d-itlab at SemEval-2026 Task 12: Per-Option Surprisal and Multi-Stage Gating for Precision-Oriented Causal Reasoning

Yasunori Terao and Yuuki Tachioka
DENSO IT Laboratory, Inc.
{terao.yasunori, tachioka.yuki}@core.d-itlab.co.jp

Abstract

We describe the system submitted by d-itlab to SemEval-2026 Task 12 (Abductive Event Reasoning), which requires selecting the most plausible direct cause(s) of an observed event from candidate options grounded in reference documents. Our approach combines (i) per-option multi-stage LLM inference that evaluates each option independently with progressively stricter verification, (ii) surprisal-based features obtained by teacher-forcing candidate sentences and measuring token-level negative log-likelihood, and (iii) an XGBoost ensemble trained on these heterogeneous features to produce a precision-oriented final prediction. In the official test set, our system scored 0.91, ranking third among 116 participating teams.

1 Introduction

Abductive Event Reasoning (AER) is the SemEval-2026 Task 12, aiming at real-world causal inference in English: given a short description of an observed event and a set of reference documents that may include distractors, a system must select the option label(s) that constitute the most plausible direct cause(s) (Task Organizers, 2026). Each instance provides four candidate options (A–D), each describing a potential cause; systems must predict the correct option labels. AER provides a standardized benchmark to probe LLMs' robustness on causal reasoning under incomplete and noisy evidence.

Our system adopts a staged, per-option pipeline that evaluates candidate options individually through multi-stage LLM gating and teacher-forcing surprisal, then integrates these signals via an ensemble of XGBoost classifiers for final prediction. Unlike approaches that directly select the highest-likelihood option, we treat token-level likelihood statistics as calibrated uncertainty signals and combine them with evidence-oriented gate decisions in a supervised meta-classifier.

Using this pipeline, our system scored 0.91 on the official test set, ranking third among 116 teams. Ablation experiments show that the XGBoost ensemble contributes approximately +0.02 over a single-model baseline (0.89). Our system is particularly conservative on false positives by design, which suits the official metric well; however, 56 out of 83 errors on the test set are partial-score instances caused by under-predicting on multi-option instances, as we discuss in Section 6.4.

2 Related work

Abductive Event Reasoning (AER) is related to abductive inference benchmarks such as αNLI (Bhagavatula et al., 2020) and multi-step reasoning QA datasets such as StrategyQA (Geva et al., 2021). Unlike settings that rely primarily on plausibility, AER requires document-grounded causal selection under noisy evidence.

Our work is also related to document-based and retrieval-augmented QA (Lewis et al., 2020; Kwiatkowski et al., 2019), where systems must remain robust to the distractor context. In addition, recent studies have highlighted LLM reasoning failures due to over-reliance on parametric knowledge or spurious correlations (Huang and Chang, 2023; Longpre et al., 2021), motivating conservative evidence-based verification strategies such as our multi-stage gating.

Finally, our use of teacher-forcing surprisal relates to recent work on uncertainty estimation and calibration in large language models. Previous studies have shown that model likelihoods and predicted probabilities are often miscalibrated with respect to true correctness in QA settings (Jiang et al., 2021), and have explored confidence estimation and self-evaluation mechanisms in LLMs (Kadavath et al., 2022). In this line of work, conditional log-likelihood, entropy, and probability margins are commonly used as confidence indi-

Proceedings of the The 20th International Workshop on Semantic Evaluation (2026), pages 1791–1801

cators or reranking signals. Rather than directly selecting the highest-likelihood option, we treat likelihood-derived token-level statistics as calibrated uncertainty features and combine them with evidence-oriented gate outputs via supervised meta-classification.

3 Task overview

The AER task is formulated as multiple-choice question answering for abductive causal inference, where the goal is to identify the most plausible direct cause(s) of a given real-world event from four candidate options. Each instance provides a short description of the target event, four natural-language options labeled A to D, and a set of reference documents; systems must produce the set of correct options. The organizers provide a training and development split with 1,819 and 400 instances, respectively, while the official evaluation uses a test split of 612 instances. Reference documents are news-style articles spanning diverse domains and include distractors, requiring evidence-based reasoning rather than relying on option plausibility alone.

For example, the target event is *"US Treasury Secretary Janet Yellen called financial regulators together to discuss the situation"* with four options (A) *"Many tech firms withdrew deposits from SVB following advice from venture capital firms"*, (B) *"California regulators closed SVB and placed it under the control of the US Federal Deposit Insurance Corporation (FDIC)"*, (C) *"Silicon Valley Bank's stock cratered"*, and (D) *"The Federal Reserve raised interest rates over the past year"*. Both B and C directly triggered the emergency meeting, while A is a contributing factor and D is the background; the correct options are B, C. Instances may also have a single correct option or a special *None of the others are correct causes* option.

