

Appendix

Contents

929	A Supporting Subgraph Extraction Algorithm	22
930	B Details on Unlearning Methods	22
931	C Prompt Template	24
932	C.1 Prompt Template for LLM Judge Rating	24
933	C.2 Prompt Template for Query Target LLM	27
934	D Detailed Information of Experiments	28
935	D.1 Compute Configurations	28
936	D.2 Experimental Details	28
937	E Additional Examples for LLM Judge	29
938	E.1 Examples of LLM Judge Adhering to Instructions	29
939	E.2 Examples Comparing LLM and Human Judgments	30
940	E.3 Case Studies of Supporting Subgraph Inferability	31
941	F Additional Experimental Results	32
942	F.1 In-Depth Analysis of Local Consistency	32
943	F.2 Evaluating Shifts in LLM-Inferred Scores Pre- and Post-Unlearning	34
944	F.3 Additional Results on the Impact of Confidence Scores on Unlearning Effectiveness	35
945	F.4 Additional Results on Performance Across Unlearning Epochs	36
946	G Mapping Between Entropy Threshold u^* and Yes Token Probability	36
947	H Additional Details and Illustrations for YAGO3-10	37
948	I Limitations and Broader impacts	38

A Supporting Subgraph Extraction Algorithm

In this section, we detail the algorithm used to construct the supporting subgraph described in Sec. 4.2. As noted in Sec. 5, we constrain the subgraph to a maximum path length $l = 3$. The full extraction procedure is provided in Alg. 1.

Algorithm 1 Subgraph Extraction

Input: Reference Knowledge graph \mathcal{G}_{ref} , target triple $e = (s, r, o)$, hop k , entropy threshold u^*
Output: Subgraph G_e containing valid paths from s to o
Process:
Initialize $\hat{G}_e = \emptyset, \mathcal{N}_s^* = \emptyset, \mathcal{N}_e^* = \emptyset, \mathcal{N}_o^* = \emptyset$
Phase 1: Find high-confidence neighbors of s
for $v \in k\text{-hop neighbors of } s \text{ in } \mathcal{G}_{\text{ref}} \setminus \{s, o\}$ **do**
 for $(s, r', v, u) \in \text{LLM-query}(s, v)$ where $u \leq u^*$ and $\text{yes} = \arg \max\{\text{yes}, \text{no}, \text{unknown}\}$ **do**
 $\mathcal{N}_s^* = \mathcal{N}_s^* \cup \{v\}, \hat{G}_e = \hat{G}_e \cup \{(s, r', v, u)\}$
 end for
end for
Phase 2: Expand from high-confidence neighbors
for $v \in \mathcal{N}_s^*$ **do**
 for $w \in k\text{-hop neighbors of } v \setminus \{\{s, o, v\} \cup \mathcal{N}_s^*\}$ **do**
 for $(v, r', w, u) \in \text{LLM-query}(v, w)$ where $u \leq u^*$ and $\text{yes} = \arg \max\{\text{yes}, \text{no}, \text{unknown}\}$ **do**
 $\mathcal{N}_e^* = \mathcal{N}_e^* \cup \{v\}, \hat{G}_e = \hat{G}_e \cup \{(v, r', w, u)\}$
 end for
 end for
end for
Phase 3: Verify connections to o
 $\mathcal{N}_{\text{check}} = \mathcal{N}_s^* \cup \mathcal{N}_e^* \cup \{s\} \setminus \{o\}$
for $v \in \mathcal{N}_{\text{check}}$ **do**
 for $(v, r', o, u) \in \text{LLM-query}(v, o)$ where $u \leq u^*$ and $\text{yes} = \arg \max\{\text{yes}, \text{no}, \text{unknown}\}$ **do**
 $\mathcal{N}_o^* = \mathcal{N}_o^* \cup \{v\}, \hat{G}_e = \hat{G}_e \cup \{(v, r', o, u)\}$
 end for
end for
Phase 4: Prune to retain only paths connecting s to o
 $G_e = \emptyset$
for each $(h, r', t, u) \in \hat{G}_e$ **do**
 if $t \in \mathcal{N}_o^*$ **or** $t = o$ **then**
 $\mathcal{N}_o^* = \mathcal{N}_o^* \cup \{h\}, G_e = G_e \cup \{(h, r', t, u)\}$
 end if
end for
return G_e

