RECCON: MITIGATING LLM HALLUCINATIONS VIA FINE-GRAINED REASONING-CHAIN CONSISTENCY

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we propose RECCON, an LLM hallucination mitigation framework that combines retrieval-augmented generation (RAG) with consistency analysis across multiple sampled answers and their reasoning chains. Existing self-consistency methods operate at a coarse-grained level, simply aggregating sampled final answers into a single prediction without pinpointing and inspecting the specific reasoning steps at which inconsistencies arise, thereby discarding valuable information for assessing the trustworthiness of each answer. Yet, performing fine-grained consistency analysis at the reasoning-step level requires reasoning over complex logical dependencies among the reasoning steps. RECCON addresses this challenge by leveraging the increasingly strong capabilities of LLMs for analyzing complex logical relationships. Specifically, it employs a multi-step, LLM-driven procedure in which inconsistent reasoning steps are first identified by an LLM, and their truthfulness is then assessed through a dual-layer verification strategy combining RAG and self-consistency. Each sampled answer is subsequently reweighted based on the estimated truthfulness of its sub-claims before the answers are aggregated into a single prediction. Across four datasets and two model settings, RECCON achieves the best AUROC on every dataset. On statements where the sampled chains disagree, it improves over the self-consistency baseline by 3.9 and 5.2 AUROC points on average.

1 INTRODUCTION

Despite the rapid progress of large language models (LLMs), hallucination remains an unresolved challenge. Hallucination refers to the behavior of an LLM generating false statements that are not grounded in the model's internal knowledge or external evidence, and can have significant consequences in applications that require precise and informed judgment. Although retrieval-augmented generation (RAG) is a common approach to mitigating hallucination, previous studies have found that LLMs can sometimes miss relevant evidence or misinterpret retrieved evidence (Wei et al., 2024; Xie et al., 2025), leading to contextual hallucination. Therefore, it is important to develop complementary approaches that work alongside RAG, particularly when RAG alone is insufficient to mitigate hallucination.

Consistency has emerged as a promising signal for detecting or mitigating hallucination. Specifically, prior studies have found that when an LLM hallucinates, it is more likely to exhibit logical inconsistencies across related statements or across multiple runs (Farquhar et al., 2024; Ghosh et al., 2025; Liu et al., 2025). For example, if an LLM hallucinates that *"Albert Einstein was born in Switzerland"*, this erroneous belief may be less consistently maintained when the same proposition is expressed differently, *e.g.*, *"Switzerland is the birth country of Albert Einstein."*

There are two main classes of hallucination mitigation methods that leverage consistency. The first class is based on *self-consistency* (Wang et al., 2023; Taubenfeld et al., 2025; Wan et al., 2025), *i.e.*, sampling multiple answers to the same question and aggregating their predictions. However, for complex questions involving multiple reasoning steps, such coarse-grained aggregation misses the opportunity to pinpoint and inspect the specific steps at which inconsistencies arise, thereby discarding valuable information for assessing the trustworthiness of each answer and adjusting its weight accordingly.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The second class of methods is based on *belief trees* (Kassner et al., 2023; Hou et al., 2025; Marinescu et al., 2025), which decompose target answers into atomic statements, recursively expand them into tree or graph structures of logically related statements, and measure the LLM's beliefs in these statements. By structurally comparing these beliefs, such methods can identify inconsistencies at the fine-grained atomic-statement level and thereby produce more reliable aggregated predictions. However, these methods model only relatively simple logical relationships among statements, such as entailment and contradiction, limiting their ability to capture the complex logical dependencies among reasoning steps in long chains of thought.

The complementary limitations of these two classes of methods reveal a fundamental tension in leveraging consistency to mitigate hallucinations in complex questions: achieving fine-grained consistency analysis requires reasoning over the very complex logical dependencies that existing fine-grained methods struggle to capture. Fortunately, the rapid advances in LLMs offer a potential opportunity to resolve this tension: Can we leverage the increasingly strong capabilities of LLMs for understanding complex logical relationships to analyze fine-grained inconsistencies across multiple complex reasoning chains, and thereby effectively mitigate hallucinations?

Motivated by this, we propose RECCON (**Re**asoning-**C**hain **Con**sistency), a fine-grained reasoning-chain self-consistency framework for LLM hallucination mitigation. As shown in Figure 1, RECCON consists of four steps: **sample**, **contrast**, **verify**, and **reweight**. First, given a statement, we sample multiple answers together with their chains of thought from the target LLM, analogous to self-consistency methods. Second, an LLM-based inspection module contrasts these reasoning chains to identify inconsistent sub-claims. Third, a claim verification module verifies the identified inconsistent sub-claims, with the help of external evidence and internal consistency, and assigns each a truthfulness score. Finally, a weighting module estimates the credibility of each answer based on the truthfulness scores of its sub-claims. The answers are then aggregated into a single prediction, weighted by their credibility scores. In this way, RECCON combines the strengths of RAG and consistency-based methods, while addressing the complementary limitations of the two classes of consistency methods: it performs fine-grained inconsistency analysis at the sub-claim level, while accommodating the complex logical dependencies among reasoning steps.

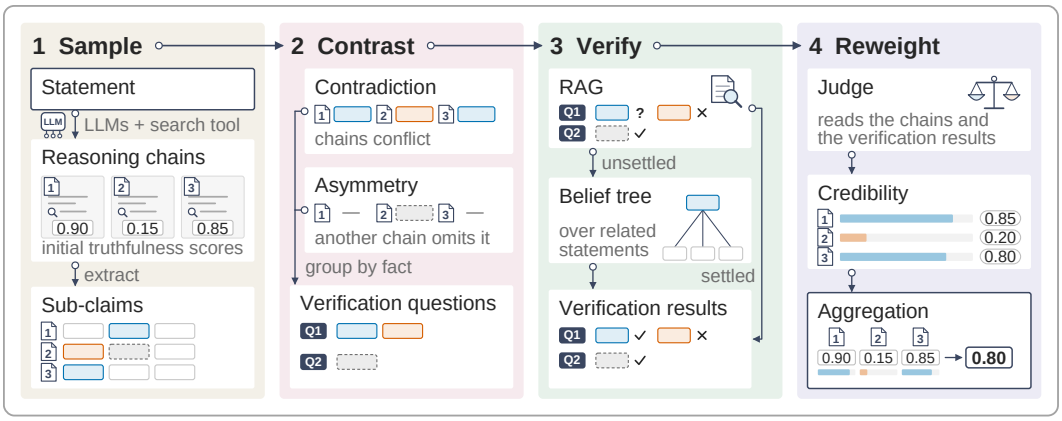


Figure 1: Overview of RECCON. Sub-claims on which the sampled reasoning chains disagree become verification questions, resolved by RAG and, if unsettled, a belief tree. A judge rates each chain's credibility from the results, and the prediction is the credibility-weighted mean of the initial truthfulness scores. Page icons number the chains. Blue and orange mark opposing sub-claims, and grey dashed ones are not mentioned by at least one other chain.

We evaluate RECCON on four datasets covering fact checking, counterfactual reasoning, and future-event prediction, in two model settings. RECCON achieves the best AUROC on every dataset, above both consistency-based and RAG-based baselines. Its gains are largest on the statements where the sampled chains disagree, on which averaging them no longer helps reliably. There, RECCON improves over RAG + self-consistency, which averages the same chains, by 3.9 and 5.2 AUROC points in the two settings.

2 Related Work

Claim and reasoning verification. FActScore and SAFE check factual claims against external sources (Min et al., 2023; Wei et al., 2024). FIRE interleaves retrieval and verification (Xie et al., 2025), and Chain-of-Verification plans questions that check a draft answer (Dhuliawala et al., 2024). Sadeddine & Suchanek (2025) check deductive steps and trace their premises to the supplied context or earlier conclusions. Our focus is selecting verification targets from differences between reasoning chains. Each finding informs the assessment of every chain that relies on the disputed information, even when their final answers agree.

Structured beliefs. BTProp represents a statement and its logically related claims as a hidden Markov tree (Hou et al., 2025), treating claim truth as latent and model confidence as a noisy observation. Our method instead uses the tree to check multiple reasoning chains against external evidence. When retrieval cannot resolve a claim, we draw on BTProp's idea of combining model beliefs about related claims to obtain an auxiliary estimate.

Reasoning-aware aggregation. Self-consistency aggregates sampled answers by their frequency (Wang et al., 2023). Confidence-Informed Self-Consistency weights votes by the model's confidence (Taubenfeld et al., 2025), while Reasoning-Aware Self-Consistency scores paths using reasoning features and answer agreement (Wan et al., 2025). Universal Self-Consistency asks an LLM to select the most consistent response (Chen et al., 2023), and multi-agent debate lets models revise their answers after reading each other's (Du et al., 2024). These methods aggregate the final answers without inspecting the reasoning steps at which the samples disagree. We score the chains' claims using shared verification findings and auxiliary model beliefs. A judge then weights each chain by whether the claims its conclusion rests on survived these findings, without rating the quality of its reasoning.

Sampling-based hallucination detection. SelfCheckGPT compares a response with other samples sentence by sentence (Manakul et al., 2023), Mündler et al. (2024) detect self-contradictions between sampled sentences, and Atomic Self-Consistency keeps the atomic facts that recur across samples (Thirukovalluru et al., 2024). These methods check whether the stated facts agree across samples, whereas we trace a disagreement back to the reasoning steps behind it and verify those steps.

3 Method

In this section, we introduce the methodology behind ReCCon. Section 3.1 formulates our research problem and provides an overview of our method, and the subsequent sections discuss each component in detail.

3.1 Problem Formulation and Algorithm Overview

Although ReCCon can be applied to a wide range of LLM answers, in this paper, we focus on the statement-level truthfulness estimation problem. Given a self-contained declarative statement s, the task is to estimate the probability of s being true, $p_{\text{true}} \in [0, 1]$. The performance of the system is evaluated by prediction accuracy against the truthfulness label and the calibration of p_{true}.

ReCCon combines the strengths of RAG and consistency-based methods, and is designed to identify fine-grained inconsistencies in reasoning steps across multiple reasoning chains. Specifically, given s, ReCCon estimates its truthfulness via four steps.

- **Sample.** Given the statement s, we sample multiple answers together with their chains of thought from the target LLM, analogous to self-consistency methods.
- **Contrast.** An LLM-based inspection module contrasts these reasoning chains to identify inconsistent sub-claims.
- **Verify.** A claim verification module verifies the identified inconsistent sub-claims, with the help of external evidence and internal consistency, and assigns each a truthfulness score.

- **Reweight.** A weighting module estimates the credibility of each answer based on the truthfulness scores of its sub-claims. The resulting credibility scores of the answers serve as the weights with which the answers are aggregated into a single prediction.

Figure 1 provides an overview of these four steps. The subsequent subsections discuss each step in detail, using a running example from IfQA (Yu et al., 2023), the statement *"If the Citi Bike system had been operational as initially planned, it would have been available for public use in fall 2011"*, which is labeled true. Figure 2 follows this statement from the extracted sub-claims to the verification questions. Algorithm 1 in Appendix A summarizes the complete procedure, and Appendix B gives its models, decoding settings, and budgets.

3.2 Sampling Reasoning Chains

Given a statement s, we prompt LLMs to generate k answers on the truthfulness of s. Each answer consists of a reasoning chain (denoted as $\mathcal{R}_i$ for the i-th answer), a short rationale summarizing its reasoning, and an initial truthfulness score estimating the probability that s is true (denoted as $\hat{p}_{\text{true},i}$). $k = 3$ in most settings in this paper. These answers can be generated from the same or different LLMs. To ensure reliable reasoning chains, we equip the LLMs with a search engine API, which they can call with their own crafted queries to gather external evidence throughout the reasoning process.

To make the reasoning chain better formulated for the subsequent steps, we introduce an extractor LLM, which reads each reasoning chain $\mathcal{R}_i$ and its initial truthfulness score $\hat{p}_{\text{true},i}$, and is prompted to extract a set of atomic sub-claims, denoted as $\mathcal{C}_i = \{\boldsymbol{h}_{ij}\}_j$ with $\boldsymbol{h}_{ij}$ the j-th sub-claim of chain i, that collectively support its judgment, including factual premises and checkable intermediate conclusions. An example of a reasoning chain and its extracted sub-claims is shown in Figure 2. The extracted sub-claims of each reasoning chain and its initial truthfulness score, *i.e.*, $\{(\mathcal{C}_i, \hat{p}_{\text{true},i})\}$, are fed to the subsequent steps.

3.3 Contrasting Reasoning Chains

In this step, an inspector LLM inspects the sub-claims of each chain, $\{(\mathcal{C}_i, \hat{p}_{\text{true},i})\}$, and flags inconsistencies for subsequent verification. The inspector checks for inconsistencies only when the chains' initial truthfulness scores differ substantially: $d_s = \max_i \hat{p}_{\text{true},i} - \min_i \hat{p}_{\text{true},i} > 0.25$. When the scores are close, disagreements usually concern minor details, and reweighting has little effect on the final prediction, so we skip the remaining steps and return the mean of the initial truthfulness scores. Otherwise, the process involves two steps, as described below.

Step 1: Flagging inconsistent sub-claims. The inspector LLM is prompted to flag sub-claims across different reasoning chains that are inconsistent with others. The following two types of inconsistencies are considered.

- ***Contradiction.*** A sub-claim that is contradictory to at least one sub-claim in another reasoning chain is flagged as a contradiction. For example, in Figure 2a, chains $\mathcal{R}_1$ and $\mathcal{R}_3$ state that Citi Bike was first supposed to start in fall 2011, whereas chain $\mathcal{R}_2$ considers two alternatives, stating in one sub-claim that the initial plan targeted 2012 and in another that it targeted spring 2013. All four sub-claims are flagged as contradictions.
- ***Asymmetry.*** A sub-claim making a statement that is not mentioned in at least one of the other reasoning chains is flagged as an asymmetry. In the same Citi Bike example, Figure 2b shows that chain $\mathcal{R}_2$ states that Citi Bike actually launched in May 2013. Neither $\mathcal{R}_1$ nor $\mathcal{R}_3$ includes this fact among its extracted sub-claims, so it is flagged as an asymmetry.