In the development split, 52.5 % of instances have a single correct option, 35.0 % have two options, and 12.5 % have three options; 11.5 % have the "None" option. The evaluation penalizes false positives heavily. Exact match scores 1.0, a correct subset with no false positives scores 0.5, and any false positive yields 0.0; the final metric averages over instances.

4 System Overview

Figure 1 illustrates the pipeline. Since the evaluation awards a zero score for any false positive, our design prioritizes precision through three principles: (i) evaluate each option independently, (ii) derive complementary signals reflecting evidence support and model uncertainty, and (iii) learn a conservative decision boundary with supervised meta-classification and ensembling.

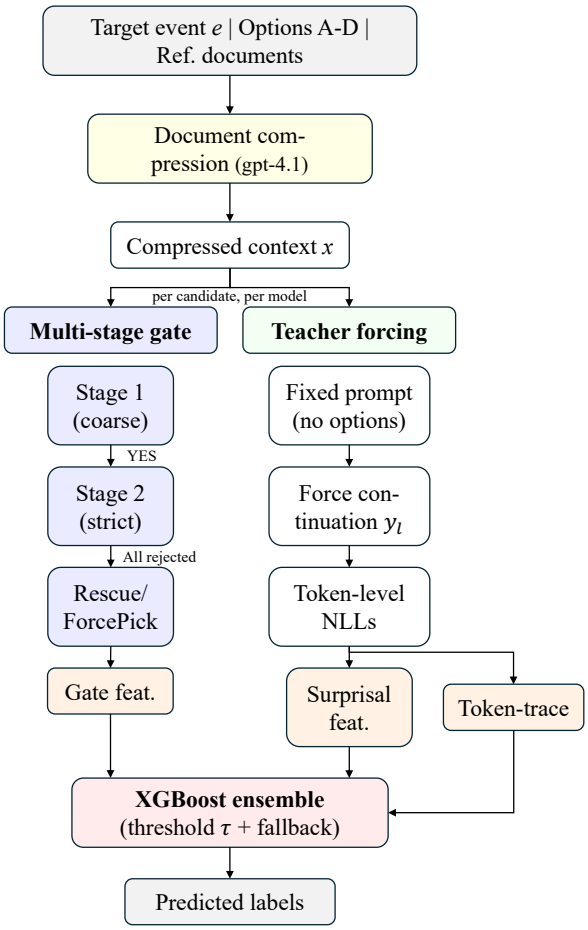

Figure 1: System overview. Reference documents are compressed, then each option is evaluated through two parallel paths—multi-stage LLM gating and teacher-forcing surprisal. The resulting features are combined in an XGBoost ensemble for final prediction.

4.1 Reference document compression

Each question is associated with reference documents via `topic_id`, averaging about 17 documents in the test split. Since raw documents can exceed 40,000 characters, we compress each document into a concise factual summary (∼500 characters) using the GPT-4.1[1], retaining salient entities, dates, numbers, and facts that may serve as causal evidence. The compressed summaries are concatenated with titles and snippets to form the context

[1]OpenAI API model `gpt-4.1`; https://platform.openai.com/docs

input for all downstream stages (see Appendix B.1 for the prompt).

4.2 Per-option multi-stage inference

For each question, we evaluate candidate options independently. We first detect the special option of *None of the others are correct causes.* and exclude it from the per-option checks. All remaining options are processed by a multi-stage gating procedure designed to suppress false positives.

The procedure has two main stages. **Stage 1** is a coarse filter that asks the LLM whether the option is a plausible direct cause under the provided documents, using a conservative instruction that prefers NO when uncertain. **Stage 2** is a strict verifier applied only to options that passed Stage 1; it answers YES only when the documents explicitly state, or unambiguously imply, a direct causal link that occurs before the target event. In addition, we implement a **limited rescue rule**: when Stage 1 yields exactly one YES but Stage 2 rejects it, we recheck that single option with a medium-strict verifier to reduce false negatives caused by overly strict gating.

At decoding time, we set the sampling temperature to zero and cap the generation length so that the model outputs only the required token, `YES` or `NO`. We also store the raw output of two stages for downstream feature construction. The full pseudocode is given in Algorithm 1 in the appendix.