B Details on Unlearning Methods

In this section, we provide an overview of all unlearning methods considered in this paper along with their implementation details. These methods directly modify LLM parameters to unlearn knowledge and are representative of approaches commonly adopted in the existing literature [14, 27, 28, 57]. Recall that D_{forget} denotes the set of target triples to be unlearned. For methods that additionally use a retain set to preserve general model functionality, we denote this set as D_{retain} . We now briefly describe each method and its specific hyperparameters as follows:

- **Gradient Ascent (GA)** [46–48]: GA seeks to remove the influence of D_{forget} by reversing its gradient updates. However, this popular approach may induce significant utility degradation [14, 57, 58].
- **Random Labels (RL)** [46, 49]: RL disrupts memorization by replacing the labels in D_{forget} with random tokens during next-token prediction.
- **Negative Preference Optimization (NPO)** [50]: NPO encourages the model to assign lower likelihood to the forget set compared to its original state, while controlling deviation from the

966 pretrained model. It frames unlearning as a preference reversal task and adopts a log-ratio-based
 967 objective derived from direct preference optimization. The objective is defined as

$$\mathcal{L}_{\text{NPO}}(\theta) = -\frac{2}{\beta_{\text{NPO}}} \mathbb{E}_{x \sim D_{\text{forget}}} \left[\log \sigma \left(-\beta_{\text{NPO}} \log \frac{\mathcal{M}(x)}{\mathcal{M}_{\text{pretrain}}(x)} \right) \right], \quad (1)$$

968 where σ denotes the sigmoid function, and parameter β_{NPO} controls the sensitivity to preference
 969 changes. A smaller β_{NPO} leads to stronger alignment with the original model. Following previous
 970 literature [28, 50], we set $\beta_{\text{NPO}} = 0.1$.

971 • **NegGrad+** [51]: NegGrad+ jointly applies gradient ascent on D_{forget} and gradient descent on
 972 D_{retain} , optimizing

$$\beta_{\text{NegGrad+}} \cdot \mathbb{E}_{x \sim D_{\text{retain}}} [\mathcal{L}(\mathcal{M}; x)] - (1 - \beta_{\text{NegGrad+}}) \cdot \mathbb{E}_{x \sim D_{\text{forget}}} [\mathcal{L}(\mathcal{M}; x)]. \quad (2)$$

973 By simultaneously “reviewing” loss \mathcal{L} over D_{forget} and D_{retain} , NegGrad+ mitigates the utility
 974 degradation typically caused by pure gradient ascent. In the experiments, we set $\beta_{\text{NegGrad+}} = 0.999$.

975 • **SCRUB** [51]: Scalable Remembering and Unlearning unBound (SCRUB) employs a student-
 976 teacher architecture to guide model updates via the following objective:

$$\begin{aligned} & \mathbb{E}_{x \sim D_{\text{retain}}} [\alpha_{\text{SCRUB}} \cdot \mathcal{D}_{\text{KL}}(\mathcal{M}_{\text{pretrain}}(x) \parallel \mathcal{M}(x)) + \beta_{\text{SCRUB}} \cdot \mathcal{L}(\mathcal{M}; x)] \\ & - \mathbb{E}_{x \sim D_{\text{forget}}} [\gamma_{\text{SCRUB}} \cdot \mathcal{D}_{\text{KL}}(\mathcal{M}_{\text{pretrain}}(x) \parallel \mathcal{M}(x))], \end{aligned} \quad (3)$$

977 where \mathcal{D}_{KL} is the divergence and parameters $(\alpha_{\text{SCRUB}}, \beta_{\text{SCRUB}}, \gamma_{\text{SCRUB}})$ control the trade-off be-
 978 tween forgetting effectiveness and utility preservation. We set $(\alpha_{\text{SCRUB}}, \beta_{\text{SCRUB}}, \gamma_{\text{SCRUB}}) =$
 979 $(0.999, 1, 0.99)$ (*unlearn with QA*) and $(\alpha_{\text{SCRUB}}, \beta_{\text{SCRUB}}, \gamma_{\text{SCRUB}}) = (1e-4, 1, 1e-4)$ (*unlearn*
 980 *with sentence*).

981 **Computation Budget.** To ensure fair comparisons across unlearning methods, we adopt a unified
 982 computational budget protocol inspired by [14]. Methods are categorized into two groups: (i) those
 983 using only the forget set (D_{forget}) (e.g., GA, NPO, RL), and (ii) those requiring both forget and retain
 984 sets (e.g., NegGrad+, SCRUB). Since unlearning involves trade-offs among effectiveness, utility, and
 985 efficiency [59–61], we standardize training epochs across methods. Specifically, methods using only
 986 D_{forget} are allowed up to 10 unlearning epochs, while those that additionally access D_{retain} (matched
 987 in size to D_{forget}) are limited to 5 epochs due to the increased computational cost per step. This
 988 setup ensures comparable computational complexity across methods, enabling a fair evaluation of
 989 unlearning effectiveness.