Step 2: Formulating verification questions. For the subsequent modules to better resolve the identified inconsistencies, the inspector LLM is further prompted to group the flagged sub-claims that concern the same fact into inconsistency groups $\{\mathcal{G}_\ell\}$. For each inconsistency group $\mathcal{G}_\ell$, the inspector constructs a verification question $\boldsymbol{q}_\ell$ and its candidate-answer set $\mathcal{A}_\ell$. The following two types of verification questions are generated.

- ***Group-specific questions.*** Since each inconsistency group $\mathcal{G}_\ell$ concerns a single fact, it is straightforward to turn that fact into a question. Therefore, for each inconsistency group, the inspector

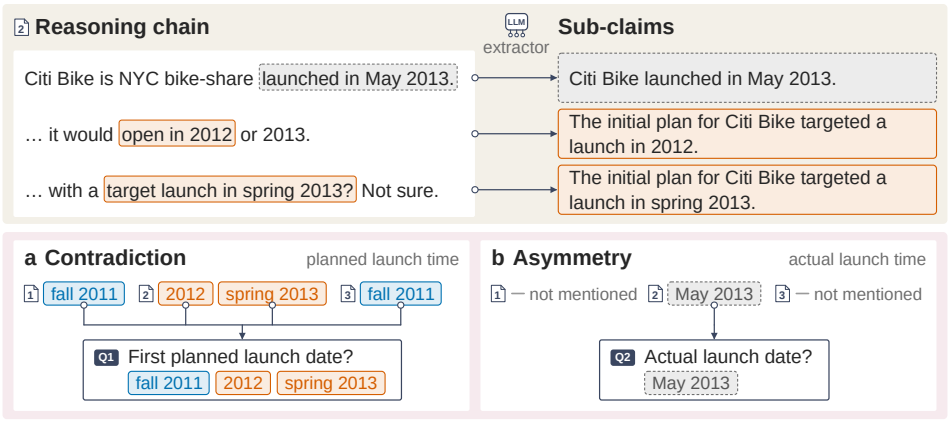


Figure 2: The running Citi Bike example. Top: sub-claims extracted from a verbatim excerpt of chain 2, three of five shown. **a** Chains 1 and 3 say the system was first planned for fall 2011, and chain 2 says 2012 or spring 2013. **b** Only chain 2 states the actual launch in May 2013. Each group becomes a verification question with its candidate answers.

LLM is prompted to rewrite the disputed fact into a question, and summarize the candidate answers based on the sub-claims in that group. For example, the contradiction group in Figure 2a becomes a question about the initial launch season or date announced for Citi Bike before any delays, with candidates *"fall 2011"*, *"2012"*, and *"spring 2013"*.

- ***Crux questions.*** In rare cases, the reasoning chains may all fail to discuss some key premises needed to estimate the truthfulness of the statement. To address this failure mode, the inspector LLM is prompted to generate crux questions that need to be answered before one can estimate the truthfulness of the statement, *regardless of* whether they have been discussed in the sub-claims or whether there are inconsistencies on these topics. The candidate answers to a crux question are extracted from all the reasoning chains if they discuss it, and otherwise the inspector proposes the most plausible answer.

By construction, the crux questions can overlap with the group-specific questions. Therefore, all verification questions with their candidate-answer sets $\{(\boldsymbol{q}_\ell, \mathcal{A}_\ell)\}$ are de-duplicated by an LLM before being sent to the verification step. The question about the originally planned launch in Figure 2 is generated through both routes, and only one copy is retained. This question is decisive for the statement, whereas the actual launch date concerns a different fact.

3.4 Verifying Sub-Claims

The previous step turns the inconsistencies among the reasoning chains into a short list of questions with their candidate-answer sets $(\boldsymbol{q}_\ell, \mathcal{A}_\ell)$. We verify the candidate answers in two cascaded layers. The first layer uses RAG, and the second layer applies self-consistency over paraphrases with a belief-tree method (Hou et al., 2025) to the questions that the first layer cannot resolve. Based on the verification results, each flagged sub-claim $\boldsymbol{h}_{ij}$ receives a truthfulness score, denoted as $b(\boldsymbol{h}_{ij}) \in [0, 1]$, which estimates the probability that $\boldsymbol{h}_{ij}$ is true.

3.4.1 RAG-Based Verification

In the first layer, for each question and its set of candidate answers $(\boldsymbol{q}_\ell, \mathcal{A}_\ell)$, a verifier searches for evidence supporting or refuting each answer. Specifically, for each question, it returns ❶ a *verdict* in $\{\text{SUPPORTED}, \text{REFUTED}, \text{UNSETTLED}\}$ for each of its candidate answers, ❷ a *truthfulness score* $b(\boldsymbol{a})$ estimating the probability that $\boldsymbol{a}$ is correct for each candidate answer $\boldsymbol{a}$, and ❸ a short *finding* that cites the retrieved evidence behind it.

For the verdict, the verifier LLM is prompted to assign SUPPORTED or REFUTED if it determines the retrieved evidence is sufficient to support or contradict $\boldsymbol{a}$, and UNSETTLED if the evidence is inconclusive. Regardless of the verdict, it is also prompted to estimate the truthfulness score $b(\boldsymbol{a})$.

For unsettled candidates, the verifier may return uncertain or unreliable truthfulness scores $b(\boldsymbol{a})$. Therefore, we pass these candidate answers, together with the corresponding verification questions, to the second layer of verification to refine the $b(\boldsymbol{a})$ estimates, as detailed in the next subsection.

3.4.2 BELIEF-TREE-BASED VERIFICATION

For each unsettled candidate answer $\boldsymbol{a}$, we adopt the BTProp algorithm (Hou et al., 2025) to produce an improved $b(\boldsymbol{a})$ estimate. Here we provide a brief introduction to BTProp. Readers can refer to the original paper for more information.

We first turn the candidate answer $\boldsymbol{a}$ and its corresponding question $\boldsymbol{q}$ into a declarative statement, denoted as $\boldsymbol{H}_0$, and regard $b(\boldsymbol{H}_0) = b(\boldsymbol{a})$ as the prior probability of $\boldsymbol{H}_0$ being true. BTProp involves four steps. *First*, a generator LLM generates $n \leq 3$ statements, denoted as $\boldsymbol{H}_{1:n}$, that are paraphrases of $\boldsymbol{H}_0$. *Second*, a relation-labeler LLM is prompted to determine the logical relationship between $\boldsymbol{H}_0$ and each generated $\boldsymbol{H}_i$ ($\forall i \in \{1, \ldots, n\}$). The logical relationship can be SYNONYM, SUPPORT, CONTRADICT, or UNRELATED. *Third*, the verifier LLM is asked to also estimate the truthfulness of the generated statements, *i.e.*, $b(\boldsymbol{H}_i)$. Finally, we use Bayes' rule to compute the posterior probability of $\boldsymbol{H}_0$ being true given the verifier LLM's belief in the generated statements,

$$p(\boldsymbol{H}_0 \text{ is true} | \{b(\boldsymbol{H}_i)\}), \tag{1}$$

based on a hidden Markov tree model. This process can be understood as utilizing the consistencies among the truthfulness estimates of $\boldsymbol{H}_{1:n}$ to refine that of $\boldsymbol{H}_0$. Appendix C details how to compute Eq. (1), which becomes the updated truthfulness score $b(\boldsymbol{a})$.

The generated statements $\boldsymbol{H}_{1:n}$ play a crucial role in this process. Good generation can effectively expose the inconsistencies among the verifier LLM's beliefs, and thereby better correct any wrong estimates. Empirically, we find two generation strategies effective: ❶ prompting the generator LLM to always generate synonyms, and ❷ fine-tuning the generator LLM such that the generated statements maximize the agreement between Eq. (1) and the ground-truth truthfulness on a labeled training set. Appendix D describes the reinforcement-learning procedure for the generator LLM.

3.5 REWEIGHTING REASONING CHAINS

After the above verification step, for each candidate answer $\boldsymbol{a}$ to a verification question, we already have a truthfulness score, $b(\boldsymbol{a})$. We then map $b(\boldsymbol{a})$ back to the sub-claim from which the candidate answer was derived, resulting in $b(\boldsymbol{h}_{ij})$, which is used to compute the credibility of each chain.

Specifically, for reasoning chain $\mathcal{R}_i$, we prompt a judge LLM to assess its credibility. The input to the judge LLM consists of the initial truthfulness score of the reasoning chain $\hat{p}_{\text{true},i}$, a summary of the reasoning chain, the truthfulness scores of its flagged sub-claims, and the findings in verifying the questions. The output of the judge LLM is a credibility score, denoted as w_i, estimating the probability that this reasoning chain holds. Appendix I.6 provides the judge's instructions.

The resulting $\{w_i\}$ serve as the weights for aggregating the truthfulness estimates of all the reasoning chains. The final truthfulness estimate, $\hat{p}_{\text{true}}$, is computed by

$$\hat{p}_{\text{true}} = \frac{\sum_{i=1}^{k} w_i \hat{p}_{\text{true},i}}{\sum_{i=1}^{k} w_i}. \tag{2}$$

4 EXPERIMENTS

We test whether examining why reasoning chains disagree helps decide which answers to trust. Across four datasets and two model settings, RECCON achieves the best AUROC on every dataset (Section 4.2), and its gains grow on the harder statements where the chains disagree (Section 4.3). Section 4.4 then examines how belief-tree verification and chain reweighting contribute.

4.1 EXPERIMENTAL SETUP

Datasets. We use four datasets, from fact checking and counterfactual reasoning to future prediction, and each input is a self-contained statement with a binary label. FactCHD (Chen et al., 2024) supplies factual and non-factual responses, which we combine with their questions to form statements.

HoVer (Jiang et al., 2020) requires combining evidence from multiple Wikipedia articles to determine whether a claim is supported. We use the original claims and their supported/not-supported labels. FutureX (Zeng et al., 2025) supplies resolved events for a retrospective prediction evaluation, with statements asserting candidate outcomes. IfQA (Yu et al., 2023) poses open-domain questions under counterfactual premises. We combine each premise with its gold answer and a plausible wrong answer to form true and false statements. We refer to FactCHD, HoVer, and IfQA statements as fact-checking statements and to FutureX statements as future-event statements. Appendix F gives the evaluation counts and describes statement conversion.

Models and chain generation. The small model setting uses backbones with roughly 30B parameters, and the large model setting uses backbones with roughly 120B parameters. Both generate three chains: the Qwen backbone produces $\mathcal{R}_1$ and $\mathcal{R}_3$, and the gpt-oss backbone produces $\mathcal{R}_2$. Qwen also serves as the Direct ask baseline and as the extractor, inspector, verifier, relation labeler, and judge. Both model settings use Qwen3-8B as the generator LLM of the belief tree, which we train with reinforcement learning. For $\mathcal{R}_1$ and $\mathcal{R}_3$, we use different random seeds to generate different chains. Appendix B lists the models for each role, the sampling parameters, and the agent budgets.

Retrieval system. Our retrieval system uses English Wikipedia as its data source and implements BM25 retrieval with Pyserini (Lin et al., 2021). The corpus is the structured Wikipedia snapshot of 13 May 2026, which contains 7.1 million articles. Every evaluated FutureX statement resolves on or after 14 May 2026, so the snapshot predates all the outcomes that the prediction task asks about, and retrieval cannot reveal them. Implementation details are provided in Appendix E.

Baseline methods. Without retrieval, Direct ask scores a statement with one chain, Self-consistency averages $k = 3$ such chains, and *BTProp* (Hou et al., 2025) infers its truthfulness from a belief tree of related statements. With retrieval, RAG scores it with the chain $\mathcal{R}_1$, and RAG + self-consistency averages the same three chains as RECCON. *FactReasoner-FR1* (Marinescu et al., 2025) and *VeriScore* (Song et al., 2024) verify atomic claims against evidence from the same Wikipedia snapshot as our method, and *FAVA* (Mishra et al., 2024) is a fine-grained hallucination detector that we run from the authors' 7B checkpoint. All baselines except FAVA and RAG + self-consistency use the Qwen model of each model setting. Appendix G.1 gives further details.

Evaluation metrics. AUROC, our main metric, measures how well the predicted truthfulness scores rank true statements above false ones, independent of any threshold.

4.2 MAIN RESULTS

Table 1 compares RECCON with its baselines on the full evaluation sets, reporting AUROC for each dataset and the macro average. Appendix G.3 reports PR-AUC, Acc@0.5, and Brier.

RECCON outperforms Self-consistency and BTProp, which do not use retrieval, and improves over RAG and RAG + self-consistency, across all four datasets in both model settings. The comparison with RAG + self-consistency is particularly informative because it averages the same initial truthfulness scores. Against this baseline, the macro average of AUROC gains 0.6 points in the small model setting and 0.5 points in the large model setting, both significant under a paired bootstrap ($p \leq 0.002$, Appendix G.2). These gains show that examining the sub-claims behind each answer remains useful when both retrieval and averaging are already in place. RECCON also surpasses FactReasoner-FR1 and VeriScore on all datasets.

4.3 DIVERGENCE SET

Statements on which the chains disagree are harder to judge, and averaging the chains helps less on them. We therefore also evaluate on the divergence set, the statements whose initial truthfulness scores differ by more than 0.25. Each model setting defines its own divergence set, which covers 28% of the statements in the small model setting and 24% in the large model setting.

