A Detailed Labeling Algorithm

Algorithm 2: EntityLabeling

Input: Event set $\mathcal{F} = \{(s, r, o, t)\}$; graph $\mathcal{G}$
Output: Labels $\mathcal{C}(s)$ for each $s \in \mathcal{E}$

Build $\mathrm{RelSet}[s] := \{r \mid (s, r, *, *) \text{ or } (*, r, s, *) \in \mathcal{F}\}$ for all $s \in \mathcal{E}$ ▷ Construct the relation set for entities
$\mathcal{F} \leftarrow \textsc{Apriori}([\,\mathrm{RelSet}[e]\,]_{e \in \mathcal{E}})$ ▷ Apply Apriori algorithm mine frequent relation subsets
Sort $\mathcal{F}$ and assign the TypeID for each combination of relations
foreach $s \in \mathcal{E}$ **do**
  $L \leftarrow [\,\mathrm{TypeID}[s] \mid p \in \mathcal{F},\ p \subseteq \mathrm{RelSet}[s]\,]$
  if L *is empty* **then**
    $p^\star \leftarrow \mathrm{RelSet}[s]$
    $L \leftarrow [\,\mathrm{TypeID}[p^\star]\,]$ ▷ We treat the relation set of s as L if L is empty
  $\mathcal{C}(s) \leftarrow$the first K_{type} combinations in L ▷ We only keep the Top-K_{type} combinations as the labels of s
return $\mathcal{C}(s)$ for each $s \in \mathcal{E}$

B Details of Computing Model Cost of the Rule Graph

Notations. Let $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{F})$ be the TKG. Let $\mathcal{A}$ be the finite set of category labels and $\mathcal{C} : \mathcal{E} \to \mathcal{A}$ map each entity to a label; write $A := |\mathcal{A}|$. Our rule graph is $M = (\mathcal{U}, \mathcal{E}_{\text{rule}})$ with nodes $u = \langle a_s, r, a_o \rangle \in \mathcal{A} \times \mathcal{R} \times \mathcal{A}$ and directed *chain* edges $(u \to v)$. Candidate edges are restricted to Hamming-1 neighbors:

$$\mathcal{W} = \big\{\{u, v\} : u \neq v,\ d_H(u, v) \leq 1\big\}, \quad d_H(u, v) = \mathbf{1}[a_s \neq b_s] + \mathbf{1}[r \neq r'] + \mathbf{1}[a_o \neq b_o].$$

What $L(M)$ measures. $L(M)$ counts the bits to (i) choose nodes from all atomic-rule options, (ii) choose *directed* chain edges from admissible pairs, and (iii) encode the selected nodes and edges using optimal prefix codes from empirical frequencies.

We adopt the two-part MDL form $L(M) + L(\mathcal{G} \mid M)$ and detail $L(M)$:

$$\boxed{L(M) = \underbrace{\log_2\big(A^2|\mathcal{R}|\big)}_{\text{candidate atomic rules}} + \underbrace{\log_2\big(2\,|\mathcal{W}|\big)}_{\text{candidate } \textit{directed} \text{ chain edges}} + \sum_{u \in \mathcal{U}} L(u) + \sum_{(u \to v) \in \mathcal{E}_{\text{rule}}} L(u \to v)}.$$

Node code length. From $\mathcal{F}$, estimate empirical probabilities $p_s(a)$ and $p_o(a)$ for subject/object categories $a \in \mathcal{A}$, and $p_r(r)$ for relations $r \in \mathcal{R}$. Then

$$L(u) = -\log_2 p_s(a_s) \ - \ \log_2 p_r(r) \ - \ \log_2 p_o(a_o), \quad u = \langle a_s, r, a_o \rangle.$$

Edge code length. Let $d^{\text{out}}(u)$ and $d^{\text{in}}(u)$ be the out-/in-degrees of u in $\mathcal{E}_{\text{rule}}$ and define the endpoint distribution

$$p_V(u) = \frac{d^{\text{out}}(u) + d^{\text{in}}(u)}{2\,|\mathcal{E}_{\text{rule}}|}.$$