4.3 Teacher-forcing surprisal

In addition to the multi-stage gate outputs, we compute surprisal-based features for each option via teacher forcing to obtain a comparable likelihood signal across options under an identical context.

For each question, we build a single prompt that contains the target event e and the compressed context x, but without any option text, where o_l is the text of the option label $l \in \{A, B, C, D\}$. We then score each option by teacher-forcing a candidate-specific continuation in the following format:

$$y_l = \text{“The direct cause of } e \text{ is } o_l\text{.”}$$

This design keeps the prompt identical across options and isolates the likelihood differences from the candidate string.

Let T_{y_l} be the token sequence of y_l. We compute the average negative log-likelihood of the continuation tokens:

$$\sigma(T_{y_l} \mid x) = -\frac{1}{|T_{y_l}|} \sum_{t=1}^{|T_{y_l}|} \log p\Big(T_{y_l}^{(t)} \mid x, T_{y_l}^{(<t)}\Big).$$

Lower $\sigma(T_{y_l} \mid x)$ indicates that the model assigns a higher probability to the candidate sentence in the same context. We treat this value as an uncertainty signal and use it as an input feature to the downstream XGBoost classifiers, rather than as a standalone decision rule. The pseudocode is given in Algorithm 2 in the appendix.

4.4 Feature construction and XGBoost meta-classification

We cast AER as an option-level binary classification (one row per option, target 1 if correct) and concatenate three groups of features per option:

Surprisal features from teacher forcing (Algorithm 2), including the average negative log-likelihood $\sigma(T_{y_l} \mid x)$, within-question normalized variants (e.g., deviation from the minimum or mean across the four options), and a percentile rank.

Gate features from multi-stage inference (Algorithm 1), including per-stage decisions, estimated probabilities when available, cross-stage probability deltas, and agreement statistics when multiple LLM runs are used.

Token-trace features derived from token-level statistics of the teacher-forcing pass, such as tail-risk indicators (high quantiles and maxima of token NLL), top-k match rates, and the gap between the forced-token log-probability and the top-1 prediction.

The complete feature list and normalization details are provided in Appendix C.4.

We train XGBoost binary classifiers (Chen and Guestrin, 2016) on these features. At inference time, the classifier outputs a probability $\hat{p}(l)$ for each option. We select all non-*none* options whose probability exceeds a threshold τ, which is tuned on the validation partition by sweeping a fixed grid and maximizing the official instance-level score. If no option passes the threshold, we output the *none* option if present, or the highest-probability non-*none* option as a fallback. We train N_{cl} XGBoost classifiers with identical feature configurations but different random seeds and group-based train/validation splits. Each model is trained independently,

and at inference time we average their predicted probabilities for each option, before a single threshold τ is applied to the averaged scores.

5 Experimental Setup

5.1 Data usage and splits

During the competition period, we did not use the official training split due to limited time. Instead, we used the official development split (400 instances) for supervised training and internal validation by creating a group-based partition by question ID. The official test split (612 instances) was reserved for the shared-task evaluation, and we additionally used it for post-hoc analysis after the official results were released. We do not use additional labeled datasets, external retrieval corpora, or manually curated knowledge bases.

5.2 Multi-stage inference configuration

We run the per-option multi-stage inference (Algorithm 1) using four LLMs. Three are locally hosted via Hugging Face Transformers[2]: Qwen2.5-32B-Instruct[3], Qwen3-32B with thinking disabled[4], and Mistral-Small-24B-Instruct[5]. The fourth is the GPT-5.2[6], accessed via the OpenAI API. For all models, we set the sampling temperature to zero and cap the maximum generation length to produce only `YES` or `NO`. The stage-specific prompt templates are provided in the Appendix B.2.

Teacher-forcing surprisal scores (Algorithm 2) are computed using only the three locally hosted models, since teacher forcing requires access to per-token log-probabilities for an arbitrary forced continuation, which we implemented via local inference. The template for the prompt is given in Appendix B.3.

5.3 XGBoost training and tuning

We train binary XGBoost classifiers using `xgboost`[7] to predict whether each candidate option is correct. Due to limited time during the competition period, we tuned only a small set of hyperparameters on the development split. The key parameters include `eta`, `max_depth`, `min_child_weight`, `subsample`, `colsample_bytree`, `num_round`, `reg_lambda`, `reg_alpha`, and early stopping settings; the final values are reported in the Appendix C.1. We use group-based splitting by question ID when creating internal partitions, so that all four options of a question remain in the same fold.