Table 2 shows that these statements are harder. The strongest baseline drops from 79.0 to 64.1 macro-averaged AUROC in the small model setting and from 79.5 (RAG + self-consistency) to 61.8 (RAG) in the large model setting. Averaging the chains also stops helping reliably. In the large model setting, RAG + self-consistency falls below RAG on three of the four datasets and on the macro average (61.0 vs. 61.8), and on FactCHD in the small model setting, Direct ask outperforms

Table 1: AUROC on four datasets and their macro average. *FAVA uses the authors' fine-tuned 7B checkpoint.

Method	FactCHD	HoVer	FutureX	IfQA	Avg
Small model setting: Qwen3.6-35B-A3B / gpt-oss-20b					
Direct ask	78.8	62.8	61.1	78.9	70.4
Self-consistency	81.0	65.9	62.3	83.0	73.0
RAG	79.8	72.8	64.0	90.3	76.7
RAG + self-consistency	82.0	75.1	65.8	93.2	79.0
BTProp (Hou et al., 2025)	73.7	64.6	54.4	74.2	66.7
FactReasoner-FR1 (Marinescu et al., 2025)	67.5	64.6	50.9	65.1	62.1
VeriScore (Song et al., 2024)	62.7	57.9	48.8	52.4	55.4
FAVA* (Mishra et al., 2024)	58.6	56.6	50.5	60.1	56.5
RECCON (ours)	**82.1**	**75.9**	**66.9**	**93.5**	**79.6**
Large model setting: Qwen3.5-122B-A10B / gpt-oss-120b					
Direct ask	80.7	69.2	60.3	85.6	74.0
Self-consistency	81.9	71.3	59.8	87.5	75.1
RAG	82.7	76.6	63.2	92.1	78.6
RAG + self-consistency	83.4	77.7	63.0	94.0	79.5
BTProp (Hou et al., 2025)	75.9	67.3	46.4	73.0	65.7
FactReasoner-FR1 (Marinescu et al., 2025)	75.7	67.8	50.7	75.7	67.5
VeriScore (Song et al., 2024)	68.6	62.0	50.0	57.7	59.6
FAVA* (Mishra et al., 2024)	58.6	56.6	50.5	60.1	56.5
RECCON (ours)	**83.7**	**78.0**	**64.2**	**94.2**	**80.0**

Table 2: AUROC on the divergence set. n is its size out of all statements.

Method	FactCHD	HoVer	FutureX	IfQA	Avg
Small model setting: Qwen3.6-35B-A3B / gpt-oss-20b					
n	234/971	324/1000	128/421	352/1273	–
Direct ask	61.8	53.1	54.7	61.9	57.9
Self-consistency	61.1	53.7	54.6	64.1	58.4
RAG	60.0	62.1	54.8	70.3	61.8
RAG + self-consistency	61.2	66.0	55.8	73.4	64.1
RECCON	**63.3**	**70.8**	**61.6**	**76.4**	**68.0**
Large model setting: Qwen3.5-122B-A10B / gpt-oss-120b					
n	185/971	278/1000	104/421	310/1273	–
Direct ask	60.9	53.5	58.6	68.1	60.3
Self-consistency	**64.4**	54.1	56.6	70.0	61.3
RAG	61.9	61.8	57.7	65.8	61.8
RAG + self-consistency	58.1	56.5	53.5	75.8	61.0
RECCON	62.5	**63.5**	**61.4**	**77.4**	**66.2**

both Self-consistency and RAG + self-consistency. On this harder set, RECCON improves the macro average of AUROC over RAG + self-consistency by 3.9 points in the small model setting (68.0 vs. 64.1) and by 5.2 points in the large model setting (66.2 vs. 61.0), both with $p \leq 0.001$, and it improves on all four datasets in both settings. When the chains conflict, averaging them cannot tell which one to trust, whereas verifying the sub-claims behind each answer can. Appendix H examines one success and one failure in detail.

4.4 COMPONENT ANALYSIS

Belief tree. Belief-tree verification provides additional signals for assessing candidate answers that RAG leaves unsettled. On the divergence set, Table 3 compares verification without the belief tree with belief-tree verification using the base or RL-trained Qwen3-8B generator.

Table 3: AUROC on the divergence set ($d_s > 0.25$) without belief-tree verification and with a base or RL-trained generator, other components fixed.

Configuration	FactCHD	HoVer	FutureX	IfQA	Avg
Small model setting					
w/o belief tree	**63.6**	69.4	60.5	75.8	67.3
w/ base generator	63.2	70.1	**61.7**	**76.4**	67.8
w/ RL-trained generator	63.3	**70.8**	61.6	**76.4**	**68.0**
Large model setting					
w/o belief tree	61.1	**63.6**	59.6	**78.9**	65.8
w/ base generator	**62.5**	63.2	61.0	77.5	66.1
w/ RL-trained generator	**62.5**	63.5	**61.4**	77.4	**66.2**

Both generators raise the macro average of AUROC over verification without the belief tree in both model settings, and the RL-trained generator gives the best average in each (68.0 vs. 67.3 and 66.2 vs. 65.8). On FutureX, both generators improve in both settings, by 1.1 to 1.8 points. On the fact-checking datasets, the belief tree acts as a refinement whose effect varies between the settings.

Reweighting. A simpler way to use the verification results is to discard the chains they refute. The drop rule in Table 4 removes every chain containing a flagged sub-claim with $b(\boldsymbol{h}_{ij}) < 0.5$ and averages the initial truthfulness scores of the remaining chains, or of all chains if none remain.

Table 4: AUROC on the divergence set ($d_s > 0.25$) of three ways to aggregate the same chains and verification findings.

Method	FactCHD	HoVer	FutureX	IfQA	Avg
Small model setting					
RAG + self-consistency	61.2	66.0	55.8	73.4	64.1
Drop rule	60.7	63.7	58.1	70.1	63.1
RECCON	**63.3**	**70.8**	**61.6**	**76.4**	**68.0**
Large model setting					
RAG + self-consistency	58.1	56.5	53.5	75.8	61.0
Drop rule	53.8	61.0	54.6	74.9	61.1
RECCON	**62.5**	**63.5**	**61.4**	**77.4**	**66.2**

Discarding chains gains little over averaging them. The drop rule falls below RAG + self-consistency in the small model setting (63.1 vs. 64.1) and matches it in the large model setting (61.1 vs. 61.0), whereas RECCON exceeds the drop rule by 4.9 and 5.1 points. A sub-claim with a low truthfulness score need not invalidate the entire answer, so estimating the credibility of each answer from the sub-claims on which it depends makes better use of the verification results than discarding the answer.

5 CONCLUSION

We presented RECCON, which estimates the truthfulness of a statement by examining why the sampled reasoning chains disagree. It verifies their contradictory and asymmetric sub-claims with RAG and a belief tree, and weights each chain by the credibility that a judge assigns from the results. RECCON achieves the best AUROC on all four datasets in both model settings. Where the chains disagree and averaging them no longer helps reliably, it improves over RAG + self-consistency on every dataset. RECCON also returns the verification questions and findings behind each prediction.

AI USE STATEMENT

Generative AI models were used as components of the experimental pipeline to generate reasoning chains, serve as the extractor, inspector, verifier, and judge, and convert dataset items into statements, as described in Section 3 and Appendices F and I. We also used generative AI tools to assist with writing and debugging the experiment code, analyzing results and case studies, preparing figures, and drafting and revising the manuscript. The authors reviewed all AI-assisted outputs incorporated into this work and take full responsibility for its content, including all claims and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work uses public datasets and a public Wikipedia snapshot. It involves no human participants and collects no personal data, and we are not aware of any ethical concerns it raises.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 describe the full procedure. The appendices give the models, decoding settings, and budgets of every agent (Appendix B), the agent instructions (Appendix I), the generator training (Appendix D), the retrieval corpus (Appendix E), and the dataset conversion (Appendix F). Upon acceptance, we will release the code, the configurations, and the data-processing scripts.

REFERENCES

Xiang Chen, Duanzheng Song, Honghao Gui, Chenxi Wang, Ningyu Zhang, Yong Jiang, Fei Huang, Chengfei Lyu, Dan Zhang, and Huajun Chen. FactCHD: Benchmarking fact-conflicting hallucination detection. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024. doi: 10.24963/ijcai.2024/687.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation, 2023. URL https://arxiv.org/abs/2311.17311.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3563–3578, 2024. URL https://aclanthology.org/2024.findings-acl.212/.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024. doi: 10.1038/s41586-024-07421-0.

Bishwamittra Ghosh, Sarah Hasan, Naheed Anjum Arafat, and Arijit Khan. Logical consistency of large language models in fact-checking. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=SimlDuN0YT.

Bairu Hou, Yang Zhang, Jacob Andreas, and Shiyu Chang. A probabilistic framework for LLM hallucination detection via belief tree propagation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3076–3099, 2025. doi: 10.18653/v1/2025.naacl-long.158.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3441–3460, 2020. doi: 10.18653/v1/2020.findings-emnlp.309.

Nora Kassner, Oyvind Tafjord, Ashish Sabharwal, Kyle Richardson, Hinrich Schuetze, and Peter Clark. Language models with rationality. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14190–14201, 2023. doi: 10.18653/v1/2023.emnlp-main.877.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2356–2362, 2021. doi: 10.1145/3404835.3463238.

Qiang Liu, Xinlong Chen, Yue Ding, Bowen Song, Weiqiang Wang, Shu Wu, and Liang Wang. Attention-guided self-reflection for zero-shot hallucination detection in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21005–21021, 2025. doi: 10.18653/v1/2025.emnlp-main.1063.

Huanhuan Ma, Weizhi Xu, Yifan Wei, Liuji Chen, Liang Wang, Qiang Liu, Shu Wu, and Liang Wang. EX-FEVER: A dataset for multi-hop explainable fact verification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9340–9353, 2024. doi: 10.18653/v1/2024.findings-acl.556.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017, 2023. doi: 10.18653/v1/2023.emnlp-main.557.

Radu Marinescu, Debarun Bhattacharjya, Junkyu Lee, Tigran T. Tchrakian, Javier Carnerero-Cano, Yufang Hou, Elizabeth M. Daly, and Alessandra Pascale. FactReasoner: A probabilistic approach to long-form factuality assessment for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 14547–14577, 2025. doi: 10.18653/v1/2025.findings-emnlp.785.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12076–12100, 2023. doi: 10.18653/v1/2023.emnlp-main.741.

Abhika Mishra, Akari Asai, Vidhisha Balachandran, Yizhong Wang, Graham Neubig, Yulia Tsvetkov, and Hannaneh Hajishirzi. Fine-grained hallucination detection and editing for language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=dJMTn3QOWO.

Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. In *The Twelfth International Conference on Learning Representations*, 2024.

Zacchary Sadeddine and Fabian M. Suchanek. Verifying the steps of deductive reasoning chains. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 456–475, 2025. doi: 10.18653/v1/2025.findings-acl.25.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Yixiao Song, Yekyung Kim, and Mohit Iyyer. VeriScore: Evaluating the factuality of verifiable claims in long-form text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9447–9474, 2024. doi: 10.18653/v1/2024.findings-emnlp.552.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. Confidence improves self-consistency in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20090–20111, 2025. doi: 10.18653/v1/2025.findings-acl.1030.

Raghuveer Thirukovalluru, Yukun Huang, and Bhuwan Dhingra. Atomic self-consistency for better long form generations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 12681–12694, 2024.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7534–7550, 2020. doi: 10.18653/v1/2020.emnlp-main.609.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3613–3635, 2025. doi: 10.18653/v1/2025.naacl-long.184.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. Long-form factuality in large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2024.

Zhuohan Xie, Rui Xing, Yuxia Wang, Jiahui Geng, Hasan Iqbal, Dhruv Sahnan, Iryna Gurevych, and Preslav Nakov. FIRE: Fact-checking with iterative retrieval and verification. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2901–2914, 2025. URL https://aclanthology.org/2025.findings-naacl.158/.

Wenhao Yu, Meng Jiang, Peter Clark, and Ashish Sabharwal. IfQA: A dataset for open-domain question answering under counterfactual presuppositions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8276–8288, 2023. doi: 10.18653/v1/2023.emnlp-main.515.

Zhiyuan Zeng, Jiashuo Liu, Siyuan Chen, Tianci He, Yali Liao, Yixiao Tian, Jinpeng Wang, Zaiyuan Wang, Yang Yang, Lingyue Yin, Mingren Yin, Zhenwei Zhu, Tianle Cai, Zehui Chen, Jiecao Chen, Yantao Du, Xiang Gao, Jiacheng Guo, Liang Hu, Jianpeng Jiao, Xiangsheng Li, Jingkai Liu, Shuang Ni, Zhoufutu Wen, Ge Zhang, Kaiyuan Zhang, Xin Zhou, Jose Blanchet, Xipeng Qiu, Mengdi Wang, and Wenhao Huang. FutureX: An advanced live benchmark for LLM agents in future prediction, 2025. URL https://arxiv.org/abs/2508.11987.

APPENDIX CONTENTS

A Inference procedure **15**

B Implementation details **15**

C Belief-tree propagation for unresolved claims **16**

D Generator training **17**

E Retrieval corpus **18**

F Datasets and statement conversion **18**

G Baselines and additional results **20**

G.1 Baselines 21

G.2 Significance 21

G.3 All metrics 22

H Case study **22**

I Prompt templates **23**

I.1 Chain generation 23

I.2 Claim extraction 24

I.3 Contrasting reasoning chains 24

I.4 Verification 26

I.5 Belief-tree verification 26

I.6 Chain reweighting 28

I.7 Direct ask 29

I.8 Future-event prompts 29

A INFERENCE PROCEDURE

Algorithm 1 summarizes the inference procedure of Section 3.

