on Table 1. Ctx. 1/2, Same, Avg. and Cov. denote F1 for Context 1, Context 2, Same classes, micro-averaged F1 and coverage

The “–” in the Coverage column indicates that the variant did not affect coverage. While coverage remains near 100% for most variants, the dramatic drop in (2) confirms that unstructured factor expansion is insufficient. The performance declines in (4) and (3) underscore the importance of accurate parameter elicitation and our two-level hierarchy, respectively. Finally, both aggregation strategies—LOP and BMA—achieve similarly high coverage and comparable F1 scores, demonstrating that effective fusion, combined with hierarchical mapping and precise elicitation, is key to HARBOR’s robust performance.

On the Common2sense dataset with Qwen2.5-72B, we plug the original BIRD[†] pipeline into HARBOR’s factor space while keeping BIRD-style CPT training. Within this unified setup, we compare four variants: the BIRD[†] baseline with HARBOR’s generated factors, HARBOR’s factors and mapping, and the latter further augmented with a CBN combined with Naïve Bayes via a linear opinion pool. Table 4 reports Avg (global micro-averaged F1 over CONTEXT1, CONTEXT2, and SAME, as in Table 1), coverage, and mapping-stage cost normalized by the BIRD[†] baseline (Tok./B, Time/B). Detailed training and sampling procedures are given in Appendix A.10.

As shown in Table 4, replacing BIRD[†]’s original factors with HARBOR’s denser factor space improves Avg from 0.513 to 0.532 but increases mapping cost by almost 3×. Using HARBOR’s context-aware factor mapping on the same factor space recovers efficiency while still outperforming the BIRD[†] baseline. Adding CBN aggregation on top of HARBOR’s factors and mapping yields the strongest variant at essentially the same reduced cost, indicating that factor-space construction, mapping, and CBN aggregation all contribute beyond a stronger factor generator alone.

Setting	Avg	Cov.	Tok./B	Time/B
BIRD [†] baseline	0.513	99.99%	1.00	1.00
+F	0.532	99.12%	2.98	2.87
+F+M	0.526	98.81%	0.17	0.55
+F+M+CBN	0.568	98.81%	0.17	0.55

Table 4: BIRD[†] variants on HARBOR’s factor space and mapping on Common2sense.

5.4 ANALYSIS

Figure 3 reports the computation time, token usage, and API calls on the Common2Sense dataset, using Qwen2.5-72B under the same experimental setup as the main results in Table 1. On average, per scenario, HARBOR runs in about 0.74× the time, uses 0.24× the tokens, and makes 0.37× the API calls of BIRD on this benchmark. While HARBOR invests more in factor-space construction,

its hierarchical condition–factor mapping significantly reduces downstream cost. For HARBOR, as more factors are added, the unknown rate falls and the average F1 generally improves, reflecting a trade-off between coverage and accuracy. More detailed analyses are provided in Appendix A.7 (e.g., retrieval algorithms, the effects of K_1 and K_2 and analyses of the probability-estimator LLM).

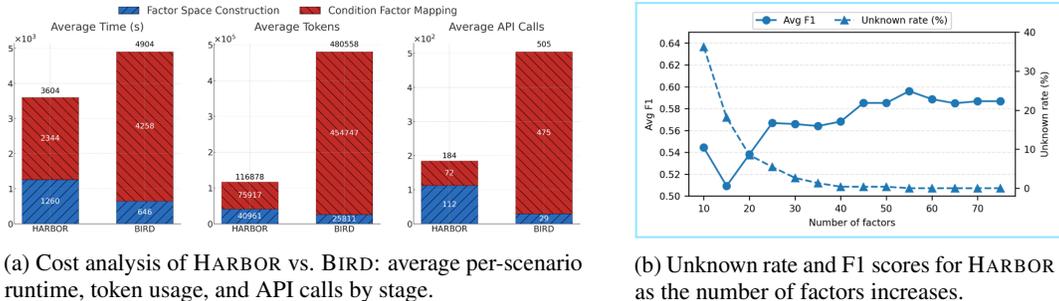


Figure 3: Cost and coverage–accuracy analysis for HARBOR and BIRD.

5.4.1 UNCERTAINTY ESTIMATES AND STATISTICAL ANALYSIS

To quantify uncertainty in our main results, we run both HARBOR and BIRD five times under the same settings as Tables 1 and 2, and report the mean, standard deviation, and 95% confidence intervals using a t -distribution (Student, 1908). Table 5 summarizes F1 for the preference-based evaluation, and Table 6 reports (balanced) accuracy on the decision-making benchmarks. Across all preference metrics, HARBOR consistently outperforms BIRD with gains larger than the corresponding standard deviations; on the decision-making datasets, HARBOR with Qwen2.5-72B likewise achieves uniformly higher accuracy with small variance. The single-run scores in Tables 1 and 2 all fall within the 95% confidence intervals in Tables 5 and 6, indicating that those point estimates are representative rather than outliers.

Model	Metric	Context1	Context2	Same	Average
BIRD · Qwen-72B	Mean ± Std	0.568 ± 0.012	0.523 ± 0.014	0.306 ± 0.011	0.526 ± 0.013
	95% CI	[0.553, 0.584]	[0.505, 0.540]	[0.293, 0.319]	[0.511, 0.542]
HARBOR · Qwen-32B	Mean ± Std	0.602 ± 0.012	0.583 ± 0.012	0.146 ± 0.018	0.563 ± 0.017
	95% CI	[0.587, 0.617]	[0.569, 0.598]	[0.124, 0.169]	[0.542, 0.583]
HARBOR · Qwen-72B	Mean ± Std	0.610 ± 0.021	0.611 ± 0.035	0.278 ± 0.014	0.606 ± 0.030
	95% CI	[0.584, 0.637]	[0.567, 0.655]	[0.260, 0.295]	[0.570, 0.643]

Table 5: Preference-based evaluation (F1) with mean ± standard deviation and 95% confidence intervals over 5 runs.

Model	Metric	expertqa	xsum	covid	cnn	today	plasma
HARBOR · Qwen-72B	Mean ± Std	0.611 ± 0.019	0.564 ± 0.016	0.726 ± 0.013	0.621 ± 0.029	0.817 ± 0.031	0.764 ± 0.022
	95% CI	[0.587, 0.634]	[0.544, 0.583]	[0.710, 0.742]	[0.585, 0.657]	[0.778, 0.856]	[0.736, 0.791]
BIRD · Qwen-72B	Mean ± Std	0.582 ± 0.023	0.511 ± 0.022	0.664 ± 0.020	0.540 ± 0.027	0.754 ± 0.035	0.718 ± 0.022
	95% CI	[0.552, 0.611]	[0.484, 0.539]	[0.639, 0.689]	[0.507, 0.574]	[0.711, 0.797]	[0.691, 0.746]

Table 6: Decision-making accuracy / balanced accuracy with mean ± standard deviation and 95% confidence intervals over 5 runs.

6 CONCLUSION

We introduced the HARBOR framework to generate reliable and calibrated probability estimates from Large Language Models for critical decision-making. By integrating a hierarchical factor-space construction with the orchestration of Naïve Bayes and a Causal Bayesian Network, our framework showed notable enhancements in decision-making accuracy and alignment with human judgment. Our experimental results underscore the effectiveness of HARBOR, thereby increasing the practical utility of LLMs in high-stakes scenarios and fostering more trustworthy autonomous systems.

540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593

ETHICS STATEMENT

This work aims to improve reliable probability estimation and interpretable decision-making under incomplete information, with clear social value in domains such as emergency response and public governance. HARBOR reduces the risks of “unknown” predictions and overconfidence through hierarchical abduction and Bayesian orchestration, while providing an auditable factor space as the basis of decisions. However, HARBOR measures reasoning consistency and uncertainty calibration, which is not equivalent to factual correctness or fairness; if training data or prompts contain bias, the model may still produce highly confident yet biased conclusions. Therefore, HARBOR should be used in conjunction with fact verification, bias detection, and human oversight, particularly in high-stakes or sensitive scenarios. No personal or sensitive data are used or released in this study, and all resources comply with platform and licensing requirements.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we provide methodological details and experimental configurations, with additional resources in the appendix:

- **Algorithmic details:** Section 4 describes the three-stage pipeline (factor-space construction, hierarchical retrieval and mapping, probabilistic inference), with structured pseudocode in Appendix A.9.
- **Hyperparameters:** All key hyperparameters (e.g., $K_1=3$, $K_2=5$, α , R , ϵ) are listed in Table 8 (Appendix A.6), together with their values and rationale.
- **Datasets and preprocessing:** Data sources and construction are described in Section 5.1 and Appendix A.5, with dataset statistics summarized in Table 7.
- **Experimental settings:** Model configurations are detailed in the Section 5.1.
- **Evaluation protocol:** Pairwise preference and decision evaluations are defined in Section 5.1, with main and ablation results presented in Tables 1 and 3.
- **Implementation and resources:** Key components for retrieval, clustering, and inference are detailed in Appendices A.8 and A.9. All source code and scripts required to reproduce our experiments are included in the supplementary material. In addition, prompt examples are provided in Appendix B.3, comparative examples between HARBOR and BIRD are given in Appendix B.2, and a statement of LLM usage in this work is presented in Appendix B.1.