[2] https://github.com/huggingface/transformers

[3] https://huggingface.co/Qwen/Qwen2.5-32B-Instruct

[4] https://huggingface.co/Qwen/Qwen3-32B

[5] https://huggingface.co/mistralai/Mistral-Small-24B-Instruct-2501

[6] OpenAI API model `gpt-5.2`.

[7] https://xgboost.readthedocs.io/

Table 1: Top-5 teams on the official test set (best submission per team). Our team ranked third among 116 teams.

Rank	Team	Score
1	nickaraf	0.95
2	zayme	0.94
3	ytachioka, **d-itlab (ours)**	**0.91**
4	pamekalws	0.91
5	essiez	0.90

Table 2: Ablation results on the test set. Ensembling 20 XGBoost models with different train / valid split improves the score by approximately +0.02.

Configuration	Score
Single XGBoost (all features)	0.89
XGBoost ensemble (all features)	**0.91**

6 Results

6.1 Official results

Table 1 shows the official results. Our system scored 0.91, which is third among the 116 participating teams. For the best result, we trained $N_{cl} = 20$ XGBoost classifiers with different random seeds and group-based train/validation splits.

6.2 Ablation study

Table 2 compares the task score of a single XGBoost classifier result with that of the 20 classifiers' ensemble, which improves the score by approximately +0.02. To assess the robustness of our ensemble approach, we tested each XGBoost run with its own threshold τ tuned in the validation portion. Across the 20 runs, the validation score averaged 0.885 (std 0.020), and the selected thresholds concentrated around high values (mean $\tau = 0.773$), reflecting the precision-oriented nature of the task. More details on threshold stability are provided in the Appendix C.2.

6.3 Feature contribution analysis

We analyze the contribution of features of the final XGBoost classifier using SHAP values (see Appendix C.3). Gate-related features, particularly the

probability of Stage 1 and the cross-model agreement signals, dominate the predictions. Teacher-forcing token-level statistics (e.g., top-k match rates and tail NLL indicators) provide complementary uncertainty signals, while surprisal features alone are less decisive. This pattern is consistent with our precision-oriented design, where conservative gating serves as the primary decision mechanism and likelihood-derived features refine borderline cases.

6.4 Error analysis

Of 612 test instances, 529 (86.4 %) received a full score of 1.0, 56 (9.2 %) received a partial score of 0.5, and 27 (4.4 %) received a score of 0.0. We analyze the two error groups below.

Partial-score errors. Of the 56 partial errors, 40 (71 %) have two correct options but the system predicted only one, 7 (13 %) have three correct options with only one predicted, and 9 (16 %) have three correct options with two predicted. The multi-stage gate and the XGBoost threshold both suppress marginal options, improving precision but reducing recall on multi-option instances.

Zero-score errors. All 27 zero-score errors involve at least one false positive. We observe two main failure modes depending on whether the multi-stage gate produced any Stage 1 YES.

In 20 of the 27 cases (74 %), the gate rejected all options at Stage 1, so the final decision fell entirely to the XGBoost meta-classifier operating on surprisal and token-trace features. Among these, the gold option had the lowest average NLL (i.e., was ranked first by the surprisal signal) in only 5 cases; in the remaining 15, the NLL signal itself did not favor the correct option, suggesting that the language models' likelihood estimates can be unreliable when the causal link requires reasoning beyond surface-level plausibility.

In the remaining 7 cases (26 %), the gate did produce some Stage 1 YES, but the XGBoost classifier either added a false-positive option that had failed the strict Stage 2 check, or selected a different option that bypassed the gate entirely.

Error concentration and topic-level patterns. Errors are not uniformly distributed between topics. The five topics with the most errors account for 42 of the 83 total errors, and we identify two recurring patterns that explain the concentration.

The first pattern, which dominates partial-score errors, is *root-cause capture with secondary-cause omission*. Of the 56 partial errors, 32 (57 %) come from just four "causal chain" topics: the CrowdStrike outage (topic 55, 9 errors), AlphaFold development (topic 50, 9), the Taiwan earthquake (topic 51, 9), and the Papua New Guinea landslide (topic 53, 5). These topics share a common structure: a single root event triggers a cascade of downstream effects, and the gold set includes both the root cause and one or more proximate causes. Our system reliably identifies the root cause but misses secondary causes, because conservative gating and thresholding suppress options that appear less directly causal even when the task treats proximate effects as valid.

The second pattern, which dominates zero-score errors, is *the failure of causal disambiguation in complex narratives*. The Israel–Lebanon conflict topic (topic 57) accounts for 6 of the 27 zero-score errors and zero partial errors: all options describe causally plausible military events interleaved in dense reference documents, and the system consistently selects the wrong event from the same conflict. More broadly, 22 of the 27 zero-score predictions have no overlap with the gold set at all, confirming that when the system fails on these topics, the failure is categorical rather than a near-miss.