Algorithm 1 RECCON inference

Require: statement s, k chain generators, extractor, inspector, verifier, generator LLM π_θ, relation labeler, judge
Ensure: truthfulness score $\hat{p}_{\text{true}}$

Sample reasoning chains
for $i = 1, \dots, k$ **do**
  generate chain $\mathcal{R}_i$ for s with search and keep its initial truthfulness score $\hat{p}_{\text{true},i}$
  extract the sub-claims $\mathcal{C}_i = \{\boldsymbol{h}_{ij}\}_j$ of $\mathcal{R}_i$
end for
if $d_s = \max_i \hat{p}_{\text{true},i} - \min_i \hat{p}_{\text{true},i} \leq 0.25$ **then**
  return $k^{-1} \sum_i \hat{p}_{\text{true},i}$
end if
Contrast reasoning chains
flag inconsistent sub-claims, group them into $\{\mathcal{G}_\ell\}$, and write a question $\boldsymbol{q}_\ell$ with candidate answers $\mathcal{A}_\ell$ for each group
add up to three crux questions with their candidate answers
de-duplicate the questions
Verify sub-claims
for each question $\boldsymbol{q}_\ell$ **do**
  obtain a finding and, for each $\boldsymbol{a} \in \mathcal{A}_\ell$, a verdict and a truthfulness score $b(\boldsymbol{a})$
  for each candidate answer $\boldsymbol{a} \in \mathcal{A}_\ell$ that is UNSETTLED **do**
    $\boldsymbol{H}_0 \leftarrow$ declarative claim that the verifier writes from $\boldsymbol{q}_\ell$ and $\boldsymbol{a}$, with prior $b(\boldsymbol{H}_0) \leftarrow b(\boldsymbol{a})$
    $\boldsymbol{H}_{1:n} \leftarrow$ paraphrases of $\boldsymbol{H}_0$ from π_θ, without exact copies of $\boldsymbol{H}_0$
    $\mathcal{E} \leftarrow$ evidence retrieved with $\boldsymbol{H}_0$ and each $\boldsymbol{H}_i$
    $b(\boldsymbol{H}_1), \dots, b(\boldsymbol{H}_n) \leftarrow$ verifier truthfulness scores against $\mathcal{E}$
    $r_1, \dots, r_n \leftarrow$ relations of $\boldsymbol{H}_1, \dots, \boldsymbol{H}_n$ to $\boldsymbol{H}_0$
    $b(\boldsymbol{a}) \leftarrow p(\boldsymbol{H}_0 \text{ is true} \mid \{b(\boldsymbol{H}_i)\})$ (Appendix C)
  end for
  $b(\boldsymbol{h}_{ij}) \leftarrow b(\boldsymbol{a})$, for every flagged $\boldsymbol{h}_{ij}$ that asserts $\boldsymbol{a} \in \mathcal{A}_\ell$
end for
Reweight reasoning chains
obtain a weight w_i for every chain from one judge call
return $\hat{p}_{\text{true}} = \sum_i w_i \hat{p}_{\text{true},i} / \sum_i w_i$

B IMPLEMENTATION DETAILS

Both model settings share the decoding settings and budgets of Table 6 and differ only in the models of Table 5.

Table 5: Models for each role.

Role	Small model setting	Large model setting
Chains $\mathcal{R}_1$, $\mathcal{R}_3$	Qwen3.6-35B-A3B	Qwen3.5-122B-A10B
Chain $\mathcal{R}_2$	gpt-oss-20b	gpt-oss-120b
Direct ask	Qwen3.6-35B-A3B	Qwen3.5-122B-A10B
Extractor, inspector, verifier, judge, relation labeler	Qwen3.6-35B-A3B	Qwen3.5-122B-A10B
Generator LLM	Qwen3-8B	Qwen3-8B

Table 6: Decoding settings and budgets.

Component	Setting	Value
Qwen sampling	temperature	1.0
	top-p	0.95
	top-k	20
	presence penalty	1.5
gpt-oss sampling	temperature	1.0
	top-p	1.0
Request seeds	chains $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$	1, 2, 3
	belief-tree calls	42
	other calls	none
Chain	searches before the answer is forced	4
	passages per search	3 by default, at most 5
	characters kept from each search result	4000
	tokens per response	32768
Extractor	characters of reasoning read	20000
	tokens per response	8192
Inspector	spread threshold on d_s	0.25
	crux questions	at most 3
	questions after de-duplication	at most 12
	tokens per response	32768
Verifier	searches before the answer is forced	4
	characters kept from each search result	4000
	tokens per response	32768
Belief tree	paraphrases requested	3
	generator temperature	0.6
	generator tokens per response	16384
	passages retrieved per query	4
	characters in the evidence block	6000
	labeling and scoring tokens per response	8192
Reweighting judge	tokens per response	16384

C BELIEF-TREE PROPAGATION FOR UNRESOLVED CLAIMS

This appendix computes the posterior of Eq. (1) for an unresolved claim $\boldsymbol{H}_0$ in Section 3.4.2.

BTProp. BTProp (Hou et al., 2025) estimates the truth of a statement from a belief tree. The root is the statement, and each child is a statement logically related to its parent. Every node u has a hidden truth value $Z_u \in \{T, F\}$ and an observed truthfulness score $b(u)$ from the model. BTProp models the tree as a hidden Markov tree with three distributions. The emission probability $\Pr(b(u) \mid Z_u)$ describes how the score depends on the truth of u. The transition probability $\Pr(Z_v \mid Z_u)$ describes how the truth of a child v depends on the truth of its parent u, and it is fixed by their logical relation. The prior $\Pr(Z_{\text{root}})$ of the root is uniform. The belief in the root is the posterior $\Pr(Z_{\text{root}}{=}T \mid \{b(u)\})$, which BTProp computes with an upward pass. Let $\beta(z, u)$ be the probability of all scores in the subtree $\mathcal{T}(u)$ of u when u has truth value z. For a leaf, $\beta(z, u) = \Pr(b(u) \mid Z_u{=}z)$. For an inner node whose children $\text{ch}(u)$ are independent given the node,

$$\beta(z, u) = \Pr(b(u) \mid Z_u{=}z) \prod_{v \in \text{ch}(u)} \sum_{z' \in \{T,F\}} \Pr(Z_v{=}z' \mid Z_u{=}z)\, \beta(z', v). \tag{3}$$

The pass starts at the leaves and carries the evidence in the children's scores up to the root, where the uniform prior gives the belief $\beta(T, \text{root})/[\beta(T, \text{root}) + \beta(F, \text{root})]$.

Our belief tree. For an unresolved claim $\boldsymbol{H}_0$, the tree has one level. $\boldsymbol{H}_0$ is the root, and the generated statements $\boldsymbol{H}_1, \ldots, \boldsymbol{H}_n$ are its leaves. Exact copies of $\boldsymbol{H}_0$ are dropped, so $n \leq 3$. The root keeps the truthfulness score that the verifier LLM gave the candidate answer in the first layer, $b(\boldsymbol{H}_0) = b(\boldsymbol{a})$. The evidence retrieved with $\boldsymbol{H}_0$ and with each $\boldsymbol{H}_i$ is merged into one block,

against which the verifier LLM gives the truthfulness scores $b(\boldsymbol{H}_i)$. We depart from BTProp in two places. First, the relation r_i of $\boldsymbol{H}_i$ to $\boldsymbol{H}_0$ is a label from the LLM rather than the output of an NLI model. The labels are SYNONYM, SUPPORT, CONTRADICT, and UNRELATED, and Table 7 maps each label to BTProp's transition probabilities. Second, we take the truthfulness score directly as the emission probability. With a constant κ shared by both truth values, $\Pr(b(u) \mid Z_u{=}T) = \kappa\, b(u)$ and $\Pr(b(u) \mid Z_u{=}F) = \kappa\,[1 - b(u)]$. BTProp instead estimates the emission probability as a histogram of confidence scores over labeled true and false statements, and unresolved claims have no labels. Using the truthfulness score directly treats the verifier LLM as calibrated. The constant κ cancels from the posterior, so we omit it below.

Table 7: Transition probabilities of each relation label and the resulting odds factor λ_i for a child $\boldsymbol{H}_i$ with truthfulness score $b = b(\boldsymbol{H}_i)$.

Label r_i	BTProp relation	$\Pr(Z_{\boldsymbol{H}_i}{=}T \mid Z_{\boldsymbol{H}_0}{=}T)$	$\Pr(Z_{\boldsymbol{H}_i}{=}T \mid Z_{\boldsymbol{H}_0}{=}F)$	λ_i
SYNONYM	equivalence, $\boldsymbol{H}_0 \Leftrightarrow \boldsymbol{H}_i$	1	0	$b/(1-b)$
SUPPORT	entailment, $\boldsymbol{H}_0 \Rightarrow \boldsymbol{H}_i$	1	$1/2$	$2b$
CONTRADICT	contradiction, $\boldsymbol{H}_0 \Rightarrow \neg\boldsymbol{H}_i$	0	$1/2$	$2(1-b)$
UNRELATED	neutral, discarded	$1/2$	$1/2$	1

Upward pass. The score $b(\boldsymbol{H}_i)$ judges $\boldsymbol{H}_i$, not $\boldsymbol{H}_0$, so it bears on $\boldsymbol{H}_0$ only through the hidden truth of $\boldsymbol{H}_i$, which Eq. (3) sums out. For a truth value z of $\boldsymbol{H}_0$, let $\tau_{i,z} = \Pr(Z_{\boldsymbol{H}_i}{=}T \mid Z_{\boldsymbol{H}_0}{=}z)$ be the transition probability of the relation r_i. The child $\boldsymbol{H}_i$ contributes

$$\ell_{i,z} = \sum_{z' \in \{T,F\}} \Pr(Z_{\boldsymbol{H}_i}{=}z' \mid Z_{\boldsymbol{H}_0}{=}z)\, \beta(z', \boldsymbol{H}_i) = b(\boldsymbol{H}_i)\, \tau_{i,z} + [1 - b(\boldsymbol{H}_i)](1 - \tau_{i,z}). \tag{4}$$

With $\beta(z, \boldsymbol{H}_0) = \Pr(b(\boldsymbol{H}_0) \mid Z_{\boldsymbol{H}_0}{=}z) \prod_i \ell_{i,z}$ and the uniform prior, the posterior of Eq. (1) is

$$p(\boldsymbol{H}_0 \text{ is true} \mid \{b(\boldsymbol{H}_i)\}) = \frac{b(\boldsymbol{H}_0) \prod_i \ell_{i,T}}{b(\boldsymbol{H}_0) \prod_i \ell_{i,T} + [1 - b(\boldsymbol{H}_0)] \prod_i \ell_{i,F}}, \tag{5}$$

so $b(\boldsymbol{H}_0)$ acts as the prior of $\boldsymbol{H}_0$, and each child multiplies the odds of $\boldsymbol{H}_0$ by $\lambda_i = \ell_{i,T}/\ell_{i,F}$, listed in the last column of Table 7. This factor depends on both transition probabilities. SYNONYM and SUPPORT share $\tau_{i,T} = 1$, because a true $\boldsymbol{H}_0$ makes both an equivalent statement and a weaker statement it implies true. When $\boldsymbol{H}_0$ is false, an equivalent statement is false, so $\tau_{i,F} = 0$, whereas a weaker statement can still hold, so $\tau_{i,F} = 1/2$. If $\boldsymbol{H}_0$ states that a person was born in Paris in 1950, the child "born in Paris" survives a wrong year. A child scored 0.9 therefore multiplies the odds of $\boldsymbol{H}_0$ by 9 as a synonym but only by 1.8 as a support, and with $\tau_{i,T}$ alone the two would count the same.

D GENERATOR TRAINING

We train the generator LLM π_θ with a variant of GRPO (Shao et al., 2024) on claims $\boldsymbol{H}_0$ with binary labels $y \in \{0, 1\}$, with the verifier LLM frozen. Algorithm 2 gives the training loop.

Training data. The training pool holds 1946 non-decomposable claims from training splits, 699 from EX-FEVER (Ma et al., 2024), 700 from HoVer (Jiang et al., 2020), 402 from SciFact (Wadden et al., 2020), and 145 from IfQA (Yu et al., 2023). Of these, 88 are held out for validation. None of them overlaps with our evaluation sets. The evidence $\mathcal{E}_{\boldsymbol{H}_0}$ of a claim is retrieved by a Qwen3-8B search agent, with the passages that directly answer the claim removed, so that, as at inference, the evidence does not settle the claim. A frozen Qwen3-8B verifier gives $b(\boldsymbol{H}_0)$, labels the relations, and scores the generated statements.

Reward. For each claim we sample $G = 8$ outputs of three generated statements each and compute the posterior p_g of output g by Eq. (5). Dropping the index g, the reward measures the reduction in absolute error,

$$\Delta = |b(\boldsymbol{H}_0) - y| - |p - y|. \tag{6}$$

Algorithm 2 Generator training

Require: training claims $\{(\boldsymbol{H}_0, y, \mathcal{E}_{\boldsymbol{H}_0}, b(\boldsymbol{H}_0))\}$, generator π_θ, frozen verifier LLM, group size G
Ensure: trained generator π_θ
for each training batch **do**
  $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
  for each claim $\boldsymbol{H}_0$ and $g = 1, \ldots, G$ **do**
    sample generated statements $\boldsymbol{H}_{1:n}^{(g)} \sim \pi_{\theta_{\text{old}}}[\cdot \mid \boldsymbol{H}_0]$ and drop empty texts and exact copies of $\boldsymbol{H}_0$
    $p_g \leftarrow$ score the retained statements against $\mathcal{E}_{\boldsymbol{H}_0}$ and apply Eq. (5)
    $R_g \leftarrow$ reward for the change from $|b(\boldsymbol{H}_0) - y|$ to $|p_g - y|$, minus format penalties
  end for
  $\widehat{A}_g \leftarrow$ standardize R_g within the group of each claim
  update θ by the clipped GRPO objective with a KL penalty toward the reference generator
end for
return π_θ

Let $e = 1$ if $b(\boldsymbol{H}_0)$ lies on the wrong side of $1/2$, that is, $b(\boldsymbol{H}_0) \geq 1/2$ and $y = 0$ or $b(\boldsymbol{H}_0) < 1/2$ and $y = 1$, and $e = 0$ otherwise. Let $\pi_{\text{err}} = 0.245$ be the fraction of initially wrong estimates in an earlier version of the pool, which roughly balances the two cases. The belief-improvement term

$$\Gamma = \begin{cases} \max\{0, \Delta\}/\pi_{\text{err}}, & e = 1, \\ \min\{0, \Delta\}/[1 - \pi_{\text{err}}], & e = 0, \end{cases} \tag{7}$$

rewards an initially wrong estimate for moving toward the label and penalizes an initially correct one for moving away. With n retained statements, n_{dup} duplicate pairs among $\boldsymbol{H}_0$ and $\boldsymbol{H}_{1:n}$, and $[x]_+ = \max\{x, 0\}$, the reward is

$$R = \Gamma - 0.1 n_{\text{dup}} - 0.1[n - 3]_+. \tag{8}$$

An output without a usable statement or with an unparseable judgment receives a fixed negative reward.