REFERENCES

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Ibrahim Al Azher, Venkata Devesh Reddy Seethi, Akhil Pandey Akella, and Hamed Alhoori. Limtopic: Llm-based topic modeling and text summarization for analyzing scientific articles limitations. In *Proceedings of the 24th ACM/IEEE Joint Conference on Digital Libraries*, pp. 1–12, 2024.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
- Faeze Brahma, Chandra Bhagavatula, Valentina Pyatkin, Jena D Hwang, Xiang Lorraine Li, Hirona J Arai, Soumya Sanyal, Keisuke Sakaguchi, Xiang Ren, and Yejin Choi. Plasma: Making small language models better procedural knowledge models for (counterfactual) planning. *arXiv preprint arXiv:2305.19472*, 2023.
- Graham Cormode, S. Muthukrishnan, and Ke Yi. Set cover algorithms for very large datasets. In *Proceedings of the 19th ACM CIKM*, pp. 479–488, 2010. doi: 10.1145/1871437.1871501.
- Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitan, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

- 594 Yu Feng, Ben Zhou, Haoyu Wang, Helen Jin, and Dan Roth. Generic temporal reasoning with
595 differential analysis and explanation. *arXiv preprint arXiv:2212.10467*, 2022.
596
- 597 Yu Feng, Ben Zhou, Weidong Lin, and Dan Roth. Bird: A trustworthy bayesian inference framework
598 for large language models. In *Proceedings of the International Conference on Learning Repre-*
599 *sentations (ICLR)*, 2025. URL [https://cogcomp.seas.upenn.edu/papers/FZLR25.](https://cogcomp.seas.upenn.edu/papers/FZLR25.pdf)
600 [pdf](https://cogcomp.seas.upenn.edu/papers/FZLR25.pdf).
- 601 Maria D. Fragoso, Wesley Bertoli, and Francisco Louzada. Bayesian model averaging: A systematic
602 review and conceptual classification. *International Statistical Review*, 86(1):1–28, 2018. doi:
603 10.1111/insr.12243.
- 604 Henry Gouk and Boyan Gao. Automated prior elicitation from large language models for bayesian
605 logistic regression. In *AutoML Workshop at ICML 2024*, 2024. URL [https://openreview.](https://openreview.net/forum?id=euLzlnU7gz)
606 [net/forum?id=euLzlnU7gz](https://openreview.net/forum?id=euLzlnU7gz).
607
- 608 Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in
609 classification. In *OTM Confederated International Conferences" On the Move to Meaningful*
610 *Internet Systems"*, pp. 986–996. Springer, 2003.
- 611 David J Hand and Keming Yu. Idiot’s bayes—not so stupid after all? *International statistical review*,
612 69(3):385–398, 2001.
613
- 614 Yulin He, Guiliang Ou, Philippe Fournier-Viger, and Joshua Zhexue Huang. Attribute grouping-based
615 naive bayesian classifier. *Science China Information Sciences*, 68(3):132106, 2025.
616
- 617 Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model
618 averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999. doi: 10.1214/ss/1009212519.
- 619 Matthew Douglas Hoffman, Du Phan, David Dohan, Sholto Douglas, Tuan Anh Le, Aaron Parisi,
620 Pavel Sountsov, Charles Sutton, Sharad Vikram, and Rif A Saurous. Training chain-of-thought via
621 latent-variable inference. In *NeurIPS*, 2023.
622
- 623 Bairu Hou, Yujian Liu, Kaizhi Qian, Jacob Andreas, Shiyu Chang, and Yang Zhang. Decomposing
624 uncertainty for large language models through input clarification ensembling. *arXiv preprint*
625 *arXiv:2311.08718*, 2023.
- 626 Edward J Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, Yoshua Bengio,
627 and Nikolay Malkin. Amortizing intractable inference in large language models. *arXiv preprint*
628 *arXiv:2310.04363*, 2023.
629
- 630 Haoyu Huang, Yongfeng Huang, Junjie Yang, Zhenyu Pan, Yongqiang Chen, Kaili Ma, Hongzhi
631 Chen, and James Cheng. Retrieval-augmented generation with hierarchical knowledge. *arXiv*
632 *preprint arXiv:2503.10150*, 2025.
- 633 Chathuri Jayaweera, Sangpil Youm, and Bonnie Dorr. Amrex: Amr for explainable fact verification.
634 In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pp. 234–244,
635 Bangkok, Thailand, 2024. Association for Computational Linguistics.
- 636 Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. Towards mitigating
637 llm hallucination via self reflection. In *Findings of the Association for Computational Linguistics:*
638 *EMNLP 2023*, pp. 1827–1843, 2023.
639
- 640 Nora Kassner, Oyvind Tafjord, Ashish Sabharwal, Kyle Richardson, Hinrich Schuetze, and Peter
641 Clark. Language models with rationality. *arXiv preprint arXiv:2305.14250*, 2023.
642
- 643 Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for
644 uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.
- 645 Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal,
646 Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented genera-
647 tion for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:
9459–9474, 2020.

- 648 Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula,
649 Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. Swiftsage: A generative agent with fast and
650 slow thinking for complex interactive tasks. *Advances in Neural Information Processing Systems*,
651 36:23813–23825, 2023.
- 652 Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao,
653 Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint*
654 *arXiv:2412.19437*, 2024.
- 655 Ankur Madaan, Wayne Xin Zhao, Lei Qin, Zhuyun Chen, and Daniel S. Weld. Self-ask: Decomposing
656 questions for complex text understanding. In *Proceedings of the 2021 Conference on Empirical*
657 *Methods in Natural Language Processing*, pp. 845–861, 2021.
- 658 Chaitanya Malaviya, Subin Lee, Sihao Chen, Elizabeth Sieber, Mark Yatskar, and Dan Roth. Expertqa:
659 Expert-curated questions and attributed answers. *arXiv preprint arXiv:2309.07852*, 2023.
- 660 John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial
661 intelligence. In Nils J. Nilsson (ed.), *Readings in Artificial Intelligence*, pp. 431–450. Morgan
662 Kaufmann, 1981. Originally published in **Machine Intelligence 4** (1969).
- 663 Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. In
664 *Journal of Open Source Software*, volume 2, pp. 205, 2017. doi: 10.21105/joss.00205.
- 665 Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and
666 projection for dimension reduction. *arXiv:1802.03426*, 2018.
- 667 Aliakbar Nafar, Kristen Brent Venable, Zijun Cui, and Parisa Kordjamshidi. Extracting probabilistic
668 knowledge from large language models for bayesian network parameterization. *arXiv preprint*
669 *arXiv:2505.15918*, 2025.
- 670 Eric Neyman and Tim Roughgarden. No-regret learning with unbounded losses: The case of
671 logarithmic pooling. *arXiv preprint arXiv:2202.11219*, 2023. URL [https://arxiv.org/
672 abs/2202.11219](https://arxiv.org/abs/2202.11219).
- 673 D Prabha, J Aswini, B Maheswari, R Siva Subramanian, R Nithyanandhan, and P Girija. A survey on
674 alleviating the naive bayes conditional independence assumption. In *2022 International Conference*
675 *on Augmented Intelligence and Sustainable Systems (ICAISS)*, pp. 654–657. IEEE, 2022.
- 676 Linlu Qiu, Fei Sha, Kelsey Allen, Yoon Kim, Tal Linzen, and Sjoerd van Steenkiste. Bayesian teaching
677 enables probabilistic reasoning in large language models. *arXiv preprint arXiv:2503.17523*, 2025.
678 URL <https://arxiv.org/abs/2503.17523>.
- 679 Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu,
680 Leila Takayama, Fei Xia, Jake Varley, et al. Robots that ask for help: Uncertainty alignment for
681 large language model planners. *arXiv preprint arXiv:2307.01928*, 2023.
- 682 Arik Reuter, Tim G. J. Rudner, Vincent Fortuin, and David Rügamer. Can transformers learn full
683 bayesian inference in context? In *Proceedings of the 42nd International Conference on Machine*
684 *Learning (ICML)*, 2025.
- 685 Arkadiy Saakyan, Tuhin Chakrabarty, and Smaranda Muresan. Covid-fact: Fact extraction and
686 verification of real-world claims on covid-19 pandemic. *arXiv preprint arXiv:2106.03794*, 2021.
- 687 Eleni Sgouritsa, Virginia Aglietti, Yee Whye Teh, Arnaud Doucet, Arthur Gretton, and Silvia Chiappa.
688 Prompting strategies for enabling large language models to infer causation from correlation. In
689 *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2024, 2024.
- 690 Fernando Simeone, Maik Olher Chaves, and Ahmed Esmín. Incremental hierarchical text clustering
691 methods: a review. *arXiv preprint arXiv:2312.07769*, 2023.
- 692 Shikhar Singh, Nuan Wen, Yu Hou, Pegah Alipoormolabashi, Te-lin Wu, Xuezhe Ma, and Nanyun
693 Peng. Com2sense: A commonsense reasoning benchmark with complementary sentences. *arXiv*
694 *preprint arXiv:2106.00969*, 2021.