7 Conclusion

We presented a pipeline for the SemEval-2026 Task 12 that combines per-option multi-stage LLM inference, teacher-forcing surprisal features, and an XGBoost ensemble to perform precision-oriented abductive causal selection. Our system achieved a score of 0.91 on the official test set, ranking 3rd among 116 teams. The ablation results confirm that the ensemble strategy contributes to the final performance.

Our error analysis reveals two dominant failure modes: multi-option recall loss, where the system captures the root cause but misses secondary causes in causal-chain topics (56 partial-score instances), and causal disambiguation failure, where the system selects a plausible but incorrect event from a dense narrative (22 of 27 zero-score predictions have no overlap with the gold set). Future work includes incorporating the training split, exploring chain-of-thought prompting for multi-option recall, and investigating adaptive thresholding strategies.

References

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen-tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language models (mostly) know what they know. *Preprint*, arXiv:2207.05221.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*.

Shayne Longpre, Lu Periez, Anthony Chen, Nikhil Mukherjee, Chris Callison-Burch, and Mark Yatskar. 2021. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7052–7063.

Task Organizers. 2026. SemEval-2026 Task 12: Abductive Event Reasoning. In *Proceedings of the 20th International Workshop on Semantic Evaluation (SemEval-2026)*. To appear.

Algorithm 1 Per-option multi-stage inference

$\mathcal{L} \leftarrow \{A, B, C, D\}$
$\mathcal{S} \leftarrow \emptyset$
$\mathcal{C} \leftarrow \mathcal{L} \setminus \{\texttt{none\_label}\}$ ▷ exclude "none" if present
$\mathcal{Y} \leftarrow \{l \in \mathcal{C} : \mathrm{LLM}_1(e, l, x) = \texttt{YES}\}$ ▷ Stage 1
for all $l \in \mathcal{Y}$ **do** ▷ Stage 2
  if $\mathrm{LLM}_2(e, l, x) = \texttt{YES}$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{l\}$
  end if
end for
if $\mathcal{S} = \emptyset \wedge |\mathcal{Y}| = 1$ **then** ▷ Rescue
  $l \leftarrow \mathrm{only}(\mathcal{Y})$
  if $\mathrm{LLM}_{2m}(e, l, x) = \texttt{YES}$ **then**
    $\mathcal{S} \leftarrow \{l\}$
  end if
end if
if $\mathcal{S} \neq \emptyset$ **then**
  return $\mathrm{sorted}(\mathcal{S})$
else if none_label exists **then**
  return none_label
else
  return $\mathrm{ForcePick}(e, \mathcal{L}, x)$
end if

Algorithm 2 Candidate-sentence teacher-forcing scoring

Build prompt x from system instruction, target event e, and compressed context
for all $l \in \{A, B, C, D\}$ **do**
  $y_l \leftarrow$ `The direct cause of` e `is` o_l `.`
  Compute $\sigma(l \mid x)$ as average NLL of tokens in y_l
end for
return $\{\sigma(l \mid x)\}_{l \in \{A,B,C,D\}}$ and token-level traces

A Algorithm pseudocode

Algorithm 1 describes the per-option multi-stage inference pipeline (Section 4.2), and Algorithm 2 describes the teacher-forcing surprisal scoring (Section 4.3).

B Prompt templates

B.1 Compression prompt

Listing 1 shows the prompt used to compress each reference document into a concise factual summary.

```
You are an information extraction engine.
    Extract ONLY concrete, verifiable facts
    stated in the article. Do NOT solve any
    downstream task. Do NOT guess. Do NOT label
    causes or reasons; do NOT explain; just list
     facts. ASCII only. Output short self-
    contained clauses separated by '; '. No
    bullets, no quotes, no opinions.

Include any facts that could serve as causal
    evidence for later reasoning, without
    stating causality: triggering actions/events
    ; sequence/temporal markers (before/after/
    then); mechanisms (how it happened); actors
    and their roles; instruments/means (weapon/
    tool/method); locations; timings; counts/
    numbers; official actions (arrest/charge/
    announcement); and immediate consequences
    explicitly stated. Avoid generic biography/
    history unless it directly describes actions
    /events in the described incident.

Length policy:
- Target: around 500 ASCII characters.
- Soft limit: if you reach the limit while
    writing a clause, you may exceed ONLY to
    finish that SAME clause.
- After you exceed ~500, DO NOT start a new
    clause. End the output.
- Max clauses: 16.
- If nothing useful, output empty.

TITLE:
{title}

SNIPPET:
{snippet}

CONTENT:
{content}
```

Listing 1: Prompt used for reference document compression with GPT-4.1.