Policy update. For a claim $\boldsymbol{H}_0$, with $\overline{R}$ and σ the mean and standard deviation of its eight rewards and a small constant ε_{adv}, the advantage of output g, shared by all of its tokens, is

$$\widehat{A}_g = \frac{R_g - \overline{R}}{\sigma + \varepsilon_{\text{adv}}}. \tag{9}$$

Let ρ_{gt} be the probability ratio of token t of output g between the current generator and the one that produced the rollouts, and $d_{gt}^{\text{KL}} = \eta - \log \eta - 1$ the per-token KL estimate, where η is the probability ratio between the reference and the current generator. The objective is

$$\mathcal{J}[\theta] = \left\langle \min\left\{ \rho_{gt} \widehat{A}_g,\ \text{clip}[\rho_{gt}, 0.8, 1.28] \widehat{A}_g \right\} - 0.005\, d_{gt}^{\text{KL}} \right\rangle_{\text{tok}}, \tag{10}$$

where the brackets average over all response tokens of all claims in the batch. We use the final checkpoint. Table 8 lists the settings.

E Retrieval corpus

Our retrieval system indexes the 7131226 English Wikipedia articles of the structured Wikipedia snapshot of 13 May 2026[1] with Pyserini (Lin et al., 2021). A query first selects a page by BM25, preferring a candidate whose title the query contains, and then ranks the sections of that page by BM25 and returns the top passages. All 421 evaluated FutureX statements resolve on or after 14 May 2026.

F Datasets and statement conversion

This appendix details the datasets introduced in Section 4.1. Table 9 gives the size and label balance of every evaluation set.

[1] https://huggingface.co/datasets/wikimedia/structured-wikipedia

Table 8: Generator training settings.

Setting	Value
Generator LLM	Qwen3-8B, all parameters
Verifier LLM during training	Qwen3-8B, frozen
Claims per batch	21
Rollouts per batch	168
Group size G	8
Statements requested per rollout	3
Sampling temperature	0.6
Sampling top-p	0.95
Optimization steps per batch	1
Optimizer	AdamW
Learning rate	10^{-6}, constant
Weight decay	0.01
Gradient clipping norm	1
Prompt budget	2048 tokens
Response budget	8192 tokens
Clipping range	$[0.8, 1.28]$
KL coefficient	0.005
Training epochs	3
Checkpoint used	step 264

Table 9: Evaluation sets.

Dataset	Size	y=1	y=0
FactCHD	971	481	490
HoVer	1000	500	500
FutureX	421	206	215
IfQA	1273	644	629

FactCHD. FactCHD (Chen et al., 2024) pairs a query with a response labeled factual or non-factual. We draw a label-balanced sample of 1000 items from its test split with a fixed seed. Claude Sonnet 4.6 folds each query into its response to form one self-contained statement that asserts what the response asserts without correcting it, and the statement keeps the item's label, with factual as $y = 1$. Merging identical statements gives 976, and removing the 5 that occur with both labels leaves 971 statements, 481 true and 490 false.

HoVer. HoVer (Jiang et al., 2020) claims from the official development split are used verbatim, with SUPPORTED as $y = 1$ and NOT-SUPPORTED, which combines refuted claims and claims with insufficient information, as $y = 0$, so evaluation on this dataset concerns evidence support rather than truth alone. We randomly sample 500 claims of each label with a fixed seed. The 1000 claims comprise 271 two-hop, 489 three-hop, and 240 four-hop claims.

FutureX. FutureX statements come from the resolved level 1 and level 2 multiple-choice, yes/no, and two-way questions of the Futurex-Past archive,[2] so this is a retrospective evaluation. Qwen3.5-122B-A10B, which does not see the labels, rewrites each candidate outcome as one sentence that predicts it with "will" by the resolution date. A statement is true if its outcome occurred, and at most one true and one false statement are kept per question. Each statement is scored on its own, without the other options. We evaluate on 421 statements, 206 true and 215 false, from 243 questions resolving between 14 May and 4 August 2026.

IfQA. IfQA (Yu et al., 2023) asks questions under counterfactual premises. We start from the 700 questions of its test split. For each question, Claude Sonnet 4.6 writes a true statement that joins the premise with the gold answer and a false statement that keeps the premise and gives a plausible wrong answer. An audit by Claude Sonnet 4.6 with web search removes 55 questions whose true statement it judged doubtful or unverifiable, and drops the false statement of 16 more because that

[2] https://huggingface.co/datasets/futurex-ai/Futurex-Past

statement may also be true. The remaining 645 questions give 1274 statements, and removing one repeated statement leaves 1273, 644 true and 629 false.

G BASELINES AND ADDITIONAL RESULTS

Table 10: All four metrics on the full evaluation sets, multiplied by 100. *FAVA uses the authors' fine-tuned 7B checkpoint in both model settings.

	Small model setting				Large model setting			
Method	**AUROC↑**	**PR-AUC↑**	**Acc@0.5↑**	**Brier↓**	**AUROC↑**	**PR-AUC↑**	**Acc@0.5↑**	**Brier↓**
FactCHD								
Direct ask	78.8	74.7	**73.2**	23.6	80.7	76.9	**75.1**	21.4
Self-consistency	81.0	78.8	72.8	21.5	81.9	80.4	73.6	20.2
RAG	79.8	74.9	72.5	24.7	82.7	80.9	72.6	20.6
RAG + self-consistency	82.0	79.8	72.7	**21.3**	83.4	82.3	72.6	**19.9**
BTProp	73.7	70.4	70.4	25.0	75.9	72.2	70.9	25.3
FactReasoner-FR1	67.5	68.5	62.2	22.4	75.7	75.7	67.6	21.4
VeriScore	62.7	58.4	59.8	30.7	68.6	62.8	65.6	25.6
FAVA*	58.6	55.2	58.4	38.6	58.6	55.2	58.4	38.6
RECCON	**82.1**	**79.9**	72.8	21.9	**83.7**	**82.4**	71.6	20.5
HoVer								
Direct ask	62.8	59.3	59.3	35.7	69.2	66.3	61.4	28.8
Self-consistency	65.9	64.1	59.2	30.6	71.3	69.1	62.3	27.6
RAG	72.8	67.0	68.4	29.2	76.6	72.8	71.4	22.8
RAG + self-consistency	75.1	72.3	69.4	24.2	77.7	74.8	70.1	21.8
BTProp	64.6	61.2	59.8	28.2	67.3	64.1	59.8	34.3
FactReasoner-FR1	64.6	59.7	59.9	**24.1**	67.8	64.0	60.2	23.5
VeriScore	57.9	54.4	52.6	28.9	62.0	56.8	59.1	29.9
FAVA*	56.6	53.7	53.0	40.6	56.6	53.7	53.0	40.6
RECCON	**75.9**	**72.7**	**70.1**	24.2	**78.0**	**74.9**	**71.9**	**21.7**
FutureX								
Direct ask	61.1	59.9	60.1	27.7	60.3	58.5	55.6	28.2
Self-consistency	62.3	63.1	58.9	27.1	59.8	59.8	55.1	28.5
RAG	64.0	63.8	**61.3**	28.5	63.2	63.1	57.0	28.8
RAG + self-consistency	65.8	68.3	58.2	27.3	63.0	64.4	57.0	28.0
BTProp	54.4	56.0	50.8	47.0	46.4	49.3	52.0	46.1
FactReasoner-FR1	50.9	51.8	49.2	**25.1**	50.7	52.0	51.3	25.2
VeriScore	48.8	50.8	48.0	25.9	50.0	50.8	50.8	**25.0**
FAVA*	50.5	51.1	51.0	45.3	50.5	51.1	51.0	45.3
RECCON	**66.9**	**69.1**	59.6	26.9	**64.2**	**65.0**	**58.4**	27.7
IfQA								
Direct ask	78.9	74.7	72.7	23.7	85.6	83.0	77.7	18.1
Self-consistency	83.0	81.9	73.9	18.8	87.5	87.9	77.9	15.5
RAG	90.3	86.3	**84.9**	13.5	92.1	90.3	83.1	12.1
RAG + self-consistency	93.2	92.5	**84.9**	**10.8**	94.0	93.9	85.4	**10.0**
BTProp	74.2	67.5	70.4	22.7	73.0	67.0	64.9	29.4
FactReasoner-FR1	65.1	61.3	57.7	23.2	75.7	72.9	64.5	20.3
VeriScore	52.4	50.7	50.7	25.9	57.7	54.2	56.0	25.6
FAVA*	60.1	55.6	60.0	39.0	60.1	55.6	60.0	39.0
RECCON	**93.5**	**92.8**	84.5	11.0	**94.2**	**94.1**	**85.7**	10.1

Table 11: All four metrics on the divergence set, multiplied by 100.

	Small model setting				Large model setting			
Method	**AUROC↑**	**PR-AUC↑**	**Acc@0.5↑**	**Brier↓**	**AUROC↑**	**PR-AUC↑**	**Acc@0.5↑**	**Brier↓**
FactCHD								
Direct ask	61.8	40.4	**63.7**	30.5	60.9	35.9	**61.1**	31.8
Self-consistency	61.1	**45.3**	61.5	27.1	**64.4**	41.5	55.7	28.8
RAG	60.0	37.3	56.4	37.3	61.9	40.4	50.3	32.8
RAG + self-consistency	61.2	41.7	57.3	**25.7**	58.1	33.3	50.8	**27.7**
RECCON	**63.3**	41.2	57.7	28.3	62.5	**44.1**	45.4	30.7
HoVer								
Direct ask	53.1	44.1	52.2	40.7	53.5	45.4	49.6	37.1
Self-consistency	53.7	43.5	51.5	35.1	54.1	47.2	49.3	34.9
RAG	62.1	49.4	59.3	36.5	61.8	**53.1**	58.3	31.3
RAG + self-consistency	66.0	57.1	62.3	**23.4**	56.5	48.8	54.0	26.5
RECCON	**70.8**	**60.9**	**64.5**	**23.4**	**63.5**	51.8	**60.4**	**26.1**
FutureX								
Direct ask	54.7	41.2	51.6	30.8	58.6	42.8	48.1	30.7
Self-consistency	54.6	40.4	52.3	29.5	56.6	45.4	45.2	31.4
RAG	54.8	39.9	**54.7**	32.4	57.7	47.3	52.9	30.7
RAG + self-consistency	55.8	43.5	47.7	29.2	53.5	40.8	50.0	29.4
RECCON	**61.6**	**52.5**	52.3	**27.9**	**61.4**	**52.0**	**55.8**	**28.2**
IfQA								
Direct ask	61.9	50.1	59.9	34.2	68.1	57.1	64.2	28.7
Self-consistency	64.1	53.2	60.2	28.5	70.0	60.6	63.5	24.0
RAG	70.3	57.3	**64.5**	31.2	65.8	59.4	56.5	28.6
RAG + self-consistency	73.4	63.6	**64.5**	**21.9**	75.8	69.6	65.8	**20.6**
RECCON	**76.4**	**69.8**	63.1	22.4	**77.4**	**69.8**	**66.8**	20.9

G.1 BASELINES

Direct ask and RAG baselines. Direct ask queries the Qwen model of each model setting once per statement without tools, with the instruction of Appendix I.7 (Appendix I.8 for future-event statements). Self-consistency averages $k = 3$ Direct ask truthfulness scores sampled with different seeds. RAG uses the initial truthfulness score $\hat{p}_{\mathrm{true},1}$ of the Qwen chain $\mathcal{R}_1$, and RAG + self-consistency averages the initial truthfulness scores $\hat{p}_{\mathrm{true},i}$ of the three chains.

External baselines. BTProp runs without retrieval and with the backbone's thinking mode on. Each node's confidence is read from the logprobs of a True/False label the model states after reasoning, and the HMM emission probabilities, which the original release estimated for ChatGPT, are re-estimated for this backbone from the root statements of the other three datasets. FactReasoner-FR1 splits a statement into atoms, retrieves three contexts for each atom from our BM25 index, and uses the mean posterior probability of its atoms as $\hat{p}_{\mathrm{true}}$. VeriScore extracts verifiable claims, verifies each against up to ten passages from the same index, and uses the fraction of supported claims as $\hat{p}_{\mathrm{true}}$. A statement that either method cannot score, for example because no claim is extracted or no context remains, gets $\hat{p}_{\mathrm{true}} = 0.5$. FAVA runs the authors' released checkpoint with its own retrieval, and an output with a factual error tag, without tags, or with only subjective or unverifiable tags gives $\hat{p}_{\mathrm{true}} = 0$, 1, or 0.5.