- 702 Mervyn Stone. The opinion pool. *Annals of Mathematical Statistics*, 32(4):1339–1342, 1961. doi:
703 10.1214/aoms/1177704873.
704
- 705 Akylas Stratigakos, Salvador Pineda, and Juan Miguel Morales. Decision-focused linear pooling for
706 probabilistic forecast combination. *International Journal of Forecasting*, 2024. doi: 10.1016/j.
707 ijforecast.2024.11.006. In press.
- 708 Student. The probable error of a mean. *Biometrika*, 6(1):1–25, March 1908. doi: 10.1093/biomet/6.1.
709 1.
710
- 711 Fengfei Sun, Ningke Li, Kailong Wang, and Lorenz Goette. Large language models are overconfident
712 and amplify human bias. *arXiv preprint arXiv:2505.02151*, 2024. URL [https://arxiv.org/
713 abs/2505.02151](https://arxiv.org/abs/2505.02151).
- 714 Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. Entailer: Answering questions with faithful
715 and truthful chains of reasoning. *arXiv preprint arXiv:2210.12217*, 2022.
716
- 717 Liyan Tang, Philippe Laban, and Greg Durrett. Minicheck: Efficient fact-checking of llms on
718 grounding documents. *arXiv preprint arXiv:2404.10774*, 2024.
- 719 Vajira Thambawita and Hugo L. Hammer. Using large language models to suggest informative prior
720 distributions in bayesian statistics. *arXiv preprint arXiv:2506.21964*, 2024.
721
- 722 Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep
723 self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.
724 arXiv:2002.10957.
- 725 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdh-
726 ery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models.
727 *arXiv preprint arXiv:2203.11171*, 2022.
728
- 729 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny
730 Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in
731 neural information processing systems*, 35:24824–24837, 2022.
- 732 Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan
733 Berant. Break it down: A question understanding benchmark. *Transactions of the Association for
734 Computational Linguistics*, 8:183–198, 2020.
735
- 736 Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms
737 express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint
738 arXiv:2306.13063*, 2023.
- 739 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan.
740 Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural
741 information processing systems*, 36:11809–11822, 2023.
742
- 743 Nayyar A Zaidi, Jesus Cerquides, Mark J Carman, and Geoffrey I Webb. Alleviating naive bayes
744 attribute independence assumption by attribute weighting. *The Journal of Machine Learning
745 Research*, 14(1):1947–1988, 2013.
- 746 Harry Zhang. The optimality of naive bayes. *Aa*, 1(2):3, 2004.
747
- 748 Nevin Lianwen Zhang and David L. Poole. A simple approach to bayesian network computations.
749 In *Proceedings of the 10th Canadian Conference on Artificial Intelligence (AI-94)*, pp. 171–178,
750 Banff, Alberta, Canada, 1994. AAAI Press / The MIT Press.
- 751 Xiaoming Zhang, Ming Wang, Xiaocui Yang, Daling Wang, Shi Feng, and Yifei Zhang. Hierarchical
752 retrieval-augmented generation model with rethink for multi-hop question answering. *arXiv
753 preprint arXiv:2408.11875*, 2024.
754
- 755 Yuwei Zhang, Zihan Wang, and Jingbo Shang. Clusterllm: Large language models as a guide for text
clustering. *arXiv preprint arXiv:2305.14871*, 2023.

756 Yuhuai Zhou, Xiang Yu, David Chern, Xu Huang, Danqi Shen, and Yiming Song. Least-to-most
 757 prompting enables complex reasoning in large language models. In *Advances in Neural Information*
 758 *Processing Systems*, 2022.

759 Zining Zhu, Tong Li, and Zhiyuan Liu. Explainability for large language models: A survey. *ACM*
 760 *Computing Surveys*, 2024. doi: 10.1145/3639372.

763 A APPENDIX A

764 A.1 UNCERTAINTY DECOMPOSITION

765 To understand how outcome probabilities are inferred from partial contextual information, we start
 766 with the law of total probability over the full factor space \mathcal{F} :

$$767 P(O_i | C) = \sum_{f \in \mathcal{F}} P(O_i | f, C) \cdot P(f | C) \quad (3)$$

768 We can divide the factor set into two disjoint parts: those consistent with context C , and those that
 769 contradict it. Denote these as:

$$770 \mathcal{F}_C = \{f \in \mathcal{F} \mid f \cap C = f\}, \quad (4)$$

$$771 \mathcal{F}_{-C} = \mathcal{F} \setminus \mathcal{F}_C.$$

772 Then the probability becomes:

$$773 P(O_i | C) = \sum_{f \in \mathcal{F}_C} P(O_i | f, C) P(f | C) + \sum_{f \in \mathcal{F}_{-C}} P(O_i | f, C) P(f | C)$$

$$774 = \sum_{f \in \mathcal{F}_C} P(O_i | f) P(f | C) + \sum_{f \in \mathcal{F}_{-C}} P(O_i | f, C) \cdot 0 \quad (5)$$

$$775 = \sum_{f \in \mathcal{F}_C} P(O_i | f) P(f | C).$$

776 The second line follows from the assumption that only context-consistent factors have non-zero
 777 posterior $P(f | C)$, and the third line uses the independence assumption $P(O_i | f, C) = P(O_i | f)$
 778 when $f \subseteq C$.

779 **Interpretation.** This formulation shows that outcome probability is a marginalization over consistent
 780 factors, where each term combines a factor’s evidential support $P(O_i | f)$ with its relevance weight
 781 $P(f | C)$. Our retrieval and voting pipeline can thus be interpreted as approximating this sum with
 782 the selected set $\mathcal{F}^*(C)$:

$$783 P(O_i | C) \approx \sum_{f \in \mathcal{F}^*(C)} P(O_i | f) P(f | C) \quad (6)$$

784 A.2 ASSUMPTIONS FOR PROBABILITY ESTIMATION IN HARBOR

785 **Weak ordering of factor odds ratios** For each factor $f_j \in \mathcal{F}$, define its odds

$$786 o_j = \frac{P(O_1 | f_j)}{1 - P(O_1 | f_j)}, \quad (7)$$

787 We write

$$788 (o_1, \dots, o_N) \succeq_E (o'_1, \dots, o'_N). \quad (8)$$

789 if a decision maker is at least as confident in outcome O_1 given odds (o_1, \dots, o_N) as given
 790 (o'_1, \dots, o'_N) . This relation is a weak ordering, satisfying:

- 791 (a) *Completeness*: for any two odds-vectors A, B , either $A \succeq_E B$ or $B \succeq_E A$;
 792 (b) *Transitivity*: if $A \succeq_E B$ and $B \succeq_E C$, then $A \succeq_E C$.

Non-interaction of factors For any disjoint factor subsets $A, B \subseteq \mathcal{F}$, the ordering induced by A alone is unchanged by the presence of B :

$$(o_j : f_j \in A) \succeq_E (o'_j : f_j \in A) \iff (o_j : f_j \in A \cup B) \succeq_E (o'_j : f_j \in A \cup B). \quad (9)$$

Self-consistency convergence Querying the LLM m times and taking a majority vote yields an error probability bounded by $\exp(-2m(q - 0.5)^2)$, where $q > 0.5$ is the accuracy of a single vote (Wang et al., 2022).

Abduction recall guarantee If each true factor is recovered in a given round with probability $p > 0$, then after T rounds the probability of missing it is $(1 - p)^T$. Thus $T = O(\log(1/\varepsilon)/p)$ rounds suffice to cover the factor space with high probability (Cormode et al., 2010).

Together, these assumptions ensure that (1) factor odds admit a well-behaved weak ordering; (2) factors do not spuriously interact in the odds ranking; and (3) both abduction and self-consistency stages have explicit convergence guarantees.

A.3 PARAMETER INITIALIZATION

Factor prior initialization Factor labels (“supports O_1 ”, “supports O_2 ”, “neutral”) are assigned via the self-consistency procedure described in our method-Iterative Abduction. We then replace the LLM-elicited posterior $\theta_{f_j} = P(O_1 | f_j)$ with

$$P_{\text{init}}(O_1 | f_j) = \begin{cases} 0.75, & f_j \text{ labeled “supports } O_1\text{”}, \\ 0.50, & f_j \text{ labeled “neutral”}, \\ 0.25, & f_j \text{ labeled “supports } O_2\text{”}. \end{cases}$$

Latent CPT computation Each latent variable L_k aggregates a subset of factors whose labels yield counts

$$\text{counts}[L_k] = \{ \text{Outcome1} : c_k^1, \quad \text{Outcome2} : c_k^2, \quad \text{Neutral} : c_k^N \}.$$

We apply Laplace smoothing (α) and split neutrals evenly:

$$\tilde{c}_k^1 = c_k^1 + \frac{1}{2} c_k^N + \alpha, \quad \tilde{c}_k^2 = c_k^2 + \frac{1}{2} c_k^N + \alpha,$$

then compute

$$P(L_k = 1 | O_1) = \frac{\tilde{c}_k^1}{\tilde{c}_k^1 + \tilde{c}_k^2}, \quad P(L_k = 1 | O_2) = \frac{\tilde{c}_k^2}{\tilde{c}_k^1 + \tilde{c}_k^2}.$$

Enumerating all 2^n latent-state vectors $x \in \{0, 1\}^n$, we form the likelihoods $\prod_k P(L_k = x_k | O_i)$, multiply by the prior $P(O_i) = 0.5$, and normalize:

$$P(x | O_i) = \frac{P(O_i) \prod_k P(L_k = x_k | O_i)}{\sum_{j=1}^2 P(O_j) \prod_k P(L_k = x_k | O_j)}.$$

Probability product approximation We approximate the marginal likelihood of evidence \mathcal{E} under model M by

$$P(\mathcal{E} | M) \approx P(O_1 | \mathcal{E}, M) \cdot P(O_2 | \mathcal{E}, M).$$

A.4 AGGREGATION METHODS

Linear Opinion Pool (LOP) Given per-model posteriors $P(O_1 | \mathcal{E}, M)$, LOP aggregates them as

$$P_{\text{LOP}}(O_1 | \mathcal{E}) = \sum_{M \in \{\text{NB}, \text{CBN}\}} w_M P(O_1 | \mathcal{E}, M).$$

We use fixed weights tailored to the underlying LLM: for Qwen2.5-32b, $w_{\text{NB}} = 0.8$ and $w_{\text{CBN}} = 0.2$; for Qwen2.5-72b, $w_{\text{NB}} = w_{\text{CBN}} = 0.5$.

Bayesian Model Averaging (BMA) Under BMA, the final posterior is a weighted sum of each model’s predictions:

$$P_{\text{BMA}}(O_1 | \mathcal{E}) = \sum_{M \in \{\text{NB, CBN}\}} P(M | \mathcal{E}) P(O_1 | \mathcal{E}, M),$$

where the model weights are proportional to the prior times the model evidence:

$$P(M | \mathcal{E}) \propto P(M) P(\mathcal{E} | M), \quad P(M) = \frac{1}{2}.$$

In practice, we approximate the evidence by the product of the two outcome posteriors, $P(\mathcal{E} | M) \approx P(O_1 | \mathcal{E}, M) \cdot P(O_2 | \mathcal{E}, M)$, and normalize these to obtain $P(M | \mathcal{E})$. Finally, the aggregated probability is

$$P_{\text{BMA}}(O_1 | \mathcal{E}) = \sum_M w_M P(O_1 | \mathcal{E}, M).$$

This “product-of-posteriors” is only an approximation: the true evidence requires summing over all latent or factor assignments, $P(\mathcal{E} | M) = \sum_x P(\mathcal{E}, x | M)$, which becomes intractable as the factor space grows.