B.2 Multi-stage inference prompts

The multi-stage per-option pipeline (Section 4.2) uses four prompt configurations. All stages share the same user-message template shown in Listing 5; only the system prompt differs between stages.

Stage 1 (coarse filter). The system prompt (Listing 2) instructs the model to determine whether a single option is a plausible direct cause using single-step reasoning. It explicitly warns that false positives are harmful and instructs the model to answer NO when uncertain.

Stage 2 (strict verifier). Applied only to options that passed Stage 1. The system prompt (Listing 3) requires the context to explicitly state or unambiguously imply that the option directly triggers the target event. Correlation, background information, consequences, and multi-step speculation are explicitly excluded.

Rescue (medium-strict verifier). Triggered when Stage 1 yields exactly one YES but Stage 2 rejects it. The system prompt (Listing 4) relaxes the evidence criterion slightly: it accepts cases where the context *strongly implies* a direct causal link, rather than requiring an explicit statement.

ForcePick fallback. Used when no option survives the gate and the special "None" option is absent. The system prompt (Listing 6) forces the

model to output at least one label from {A,B,C,D}. If the first call still fails to produce a valid label, a retry prompt (Listing 7) with minimal wording is used once.

```
You are a careful causal reasoner.
Your job: decide whether a SINGLE candidate
    option is a plausible DIRECT cause of the
    target event,
using ONLY the provided context documents (some
    may be distractors).

Definitions:
- A "direct cause" is an event or condition that
     plausibly triggers the target event.
- The cause must happen BEFORE the target event
    and should have a clear causal link.
- Do NOT answer YES if the option is merely
    correlated, background info, a consequence,
    or unrelated.

Official guidance (apply this):
- The most plausible direct cause should be
    justified via SINGLE-STEP reasoning (no
    multi-hop speculation).
- It should be directly supported by evidence in
     the provided documents (not just plausible
    guesswork).

Scoring-aware behavior (important):
- A false positive (saying YES when it's not a
    direct cause) is very harmful.
- If you are unsure, answer NO.

Output format rules (strict):
- Output ONLY YES or NO.
- Do not output any other text.
```

Listing 2: Stage 1 system prompt (coarse filter).

```
You are an extremely strict causal verifier.
Your job: verify whether the SINGLE candidate
    option is a DIRECT cause of the target event
    ,
using ONLY the provided context documents.

Strict rules (important):
- Answer YES ONLY if the context explicitly
    states, or unambiguously implies, that the
    candidate
  directly triggers/leads to the target event (a
       clear causal link).
- The candidate must happen BEFORE the target
    event.
- Do NOT answer YES for correlation, background
    info, consequences, context, or multi-step
    speculation.
- If the evidence is incomplete, indirect, or
    uncertain, answer NO.

Official guidance (apply this strictly):
- SINGLE-STEP reasoning only: the cause must be
    directly supported by evidence in the
    documents.
- The cause must be logically and
    commonsensically sufficient to trigger the
    observed event
  (not merely enabling conditions or background).


Scoring-aware behavior:
- False positives are very harmful. If unsure,
    answer NO.

Output format rules (strict):
- Output ONLY YES or NO.
- Do not output any other text.
```

Listing 3: Stage 2 system prompt (strict verifier).

```
You are a careful causal verifier.
Your job: verify whether the SINGLE candidate
    option is a DIRECT cause of the target event
    ,
using ONLY the provided context documents.

Rules:
- Use SINGLE-STEP reasoning only (no multi-hop
    speculation).
- Answer YES if the context explicitly states OR
     strongly implies that the candidate
    directly leads to the target event.
- The candidate must happen BEFORE the target
    event.
- The candidate should be logically and
    commonsensically sufficient to plausibly
    trigger the observed event
  (not merely enabling conditions or background).

- If unsure, answer NO.

Output format rules (strict):
- Output ONLY YES or NO.
- Do not output any other text.
```

Listing 4: Stage 2 system prompt (medium-strict rescue verifier).

```
Task: Determine if the candidate option is a
    DIRECT cause of the target event.

Target event:
{target_event}

Candidate option ({candidate_label}):
{candidate_text}

Context documents (some may be distractors):
{context}

Decision procedure (follow internally):
1) Identify what happened in the target event.
2) From the context, find explicit or strongly
    implied reasons/causes for the target event.
3) Check whether the candidate option happens
    BEFORE the target event and plausibly
    triggers it.
4) If clearly supported as a direct cause,
    answer YES. Otherwise answer NO.

Now output YES or NO only:
```