A directed chain edge $(u \to v)$ is encoded as

$$L(u \to v) = -\log_2 p_V(u) \ - \ \log_2 p_V(v).$$

C Detailed algorithm of Computing personalization vector

Seeded personalization vector $\boldsymbol{\gamma}$. Let $\mathcal{F}_{K_1} = \{f_1, \ldots, f_{K_1}\}$ be the Top-K_1 anchors for query q (ordered by cosine similarity). We map $\mathcal{F}_{K_1}$ to rule nodes and form the seeded rule set

$$\mathcal{U}_{\text{seed}} = \{\, u \in \mathcal{U} : \ \text{supp}(u) \cap \mathcal{F}_{K_1} \neq \emptyset \,\}.$$

Corpus coverage (counts). For each $u \in \mathcal{U}_{\text{seed}}$, define the raw coverage $c_u = |\text{supp}(u)|$ and its normalization

$$\tilde{c}_u \ = \ \frac{c_u}{\sum_{v \in \mathcal{U}_{\text{seed}}} c_v}.$$

Ranking importance (geometrically discounted hits). Let $\beta \in (0,1)$ and write f_j for the j-th anchor (rank j is 1-based). Define

$$p_u \;=\; \sum_{j:\, f_j \in \mathrm{supp}(u)} \beta^{\,j-1}, \qquad \tilde{p}_u \;=\; \frac{p_u}{\sum_{v \in \mathcal{U}_{\mathrm{seed}}} p_v}\,.$$

Blending and smoothing. With mixture weight $\theta \in [0,1]$ and Dirichlet smoothing $\tau > 0$,

$$s_u \;=\; (1-\theta)\,\tilde{c}_u \;+\; \theta\,\tilde{p}_u, \qquad \gamma_u \;=\; \frac{s_u + \tau}{\sum_{v \in \mathcal{U}_{\mathrm{seed}}} (s_v + \tau)}\,, \quad u \in \mathcal{U}_{\mathrm{seed}}.$$

In our implementation we set $\theta = 0.6$, $\beta = 0.7$, and use $\tau = 1/|\mathcal{U}_{\mathrm{seed}}|$ by default. Finally, we diffuse $\boldsymbol{\gamma}$ on the rule graph by personalized PageRank:

$$\boldsymbol{\pi} = \alpha\,\boldsymbol{\gamma} + (1-\alpha)\,\boldsymbol{\pi}\,\tilde{\mathbf{A}},$$

where $\alpha = 0.2$ is the restart probability used in our experiments.

D ADDITIONAL EXPERIMENTAL RESULTS

Here, we present the comparison of token consumption and reasoning time based on *CronQuestion* and *Forecast* datasets in Fig. 5 and 6.

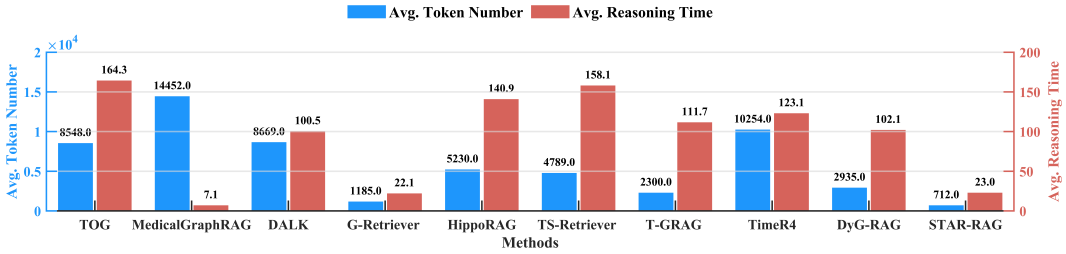

Figure 5: Comparison of token consumption and reasoning time on *CronQuestion*.