A.5 EXPERIMENT DATASET DESCRIPTION

COMMON2SENSE Singh et al. (2021) introduces a multi-domain commonsense reasoning and planning benchmark comprising true/false natural language statements. We select only the *comparative reasoning* instances in which a smaller pre-trained model shows low confidence. Each statement is paired with its GPT-4-generated opposite, yielding two outcomes per scenario. For each outcome, 10 supporting conditions are generated and then filtered via reverse verification to ensure quality. The original test split contained 350 instances; to enable a fairer evaluation, the first three authors of this paper manually expanded it to 530 instances—strictly by adding contrasting conditions to the existing scenarios and outcomes, following the annotation protocol of Feng et al. (2025). The final dataset comprises 216 scenarios and 3,822 instances, with an average of 9 conditions per outcome.

TODAY Feng et al. (2022) proposes a temporal reasoning dataset where the effect of appending an extra sentence on temporal relations is evaluated. It consists of 1,000 instances designed to probe fine-grained temporal inference under controlled scenario modifications.

PLASMA Brahman et al. (2023) focuses on plan revision under a new condition. For each of 279 scenarios, GPT-4 identifies and rewrites the altered step into two alternative outcomes. The less common outcome in each pair is selected, and 5 supporting conditions are generated to favor it. The resulting dataset comprises 279 scenarios and 1,395 instances, with exactly 5 conditions per outcome.

Fact-Checking Benchmarks In addition, we sample 100 instances (with random seed fixed to 42) from the test split of each of four widely used fact-checking datasets, treating the document as the condition U and the claim as the outcome O , with the central scenario S_{cen} left empty. The long-document contexts pose a challenge for robust factor-space construction.

- **COVID** (Saakyan et al., 2021): Scientific claims about COVID-19 paired with biomedical abstracts, reflecting high-precision domain reporting.
- **ExpertQA** (Malaviya et al., 2023): Expert-driven statements with detailed contextual explanations spanning medicine, law, and engineering.
- **CNN** (Tang et al., 2024): News-derived claims with multi-sentence evidential contexts, suitable for long-range factual consistency evaluation.
- **XSum** (Tang et al., 2024): Headline-style summaries paired with full article text, emphasizing abstractiveness under concise outcomes.

Table 7 reports the average character lengths of outcome and condition in each dataset.

Dataset	Outcome	Condition
Common2Sense	79.78	80.98
Plasma	92.69	79.76
Today	79.08	71.07
COVID	89.37	505.04
ExpertQA	163.20	2399.16
CNN	296.31	2972.89
XSum	127.25	1875.37

Table 7: Average character lengths of outcome and condition across all datasets.

A.6 HYPERPARAMETER SETTINGS

Table 8 provides a comprehensive list of the key hyperparameters used in our experiments. **Decision Stage.** For downstream fact-checking, we convert probabilistic outputs into categorical labels by applying a fixed decision threshold: a claim is marked as `support` only when its predicted probability exceeds 0.9, otherwise it is treated as `unsupported`. Similar threshold-based rules are routinely adopted in recent fact-verification systems to balance precision and recall; for instance, AMREx employs dataset-specific entailment thresholds to separate “Supports”, “Refutes”, and “Not Enough Info” classes (Jayaweera et al., 2024).

Stage	Parameter	Value	Description
Iterative Factor Generation			
Factor Generation	<i>For Common2Sense & Plasma datasets:</i>		
	Target factor count (K)	80	Target number of unique factors.
	Batch size (b)	10	Sentences generated per LLM call.
	Max rounds (T_{\max})	20	Maximum number of generation rounds.
	<i>For Today, COVID, XSum, CNN, & ExpertQA datasets:</i>		
	Target factor count (K)	40	Target number of unique factors.
	Batch size (b)	5	Sentences generated per LLM call.
	Max rounds (T_{\max})	10	Maximum number of generation rounds.
Context-Aware Factor Mapping			
Hierarchical Retrieval	Top- K_1 clusters	3	Number of top clusters to retrieve in the coarse search.
	Top- K_2 factors	5	Number of top factors to retrieve per cluster in the fine search.
	Prototype weight (α)	0.5	Weight balancing cluster theme vs. factor content in prototypes.
Self-Consistent Filtering	Voting rounds (R)	3	Number of LLM calls for the majority voting mechanism.
	Vote ratio	0.5	Ratio for calculating the vote threshold τ .
Probabilistic Inference			
Parameter Elicitation	Smoothing alpha (ϵ)	0.5	The Laplace smoothing factor for CPT stabilization.
	LLM sampling temp.	0.5	Sampling temperature for LLM-based parameter elicitation.
	LLM parse retries	20	Maximum retries for eliciting valid parameters from the LLM.
Fact Checking (COVID, XSum, CNN, & ExpertQA datasets) threshold			
Decision-making	Probability threshold (τ_{dec})	0.9	Minimum probability required to label as “support.”

Table 8: Hyperparameter settings used in our experiments. Note that settings for factor generation vary by dataset group to accommodate different complexity levels, applies a decision threshold for support classification.

A.7 SENSITIVITY TO K AND METHOD COMPARISON

Why we finally choose KNN and ($K_1=3, K_2=5$) All analyses here are conducted on the HARBOR model built on Qwen2.5-72b. KNN consistently delivers a low Unknown Rate and stable average F1 in our tests, avoiding BM25’s high unknown proportion and FAISS’s fluctuations. Very small K values (2/3) under-cover relevant factors, while enlarging them to 4/8 actually degrades performance relative to our tuned (3, 5) setting because more noisy neighbors are included. Hence, the mid-range (3, 5) strikes a better coverage–precision trade-off. Moreover, KNN’s Euclidean distance is robust to our factor-embedding scale and requires minimal tuning and engineering overhead. **Accordingly, the main results reported in Table 1 are obtained with KNN using ($K_1=3, K_2=5$).**

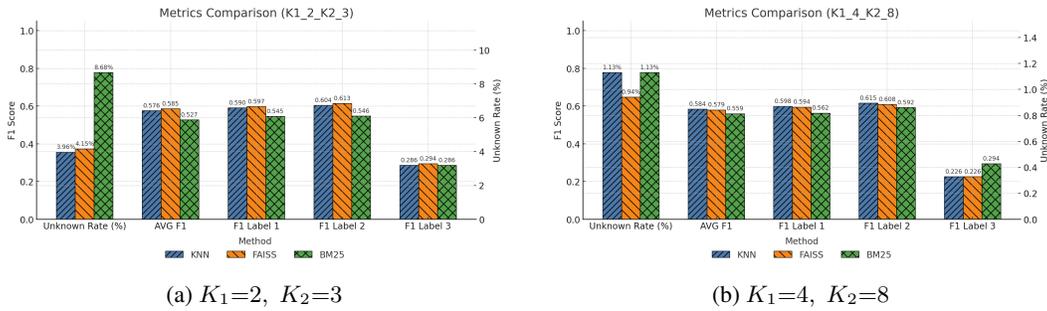


Figure 4: Unknown Rate and per-class F1 comparison across KNN, FAISS, and BM25 under two K settings.

A.8 FURTHER ANALYSIS: CLUSTERING AND LLM CALIBRATION BIAS

A.8.1 CLUSTERING QUALITY AND ALGORITHM CHOICE

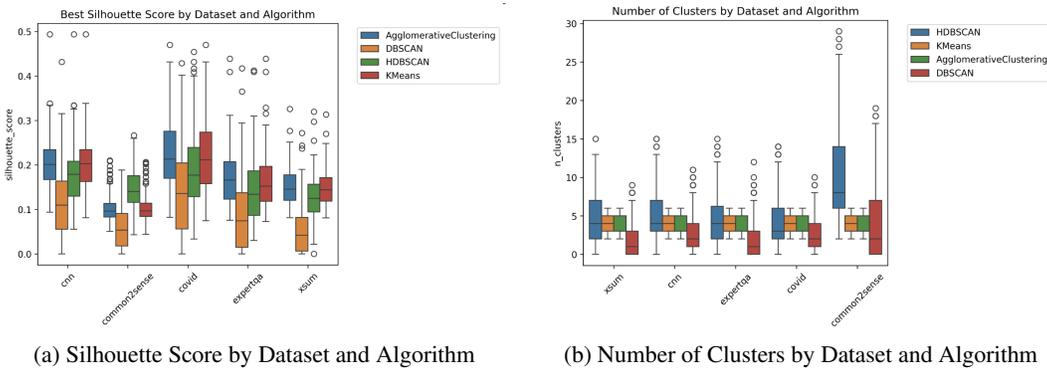


Figure 5: Comparison of clustering quality and flexibility across algorithms.

We adopt **HDBSCAN** as our clustering algorithm due to its ability to automatically determine the number of clusters based on data density, which is especially important in open-ended and semantically diverse factor spaces. As shown in Figure 5(b), HDBSCAN produces a wider and more adaptive range of cluster counts across datasets compared to other algorithms.

Although fixed- k algorithms like KMeans or Agglomerative Clustering occasionally yield higher silhouette scores on certain datasets, their performance heavily depends on the manually chosen number of clusters, which may not generalize across tasks. This fixed structure risks oversimplifying or overfragmenting the semantic space when the true number of factor types varies with input complexity.

DBSCAN also determines clusters automatically, but tends to merge semantically distinct groups or produce very few clusters, especially under high noise or sparse factor conditions.

Overall, HDBSCAN provides a good trade-off between clustering quality and adaptability, making it suitable for our pipeline that operates across datasets with diverse semantic granularity.