990 **Unlearning Epoch Selection.** As discussed above, all unlearning methods are allocated the same
 991 computation budgets. In general settings (Sec. 6.1), we discussed our epoch selection criterion: if the
 992 model’s utility (Loc) exceeds 0.8 at any point, we select the last epoch that satisfies this threshold;
 993 otherwise, we select the epoch with the highest utility below it. The selected epoch for each method
 994 under each setting is reported in Tab.2.

Unlearning Methods	LLaMA3-8B-Instruct				Qwen2.5-7B-Instruct			
	Sentence-based		QA-based		Sentence-based		QA-based	
	Full	LoRA	Full	LoRA	Full	LoRA	Full	LoRA
GA	9	10	1	1	1	6	1	1
RL	5	10	1	2	2	10	1	1
NPO	2	10	1	3	6	10	1	1
NegGrad+	6	10	2	4	10	6	2	2
SCRUB	10	10	2	10	4	8	2	4

Table 2: Epochs comparison between unlearning methods across model types, unlearn formats, and unlearn settings.

995 **Unlearning Hyperparameter Configurations.** For full-parameter unlearning, we use a learning
 996 rate of 1×10^{-6} ; for parameter-efficient unlearning with LoRA, a higher learning rate of 1×10^{-5} is
 997 applied. In both settings, training is conducted with a batch size of 2 and gradient accumulation steps
 998 of 2 across 2 GPUs, effectively simulating a batch size of 8.

LLM Judge Inference Prompt

You are a reasoning assistant that evaluates whether a **target triple** can be logically inferred from a given set of **subgraph triples**.

Task:

Using only the subgraph triples below, determine whether the target triple can be inferred. Do not use any external knowledge. All provided triples must be considered as factual ground truth. Only rely on logical reasoning from the facts.

Each edge is annotated with entropy $\in [0, 1]$, which quantifies the uncertainty of that triple. Lower entropy means higher confidence in its validity.

Entropy Interpretation:

- 0.00–0.25 → Very Confident
- 0.25–0.50 → Confident
- 0.50–0.75 → Less Confident
- 0.75–1.00 → Not Confident

Scoring Rules:

Assign a confidence score from **0 to 5** for the target triple:

- 0 → No logical path exists; inference is impossible from the given triples.
- 1 → Very weak support; entities appear but no relevant path.
- 2 → Some weak logical connection, but with high uncertainty or missing relations.
- 3 → Moderate path exists with reasonable certainty.
- 4 → Confident support; well-formed path with generally low uncertainty.
- 5 → Strong support; direct match or very strong multi-hop support with high confidence.

Evaluation Guidelines

Firstly, you need to classify the direct path and the support path:

1. **Direct Path:** A single triple that directly connects the head entity to the tail entity with a relation that is identical or logically similar (ex. `isLocatedIn` and `hasCapital`) to the target relation. The entropy of this triple determines the confidence level of the direct path. If target triple exists in the subgraph, it must be regarded as a direct path.
2. **Support Path:** A multi-hop directed path that connects the head entity to the tail entity through intermediate entities. This path should logically imply the target triple through reasoning. The entropy of all triples in this path collectively determines the confidence level of the support path.

Secondly, when you are evaluating a target triple, please:

1. Identify all direct paths between the head and tail entities
2. Identify all support paths between the head and tail entities
3. Record the number of each path type
4. Assess the entropy (uncertainty) level of each path
5. Assign a score based on the criteria below

Criteria: Scores of 3+ indicate the target triple can be reasonably inferred; scores below 3 indicate insufficient evidence.