G.2 SIGNIFICANCE

Table 12 tests whether the gain of RECCON over RAG + self-consistency in the macro average of AUROC could arise from sampling variation. We resample the statements of each dataset with replacement, recompute the gain on each dataset, and average it over the four datasets.

Table 12: Paired bootstrap of the gain of RECCON over RAG + self-consistency in the macro average of AUROC, with 10000 resamples of the statements of each dataset. p is the fraction of resamples without a gain.

	Small model setting			Large model setting		
Evaluation set	**Gain**	**95% CI**	p	**Gain**	**95% CI**	p
Full evaluation sets	0.6	[0.2, 1.0]	0.002	0.5	[0.2, 0.9]	0.001
Divergence set	3.9	[1.4, 6.3]	0.001	5.2	[2.5, 8.0]	<0.001

G.3 ALL METRICS

Table 10 reports all four metrics for every method on the full evaluation sets, and Table 11 for RECCON and the Direct ask and RAG baselines on the divergence set. Besides AUROC, PR-AUC summarizes precision across recall levels when detecting false statements, Acc@0.5 measures classification accuracy at a probability threshold of 0.5, and the Brier score measures the mean squared error between predicted truthfulness scores and binary labels (lower is better).

H CASE STUDY

We examine two statements from the small model setting on which reweighting changes the prediction of RAG + self-consistency at the threshold 0.5, a success on HoVer and a failure on IfQA. In both, $\mathcal{R}_1$ and $\mathcal{R}_3$ come from Qwen3.6-35B-A3B and $\mathcal{R}_2$ from gpt-oss-20b.

Success: targeted retrieval settles a hop the chains could not. The statement is *"Marvin John Heemeyer the muffler repair shop owner inspired a director's film that Mikhail Krichman photographed. The director was a guest at the Bergman Week 2011"*, and its label is true. All three chains establish that Heemeyer inspired *Leviathan* and that Krichman photographed it. They differ only on the last hop, whether its director Andrey Zvyagintsev was a guest at the Bergman Week 2011 (Table 13). $\mathcal{R}_1$ and $\mathcal{R}_3$ search for the director together with the event and retrieve only his biography, since retrieval prefers a page whose title the query contains (Appendix E). $\mathcal{R}_2$ never searches for the event. $\mathcal{R}_1$ and $\mathcal{R}_2$ therefore find the last hop unsupported and judge the statement false, whereas $\mathcal{R}_3$ accepts it, so RAG + self-consistency predicts 0.39. The inspector turns the disputed hop into an asymmetry question. The verifier searches for the event itself and retrieves the Bergman Week article, which lists Zvyagintsev among the guests of 2011. The judge down-weights $\mathcal{R}_1$ and $\mathcal{R}_2$, whose verdicts this finding contradicts, and the prediction rises to 0.70.

Table 13: Success case. Each search is quoted with the article that retrieval returned, and the verifier row gives its verdict and truthfulness score.

Agent	$\hat{p}_{\text{true},i}$	Search on the last hop	Last hop	w_i
$\mathcal{R}_1$	0.10	"Andrey Zvyagintsev Bergman Week 2011 guest" → Andrey Zvyagintsev	unsupported	0.15
$\mathcal{R}_2$	0.16	none	unsupported	0.15
$\mathcal{R}_3$	0.90	"Andrey Zvyagintsev Bergman Week 2011 guest" → Andrey Zvyagintsev	accepted	0.85
Verifier		"Bergman Week 2011 guests attendees" → Bergman Week	supported (1.00)	
RAG + self-consistency 0.39, RECCON 0.70, label true				

Failure: the judge is misled by the counterfactual premise. The IfQA statement is *"If Greece hadn't triumphed in the UEFA Euro 2004 final match, Portugal would have been declared the winners"*, and its label is true. $\mathcal{R}_1$ and $\mathcal{R}_3$ reason that a final between two teams must produce a winner, so the other finalist wins. $\mathcal{R}_2$ argues that a drawn final goes to extra time and a penalty shootout instead of being awarded to the opponent, and judges the statement false, so RAG + self-consistency predicts 0.67 (Table 14). The inspector groups the disagreement as a contradiction about how a drawn final is decided. The verifier correctly confirms that such a final goes to extra time and possibly a shootout, and refutes the sub-claims of $\mathcal{R}_1$ and $\mathcal{R}_3$ that the opponent is simply declared

the winner. This correction does not change the answer. Extra time and a shootout still produce a winner, so a final that Greece does not win is won by Portugal. The judge instead takes the finding as overturning $\mathcal{R}_1$ and $\mathcal{R}_3$ and trusts $\mathcal{R}_2$, and the prediction drops to 0.28. When a statement rests on a counterfactual premise, the judge can mistake a refuted step of a chain for a refutation of its conclusion.

Table 14: Failure case. Each chain is shown with the sub-claim that the judge takes as decisive and the verifier's verdict and truthfulness score for it.

Agent	$\hat{p}_{\text{true},i}$	Decisive sub-claim	Verdict	w_i
$\mathcal{R}_1$	1.00	In a binary two-team tournament final, if one team fails to win, the opposing team is automatically declared the winner.	refuted (0.05)	0.15
$\mathcal{R}_2$	0.05	In a knockout tournament final, a draw after regulation leads to extra time and a potential penalty shootout instead of automatically awarding the match to the opposing team.	supported (0.95)	0.90
$\mathcal{R}_3$	0.95	In a two-team knockout final governed by standard rules, if one team fails to win, the opposing team is declared the winner.	refuted (0.05)	0.15
RAG + self-consistency 0.67, RECCON 0.28, label true				

I PROMPT TEMPLATES

This appendix presents the prompt templates of every agent for fact-checking statements. The instructions call the statement R, which is s in the main text. Appendix I.8 gives the templates for future-event statements. Fields in braces or in double angle brackets are filled in at run time, such as the statement, a chain's reasoning, or the evidence block, and doubled braces appear to the model as single braces. Braces are literal in the templates with double angle brackets and in the last paragraph of the chain and Direct ask templates, which is appended after the fields are filled. Em dashes are printed as double hyphens, and a few pictorial symbols are rendered as plain text. Small differences across runs, such as the wording of the tool description, are not shown. The judge reads the status of each verification question, which is confirmed when the verifier supports a candidate or establishes another answer, refuted when it only rules candidates out, and not_found otherwise. A claim line marked verified (unsettled) belongs to a settled question but asserts a candidate that the verifier left unsettled. Some templates also ask for fields that no later stage reads, such as the direction of a declarative claim. The provenance of an extracted claim is read only to mark a judgment of the future-event extractor as [estimate] for the inspector.

I.1 CHAIN GENERATION

Chain

```
You are a careful fact-checker with a tool: wiki_search(query, topk=3) -- Wikipedia BM25
retrieval. Decide the probability that statement R is factually TRUE, reported as JSON.

  "p_true" = P(R is true), the truthfulness score in [0.0, 1.0]. 1.0 = certain TRUE, 0.0 =
certain FALSE, 0.5 = genuine 50-50.

HOW TO WORK:
- Use wiki_search to GROUND the real-world facts that decide R -- entities, numbers, dates,
rankings, the real direction of causation. Search the specifics; do NOT rely on memory. You
may call it up to {K} times.
- COUNTERFACTUAL / CONDITIONAL ("if A, then B"): treat the antecedent A as GIVEN /
hypothetical, even if it is contrary to fact. Reason about whether B follows UNDER A. NEVER
mark R false merely because A did not actually happen, or because B is not true in the
actual (non-A) world.
- Resolve double negatives explicitly. Be calibrated: use the full 0-1 range; reserve <0.2
for clearly false and >0.8 for clearly true.

R: {stmt}

When you have searched enough, STOP calling the tool and end your reply with EXACTLY one
line:
{{"p_true": 0.XX}}
```

```
Before that final JSON line, write a brief, self-contained explanation of your judgment in
two to five sentences. State the specific facts or estimates your conclusion relies on and
how they support or oppose R. Distinguish retrieved evidence from your own knowledge or
estimates, and preserve negations. This explanation must be visible in your final answer.
The last line must still be exactly {"p_true": 0.XX}.
```

I.2 CLAIM EXTRACTION

Claim extraction

```
A fact-checker analyzed statement R and produced the reasoning below (its private reasoning,
 then its final answer). Your job: list the ATOMIC FACTUAL CLAIMS that this reasoning
actually RELIES ON to reach its conclusion about R.

R: {R}

=== PRIVATE REASONING ===
{think}

=== FINAL ANSWER ===
{answer}

Rules:
- A claim is one self-contained, checkable factual assertion (one entity/number/date/
relation per claim). Rewrite it so it is understandable without the surrounding text.
- Only include claims the conclusion DEPENDS on -- skip restatements of R itself, meta-
commentary about the checking process, and hedged musings the reasoning ends up discarding.
- Tag each claim's provenance: "search" if the reasoning grounded it in passages it
retrieved; "memory" if it was asserted from the checker's own knowledge.
- Keep the claim's own polarity: state what the reasoning ASSERTS, even if that assertion is
 what makes it doubt R.

Output ONLY a JSON object in this exact format (no markdown, no explanation):
{{"claims": [{{"text": "...", "src": "memory"}}, {{"text": "...", "src": "search"}}]}}
```

I.3 CONTRASTING REASONING CHAINS

Inconsistency groups and questions

```
Independent fact-checking chains assessed statement R and reached different conclusions.
Below are the sub-claims each chain's reasoning rests on, grouped by chain.

R: {R}

SUB-CLAIMS:
{claims}

Find the INCONSISTENCIES among the chains, and turn each into ONE verification question.

An inconsistency is a group of sub-claims on one subject of one of two types:
- "contradiction": sub-claims from different chains state incompatible facts on the subject
(two different years, two different people).
- "asymmetry": one or more chains make a claim on a subject the remaining chains do not
discuss at all.
Sub-claims that all chains state consistently are NOT inconsistencies: leave them out.

Grouping:
- Put into one group the claims that speak to the same underlying fact, so that a single
question could settle all of them at once.
- Each claim belongs to AT MOST ONE group. When a claim bears on more than one subject, put
it in the group of the subject its chain's conclusion depends on most.
- Claims about DIFFERENT facts must stay in different groups, even when they concern the
same entity. Two claims about the same person's birth date and birthplace are two subjects,
not one.
- Preserve the granularity under dispute: if the chains differ over a specific day, figure
or attribution, the subject is that specific detail, not a coarser version of it.

Questions, one per group:
- "question" must be an actual question ending in "?", self-contained (name the entities,
dates and quantities), and must ask for the EXACT detail under dispute, never a coarser
version of it. Never restate a claim as a declarative: that puts one side's answer inside
the question and leaves the verdict nothing to attach to.
```

```
- A claim marked [estimate] is a base rate the chain assumed, not a claim about a specific
event. Ask about the historical record that would settle it ("In FIFA World Cup matches
between a top-ranked and a lower-ranked side, how often does the favourite win?"), never "
what is P(X)".
- "candidates" holds the competing answers, taken from the group's own claims. Give each
answer the ids of the claims that put it forward, copied from the brackets above. Add {{"
answer": "not established", "claim_ids": []}} when some chain holds the detail unknown. The
id is how a verdict finds its way back to the claim it judges, so an answer with no id can
be charged to nobody.
- Give every position the group's claims actually take, one candidate each. A group whose
claims take only ONE position (an asymmetry) gets ONE candidate: that is what a search
should check, and the verifier judges each candidate on its own. Do NOT invent an opponent
the claims do not contain.

Output ONLY a JSON object in this exact format (no markdown, no explanation):
{{"groups": [{{"type": "contradiction", "subject": "<the one fact this group turns on>", "
claims": ["s1:c3", "s2:c1"], "question": "...?", "candidates": [{{"answer": "...", "
claim_ids": ["s1:c3"]}}, {{"answer": "...", "claim_ids": ["s2:c1"]}}]}]}}
```

Crux questions

```
Statement R was analyzed independently by several fact-checking chains. Their conclusions
DIVERGE. Below are the sub-claims each chain's judgment rests on, with the initial
truthfulness score it assigned to R.

R: {R}

{chains}

Name the DECISIVE MISSING FACT(S): the smallest set of concrete, searchable factual
questions which, once answered, would settle the disagreement about R. Look for premises the
 chains needed and did not establish -- a fact NONE of them discusses, one only partly
pinned down, or one a single chain asserts and the others never examine.

Each question must be self-contained (name the entities, dates and quantities), end in "?",
and be checkable by a search engine. Usually one or two questions suffice.

Give each crux in the same form as an inconsistency group:
- "subject": the one fact the question turns on.
- "claims": the ids of the sub-claims above that speak to it, copied from the brackets.
Empty when no chain discusses it.
- "candidates": at least ONE candidate answer for every crux. When sub-claims speak to it,
give the competing answers they give, one per position, each with the ids of the sub-claims
that give it. A crux whose sub-claims take only ONE position gets ONE candidate. When no sub
-claim speaks to it, give the single most plausible answer, with empty "claim_ids".

Output ONLY a JSON object in this exact format (no markdown, no explanation):
{{"groups": [{{"type": "crux", "subject": "<the one fact this question turns on>", "claims":
 ["s1:s12"], "question": "...?", "candidates": [{{"answer": "...", "claim_ids": ["s1:s12
"]}}]}}, {{"type": "crux", "subject": "...", "claims": [], "question": "...?", "candidates":
 [{{"answer": "...", "claim_ids": []}}]}}]}}
```

Removing duplicate questions

```
Verification questions were written for statement R from several sources, and some of them
may ask about the same fact in different words.

R: {R}

QUESTIONS:
{qs}

Find the questions that ask about the SAME fact, so that one search answering one of them
would answer the others. Two questions about the same entity but different facts (a birth
date and a birthplace) are NOT duplicates. A broader and a narrower question (the year and
the exact day) are NOT duplicates. List only sets of two or more; a question with no
duplicate is left out.

Output ONLY a JSON object in this exact format (no markdown, no explanation):
{{"duplicates": [[0, 3], [2, 5]]}}
```