A.8.2 EFFECT OF PROBABILITY-ESTIMATOR LLM AND CALIBRATION

To test how sensitive HARBOR is to such calibration effects, we replace Qwen2.5-72B with the smaller GPT-4o-mini while keeping the factor space, mapping pipeline, and NB+CBN inference fixed, we evaluate on CNN, Today, Plasma, and ExpertQA; as shown in Table 9, all changes are

Dataset	Qwen2.5-72B	GPT-4o-mini	Δ (mini – 72B)
CNN	62.9	62.3	-0.6
ExpertQA	60.5	58.8	-1.7
Plasma	76.2	73.8	-2.4
Today	82.7	83.8	+1.1

Table 9: Task performance (%) of HARBOR with different probability-estimator LLMs.

within about ± 2.5 points, indicating limited sensitivity to the specific choice of probability-estimator LLM.

We further analyze how changing the probability-estimator LLM reshapes factor-level scores: Table 10 reports distributional distances between factor-score distributions under Qwen2.5-72B and GPT-4o-mini, and Figure 6 visualizes the corresponding smoothed factor-level probability profiles on CNN, TODAY, PLASMA, and EXPERTQA, showing that only CNN exhibits a pronounced distributional shift while overall factor patterns remain similar, consistent with the small performance differences in Table 9.

Dataset	JS div.	Wass. dist.	KS p -value	Diff.
CNN	0.356	0.074	0.0001	significant
TODAY	0.235	0.042	0.7012	not significant
PLASMA	0.205	0.040	0.7287	not significant
EXPERTQA	0.313	0.045	0.0773	not significant

Table 10: Distributional distances between factor scores under Qwen2.5-72B and GPT-4o-mini.

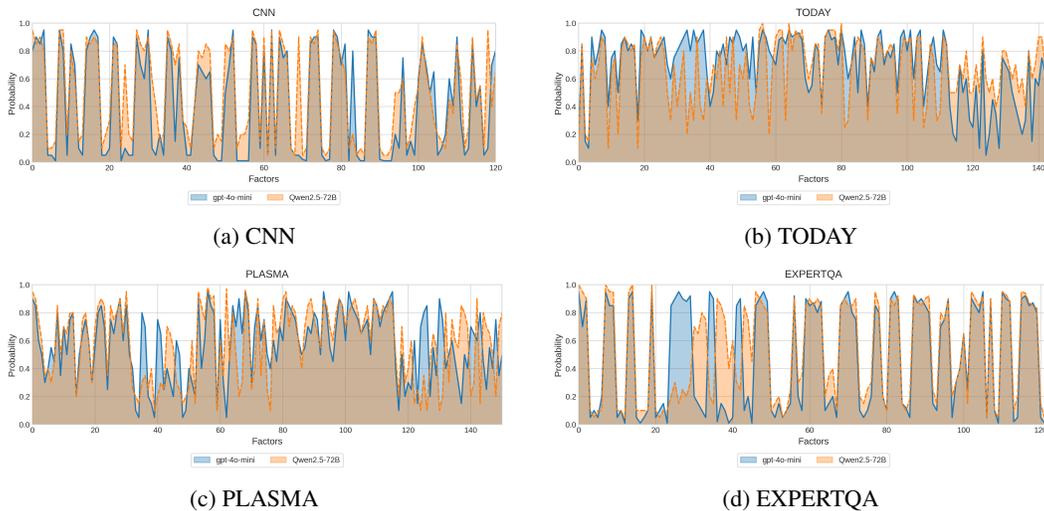


Figure 6: Smoothed factor-level probability profiles under Qwen2.5-72B and GPT-4o-mini on four datasets.

A.9 DETAILED ALGORITHM SPECIFICATIONS

This section provides the detailed pseudo-code for the three core stages of the HARBOR framework, as described in Section 4. Algorithm 1 details the construction of the hierarchical factor space. It operates by iteratively generating raw factors, validating them via self-consistency voting, and finally organizing them using a pipeline of embedding, clustering, and LLM-based thematic pruning. Algorithm 2 specifies the multi-stage process for mapping a condition to a relevant factor set. The pipeline begins with a broad, two-level hierarchical retrieval to generate candidates, followed by a voting-based filtering step and a final reflective refinement to ensure high precision. Algorithm 3

1080 **Algorithm 1:** Iterative Factor Generation via **Bottom-up Abduction**

1081 **Input:** Scenario description s , primary outcome O_1 , secondary outcome O_2 , LLM \mathcal{M} ,
1082 target factor count K , batch size b , maximum rounds T_{\max} , clustering flag C
1083 **Output:** Hierarchical factor space $\tilde{\mathcal{F}}$, factor–outcome mapping M , clustering statistics

```

1084 1  $\mathcal{F}^{(0)} \leftarrow \emptyset$ ;  

1085 2 for  $t \leftarrow 1$  to  $T_{\max}$  do  

1086 3   if  $|\mathcal{F}^{(t-1)}| \geq K$  then  

1087 4     break  

1088   // Contextual sentence generation  

1089 5    $S^{(t)} \leftarrow \text{GENERATESENTENCES}(s, O_1, O_2, b, \mathcal{M})$ ;  

1090   // Factor harvesting and validation  

1091 6    $\Delta\mathcal{F}^{(t)} \leftarrow \text{EXTRACTFACTORS}(S^{(t)}, \mathcal{M})$ ;  

1092 7    $\mathcal{F}^{(t)} \leftarrow \mathcal{F}^{(t-1)} \cup \Delta\mathcal{F}^{(t)}$ ;  

1093 8  $M \leftarrow \text{VOTESUPPORT}(\mathcal{F}^{(T)}, s, O_1, O_2, \mathcal{M})$ ;  

1094 9  $\mathcal{F}_{\text{validated}} \leftarrow \{f \in \mathcal{F}^{(T)} : M[f] \in \{\text{“}O_1\text{”}, \text{“}O_2\text{”}, \text{“}Neutral\text{”}\}\}$ ;  

1095 10 if  $C = \text{True}$  then  

1096 11   // Encode factors into dense representations  

1097 12    $E \leftarrow \{e_f : f \in \mathcal{F}_{\text{validated}}\}$  using MiniLM;  

1098 13   // Dimensionality reduction and clustering  

1099 14    $E_{\text{reduced}} \leftarrow \text{UMAP}(E)$ ;  

1100 15    $\mathcal{C}_{\text{raw}} \leftarrow \text{HDBSCAN}(E_{\text{reduced}})$ ;  

1101 16   // LLM-guided thematic organization and pruning  

1102 17    $\mathcal{C}_{\text{themed}} \leftarrow \text{ASSIGNTHEMES}(\mathcal{C}_{\text{raw}}, \mathcal{M})$ ;  

1103 18    $\tilde{\mathcal{F}} \leftarrow \text{PRUNEREDUNDANCY}(\mathcal{C}_{\text{themed}}, M, \mathcal{M})$ ;  

1104 19   compute clustering statistics;  

1105 20 else  

1106 21    $\tilde{\mathcal{F}} \leftarrow \{\text{“default”}, \mathcal{F}_{\text{validated}}\}$ ;  

1107 22   set default clustering statistics;  

1108 23 return  $(\tilde{\mathcal{F}}, M, \text{clustering statistics})$ ;
```

1111 outlines the procedure for transforming a set of mapped factors into a final probability. It involves
1112 parameterizing two parallel models—Naïve Bayes and a latent-augmented Causal Bayesian Net-
1113 work—with probabilities elicited from an LLM, and then aggregating their posteriors for a robust
1114 final estimate.

1117 A.10 ADDITIONAL DETAILS FOR THE BIRD-ON-HARBOR ABLATIONS

1118 **Complete-information spaces in BIRD vs. HARBOR.** Conceptually, BIRD assumes that each factor
1119 takes two mutually exclusive attribute values. For a given scenario, these categorical factors form a
1120 low-arity product space, so the complete-information training set can be obtained by (conceptually)
1121 enumerating this Cartesian product and sampling from it. In HARBOR, factors are standalone
1122 evidence-like statements drawn from a global pool and organized into clusters and themes, not tied
1123 to a fixed two-value choice per slot. A complete-information assignment is thus a subset of this
1124 pool, and naively forming a Cartesian product over clusters would both generate many incoherent
1125 combinations and lead to a combinatorial explosion.

1126 **Structured Monte Carlo scheme for CPT training.** To plug BIRD’s CPT-based outcome model
1127 into HARBOR’s factor space, we therefore adopt a structured Monte Carlo scheme. For each mapped
1128 condition, we fix a “base” set of factors returned by the mapping pipeline, randomly complete it with
1129 a bounded number of additional factors up to a maximum length L , and treat the resulting complete
1130 assignments