1. Score 5

- Low entropy direct path, OR
- Moderately low entropy direct path + at least one support path, OR
- Higher entropy direct path + multiple support paths (more paths needed as entropy increases), OR
- No direct path but multiple low entropy 2-3 hop paths

2. Score 4

- Moderately low entropy direct path without support paths, OR
- Relatively high entropy direct path + 1-2 support paths, OR

- High entropy direct path + numerous support paths, OR
- No direct path but moderately low entropy 2-3 hop paths

3. **Score 3**

- High entropy direct path, OR
- 1-3 high entropy support paths

4. **Score 2**

- No direct path
- Multiple high entropy support paths, none completely correct

5. **Score 1**

- No direct path
- Very few support paths with errors, OR
- Only 1-2 high entropy support paths with errors

6. **Score 0**

- No direct path
- Irrelevant or unrelated paths
- No logical connection to target triple

For each evaluation, please provide:

1. Analyze direct paths (It is very important to confirm whether a direct path exists, because an incorrect judgment will lead to a significant difference in the assigned score).
2. Analyze support paths
3. Your reasoning for the assigned score
4. The final score (0-5)

Examples:

Example for Score 5:

Subgraph Triples:

(Rome, capital_of, Italy) with entropy 0.08

(Italy, located_in, Europe) with entropy 0.12

Target Triple:

(Rome, located_in, Europe)

Reasoning:

(Rome, capital_of, Italy) and (Italy, located_in, Europe) form a clear inference path

Both triples have very low entropy, indicating high confidence

The relation "located_in" is logically implied by the combination of "capital_of" and "located_in"

This creates a strong transitive relationship between Rome and Europe

Final Confidence Score: 5

Example for Score 4:

Subgraph Triples:

(Apple, produces, iPhone) with entropy 0.35

(iPhone, runs_on, iOS) with entropy 0.15

(Apple, develops, iOS) with entropy 0.20

Target Triple:

(Apple, manufactures, iPhone)

Reasoning:

(Apple, produces, iPhone) is a direct path with moderately high entropy

"produces" and "manufactures" are very similar relations

The support path (Apple, develops, iOS) and (iPhone, runs_on, iOS) indirectly reinforces the relationship

The combination of a direct path and supporting evidence compensates for the moderate entropy.

Final Confidence Score: 4

Example for Score 3:*Subgraph Triples:*

(Einstein, worked_at, Princeton_University) with entropy 0.55
(Princeton_University, located_in, New_Jersey) with entropy 0.30
(New_Jersey, part_of, USA) with entropy 0.25

Target Triple:

(Einstein, lived_in, USA)

Reasoning:

No direct path exists between Einstein and USA

One support path exists: (Einstein, worked_at, Princeton_University) → (Princeton_University, located_in, New_Jersey) → (New_Jersey, part_of, USA)

The first triple has high entropy (0.55), creating uncertainty

The logical connection is sound (working somewhere typically implies living there)

The complete path allows reasonable inference but with moderate uncertainty due to the high entropy in the first connection

Final Confidence Score: 3

Example for Score 2:*Subgraph Triples:*

(Tiger, belongs_to, Felidae) with entropy 0.45
(Lion, belongs_to, Felidae) with entropy 0.40
(Felidae, is_carnivorous, True) with entropy 0.25

Target Triple:

(Tiger, hunts, Lion)

Reasoning:

No direct path between Tiger and Lion

Support paths only establish that both animals belong to the same family

Being in the same carnivorous family might suggest interaction but doesn't support hunting specifically

The paths have high entropy and none directly supports the target relation.

Final Confidence Score: 2

Example for Score 1:*Subgraph Triples:*

(Sun, larger_than, Earth) with entropy 0.60
(Earth, has_satellite, Moon) with entropy 0.30

Target Triple:

(Sun, orbited_by, Moon)

Reasoning:

No direct path between Sun and Moon

Only one weak support path through Earth with high entropy

The path contains a factual error - while the Moon orbits Earth, it doesn't directly orbit the Sun

The relationship is misleading for the inference task.

Final Confidence Score: 1

Example for Score 0:*Subgraph Triples:*

(Water, contains, Hydrogen) with entropy 0.25
(Tree, produces, Oxygen) with entropy 0.40
(Fire, consumes, Oxygen) with entropy 0.35

Target Triple:

(Water, extinguishes, Fire)

Reasoning:

No direct path between Water and Fire

No logical support paths connecting the entities

The existing triples discuss chemical composition but are unrelated to the fire extinguishing property

The facts, while individually correct, have no relevance to the inference task.

Final Confidence Score: 0

1004

Given Subgraph Triples:

{ facts_str }

Target Triple:

(A, relation, B)

Your Reasoning:

Explain step-by-step how the given facts lead to (or fail to lead to) the target triple.

Then output the score on a new line like:

Final Confidence Score: <integer between 0 and 5>1005 **C.2 Prompt Template for Query Target LLM****Target LLM Query Template****System Message**

You are an expert in knowledge graphs. Your task is to determine whether a given relation between two entities is correct, incorrect, or unknown.