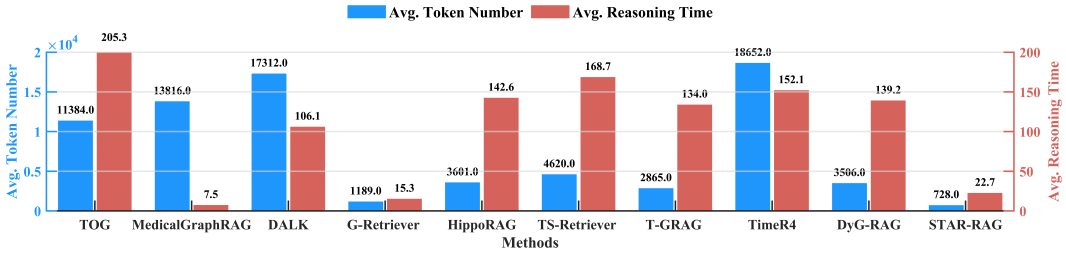

Figure 6: Comparison of token consumption and reasoning time on *Forecast*.

Then we report the rule-graph construction time and memory usage, as well as the per-query reasoning time for each dataset, in Table 5.

Table 5: Runtime and memory measurements for STAR-RAG: rule-graph construction time (s), peak memory usage (MB), and per-query reasoning time (s) on each dataset.

Measurements	CronQuestion	Forecast	MultiTQ	STAR-QA
Building Time	254.3	283.9	268.9	384.6
Building Memory	1,062	1,232	1,102	2,754
Reasoning Time	20.6	23.0	22.7	35.6

E PROMPT AND EXAMPLE FOR TEMPORAL QUESTION ANSWERING

Prompt

Instruction:
As an advanced reading comprehension assistant, your task is to analyze multiple triple facts and corresponding questions with time constraints meticulously.
Your response start after "Thought: ", where you will methodically break down the reasoning process, illustrating how you arrive at conclusions.
Keep subject/object orientation. Match the same base relation. Apply temporal operator precisely.
Conclude with "Answer: " to present return 10 short answer candidates ranked best-to-worst, devoid of additional elaborations.

One-Shot Demonstration:
Events:
Event A: On 2010-08-30, European Central Bank criticized Romania. Event B: On 2011-02-14, European Central Bank criticized government of Germany.
Question:
Before Germany, who did the European Central Bank criticize last?
Answer:
Romania.

An example to retrieve the time-aligned events

Question:
Which country was the last to accuse the UN security council before the Military Personnel of Canada did?
Retrival results:

1. `Military_personnel_(Canada) accuse Un_security_council @ 2006-02-10`
2. `Un_security_council accuse Government_sudan @ 2006-01-27`
3. `Un_security_council accuse Iran @ 2005-07-31`
4. `Lawmaker_(United_kingdom) accuse Government_(Sudan) @ 2006-01-26`
5. `Citizen_(Africa) accuse Government_(Sudan) @ 2006-02-04`
6. `Congress_(Philippines) accuse Military_personnel_(Philippines) @ 2006-01-30`
7. **`Eritrea accuse Un_security_council @ 2006-01-04`**
8. `Military_(Philippines) accuse Military_personnel_(Philippines) @ 2005-12-26`
9. `Military_personnel_(Thailand) accuse Sondhi_limthongkul @ 2005-11-28`
10. `Military_personnel_(Thailand) accuse Citizen_(Thailand) @ 2005-11-22`

Thoughts:
To determine which country was the last to accuse the UN Security Council before the Military Personnel of Canada did, we need to examine the dates provided for each entity accusing the UN Security Council and compare them to the date when the Military Personnel of Canada made their accusation, which is 2006-02-10.

The relevant accusations against the UN Security Council are:
- Eritrea on 2006-01-04
- Yemane Gebremeskel on 2005-11-24
- Foreign Affairs (Syria) on 2005-10-31
- UN Security Council itself is not relevant as it's the one being accused.