$$\{f^{(1)}, \dots, f^{(m)}\}$$

Algorithm 2: Context-Aware Factor Mapping

```

1134 Input: Hierarchical factor space  $\tilde{\mathcal{F}} = \{(C_j, F_j)\}$ , unclustered factors  $F_u$ , condition  $u$ ,
1135 cluster top- $K_1$ , factor top- $K_2$ , voting rounds  $R$ , vote threshold  $\tau$ , balance weight  $\alpha$ 
1136 Output: High-precision factor set  $\mathcal{F}^*(u)$ 
1137
1138 1  $\mathcal{F}_{\text{cand}} \leftarrow \emptyset$ ;
1139
1140 2 if  $\tilde{\mathcal{F}} \neq \emptyset$  (clustered factors exist) then
1141   // Compute cluster prototypes
1142   3 foreach  $(C_j, F_j) \in \tilde{\mathcal{F}}$  do
1143     4  $e_{C_j} \leftarrow \alpha \cdot \text{EMBED}(\text{theme}(C_j)) + (1 - \alpha) \cdot \frac{1}{|F_j|} \sum_{f \in F_j} \text{EMBED}(f)$ ;
1144     // Build KNN indices
1145     5  $\text{KNN}_{\text{clusters}} \leftarrow \text{BUILDKNN}(\{e_{C_j}\}, K_1)$ ;
1146     6 foreach  $(C_j, F_j) \in \tilde{\mathcal{F}}$  do
1147       7  $\text{KNN}_{F_j} \leftarrow \text{BUILDKNN}(\{\text{EMBED}(f) : f \in F_j\}, K_2)$ ;
1148     // Coarse-to-fine retrieval
1149     8  $C_{\text{sel}} \leftarrow \text{KNN}_{\text{clusters}}.\text{query}(\text{EMBED}(u))$ ;
1150     9 foreach  $C_j \in C_{\text{sel}}$  do
1151       10  $F_{\text{retrieved}} \leftarrow \text{KNN}_{F_j}.\text{query}(\text{EMBED}(u))$ ;
1152       11  $\mathcal{F}_{\text{cand}} \leftarrow \mathcal{F}_{\text{cand}} \cup F_{\text{retrieved}}$ ;
1153
1154 12 if  $F_u \neq \emptyset$  (unclustered factors exist) then
1155     13  $\text{KNN}_{\text{unclustered}} \leftarrow \text{BUILDKNN}(\{\text{EMBED}(f) : f \in F_u\}, K_2)$ ;
1156     14  $\mathcal{F}_{\text{cand}} \leftarrow \mathcal{F}_{\text{cand}} \cup \text{KNN}_{\text{unclustered}}.\text{query}(\text{EMBED}(u))$ ;
1157 15 initialize vote counts  $v[f] \leftarrow 0$  for all  $f \in \mathcal{F}_{\text{cand}}$ ;
1158 16 for  $r \leftarrow 1$  to  $R$  do
1159     17  $S^{(r)} \leftarrow \text{LLMSELECT}(u, \mathcal{F}_{\text{cand}}, \mathcal{M})$ ;
1160     18 foreach  $f \in S^{(r)}$  do
1161       19  $v[f] \leftarrow v[f] + 1$ ;
1162 20  $\mathcal{F}_{\text{vote}} \leftarrow \{f \in \mathcal{F}_{\text{cand}} : v[f] \geq \tau\}$ ;
1163 21  $\mathcal{F}^*(u) \leftarrow \text{REFLECTIVEREFINE}(u, \mathcal{F}_{\text{vote}}, \mathcal{M})$ ;
1164 22 return  $\mathcal{F}^*(u)$ ;

```

as samples from a distribution π over the intractable complete-information space. This allows us to approximate the BIRD training objective

$$\mathcal{L}(\theta) = \mathbb{E}_{f \sim \pi} [\ell(P_\theta(O | f), y(f))] \approx \hat{\mathcal{L}}_m(\theta) = \frac{1}{m} \sum_{i=1}^m \ell(P_\theta(O | f^{(i)}), y(f^{(i)})),$$

where $P_\theta(O | f)$ is the outcome model with parameters θ , ℓ is the training loss, and $y(f^{(i)})$ is the LLM’s coarse probability assessment under complete information $f^{(i)}$. In the experiments we match the original BIRD setting with $m = 128$ sampled assignments per scenario.

Limitations and motivation for LLM-based priors. This Monte Carlo construction is practical but imperfect: with a finite sampling budget, some factors may never appear in the training set and remain tied to their priors, and uniform completion from a large factor pool can under-cover parts of the space or induce slightly unrealistic co-occurrences. Together with prior work showing that LLMs can provide informative priors and probabilistic knowledge for Bayesian models (e.g., Gouk and Gao, 2024; Riegler et al., 2025; Nafar et al., 2025), this motivates HARBOR’s main design choice of eliciting coarse factor-level probabilities directly from the LLM instead of relying purely on CPT fitting.

```

1188 Algorithm 3: Probabilistic Inference with Elicited Parameters
1189
1190 Input: Mapped factor set  $\mathcal{F}^*(u)$ , outcome hypotheses  $\{O_1, O_2\}$ , LLM  $\mathcal{M}$ ,
1191 aggregation weights  $w_{\text{NB}}, w_{\text{CBN}}$ , smoothing parameter  $\epsilon$ 
1192 Output: Aggregated posterior probability  $P_{\text{agg}}(O_1 | \mathcal{F}^*(u))$ 
1193 // Factor-level parameters for both models
1194 1 foreach  $f \in \mathcal{F}^*(u)$  do
1195 2    $\theta_f \leftarrow \text{ELICITPOSTERIOR}(f, O_1, \mathcal{M})$  //  $\theta_f \approx P(O_1 | f)$ 
1196 3    $\theta_f \leftarrow \text{SMOOTH}(\theta_f, \epsilon)$ ;
1197 // Build NB model with conditional independence assumption
1198 4 Build NB structure:  $O \rightarrow f$  for all  $f \in \mathcal{F}^*(u)$ ;
1199 5 foreach  $f \in \mathcal{F}^*(u)$  do
1200 6   Set CPT:  $P(f | O_1) = \theta_f, P(f | O_2) = 1 - \theta_f$ ;
1201 7 Set uniform prior:  $P(O_1) = P(O_2) = 0.5$ ;
1202 8  $P_{\text{NB}} \leftarrow \text{INFERPOSTERIOR}(\text{NB}, \mathcal{F}^*(u))$  using Eq. 2;
1203 // Latent variable discovery and structure learning
1204 9  $\mathcal{L} \leftarrow \text{ELICITLATENTVARIABLES}(\mathcal{F}^*(u), \mathcal{M})$ ;
1205 10 partition  $\leftarrow \text{ASSIGNFACTORS تولاتENTS}(\mathcal{F}^*(u), \mathcal{L}, \mathcal{M})$ ;
1206 // Latent-level parameter elicitation
1207 11 foreach  $L_i \in \mathcal{L}$  do
1208 12    $p_{i1} \leftarrow \text{ELICITCONDITIONAL}(L_i, O_1, \mathcal{M})$  //  $p_{i1} = P(L_i = 1 | O_1)$ 
1209 13    $p_{i2} \leftarrow \text{ELICITCONDITIONAL}(L_i, O_2, \mathcal{M})$  //  $p_{i2} = P(L_i = 1 | O_2)$ 
1210 14    $(p_{i1}, p_{i2}) \leftarrow \text{SMOOTH}((p_{i1}, p_{i2}), \epsilon)$ ;
1211 // Build CBN structure and parameterize CPTs
1212 15 Build CBN structure:  $L_i \rightarrow f$  for  $f \in \text{partition}(L_i)$ ,  $L_i \rightarrow O$  for all  $L_i$ ;
1213 16 foreach  $L_i \in \mathcal{L}$  do
1214 17   Set uniform prior:  $P(L_i = 1) = 0.5$ ;
1215 18   foreach  $f \in \text{partition}(L_i)$  do
1216 19     Set CPT:  $P(f | L_i = 1) = \theta_f, P(f | L_i = 0) = 1 - \theta_f$ ;
1217 20 Derive outcome CPT:  $P(O | L_1, \dots, L_k)$  using  $\{p_{i1}, p_{i2}\}_{i=1}^k$  via Bayes' rule;
1218 21  $P_{\text{CBN}} \leftarrow \text{INFERPOSTERIOR}(\text{CBN}, \mathcal{F}^*(u))$  using variable elimination;
1219 22  $P_{\text{agg}} \leftarrow w_{\text{NB}} \cdot P_{\text{NB}} + w_{\text{CBN}} \cdot P_{\text{CBN}}$ ;
1220 23 return  $P_{\text{agg}}$ ;

```

B APPENDIX B

B.1 LLM USAGE STATEMENT

We clarify the role of LLMs in our research workflow. Specifically, LLMs were *only* used to assist with (i) code writing (e.g., generating boilerplate code, debugging minor errors, and improving readability), and (ii) polishing the writing of the paper text (e.g., improving clarity, grammar, and style). No experimental results, theoretical analyses, or substantive scientific claims in this paper were produced by LLMs. All methodological designs, experiments, and conclusions are solely the work of the authors.

B.2 COMPREHENSIVE EVALUATION OF FACTOR SPACES AND MAPPINGS

In this subsection, we first compare the factor spaces generated by HARBOR and BIRD (Table 11), then examine their mapping performance under specific cooking conditions (Table 12), and finally analyze the organization of latent nodes for Outcome1 (Table 13). As shown in Table 11, HARBOR produces a more hierarchically structured set of categories that cover safety and nutrition dimensions while maintaining low redundancy, whereas BIRD's space exhibits multiple synonymous and fragmented entries. Table 12 demonstrates that HARBOR successfully maps key factors—such as food safety, temperature control, and process precision—under conditions like pathogen elimination at high temperatures, large-batch cooking, and high-altitude environments, filling gaps left by BIRD. Furthermore, Table 13 presents HARBOR's five latent dimensions for Outcome1 (TextureLat, Flavor-

1242 Lat, EfficiencyLat, SafetyLat, and ProcessControlLat), each with a logically coherent and balanced
1243 distribution of factors, offering robust support for downstream factor analysis and visualization. Over-
1244 all, the HARBOR method shows clear advantages in generating a high-cohesion, low-redundancy,
1245 and practically actionable factor space.
1246

1247 **Scenario:** The efficiency of cooking noodles is being compared between hot water and warm water.
1248 **Outcome1:** Noodles cook much more efficiently in hot water than they do in warm water.
1249 **Outcome2:** Noodles cook much more efficiently in warm water than they do in hot water.
1250

1251 **HARBOR Generated Factor Space**

1252 Clumping Prevention	hard inside; reduced sticking; clumping reduction; rapid softening
1253 Efficiency Reductions	energy and time efficiency; practicality; energy efficiency
1254 Texture & Flavor	texture preservation; al dente texture; flavor; dispersing flavorings; texture inconsistency; uniform texture; longer gelatinization; palatability; better flavor infusion
1255	
1256 Cooking Efficiency	ensures even cooking and texture retention; inefficient activation of cooking process; professional kitchens; cooking environment; slow outer layer cooking; reduced cooking time; gradual cooking; difficulty controlling cooking process
1257	
1258	
1259	
1260 Starch Digestion Efficiency	effective breakdown of starches; starch breakdown inefficiency; digestibility; gluten breakdown
1261	
1262 Temperature Control	inconsistent temperature control; insufficient thermal energy; lower temperature; consistent use of hot water; noodle hydration; evaporation; temperature of water; pre-soaking benefits; delicate noodles
1263	
1264 Cooking Safety & Precision	risk of undercooking or overcooking; avoiding overcooking/mushiness; food safety (reduces risk of foodborne illness); overcooking prevention
1265	
1266	