First analyze the semantic properties of both entities, and then reason about whether the relation mentioned in the task is appropriate for these two entities.

Here is an example of a correct relation:

Example 1: For head entity 'Shakespeare', tail entity 'Hamlet', relation 'wrote', reasoning process: Shakespeare is a person, specifically an author, while Hamlet is a literary work. 'wrote' is one of the most specific relations between an author and their work, so the relation 'wrote' is correct.

Here is an example of an incorrect relation:

Example 2: For head entity 'Shakespeare', tail entity 'Hamlet', relation 'locatedIn', reasoning process: Shakespeare is a person and Hamlet is a literary work. The relation 'locatedIn' typically describes spatial or geographic relationships, which does not apply to an author and their work. Therefore, the relation 'locatedIn' is incorrect.

Here is an example of an unknown relation:

Example 3: For head entity 'Hamlet', tail entity 'Existentialism', relation 'influencedBy', reasoning process: Hamlet is a literary work, while Existentialism is a philosophical movement. Although some scholars interpret Hamlet's introspective nature as proto-existentialist, there is no widely agreed-upon or factual relationship confirming that Hamlet was directly influenced by Existentialism. Therefore, the relation 'influencedBy' is unknown.

Be deliberate and analytical in your reasoning before providing your final answer. Your answer (which is provided) should be taken as-is; the goal is to compute its log probability given the context. According to the user's task, you should provide your final answer in the format 'Answer: Yes' or 'Answer: No' or 'Answer: Unknown'.

User Templates**Qwen/Qwen2.5-7B-Instruct:**

Task: In the triple ({entity1}, ?, {entity2}), does the relation

'{relation}' correctly complete it? Answer: Yes/No/Unknown

meta-llama/Llama-3.1-8B-Instruct:

Task: Given that the head entity is '{entity1}' and the tail entity is '{entity2}', is the relationship '{relation}'? Answer: Yes/No/Unknown

1006

D Detailed Infomation of Experiments

D.1 Compute Configurations

All experiments were conducted on a hardware platform equipped with NVIDIA A100 80GB PCIe GPUs, which were used for both training and inference.

D.2 Experimental Details

For the model unlearning phase, we employed DeepSpeed ZeRO-2 optimization for efficient distributed training across multiple devices. For model evaluation, we leveraged vLLM [62] to facilitate efficient parallel inference, processing prompts with a substantial batch size of 500 to maximize throughput.

Refinement of Entropy-Based Filtering for Instruction-Following Failures. We observe that certain unlearning methods may impair the model’s ability to follow instructions, i.e., reliably producing one of the expected outputs: Yes, No, or Unknown. This degradation can result in degenerate or uninformative outputs. To address this issue, we refine the entropy-based filtering criterion by requiring that the Yes token be the most probable among the top-5 predicted tokens and that the corresponding normalized entropy falls below a predefined threshold. This modification allows us to detect instruction-following failures while preserving agreement with the original entropy-based measure when the model behaves as intended.

Additionally, we emphasize that we experimented with several alternative designs for knowledge probing. First, we tested an open-ended prompting strategy in which the target LLM is given the subject and object entities from a triple and asked to select all reasonable relations between them from a provided list. However, we observed that this approach performed poorly on our constructed validation set (Sec. 5), frequently leading to hallucinated relations. Second, we evaluated the effect of introducing an explicit Unknown option when querying the LLM. We found that including this option helped mitigate hallucination and improved the model’s accuracy on the validation set by encouraging more conservative predictions.

Model Calibration. In our experiments, we probe the target LLM to evaluate its own knowledge with associated confidence scores. We observe that directly asking the target LLM to generate this confidence score results in unreliable results without calibration. The confidence score should be reliable. For example, predictions made with 0.8 confidence should be correct approximately 80% of the time. Specifically, we discover that LLaMA3-8B-Instruct inherently generates underconfident predictions. Its generated confidence scores are much lower than the corresponding binning accuracies, as shown in Fig. 10, with an ECE of 0.475. On the other hand, Qwen2.5-7B-Instruct inherently generates more reliable confidence scores, with an ECE of 0.083, leading to more trustworthy predictions. Temperature scaling has been widely used to calibrate model predictions [41, 63–65]. We show that applying temperature scaling reduces the ECE of LLaMA3-8B-Instruct and Qwen2.5-7B-Instruct to 0.031 and 0.067 respectively.