I.4 VERIFICATION

Candidate verification

```
You are a careful fact-checker with a tool: wiki_search(query, topk=3) -- Wikipedia BM25
retrieval. Determine what the RETRIEVED EVIDENCE supports -- strictly from retrieved
passages, do NOT rely on memory. You may call the tool up to {K} times; if generic queries
do not surface the detail in question, search likely phrasings of the answer directly.

QUESTION: {q}
CANDIDATES: {cands}

Judge each CANDIDATE, not the question. A question like "who was the captain" has no truth
value of its own, so a verdict on it cannot say which answer the evidence ruled out -- and
the analysis that argued for the losing answer is the one whose reasoning is in doubt. Give
every candidate exactly one verdict:
- "supported": a retrieved passage positively establishes THIS candidate as the answer, AT
THE GRANULARITY ASKED (finding only a coarser fact -- e.g. the year when the month is asked
-- is NOT support).
- "refuted": a retrieved passage positively establishes THIS candidate is NOT the answer (it
 states an incompatible answer, or contradicts the candidate outright).
- "unsettled": the retrieved passages do not decide this candidate either way.

CRITICAL: not finding a passage that supports a candidate makes it "unsettled", NEVER "
refuted". Wikipedia's silence is not evidence against a candidate -- many true facts (
specialized biology, non-English or niche topics) are simply absent from this corpus. Use "
refuted" only when a passage AFFIRMATIVELY rules that candidate out.

Also give every candidate a "truthfulness" score: the probability, in [0,1], that THIS
candidate is the correct answer, given the retrieved passages. A supported candidate is
usually above 0.85 and a refuted one below 0.15. For an unsettled candidate, give your best
estimate from what the passages do show, and your own knowledge where they are silent;
unsettled does not mean 0.5.

If the evidence establishes an answer that is NOT among the candidates, put it in "
established_answer"; otherwise leave that field out.

When you have searched enough, STOP calling the tool and end your reply with EXACTLY one
line of JSON:
{{"finding": "<what the evidence does or does not establish, 1-2 sentences, citing the
decisive passage detail>", "candidate_verdicts": [{{"candidate": "<copy the candidate
verbatim>", "verdict": "supported|refuted|unsettled", "truthfulness": 0.0}}], "
established_answer": "<the answer, when it is none of the candidates or none were offered
>"}}
```

I.5 BELIEF-TREE VERIFICATION

Declarative claim from a question and a candidate

```
A fact-checking pipeline needs to convert a verification question and one CANDIDATE answer
into a single self-contained POSITIVE declarative statement whose truth can be checked
against a reference corpus.

Context -- the root statement under review (do NOT judge it, it is context only):
R: {R}

QUESTION: {q}
CANDIDATE ANSWER: {cand}

Write ONE declarative sentence that asserts exactly the candidate, in the candidate's own
direction, plus two labels:
- "statement": a direct assertion of the candidate, at the question's granularity.
  * If the candidate asserts the fact HOLDS, write the affirmative statement.
  * If the candidate asserts the fact does NOT hold -- including phrasings like "no", "did
not", "no indication/evidence/record of it" -- write a DIRECT negative statement about the
fact itself (subject + "did not" / "was not" / "is not"). Never write a sentence about
evidence or records ("there is no evidence that ..." becomes "<subject> did not ...").
  * If the candidate names a DIFFERENT concrete value, person, or event than the question
presumes, assert the candidate's own content (e.g. "the award went to X"), not a negation of
 the question's version.
  * MINIMAL rewrite: reuse the question's own wording wherever possible; do not paraphrase,
do not embellish, do not add facts;
  * self-contained: keep the entities, dates, and quantities the question names.
- "direction": "positive" if the statement is affirmative, "negative" if it asserts the fact
 does not hold.
```

```
- "checkable": "external" if the statement's truth rests on a fact about the world that a
reference corpus could contain; "relational" if it is definitional, tautological, or
decidable by reasoning alone with no external fact to retrieve.

Output ONLY a JSON object: {{"statement": "...", "direction": "positive|negative", "
checkable": "external|relational"}}
```

Paraphrase generation

```
You are given a single sub-claim H. Identify the SINGLE most truth-bearing key slot in H (
the one value whose change most affects whether H is true -- a number, date, place, person,
or quantity), then GENERATE 3 DIFFERENT PARAPHRASES of H that keep that slot's value EXACTLY
 unchanged. Do NOT search; rely on your own knowledge.

GOAL: denoising. Each paraphrase is scored against the same evidence and pooled with H's own
 truthfulness score, so a paraphrase is only useful if it is JUST AS EASY TO VERIFY as H. It
 is a RE-READING, not a re-styling.

!! Do NOT change the key slot's value, or any other value in H. Every number, date, place,
person and named entity must appear with the SAME value. If a paraphrase's truth could
differ from H's, it is a FAILURE.
!! Do NOT bury, hedge or generalise the key slot -- "in 2007" must not become "in the 2000s"
 and must not disappear.
!! Do NOT make the sentence harder to check than H: keep it as SHORT and as DIRECT as H,
keep the same grammatical subject where possible, and do NOT pile on subordinate clauses or
nominalise the main verb. A fluent but convoluted rewrite is worse than a plain one.
!! Do NOT add information H does not state, and do NOT drop information H does state.
[ok] DO make the paraphrases differ from EACH OTHER, not just from H.
[ok] DO reword genuinely -- a different lexical choice or a different but equally plain
construction.

**Example:**
H: The iPhone was first released by Apple in 2007.
{"slot":"year","original_value":"2007","variants":[
  {"value":"2007","statement":"Apple released the first iPhone in 2007."}]}

!! The example above is ILLUSTRATIVE ONLY -- never reuse its entities or values; answer
solely about the actual H below.

OUTPUT strict JSON only. "slot" and "original_value" must be the REAL slot and REAL value
from H:
{"slot":"<chosen slot>","original_value":"<H's value>","variants":[{"value":"<same value>","
statement":"<full statement>"}]}

H: <<STATEMENT>>
Output JSON only:
```

Relation labeling

```
You label the logical RELATION between an ORIGINAL sub-claim and each of several VARIANT sub
-claims.

!!!! INDEPENDENCE -- READ FIRST. Your judgment must be COMPLETELY INDEPENDENT of:
  - TRUTH / EVIDENCE -- you are NOT deciding which claim is factually correct. You have no
evidence and need none. Judge ONLY how the two claims' values relate.
  - PLAUSIBILITY / ESTIMATED TRUTHFULNESS -- do NOT reason about which value "seems more
likely". A value you believe is false still has a well-defined relation to the original.
  - HOW THE VARIANT WAS PRODUCED -- ignore any notion of how it was generated; look ONLY at
the two statements in front of you.

STEP 1 -- COEXISTENCE TEST (do this FIRST for every variant, before choosing a label):
  Ask: "Could ONE real subject make BOTH the original AND the variant true -- both being
true of that subject (not necessarily at the same instant)?"
  MANY attributes are REPEATABLE / MULTI-VALUED, so DIFFERENT values COEXIST (answer = YES):
    - achievements won more than once: World Cup / championship / award / title in different
 years (won in 2002 AND 1994 AND 2022 -- all true of the same team)
    - events attended, places visited or lived, roles held, works created, products shipped,
 jobs held
    - overlapping TIME PERIODS a span can cover: lived in the 20th century AND the 21st;
active in the 1990s AND the 2000s; reigned in one decade AND the next
  Only a truly SINGLE-VALUED attribute answers NO (the values are mutually exclusive):
    - birth year, death year, THE unique birthplace, the current capital, one biological
father, the single tallest mountain, the one winner of ONE specific event instance.
```

```
  !! A different-looking or far-apart value is NOT enough for contradict -- repeatable/
period attributes still COEXIST. Decide by the CHANGED piece only; keep the rest of the
sentence fixed.

STEP 2 -- pick exactly ONE relation:
  - "contradict" -- coexistence = NO (single-valued attribute; at most one of the two can be
 true). (born 1879 vs 1880; capital Paris vs Lyon; the director is Nolan vs Spielberg.)
  - "synonym"    -- SAME value, only reworded / another surface form of the identical fact (
Jan 1st = 1st of January; Bombay = Mumbai; "1,000" = "one thousand").
  - "support"    -- coexistence = YES AND the original ENTAILS the variant -- a strict
generalization / superset, so whenever the original holds the variant necessarily holds too
(California -> America; "born 1990" -> "born in the 1990s"). Variant is BROADER.
  - "unrelated"  -- coexistence = YES but NOT entailed: a repeatable / multi-valued
different value (World Cup in 2002 vs 1994; lived 20th vs 21st century; visited Paris vs
Tokyo), OR a change to a different fact entirely. Its truth does not pin down the original.

!! The bracketed examples above are ILLUSTRATIVE ONLY -- never reuse their values; judge
solely the actual ORIGINAL and VARIANTS below.

OUTPUT strict JSON only (one entry per variant, in order); include the STEP-1 coexist answer
:
{"relations":[{"variant":"<variant statement>","coexist":"yes|no","relation":"synonym|
support|contradict|unrelated","why":"<short reason>"}]}

ORIGINAL: <<STATEMENT>>
VARIANTS:
<<VARIANTS_BLOCK>>
Output JSON only:
```

Claim scoring

```
You are a careful fact-checker. Estimate the truthfulness score of the STATEMENT, the
probability that it is factually true, and report it as JSON {"p_true": 0.XX}. Do NOT search
; use the EVIDENCE below (if any) plus your own knowledge.

!! ABSENCE OF EVIDENCE IS NOT EVIDENCE OF FALSEHOOD. If the EVIDENCE is empty, off-topic, or
 does not mention the statement, that is a SEARCH MISS -- NOT a refutation. Do NOT output a
low probability for that reason; fall back to your OWN knowledge:
  confidently true -> 0.8-0.95 . leans true -> 0.6-0.75 . genuinely unsure -> ~0.5 .
  leans false -> 0.25-0.4 . confidently false -> 0.05-0.2.
Assign a low probability ONLY when the statement is actually false (per evidence or your
knowledge), NEVER merely because evidence is missing.

For a counterfactual / conditional claim, accept its antecedent as GIVEN and judge whether
the consequent follows from the real-world mechanism -- do not lower p_true just because the
 antecedent did not actually happen.

EVIDENCE:
<<EVIDENCE_BLOCK>>

STATEMENT: <<STATEMENT>>
Output ONLY JSON {"p_true": 0.XX}:
```

I.6 CHAIN REWEIGHTING

Chain weights

```
Independent reasoning chains each judged whether statement R is true. A verification step
then checked, with targeted retrieval, the facts the chains disagreed on. Your job: give
each chain a weight, the probability that its verdict on R is right.

R: {R}

{chains}

VERIFICATION QUESTIONS (targeted retrieval on the disputed facts)
{questions}

How to read a claim line: "score" is the claim's truthfulness score, the probability that
the claim itself is true. Only the claims that reached verification are listed. "src" says
where the score comes from: "verified (supported)" or "verified (refuted)" means targeted
retrieval settled the question the claim answers; "model estimate" and "unsettled" mean
retrieval did not settle it and the number is only an estimate. A score from a model
estimate or an unsettled question that is below 0.5 is NOT reliable: it only says the
```

```
evidence did not confirm the claim, not that the claim is false. Never treat such a claim as
 overturned; treat it as open.

First decide what the verification record says about R ITSELF:
- Take only SETTLED findings (status confirmed or refuted) that concern the same entities,
events and attributes R is about. A finding about a different person, work or thing that
shares a name is irrelevant. A question whose status is not_found settles nothing: missing
evidence is not evidence against a claim.
- Ask: if these settled findings are true, is R true, false, or still undetermined? Most
side details (who founded a company, an exact count, a secondary date) do not decide R even
when they are refuted.

Then, for EACH chain:
1. decisive: the claims its verdict on R actually rests on, the ones whose falsity would
flip its verdict. A side detail the chain mentions in passing is not decisive.
2. weight:
   - The settled findings make R's truth value the OPPOSITE of the chain's verdict: 0.1 to
0.2. Cite the question or claim in "overturned_by".
   - The settled findings make R's truth value MATCH the chain's verdict: 0.85 to 0.95.
   - Otherwise R is still undetermined by settled findings: weight 0.35 to 0.65. Within that
 band, lean on the decisive claims' truthfulness scores and on how directly the chain's
verdict follows from what it knows about R. A refuted side detail or a low truthfulness
score on a non-decisive claim moves the weight at most slightly.
A chain whose side details were wrong can still be right about R. Do not reward a chain for
agreeing with the others. A weight below 0.35 must cite, in "overturned_by", the settled
finding that makes R's truth value the opposite of the chain's verdict.

End your reply with EXACTLY one line of JSON:
{{"R_by_findings": "true|false|undetermined", "chains": [{{"label": "A", "decisive": ["A2"],
 "weight": 0.15, "overturned_by": "Q1", "why": "<one sentence>"}}]}}
with one entry for each of: {labels}
```

I.7 Direct ask

Direct ask

```
You are a careful fact-checker. Using ONLY your own knowledge (you have NO search tool),
decide the probability that statement R is factually TRUE, reported as JSON.

  "p_true" = P(R is true), the truthfulness score in [0.0, 1.0]. 1.0 = certain TRUE, 0.0 =
certain FALSE, 0.5 = genuine 50-50.

HOW TO WORK:
- Reason through the real-world facts that decide R -- entities, numbers, dates, rankings,
the real direction of causation.
- COUNTERFACTUAL / CONDITIONAL ("if A, then B"): treat the antecedent A as GIVEN /
hypothetical, even if it is contrary to fact. Reason about whether B follows UNDER A. NEVER
mark R false merely because A did not actually happen, or because B is not true in the
actual (non-A) world.
- Resolve double negatives explicitly. Be calibrated: use the full 0-1 range; reserve <0.2
for clearly false and >0.8 for clearly true.

R: {stmt}

End your reply with EXACTLY one line:
{{"p_true": 0.XX}}

Before that final JSON line, write a brief, self-contained explanation of your judgment in
two to five sentences. State the specific facts or estimates your conclusion relies on and
how they support or oppose R. Distinguish retrieved evidence from your own knowledge or
estimates, and preserve negations. This explanation must be visible in your final answer.
The last line must still be exactly {"p_true": 0.XX}.
```

I.8 Future-event prompts

Future-event statements use the templates below for the chains, the extractor, the verifier, the judge, and Direct ask. The other agents use the prompts of the preceding subsections.