Given these dates, the last entity to accuse the UN Security Council before the Military Personnel of Canada (on 2006-02-10) was Eritrea, which did so on 2006-01-04.
Answer:
Eritrea

F MORE RELATED WORKS

In addition to STAR-RAG, we summarize several recent temporal RAG and temporal LLM plus KG methods. DRAGIN Su et al. (2024) proposes a dynamic retrieval-augmented generation framework that adapts its knowledge store and retrieval strategy to streaming information needs. TimeRAG Yang et al. (2025) targets time series forecasting by retrieving relevant historical segments and using an LLM to generate future trajectories. RAG4DyG Wu et al. (2024c) focuses on dynamic graphs and uses retrieval-augmented modeling to improve link prediction under temporal evolution. DynaGRAG Thakrar (2024) designs a dynamic GraphRAG pipeline that updates and queries evolving subgraphs aligned with query time. MusTQ Zhang et al. (2024b) introduces a multi step temporal question answering benchmark and model that require compositional reasoning over temporal facts.

F.1 INTRODUCTION OF BASELINE METHODS

We compare STAR-RAG with a set of representative GraphRAG methods on static knowledge graphs, including: TOG (Sun et al., 2024), which organizes knowledge into topic-oriented subgraphs for efficient traversal; MedicalGraphRAG (Wu et al., 2024b), which augments GraphRAG with triple-linked graphs and a coarse-to-fine retrieval strategy that combines precise matching with iterative context refinement; G-Retriever (He et al., 2024), which performs graph retrieval via a Prize-Collecting Steiner Tree, supporting scalable multi-hop reasoning; DALK (Li et al., 2024), which employs a dual-level adaptive knowledge graph to balance semantic and structural reasoning; and HippoRAG (Gutiérrez et al., 2024), which incorporates a hippocampus-inspired memory mechanism to unify short- and long-term knowledge. In addition, we include four temporal GraphRAG methods: TS-Retriever (Wu et al., 2024a), which models event dynamics and temporal dependencies; T-GRAG (Li et al., 2025), which constructs time-stamped graphs with temporal query decomposition to address conflicts and redundancy; TimeR4 Qian et al. (2024), which uses a temporal knowledge graph with a trained time-aware retriever and a multi-stage retrieve–rewrite–rerank pipeline; DyG-RAG Sun et al. (2025), which constructs a dynamic event graph and performs time-aware graph traversal with an LLM.

G ROBUSTNESS TO SPARSE/NOISY RELATIONS AND LONG-TAIL GRAPHS

In STAR-RAG, similarity-based anchor retrieval serves as the primary robustness backbone: for each query, we first perform dense similarity search over temporal facts, and the top-k anchor events

are surfaced regardless of how well they are covered by the rule graph. While some anchors may correspond to sparse or noisy relations that are weakly represented in the rule topology, the framework is explicitly designed so that the rule graph is not a single point of failure: if an anchor event is not included in any rule node or edge, STAR-RAG simply bypasses the rule graph and injects these anchors directly into the prompt template for LLM generation. In that case, STAR-RAG effectively degenerates to a standard similarity-based RAG pipeline, behaving comparably to widely used systems such as HippoRAG Gutiérrez et al. (2024), GRAG Hu et al. (2024), and REANO Fang et al. (2024). Moreover, sparse or noisy relations and tail events in long-tail graphs typically have poor connectivity to the rest of the graph, so answering queries about them often reduces to detecting a single specific fact rather than performing complex multi-hop reasoning. This regime is in fact the most favorable case for our retrieval–generation pipeline (and for static RAG baselines), as it only requires correctly retrieving the relevant anchor event and conditioning the LLM on it, instead of relying on densely connected rule structures.

H Complexity analysis.