Runtime Analysis. We report the runtime of both the unlearning and evaluation stages for full-model unlearning on Qwen2.5-7B-Instruct using the sentence-based format, including the unlearning procedure and the construction of supporting subgraphs across all methods. Recall that D_{forget} contains 200 target triples to be unlearned; the unlearning process is conducted over 10 epochs (5 epochs for NegGrad+ and SCRUB). The subgraph construction phase also operates over the same set of target triples. On average, both stages require approximately 1 to 1.5 hours to complete, as demonstrated in Fig. 11. This corresponds to an average of roughly 30 seconds per triple for unlearning and 20 seconds for subgraph

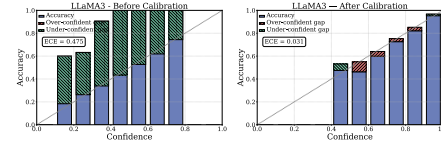


Figure 10: Target LLM LLaMA-3 Before (Left) and After (Right) Calibration.

we discover that LLaMA3-8B-Instruct inherently generates underconfident predictions. Its generated confidence scores are much lower than the corresponding binning accuracies, as shown in Fig. 10, with an ECE of 0.475. On the other hand, Qwen2.5-7B-Instruct inherently generates more reliable confidence scores, with an ECE of 0.083, leading to more trustworthy predictions. Temperature scaling has been widely used to calibrate model predictions [41, 63–65]. We show that applying temperature scaling reduces the ECE of LLaMA3-8B-Instruct and Qwen2.5-7B-Instruct to 0.031 and 0.067 respectively.

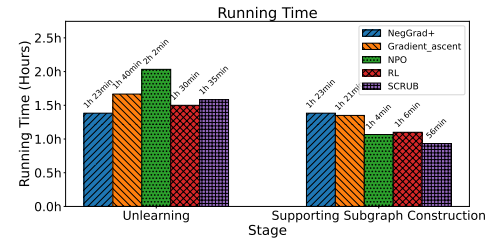


Figure 11: Runtime for full-model unlearning using the sentence-based format on Qwen2.5-7B-Instruct.

1060 construction. Other settings yield comparable or lower runtime overhead. For supporting subgraph
 1061 evaluation with the LLM judge, recall that we use GPT-o4-mini. On average, rating each supporting
 1062 subgraph takes approximately 15 seconds (In practice, the rating can be further accelerated by making
 1063 parallel API calls).

1064 **Random Seed Selection.** In all our experiments, we followed the common practice and fixed our
 1065 random seed to be 42.

1066 **Note on Knowledge Probing.** It is important to acknowledge that the knowledge probing process
 1067 may occasionally yield incomplete or imprecise factual outputs. In our framework, we adopt multiple-
 1068 choice questions instead of open-ended formats to reduce hallucination and improve consistency, and
 1069 apply temperature scaling to enhance the reliability of confidence estimates. Nevertheless, probing
 1070 results may still deviate from ground truth in some cases. During inference, we treat all extracted facts
 1071 from the supporting subgraph as assumed ground truth knowledge in the target LLM, and consider
 1072 any inference patterns identified therein as targets for unlearning.

1073 E Additional Examples for LLM Judge

1074 E.1 Examples of LLM Judge Adhering to Instructions

1075 As discussed in Sec. 6.3, for all 50 tested subgraphs, the LLM judge strictly follows the instruction
 1076 to reason solely based on the provided subgraph triples when assessing the inferability of the target
 1077 triple. Notably, its reasoning does not incorporate any external or latent internal knowledge beyond
 1078 the subgraph. Below, we present several representative examples to illustrate this behavior in Tab. 3.

Example 1: (San_Martín_Province, isLocatedIn, San_Martín)

Related Triple(s) in Supporting Subgraph:

- (San_Martín_Province, isLocatedIn, San_Martín)
- (San_Martín_Province, isLocatedIn, San_Martín_Region)
- (San_Martín_Region, isLocatedIn, San_Martín)

Reasoning: [...] exact triple in the subgraph: (San_Martín_Province, isLocatedIn, San_Martín) [...] there is also a 2-hop path: - (San_Martín_Province, isLocatedIn, San_Martín_Region) with entropy = 0.075 (Very Confident) - (San_Martín_Region, isLocatedIn, San_Martín) with entropy = 0.364 (Confident) This yields an additional, reasonably low-entropy route [...]