1267 **BIRD Generated Factor Space**

1268 Cooking Temperature and Time	Hot water significantly reduces cooking time by breaking down starches more quickly; Warm water requires a longer cooking time for noodles to become tender
1269	
1270	
1271 Noodle Hydration Process	Hot water leads to faster hydration and cooking of noodles; Warm water results in a slower but more even hydration process
1272	
1273 Recommended Cooking Time	Shorter recommended cooking time in hot water; Longer recommended cooking time in warm water
1274	
1275 Energy Efficiency	Hot water uses more energy to heat up initially but cooks faster, potentially saving overall energy; Warm water uses less energy to heat up but requires a longer cooking time, potentially using more energy overall
1276	
1277 Texture and Quality	Hot water can lead to a more uniform texture and prevent clumping of noodles; Warm water may result in a softer texture but can cause the noodles to become mushy if cooked for too long
1278	
1279	
1280 Cooking Efficiency	Significantly reduces cooking time with hot water; Requires longer cooking time with warm water
1281	
1282 Noodle Texture	Hot water ensures a more uniform texture; Warm water can lead to a softer texture
1283	
1284 Starch Breakdown	High temperature in hot water breaks down starches more quickly; Lower temperature in warm water results in slower starch breakdown
1285	
1286 Energy Consumption	Hot water uses more initial energy but cooks faster; Warm water uses less initial energy but cooks longer
1287	
1288 Hydration Process	Hot water leads to faster hydration; Warm water results in a slower but more even hydration
1289	
1290 Clumping Prevention	Hot water helps prevent noodles from clumping; Warm water may increase the risk of noodles clumping
1291	

Table 11: An example of HARBOR and BIRD generated factor spaces

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349

Example Prompt for Generating Supporting/Refuting Sentences

System
You are an AI assistant that helps people make decisions.
Generate {n} diverse supporting or refuting sentences for scenario:
{scenario}, comparing '{Outcome}' vs '{oppo_Outcome}'.

User
Scenario: Alice is training for a marathon.
Outcome: Running on a treadmill improves her endurance.
Opposite Outcome: Running on a treadmill does not improve her endurance.
Generate 2 sentences.

Assistant
1. Treadmill training allows Alice to maintain a consistent pace and monitor heart rate, boosting her aerobic capacity.
2. The treadmill's adjustable incline simulates hill workouts, increasing leg strength and stamina.

User
Scenario: Bob studies every evening.
Outcome: Studying in short, focused bursts enhances retention.
Opposite Outcome: Studying in short, focused bursts does not enhance retention.
Generate 2 sentences.

Assistant
1. Brief study sessions with breaks prevent mental fatigue and improve long-term recall.
2. Frequent reviews in small intervals reinforce memory pathways, aiding retention.

Figure 7: Example Prompt for Generating Supporting or Refuting Sentences

1350	Scenario: The efficiency of cooking noodles is being compared between hot water and warm water.		
1351	Outcome1: Noodles cook much more efficiently in hot water than they do in warm water.		
1352	Outcome2: Noodles cook much more efficiently in warm water than they do in hot water.		
1353			
1354	Condition	BIRD Mapped Factors	HARBOR Mapped Factors
1355			
1356	The higher temperature of hot water can accelerate the gelatinization process in starchy foods like noodles.	Shorter recommended cooking time in hot water; High temperature in hot water breaks down starches more quickly; Hot water can lead to a more uniform texture and prevent clumping of noodles; Hot water leads to faster hydration and cooking of noodles	noodle hydration; temperature of water; effective breakdown of starches; digestibility; better flavor infusion; dispersing flavorings; flavor
1357			
1358			
1359			
1360			
1361			
1362			
1363			
1364			
1365			
1366	Cooking in hot water reduces the amount of total time that the noodles spend in the water.	Shorter recommended cooking time in hot water; High temperature in hot water breaks down starches more quickly; Significantly reduces cooking time with hot water; Hot water leads to faster hydration and cooking of noodles; Hot water leads to faster hydration; Hot water significantly reduces cooking time by breaking down starches more quickly	reduced cooking time; ensures even cooking and texture retention; noodle hydration; temperature of water; avoiding overcooking/mushiness; overcooking prevention
1367			
1368			
1369			
1370			
1371			
1372			
1373			
1374			
1375			
1376	Hot water helps in killing any potential foodborne pathogens or microbes present in the noodles.	None	consistent use of hot water; temperature of water; food safety (reduces risk of foodborne illness)
1377			
1378			
1379			
1380	You're cooking a large quantity of noodles.	None	gradual cooking; inefficient activation of cooking process; ensures even cooking and texture retention; cooking environment; avoiding overcooking/mushiness; overcooking prevention; risk of undercooking or overcooking; starch breakdown inefficiency; effective breakdown of starches
1381			
1382			
1383			
1384			
1385			
1386			
1387			
1388			
1389			
1390	You are at a high altitude where water boils at lower temperatures than at sea level.	None	temperature of water; lower temperature; insufficient thermal energy; risk of undercooking or overcooking; slow outer layer cooking; cooking environment; inefficient activation of cooking process; consistent use of hot water; evaporation; overcooking prevention; avoiding overcooking/mushiness; gradual cooking
1391			
1392			
1393			
1394			
1395			
1396			
1397			
1398			
1399			
1400			
1401			
1402			
1403			

Table 12: An example of BIRD and HARBOR conditions factor mappings

1404
 1405
 1406
 1407
 1408
 1409
 1410
 1411
 1412
 1413
 1414
 1415
 1416
 1417
 1418
 1419
 1420
 1421
 1422
 1423
 1424
 1425
 1426
 1427
 1428
 1429
 1430
 1431
 1432
 1433
 1434
 1435
 1436
 1437
 1438
 1439
 1440
 1441
 1442
 1443
 1444
 1445
 1446
 1447
 1448
 1449
 1450
 1451
 1452
 1453
 1454
 1455
 1456
 1457

Scenario: The efficiency of cooking noodles is being compared between hot water and warm water.
Outcome1: Noodles cook much more efficiently in hot water than they do in warm water.
Outcome2: Noodles cook much more efficiently in warm water than they do in hot water.

Parent Node	Latent Node	Child Nodes (Factors)
Outcome1	TextureLat	slow outer layer cooking; texture preservation; ensures even cooking and texture retention; al dente texture; reduced cooking time; overcooking prevention; uniform texture; gradual cooking; palatability; hard inside; starch breakdown inefficiency; reduced sticking; rapid softening; delicate noodles; avoiding overcooking/mushiness; gluten breakdown; longer gelatinization; texture inconsistency
Outcome1	FlavorLat	noodle hydration; flavor; dispersing flavorings; better flavor infusion; digestibility; food safety (reduces risk of foodborne illness)
Outcome1	EfficiencyLat	energy and time efficiency; energy efficiency; evaporation; practicality; pre-soaking benefits; consistent use of hot water; cooking environment
Outcome1	SafetyLat	temperature of water; lower temperature; inconsistent temperature control; insufficient thermal energy
Outcome1	ProcessControlLat	clumping reduction; effective breakdown of starches; inefficient activation of cooking process; risk of undercooking or overcooking; difficulty controlling cooking process; professional kitchens

Table 13: An example of latent nodes organized under parent Outcome1 with their corresponding factors

1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511

Few-Shot Examples for Factor Extraction

System
Extract distinct factors from these sentences. Think step by step about what factors are mentioned, then provide your final answer as a JSON array.

User
Extract distinct factors from these sentences as a JSON array.
1. Treadmill training allows maintaining a consistent pace and monitoring heart rate, boosting aerobic capacity.
2. The treadmill's adjustable incline simulates hill workouts, increasing leg strength and stamina.

Assistant
Let me analyze these sentences to identify the key factors:

- Pace consistency (ability to maintain steady speed)
- Heart rate monitoring (tracking cardiovascular response)
- Adjustable incline (variable difficulty/terrain simulation)
- Leg strength (muscle development)

Final answer: ["Pace consistency", "Heart rate monitoring", "Adjustable incline", "Leg strength"]

User
Extract distinct factors from these sentences as a JSON array.
1. Group work requires coordination between team members.
2. Individual tasks allow for personal control and efficiency.

Assistant
Let me identify the key factors from these sentences:

- Coordination requirements (need for team synchronization)
- Personal control (individual autonomy)
- Efficiency (productivity/effectiveness)
- Team collaboration (working together)

Final answer: ["Coordination requirements", "Personal control", "Efficiency", "Team collaboration"]

Figure 8: Few-shot prompt–response pairs for factor extraction.

1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565

Few-Shot Examples for Factor–Outcome Voting

System
Decide which outcome the factor supports. Reason briefly (1-2 sentences), then provide your final answer as a JSON object. Keep the explanation as short as possible--no extra commentary.

User
Scenario: Alice trains for a marathon.
Outcome1: Treadmill running improves endurance.
Outcome2: Treadmill running does not improve endurance.
Factor: Pace consistency
Decide which outcome this factor supports: Outcome1, Outcome2, or Both. Think step by step, then answer in JSON.

Assistant
Pace consistency forces a steady speed, building aerobic capacity and sustained effort.
Final answer: {"Pace consistency": "Outcome1"}

User
Scenario: Alice trains for a marathon.
Outcome1: Treadmill running improves endurance.
Outcome2: Treadmill running does not improve endurance.
Factor: Weather conditions
Decide which outcome this factor supports: Outcome1, Outcome2, or Both. Think step by step, then answer in JSON.