Let $\mathcal{G}$ be a temporal knowledge graph with event set $\mathcal{F}$, candidate rule-node set $\mathcal{U}$, and candidate rule-edge set $\mathcal{W}$, and let $|\mathcal{F}|$, $|\mathcal{U}|$, and $|\mathcal{W}|$ denote their sizes. Let $C_{\max}$ be the maximum number of structural labels assigned to any entity. The offline rule-graph construction is executed once per dataset. Mapping each event $(s, r, o, t) \in \mathcal{F}$ into rule-node candidates $\langle c_s, r, c_o \rangle$ costs at most $\mathcal{O}(C_{\max}^2 \, |\mathcal{F}|)$, and generating candidate temporal edges in $\mathcal{W}$ using local time windows costs $\mathcal{O}(f_{\max}^2 \, |\mathcal{F}|)$, where $f_{\max}$ is the maximum number of events within such a window. Ranking and selecting rule nodes and edges under the MDL objective then adds $\mathcal{O}(|\mathcal{U}| \log |\mathcal{U}| + |\mathcal{W}| \log |\mathcal{W}|)$, so the total offline cost is

$$\mathcal{O}\big(C_{\max}^2 \, |\mathcal{F}| + f_{\max}^2 \, |\mathcal{F}| + |\mathcal{U}| \log |\mathcal{U}| + |\mathcal{W}| \log |\mathcal{W}|\big),$$

which is near-linear in $|\mathcal{F}|$ in our setting (small $C_{\max}$, $f_{\max}$ and $|\mathcal{U}|, |\mathcal{W}| \ll |\mathcal{F}|$). At query time, STAR-RAG first performs dense similarity search over all events and keeps the top-K_1 anchor events, which is $\mathcal{O}(|\mathcal{F}|)$ with a linear scan plus $\mathcal{O}(K_1 \log K_1)$ for selection; mapping these K_1 anchors to rule nodes is $\mathcal{O}(K_1 C_{\max})$. Personalized PageRank on the rule graph (with a sparse adjacency over $\mathcal{W}$) costs $\mathcal{O}(I \, |\mathcal{W}|)$ per query, where I is the number of iterations. We then select the top-K_2 rule nodes and gather their supported events into a candidate set $\mathcal{F}_{\text{cand}}$ with size at most $|\mathcal{F}_{\text{cand}}| \leq K_2 \, \bar{s}$, where $\bar{s}$ is the average support size of a selected rule node; re-ranking these candidates against the query costs $\mathcal{O}(|\mathcal{F}_{\text{cand}}|)$. Overall, the per-query complexity can be written as

$$\mathcal{O}\big(|\mathcal{F}| + I \, |\mathcal{W}| + K_1 C_{\max} + K_2 \bar{s}\big),$$

and since K_1, K_2, $C_{\max}$, and $\bar{s}$ are small constants in our experiments, the dominant terms in practice are a single pass over $\mathcal{F}$ for similarity search and a sparse PPR over the much smaller rule graph, while the MDL-based construction (including ranking and selection) is amortized as a one-time offline cost.

I Details for building STAR-QA

To evaluate the performance of STAR-RAG on large-scale KGs, we additionally reconstruct the evaluation data by ourselves from ICEWS 2005–2015, 2018, 2021 and augment it with 1,000 LLM-generated QA pairs that are explicitly designed to be more human-written, out-of-template, and to include adversarial temporal confounds. Following the question-generation process of FAITH Jia et al. (2024), we prompt the LLM (Claude) with event facts to synthesize new questions, but we design our own dataset-specific prompt (shown as follows) to explicitly encourage more human-written phrasings and adversarial temporal confounds such as distractor years and plausible but incorrect time periods.

Prompt

Instruction:

You are constructing a temporal QA dataset from a *series of events*. For each question, you are given:

- a unique question ID `quid` (integer);
- a list of time-stamped events, where each event has a subject entity, an object entity, a relation description, and an event date in the form `YYYY-MM-DD`;
- a correct answer entity, associated with one specific *target event* in this list (exactly one event justifies the answer).