Example 2: (Voldemārs_Plade, isAffiliatedTo, Rīgas_FK)

Related Triple(s) in Supporting Subgraph:

- (Voldemārs_Plade, isAffiliatedTo, Rīgas_FK)
- (Voldemārs_Plade, isKnownFor, Rīgas_FK)

LLM Reasoning: [...] The target triple (Voldemārs_Plade, isAffiliatedTo, Rīgas_FK) is directly present in the subgraph with an entropy of 0.862. [...] Other related triples include (Voldemārs_Plade, isKnownFor, Rīgas_FK), which has a lower entropy of 0.385 and suggests a confident link in terms of recognition or fame [...]

Example 3: (Emad_Hamdy, wasBornIn, Sohag)

Related Triple(s) in Supporting Subgraph:

- (Emad_Hamdy, wasBornIn, Sohag_Governorate)
- (Sohag_Governorate, isLocatedIn, Sohag)

LLM Reasoning: [...] The only “wasBornIn” triple for Emad_Hamdy is (Emad_Hamdy, wasBornIn, Sohag_Governorate) with entropy 0.849. [...] We look for a multi-hop path from Emad_Hamdy to Sohag that could imply “wasBornIn.” The plausible 2-hop path is: a. (Emad_Hamdy, wasBornIn, Sohag_Governorate) entropy = 0.849; b. (Sohag_Governorate, isLocatedIn, Sohag) entropy = 0.044. Reasoning for path: If Emad_Hamdy was born in Sohag_Governorate, and Sohag_Governorate is located in the city Sohag, then it follows that Emad_Hamdy was born in Sohag. [...]

Example 4: (Ptolemy_VIII_Physcon, isMarriedTo, Cleopatra_II_of_Egypt)

Related Triple(s) in Supporting Subgraph:

– (Ptolemy_VIII_Physcon, isMarriedTo, Cleopatra_II_of_Egypt)

LLM Reasoning: [...] We need to find a direct match or a logical path that supports this relationship using the subgraph triples provided. The subgraph contains the triple (Ptolemy_VIII_Physcon, isMarriedTo, Cleopatra_II_of_Egypt) with an entropy of 0.181. This directly matches the target triple. [...]

Table 3: Examples of LLM reasoning grounded in the extracted supporting subgraph. Blue indicates the utilized knowledge triple from the subgraph. We verified no external knowledge is used.

1079 E.2 Examples Comparing LLM and Human Judgments

1080 As discussed in Sec. 6.3, our comparison between LLM and human ratings reveals a strong overall
1081 correlation, with the majority of samples receiving similar scores. Nonetheless, there are some
1082 discrepancies in certain cases. Specifically, human ratings tend to be higher when the LLM overlooks
1083 multi-hop inference pathways that human evaluators are able to identify. Conversely, in instances
1084 where the LLM assigns higher scores, the supporting subgraph often contains multiple relation types,
1085 making it more likely for human annotators to overlook critical patterns relevant to the inference.
1086 To further illustrate these observations, we present representative examples in Tab. 4. Moreover,
1087 based on the LLM’s reasoning over these examples, we also observe that it does follow the prompt
1088 instructions on carefully considering both direct and multi-hop inference patterns, while taking the
1089 associated entropy scores into account.

Example 1: (Benjamin_Butler_(politician), isAffiliatedTo, Greenback_Party)

LLM: 1. Direct Path Analysis – We look for a triple linking Benjamin_Butler_(politician) directly to Greenback_Party with relation “isAffiliatedTo.” There is no exact “isAffiliatedTo” triple. However, we do have: (Benjamin_Butler_(politician), participatedIn, Greenback_Party) with entropy 0.145. “participatedIn” is not identical to “isAffiliatedTo,” but in the context of political parties it is a very close, logically similar relation: participating in a party generally implies affiliation. Entropy 0.145 falls in the “very confident” range (0.00–0.25). 2. Support Path Analysis – A possible multi-hop path: (Benjamin_Butler_(politician), isPoliticianOf, Portsmouth_NH) [0.311] → (Portsmouth_NH, created, Grassroots_Campaigns_Inc.) [0.644] → (Grassroots_Campaigns_Inc., isAffiliatedTo, Greenback_Party) [0.632]. This path is weak: it infers Butler’s personal affiliation from the municipality’s creation of an organization that itself is affiliated, which is a tenuous link. Entropies on that path are moderate to high, further reducing confidence. 3. Reasoning for Score – We have one very low-entropy direct path (0.145) via “participatedIn,” a relation closely aligned with “isAffiliatedTo.” No equally strong support paths are needed when a direct path is already very confident. According to the scoring rubric, a “low entropy direct path” alone is sufficient for a score of 5. **Final Confidence Score: 5**

Human: In the supporting subgraph, there exists a direct correlated target triple that is closely aligned with and supports the target fact. Furthermore, there are also indirect paths that can partially support the inference. Therefore, the final confidence score is 5.