Chain (future events)

```
You are a careful forecaster with an OPTIONAL tool: wiki_search(query, topk=3) -- Wikipedia
BM25 retrieval. Statement R asserts the outcome of a real-world event, settled by the date
it names. Estimate the probability that the asserted outcome actually occurred, reported as
JSON.

  "p_true" = P(R is true), the truthfulness score in [0.0, 1.0]. 1.0 = certain TRUE, 0.0 =
certain FALSE, 0.5 = genuine 50-50.

YOUR OWN KNOWLEDGE IS THE DEFAULT; SEARCH ONLY ADDS.
- The corpus predates the event and CANNOT contain the outcome. Your own world knowledge and
 reasoning -- who the parties are, their track record, the constraints they were under, how
such situations usually resolve -- are the PRIMARY basis for the forecast. First form your
best estimate as if you had no tool, and treat that as your working answer.
- Use wiki_search only to pin down a SPECIFIC decisive fact you are unsure of: an identity,
a rule, a record, a standing, a reference-class distribution. You may call it up to {K}
times. Move off your working estimate ONLY when a passage that is clearly ABOUT the entity
in R supplies such a fact.
- Absence changes nothing. If search returns nothing relevant, or only off-topic / same-name
 noise, KEEP your internal estimate -- that is the expected case, not evidence. "I could not
 verify it" is NEVER a reason to lower p_true; do not forecast FALSE for lack of
confirmation.
- Reserve p_true < 0.2 only for outcomes your knowledge or a decisive retrieved fact
positively RULES OUT. When you genuinely cannot separate the asserted outcome from its
alternatives, price it at its base rate in comparable cases -- for one option among N
mutually exclusive outcomes, about 1/N -- not at zero.
- WHAT R NAMES IS GIVEN. R names an event, fixture, or matchup and asks about a PROPERTY of
it ("the match between X and Y will have ...", "the winner of Z will be ..."). Take the
named event as having happened and answer only about the property. Do NOT multiply your
answer by the probability that the fixture occurs, that the teams meet, or that the event
takes place -- R presupposes it, so that probability is not part of the question and folding
 it in answers something much narrower than what was asked.
- Resolve double negatives explicitly. Be calibrated: use the full 0-1 range in BOTH
directions.

R: {stmt}

When you have searched enough, STOP calling the tool and end your reply with EXACTLY one
line:
{{"p_true": 0.XX}}

Before that final JSON line, write a brief, self-contained explanation of your judgment in
two to five sentences. State the specific facts or estimates your conclusion relies on and
how they support or oppose R. Distinguish retrieved evidence from your own knowledge or
estimates, and preserve negations. This explanation must be visible in your final answer.
The last line must still be exactly {"p_true": 0.XX}.
```

Claim extraction (future events)

```
A forecaster analyzed statement R -- a prediction about an event settled by the date R names
 -- and produced the reasoning below (its private reasoning, then its final answer). Your
job: list the EVIDENCE this reasoning actually RELIES ON to reach its conclusion about R.

R: {R}

=== PRIVATE REASONING ===
{think}

=== FINAL ANSWER ===
{answer}

WHAT COUNTS AS EVIDENCE. A prediction is built from an evidence chain, and BOTH of these are
 first-class evidence of equal standing:
- what the forecaster knows or retrieved about the world -- entities, records, rules,
standings, prior results;
- the judgments it made under acknowledged uncertainty -- a base rate, a reference class, a
trend, "such situations usually resolve this way".
The second kind is NOT hedging to be discarded. For an event no corpus can contain it is
often the load-bearing basis, and dropping it would misrepresent the chain as having no
basis at all.

Rules:
- One self-contained, checkable entry each (one entity/number/date/relation, or one stated
base rate / reference class). Rewrite it so it is understandable without the surrounding
text.
```

```
- Only include what the conclusion DEPENDS on -- skip restatements of R itself and meta-
commentary about the checking process.
- Tag each entry's provenance: "search" if grounded in a retrieved passage, "memory" if
asserted from the forecaster's own knowledge, "judgment" if it is an estimate, base rate, or
 reference-class assessment rather than a fact.
- Keep the entry's own polarity: state what the reasoning ASSERTS, even if that assertion is
 what makes it doubt R.
- Do NOT record "the event has not happened yet", "no record exists", or "this could not be
verified" as evidence. Those describe the corpus, not the world, and they are not a basis
for any conclusion about R.

Output ONLY a JSON object in this exact format (no markdown, no explanation):
{{"claims": [{{"text": "...", "src": "memory"}}, {{"text": "...", "src": "judgment"}}]}}
```

Candidate verification (future events)

```
You are settling a disputed fact inside a forecast. You have a tool: wiki_search(query, topk
=3) -- Wikipedia BM25 retrieval. You may call it up to {K} times; if generic queries do not
surface the disputed detail, search the candidate phrasings directly.

THE PREDICTION BEING ASSESSED -- read this before anything else:

    R: {R}

R is a prediction, and it RESOLVES ON THE DATE IT NAMES. Read that date off R now and hold
it: it is the moment the QUESTION below is being asked about, and it lies after the point
where both your own knowledge and the corpus end. So the QUESTION is not "what is the case
as of what I know" -- it is "what was the case as of R's date". Anything you know or
retrieve reflects an EARLIER world than the one R resolves in.

Independent analyses disagreed on the QUESTION below; the CANDIDATES are the competing
answers they put forward.

QUESTION: {q}
CANDIDATES: {cands}

SEARCH FIRST, ALWAYS. You are the verification step. The chains already wrote down what they
 believe, and your job is to TEST it against sources, not to restate it from the same
knowledge that produced it. Spend your search budget on every question before settling
anything: try the entities, the candidate phrasings, the dates. If a retrieved passage
settles the question, cite the passage in your finding.

Only when the searches come back empty, off-topic, or about a same-named different entity
may you settle from your own reliable knowledge, and then only on a DURABLE fact: a rule, a
format, an identity, a completed historical record. A finding settled that way carries the
same weight as one settled by retrieval. A finding that never tried retrieval is not
verification.

Judge each CANDIDATE, not the question. A question like "who wins the group" has no truth
value of its own, so a verdict on it cannot say which answer the evidence ruled out -- and
the analysis that argued for the losing answer is the one whose reasoning is in doubt. Give
every candidate exactly one verdict:
- "supported": a retrieved passage OR your own reliable knowledge positively establishes
THIS candidate as the answer, AT THE GRANULARITY ASKED (a coarser fact -- e.g. the year when
 the month is asked -- is NOT support).
- "refuted": a retrieved passage OR your own reliable knowledge positively establishes THIS
candidate is NOT the answer.
- "unsettled": neither route decides this candidate either way.

Also give every candidate a "truthfulness" score: the probability, in [0,1], that THIS
candidate is the correct answer as of R's date. A supported candidate is usually above 0.85
and a refuted one below 0.15. For an unsettled candidate, give your best estimate from what
the passages and your reliable knowledge do show; unsettled does not mean 0.5.

If the evidence establishes an answer that is NOT among the candidates, put it in "
established_answer"; otherwise leave that field out.

CRITICAL -- silence is "unsettled", never "refuted". A fact the corpus omits but that you
reliably know is "supported" or "refuted" on the strength of that knowledge alone. Absence
via BOTH routes is still "unsettled": assert "refuted" only when something AFFIRMATIVELY
rules a candidate out on a DURABLE fact that still holds at R's date.

CRITICAL -- APPLY THE DATE YOU READ OFF R. Before writing a status, ask: is what I am about
to report a fact about the world AT R'S DATE, or only a fact about where my information
stops? If the latter, the status is "not_found". Concretely, these describe your own horizon
 and are NOT refutations:
- "as of my knowledge the draw / announcement / match had not yet taken place";
```

```
- "no teams had been assigned / no line-up was known yet";
- "I know of no credible reports, announcements, or coverage of it".
Every one of those is TRUE BY CONSTRUCTION for anything resolving at R's date, and would
refute every prediction equally -- which is exactly why it separates nothing and must carry
no weight. If the question asks after something that post-dates the corpus, its silence is
not refutation: every candidate is "unsettled" and the question moves on.

When you have searched enough, STOP calling the tool and end your reply with EXACTLY one
line of JSON:
{{"finding": "<what the evidence does or does not establish, 1-2 sentences; name the
decisive passage detail, or say the basis is your own knowledge>", "candidate_verdicts":
[{{"candidate": "<copy the candidate verbatim>", "verdict": "supported|refuted|unsettled", "
truthfulness": 0.0}}], "established_answer": "<only if the answer is none of the candidates
>"}}
```

Chain weights (future events)

```
Independent reasoning chains each judged whether statement R is true. A verification step
then checked, with targeted retrieval, the facts the chains disagreed on. Your job: give
each chain a weight, the probability that its verdict on R is right, judged by whether the
facts its verdict rests on survived verification.

R: {R}
R predicts an outcome that had not happened when the chains reasoned. Judge each chain on
the facts, trends and plans its forecast rests on.

{chains}

VERIFICATION QUESTIONS (targeted retrieval on the disputed facts)
{questions}

How to read a claim line: "score" is the claim's truthfulness score, the probability that
the claim itself is true, on one scale for every claim. Only the claims that reached
verification are listed. "src" says where the score comes from: "verified" means targeted
retrieval settled the question the claim answers; "model estimate" means retrieval could not
 settle it and the number is a calibrated model belief, not a retrieved fact; "unsettled"
means verification could not settle it and the number is the verifier's own estimate. A
score from a model estimate or an unsettled question that is below 0.5 is NOT reliable: it
only says the evidence did not confirm the claim, not that the claim is false. Never treat
such a claim as overturned; treat it as open.

For EACH chain:
1. decisive: read its rationale and find the claims its verdict actually RESTS ON, the ones
that, if false, would overturn the verdict. List their ids. If its verdict rests on a fact
none of its listed claims states, say so with an empty list.
2. For each decisive claim, read its score and source: a verified claim scored above 0.85 is
 established, one below 0.15 is overturned, anything else is open.
3. Check the whole verification record: a settled finding from ANY question, including one
raised about another chain, can establish or overturn the fact a verdict rests on.
4. weight:
   - A decisive claim overturned, or a settled finding that rules the verdict out: 0.1 to
0.2.
   - Every decisive claim established, and together they carry the verdict: 0.85 to 0.95.
   - Decisive claims open, or the verdict rests on facts verification never reached: 0.5,
adjusted only as far as the model estimates warrant.
Do NOT judge how well the chain reasons. A chain that reasons carefully from a premise
verification overturned is wrong, and gets a low weight. A terse chain whose premises were
established gets a high one. Do not reward a chain for agreeing with the others.
A weight below 0.5 must cite, in "overturned_by", the claim id or the question whose finding
 overturns the verdict.

End your reply with EXACTLY one line of JSON:
{{"chains": [{{"label": "A", "decisive": ["A2"], "weight": 0.15, "overturned_by": "A2", "why
": "<one sentence>"}}]}}
with one entry for each of: {labels}
```

Direct ask (future events)

```
You are a careful forecaster. You have NO search tool and NO external evidence. Statement R
asserts the outcome of a real-world event, settled by the date it names. Using ONLY your own
 knowledge and reasoning, estimate the probability that the asserted outcome actually
occurred, reported as JSON.
```

```
  "p_true" = P(R is true), the truthfulness score in [0.0, 1.0]. 1.0 = certain TRUE, 0.0 =
certain FALSE, 0.5 = genuine 50-50.

HOW TO WORK:
- Your own world knowledge and reasoning -- who the parties are, their track record, the
constraints they were under, how such situations usually resolve -- are the basis for the
forecast. Form your best estimate from them.
- Not knowing the outcome is the expected case, not evidence. "I cannot verify it" is NEVER
a reason to lower p_true; do not forecast FALSE for lack of confirmation.
- Reserve p_true < 0.2 only for outcomes your knowledge positively RULES OUT. When you
genuinely cannot separate the asserted outcome from its alternatives, price it at its base
rate in comparable cases -- for one option among N mutually exclusive outcomes, about 1/N --
 not at zero.
- WHAT R NAMES IS GIVEN. R names an event, fixture, or matchup and asks about a PROPERTY of
it ("the match between X and Y will have ...", "the winner of Z will be ..."). Take the
named event as having happened and answer only about the property. Do NOT multiply your
answer by the probability that the fixture occurs, that the teams meet, or that the event
takes place -- R presupposes it, so that probability is not part of the question and folding
 it in answers something much narrower than what was asked.
- Resolve double negatives explicitly. Be calibrated: use the full 0-1 range in BOTH
directions.

R: {stmt}

End your reply with EXACTLY one line:
{{"p_true": 0.XX}}

Before that final JSON line, write a brief, self-contained explanation of your judgment in
two to five sentences. State the specific facts or estimates your conclusion relies on and
how they support or oppose R. Distinguish retrieved evidence from your own knowledge or
estimates, and preserve negations. This explanation must be visible in your final answer.
The last line must still be exactly {"p_true": 0.XX}.
```