Your task is to write *one* natural, conversational English question whose only correct answer is exactly the given answer entity, and whose meaning depends on the target event and its time. Other events in the list may be used only as contextual or adversarial distractors.

Style and temporal requirements:

- Rewrite names of the form `"X (Country)"` into natural phrases, e.g., "Foreign Affairs (Iran)" → "the Foreign Affairs ministry in Iran", "Police (Russia)" → "the police in Russia", "Citizen (Australia)" → "citizens of Australia".
- Rewrite dates like `"2021-01-01"` in natural English, e.g., "January 1st, 2021".
- Use conversational contractions where reasonable (e.g., "Who's" for "Who is", "gonna" for "going to"), and you may optionally add light discourse markers such as "So," or "Well," at the beginning to sound natural.
- Add adversarial temporal confounds: you may mention other events from the list or introduce extra, plausible but incorrect time expressions (e.g., another year, a broader time range, or vague phrases like "around that time") so that a model that only matches surface dates or counts mentions is misled. The true date of the target event and the correct answer must remain unchanged, and the question must still be answerable only by reasoning about the real target event and its time.

Do not invent facts that contradict any of the input events. There must be exactly one correct answer.

Output format:

Return your output as a single JSON object on one line, with no additional text: `{"quid": <QUID>, "question": "<YOUR_QUESTION>", "answers": ["<ANSWER_ENTITY>"], "qlabel": "Single"}`.

J FUTURE WORK

In this section, we also highlight a natural extension of our design as a direction for future work. At present, STAR-RAG uses a globally fixed parameter θ in the personalization vector (Appendix C), which implicitly balances temporal information (through L_{time}) and structural or coverage information (through L_{cov}) during rule-graph propagation. A more flexible variant would equip STAR-RAG with a query-adaptive controller that adjusts a query-dependent weight $\theta(q)$ according to the time sensitivity and structural complexity of the input question. This idea is consistent with recent work on time-aware retrieval-augmented generation such as TimeR^4, where the system explicitly rewrites questions to reveal temporal constraints before retrieval Qian et al. (2024), and with adaptive expert routing in retrieval-augmented Mixture-of-Experts language models such as Expert-RAG Zhou et al. (2024). Concretely, one could estimate a time-sensitivity score for each query by using simple rule-based heuristics (for example, detecting words like "before", "after", or explicit years), or by using an

optional LLM-based classifier, and then use this score to emphasize temporally aligned propagation for strongly temporal questions while giving more weight to structural and semantic coverage for more static questions. However, this design naturally brings a trade-off between computational cost and accuracy: richer controllers based on LLM scoring or expert routing can provide finer-grained adaptation but introduce additional token and latency overhead, whereas simple heuristic controllers are efficient but less expressive. In the present work we favor simplicity and efficiency by using a global θ, and we leave the design and evaluation of such query-adaptive controllers, together with a systematic study of their trade-offs, as an important avenue for future research.

Another limitation of STAR-RAG is that the rule graph is constructed offline from a static temporal KG and then kept fixed, without efficient support for incremental updates when new events arrive. An important direction for future work is to develop an incremental maintenance mechanism for the rule graph, inspired by dynamic PPR-based systems such as Instant Zheng et al. (2022) and IDOL Zhu et al. (2024). In these methods, representations are updated under graph changes by locally refreshing Personalized PageRank or embeddings in the affected region instead of recomputing the entire model. Analogously, for STAR-RAG, new temporal events could update only the supports and temporal statistics of the impacted rule nodes and a small neighborhood of rule edges whose coverage or time patterns change significantly, while preserving the rest of the rule graph. This could be coupled with topology-monitorable triggers in the spirit of Instant and IDOL that decide when accumulated local changes warrant structural adjustments such as splitting or merging rule nodes or adding and pruning edges. Developing such an incremental rule-graph maintenance scheme would make STAR-RAG more suitable for frequently updated temporal KGs, while retaining its interpretable, rule-level temporal structure.