Listing 5: User-message template shared by all per-option stages. Placeholders are filled per candidate.

```
You are a careful causal reasoner.

Task:
Choose the option label(s) that are plausible
    DIRECT cause(s) of the target event, using
    ONLY the provided context.

Key requirements:
- SINGLE-STEP reasoning only (no multi-hop
    speculation).
- The cause must be directly supported by
    evidence in the documents.
- The cause must be logically and
    commonsensically sufficient to trigger the
    target event.
- Prefer selecting fewer labels (false positives
     are harmful).
- You MUST output at least one label.

Output format rules (strict):
- Output ONLY a comma-separated list of labels
    from {A,B,C,D}. Examples: A | B,D | A,C,D
- Do NOT output any other text.
- Do NOT output phrases like "None of the
    provided options...".
```

Listing 6: ForcePick system prompt.

```
Output ONLY option labels.

Rules:
- Output ONLY a comma-separated list of labels
    from A,B,C,D (no spaces).
- No other text.
- Must output at least one label.
```

Listing 7: ForcePick retry system prompt (used once if the first call fails to produce valid labels).

B.3 Teacher-forcing prompt template

The teacher-forcing surprisal pipeline (Section 4.3) uses a fixed system–user prompt pair that deliberately omits all option text, so that the prompt is identical across options. The system prompt (Listing 8) instructs the model to state the direct cause of the target event in exactly one sentence. The user message (Listing 9) specifies the output format; at inference time, the model's generation is replaced by the teacher-forced candidate continuation $y_l = \texttt{The direct cause of } e \texttt{ is } o_l\texttt{.}$

```
You are a careful causal reasoner.

Task:
State the DIRECT cause of the target event using
     ONLY the provided context documents (some
    may be distractors).

Definitions:
- A "direct cause" is an event or condition that
     plausibly triggers the target event.
- The cause must happen BEFORE the target event
    and should have a clear causal link.
- Do NOT output correlation, background info,
    consequences, or unrelated facts as the
    cause.

Reasoning constraints:
- Use SINGLE-STEP reasoning only (no multi-hop
    speculation).
- The cause should be supported by evidence in
    the context.

Output format rules (strict):
- Output MUST be exactly ONE sentence.
- The sentence MUST follow the exact format
    shown in the user message.
- Output nothing else (no explanation, no quotes,
     no extra punctuation).
```

Listing 8: Teacher-forcing system prompt.

```
Task: State the direct cause of the target event
     using ONLY the context.

Target event:
{target_event}

Context documents (some may be distractors):
{context}

Output format (exactly one sentence; follow
    exactly):
The direct cause of {target_event} is <CAUSE>.
```

Listing 9: Teacher-forcing user-message template.

C Complementary materials for experiments

C.1 XGBoost hyperparameters

Table 3 reports the hyperparameter values used for the final submission.

C.2 Stability of Decision Threshold

To further analyze the calibration stability, we examine the distribution of the selected decision

Table 3: XGBoost hyperparameters used for the final submission.

Parameter	Value
eta	0.03
max_depth	4
min_child_weight	10
subsample	0.8
colsample_bytree	0.8
max_round	10000
early_stopping_round	50
reg_lambda	5.0
reg_alpha	0.0
Threshold τ	0.8

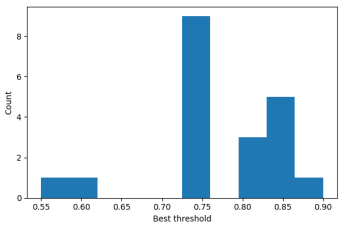

Figure 2: Histogram of best validation thresholds across 20 runs.

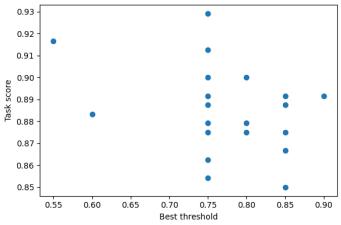

Figure 3: Scatter plot of best threshold vs. task score across 20 runs.

threshold τ between the $N_{cl} = 20$ runs with different random seeds and training / validation splits. For threshold calibration, we performed a grid search in $\tau \in \{0.05, 0.10, \ldots, 0.95\}$ and selected the value that maximized the official instance-level score.