Assistant
Treadmill gives consistent conditions, yet outdoor weather readies race adaptability.
Final answer: {"Weather conditions": "Both"}

Figure 9: Few-shot prompt–response pairs for factor–outcome voting.

Few-Shot Examples for Theme Name Generation

System
Generate a concise English theme name (1-3 words) that captures the common topic of these factors.
Return only the theme name, no explanation.

User
Generate a theme name for these related factors:
["energy expenditure", "energy transfer efficiency"]

Assistant
Energy Efficiency

User
Generate a theme name for these related factors:
["precision control", "better control (accuracy)"]

Assistant
Control Precision

Figure 10: Few-shot prompt–response pairs for generating concise theme names.

1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619

Few-Shot Examples for Factor-Condition Mapping

System
You are an expert at analyzing logical relationships between conditions and factors. Given a condition and a list of candidate factors, select factors that have reasonable connections to the condition. Be inclusive rather than restrictive.

User
Scenario: A student is preparing for final exams.
Condition: The student spends more time in the library.
Candidate factors: ["Better time management", "More stress", "Increased social activities"]
Task: Select the factor list that is most reasonably related to the given condition. Think step by step about each factor's relevance, then provide your selection. Please respond with your reasoning followed by Final answer: and a JSON object in this exact format: {"answer": ["factor1", "factor2", ...]}.

Assistant
Let me analyze each factor: • "Better time management": Spending more time in the library suggests the student is organizing their study schedule better and dedicating focused time to studying. This is directly related. • "More stress": While exam preparation can be stressful, spending more time in the library is typically a positive study behavior, not necessarily indicating increased stress. • "Increased social activities": Libraries are generally quiet study spaces, not social venues. More library time would likely mean less social activities. Final answer: {"answer": ["Better time management"]}

User
Scenario: A company introduces remote working policies.
Condition: Employees can work from home twice a week.
Candidate factors: ["Improved work-life balance", "Reduced office costs", "More commuting"]
Task: Select the factor list that is most reasonably related to the given condition. Think step by step about each factor's relevance, then provide your selection. Please respond with your reasoning followed by Final answer: and a JSON object in this exact format: {"answer": ["factor1", "factor2", ...]}.

Assistant
Let me evaluate each factor: • "Improved work-life balance": Working from home twice a week allows employees to save commute time, have more flexibility, and better integrate work with personal life. This is directly related. • "Reduced office costs": With employees working from home part-time, the company needs less office space, utilities, and resources. This is a logical consequence. • "More commuting": This is contradictory--working from home twice a week would actually reduce commuting, not increase it. Final answer: {"answer": ["Improved work-life balance", "Reduced office costs"]}

Figure 11: Few-shot prompt-response pairs for factor-condition mapping.

1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673

Few-Shot Examples for Factor Reflection

System
You are performing a self-reflection task. Given a condition and a list of initially selected factors, review each factor with a LENIENT approach. Keep factors that have ANY reasonable connection to the condition. Only remove factors that are clearly irrelevant or contradictory. When uncertain, err on the side of keeping the factor. Think step by step about each factor, then provide your reasoning followed by 'Final answer:' and a JSON array of the factors to keep.

User
Condition: City implements a bike-sharing program.
Initially selected factors: ["Increased bike usage", "Higher car sales", "More traffic jams"] Task: Review and keep factors reasonably related to the condition. Think step by step about each factor's relevance, then 'Final answer:' with a JSON array.

Assistant
"Increased bike usage" is the direct result of more shared bikes--keep it.
"Higher car sales" has no clear link to bike sharing--remove it.
"More traffic jams" could occur if road space shifts to bikes and cars interact--keep it.
Final answer: ["Increased bike usage", "More traffic jams"]

User
Condition: Students study in a quiet library.
Initially selected factors: ["Better concentration", "Distractions from phones", "Improved retention", "Reduced social interaction"] Task: Review and keep factors reasonably related to the condition. Think step by step about each factor's relevance, then 'Final answer:' with a JSON array.

Assistant
"Better concentration" follows from a quiet space aiding focus--keep it.
"Distractions from phones" contradicts the library's purpose--remove it.
"Improved retention" naturally arises from focused study--keep it.
"Reduced social interaction" is a likely side-effect of silence--keep it.
Final answer: ["Better concentration", "Improved retention", "Reduced social interaction"]

Figure 12: Few-shot prompt-response pairs for lenient self-reflection on factor relevance.

1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727

Few-Shot Examples for Probability Estimation of Factor Support

User
 Given the scenario: "Comparing LED bulbs vs incandescent bulbs in home lighting."
 For each of the following factor values, please estimate the probability (a float between 0 and 1) that it supports Outcome1 (LED bulbs are more advantageous) rather than Outcome2 (incandescent bulbs are more advantageous).
 Return a JSON mapping.
 Factor values:
 ["Initial cost per bulb", "Energy consumption per hour", "Lifespan hours", "Color rendering index", "Warm color temperature", "Instant full brightness", "Dimmable compatibility", "Mercury content", "Heat generation", "Availability"]

Assistant
 Thought: LEDs excel in low energy use, longevity, low heat output, and no mercury, but cost more upfront, may have lower CRI, and vary in warmth, dimmability, and availability.
 Final answer:
 {
 "Initial cost per bulb": 0.30, "Energy consumption per hour": 0.95,
 "Lifespan hours": 0.90, "Color rendering index": 0.25,
 "Warm color temperature": 0.40, "Instant full brightness": 0.50,
 "Dimmable compatibility": 0.35, "Mercury content": 0.85,
 "Heat generation": 0.88, "Availability": 0.45
 }

User
 Given the scenario: {scenario}
 For each of the following factor values, please estimate the probability (a float between 0 and 1) that it supports Outcome1: {outcome1} rather than Outcome2: {outcome2}.
 As reference (but not absolute) here are some initial estimates: {prior_text}
 Think step by step about each factor's relation to the outcomes, and provide your probability estimates. Return a JSON mapping.
 Factor values:
 {Factor values}

Figure 13: Few-shot prompt–response pairs for estimating the probability that a factor supports one outcome over another.

1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781

Few-Shot Examples for Latent Variable Identification with CoT

System
Please perform a brief chain-of-thought (step-by-step reasoning) before outputting the final JSON.
You are an AI assistant tasked with identifying latent variables and assigning each latent only factors drawn from the provided list. Do NOT output any edges. Return a JSON object with a single field:
 latents: an array of objects, each with:
 - name: string
 - factors: array of strings (each chosen from the provided Factors list)
Ensure the JSON parses correctly and strictly follows this schema.

User
Please identify latent variables and assign each factor to a latent. Then return JSON with fields:
 latents: [{"name": string, "factors": [...]}, ...]
Factors: ["Nutrition", "Vitamins", "Taste", "Convenience"]

Assistant
Thought: Nutrition and Vitamins both relate to health aspects of food, while Taste and Convenience relate to user enjoyment and practicality.
Final answer:
{
 "latents": [
 {"name": "HealthLat", "factors": ["Nutrition","Vitamins"]},
 {"name": "EnjoyLat", "factors": ["Taste","Convenience"]}
]
}

User
Please identify latent variables and assign each factor to a latent. Then return JSON with fields:
 latents: [{"name": string, "factors": [...]}, ...]
Factors: ["Usability", "Security", "Maintainability", "Portability", "Reliability"]

Assistant
Thought: Usability and Portability focus on user experience and access, Reliability and Maintainability focus on software quality over time, and Security is a distinct concern.
Final answer:
{
 "latents": [
 {"name": "UXLat", "factors": ["Usability","Portability"]},
 {"name": "QualityLat", "factors": ["Reliability","Maintainability"]},
 {"name": "SecurityLat", "factors": ["Security"]}
]
}

Figure 14: Few-shot prompt–response pairs for latent variable identification with chain-of-thought reasoning.

1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835

Few-Shot Examples for Latent Probability Estimation

System
Please perform a brief chain-of-thought (step-by-step) before outputting the final JSON:
You are an AI assistant. You will be given:

- A list of latents where each latent has:
 - its name
 - the list of factor descriptions it groups
- Two competing outcomes (Outcome1 vs. Outcome2)

Your task: Think through the semantic content of the factors relative to the outcomes,
and estimate for each latent a probability pair [p1, p0]:

- p1 = probability the latent supports Outcome1
- p0 = probability the latent supports Outcome2

Begin your answer with "Thought:" to show your reasoning, then output exactly a JSON object mapping each latent name to its [p1, p0] (no extra text).

User
Latents with factors:
[
 {"name": "Performance", "factors": ["Faster processing", "Efficient resource use"]},
 {"name": "Stability", "factors": ["Crash reports", "Memory leaks"]}
]
Outcome1: The system improves performance.
Outcome2: The system does not improve performance.

Assistant
Thought: The Performance latent groups factors that directly indicate faster and more efficient operation, which strongly supports Outcome1. The Stability latent lists issues that undermine reliability, which indirectly suggests performance might not improve overall. I'd estimate Performance around [0.85, 0.15] and Stability around [0.30, 0.70].
Final answer:
{
 "Performance": [0.85, 0.15],
 "Stability": [0.30, 0.70]
}

User
Latents with factors:
[
 {"name": "HealthLat", "factors": ["Nutrition benefits"]},
 {"name": "EnjoyLat", "factors": ["Taste appeal", "Fun presentation"]}
]
Outcome1: Healthy eating is fun.
Outcome2: Healthy eating is not fun.

Assistant
Thought: Nutrition benefits relate to health but don't guarantee fun, so HealthLat feels neutral to slightly positive. EnjoyLat clearly centers on taste and fun aspects, strongly supporting Outcome1. I'd give HealthLat [0.55, 0.45] and EnjoyLat [0.85, 0.15].
Final answer:
{
 "HealthLat": [0.55, 0.45],
 "EnjoyLat": [0.85, 0.15]
}

Figure 15: Few-shot prompt–response pairs for latent probability estimation.