Figure 2 shows the histogram of the best validation thresholds. The most frequent value is $\tau = 0.75$ (9 out of 20 runs), and the mean threshold is 0.773 ± 0.083.

Figure 3 plots the relationship between the selected threshold and the corresponding task score. Higher-scoring runs tend to select moderately high thresholds (around 0.75–0.85), consistent with the precision-oriented evaluation metric, where false positives are heavily penalized.

Overall, the results indicate that while the optimal threshold varies slightly depending on the data split, the decision boundary consistently remains conservative, reflecting the precision-oriented design of our system.

C.3 Feature contributions

We further analyze feature contributions of the final XGBoost meta-classifier using SHAP. Fig. 4 shows that gate-related features (e.g., Stage-1 YES probability and cross-model voting signals) dominate the decision, consistent with our precision-oriented design. Teacher-forcing token-trace features (e.g., top-k match rates, top1–top2 margin statistics, and tail NLL indicators) also contribute as complementary uncertainty signals, although with a smaller overall impact. Note that SHAP reflects the contribution to the meta-classifier under correlated features and does not imply causal importance.

Feature	Description
Surprisal features (per model)	
`surp_mean`	Mean NLL over continuation tokens
`surp_min_delta`	Deviation from within-question min
`surp_mean_delta`	Deviation from within-question mean
`surp_rank`	Within-question percentile rank
Gate features (per model)	
`stage1_yes`	Stage 1 YES/NO (binary)
`stage2_yes`	Stage 2 YES/NO (binary)
`rescue_triggered`	Rescue applied (binary)
`stage1_prob`	Estimated Stage 1 YES probability
`stage2_prob`	Estimated Stage 2 YES probability
`gate_delta`	Stage 1 prob − Stage 2 prob
Token-trace features (per model)	
`nll_p90`	90th percentile of token NLLs
`nll_max`	Max token NLL
`top1_match_rate`	Fraction of tokens matching top-1
`top5_match_rate`	Fraction matching top-5
`top1_gap_mean`	Mean gap: forced vs. top-1 log-prob
Cross-model aggregates	
`*_model_mean`	Mean of per-model values
`*_model_std`	Std of per-model values
`agreement_count`	#models with Stage 2 YES

Table 4: Features used in XGBoost meta-classification. Gate features are computed for each of the four LLMs; surprisal and token-trace features are computed for each of the three locally hosted LLMs. Cross-model aggregates are derived from the per-model values.

C.4 Full feature list

Table 4 lists all features used in the XGBoost meta-classifier, grouped by category. For each question, the features are computed per option ($l \in \{A, B, C, D\}$) and, where noted, normalized within the question.

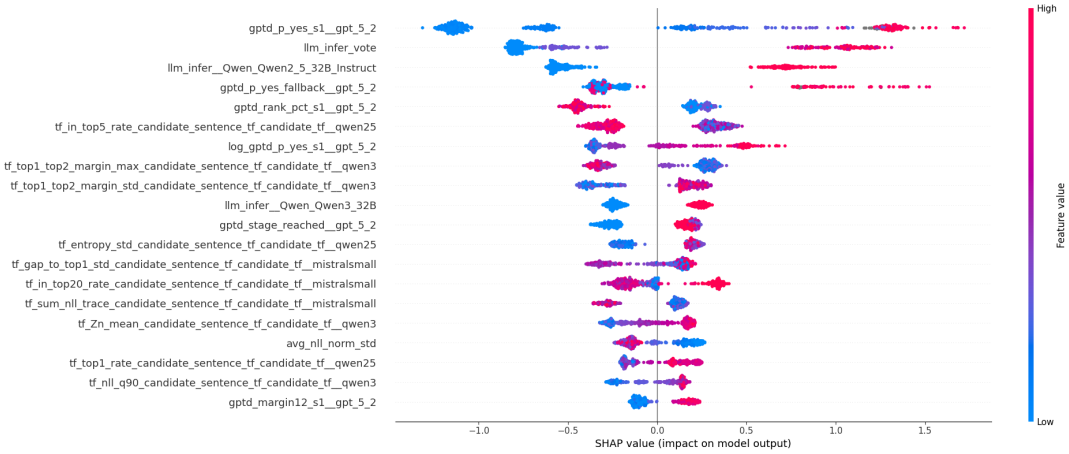

Figure 4: SHAP summary plot for the final XGBoost meta-classifier (option-level). Each point shows the SHAP value of a feature for one option; color indicates the feature value (high/low). Positive SHAP values increase the predicted probability that the option is a correct direct cause.