Example 2: (Mooresville_North_Carolina, isLocatedIn, Iredell_County_North_Carolina) - LLM fails to capture certain inferential pathways.

LLM: 1. Analysis of Direct Paths — We look for any triple of the form (Mooresville_North_Carolina, isLocatedIn, Iredell_County_North_Carolina) in the subgraph. **No such direct triple exists.** 2. Analysis of Support Paths — We look for multi-hop paths that would imply Mooresville is located in Iredell County. The **only immediate location fact** about Mooresville is: (Mooresville_North_Carolina, isLocatedIn, Mecklenburg_County_North_Carolina) — entropy 0.465 (Less Confident). To bridge from Mecklenburg County to Iredell County, one might try: (Mecklenburg_County_North_Carolina, hasCapital, Concord_North_Carolina) — entropy 0.268 (Very Confident), and (Concord_North_Carolina, isLocatedIn, Iredell_County_North_Carolina) — entropy 0.205 (Very Confident). This yields the chain: Mooresville —isLocatedIn→ Mecklenburg County —hasCapital→ Concord —isLocatedIn→ Iredell County. However, *hasCapital* does not imply that every part of Mecklenburg County is in Iredell County, nor that Mecklenburg County itself is located in Iredell County. It simply states that Concord is the capital of Mecklenburg County and happens to be located in Iredell County. This is a semantic mismatch and **does not logically support** that Mooresville, which is in Mecklenburg County, lies in Iredell County. **No other viable paths exist** (all other *isLocatedIn* links from Mecklenburg County point to other cities, or the directionality is reversed). 3. Reasoning for Score — There is no direct (Mooresville_North_Carolina, isLocatedIn, Iredell_County_North_Carolina) triple. **Only one multi-hop chain exists**, but it **misuses *hasCapital*** and therefore cannot validly establish the target relation. Although entropies along that chain are low, the relation semantics are incorrect for location inference. This constitutes very weak, semantically flawed support. **Final Confidence Score: 1**

Human: The subgraph contains several indirect paths (each requiring three hops to infer the target triple), and all paths exhibit medium-to-low entropy, indicating that the target LLM has only moderate confidence in this knowledge. For example, one representative reasoning chain is: (Mooresville_North_Carolina, isLocatedIn, Mecklenburg_County_North_Carolina) (entropy = 0.465) → (Mecklenburg_County_North_Carolina, isLocatedIn, Pineville_North_Carolina) (entropy = 0.358) → (Pineville_North_Carolina, isLocatedIn, Iredell_County_North_Carolina) (entropy = 0.106), which forms a plausible 3-hop inference path supporting the target triple. Therefore, the **final confidence score is 4.**

Example 3: (Gmina_Cybinka, isLocatedIn, Słubice_County) - Human missed certain supportive triples

LLM: 1. Direct Path Analysis — There is an **exact matching triple**: (Gmina_Cybinka, isLocatedIn, Słubice_County) with entropy 0.011. This **directly corresponds** to the target relation “isLocatedIn” and has **very low entropy** (0.00–0.25), indicating a highly confident assertion. 2. Support Path Analysis — **Not required** given the strength of the direct match. (For completeness: there exist location or adjacency triples involving Gmina_Cybinka and Słubice_County, but they do not add meaningful certainty beyond the direct fact.) 3. Reasoning — A direct, identical relation exists in the subgraph with extremely low entropy. According to the scoring rubric, such highly confident direct evidence **warrants the highest possible score.** **Final Confidence Score: 5**

Human: Since **no direct or multi-hop relational evidence** supporting the inference is found within the supporting subgraph, **the final confidence score is given 0.** **[Reason:** The primary reason for the low human-assigned scores lies in the substantial number of triples present in the subgraph (47 triples in this case), many of which involve diverse relation types (e.g., isLocatedIn, isAffiliatedTo, hasNeighbor, hasCapital, isConnectedTo, owns) without forming clear or effective supporting inference patterns. This information overload substantially increases the cognitive complexity of the evaluation task, making human annotators more susceptible to being misled or overlooking crucial triples, ultimately resulting in incorrect or inconsistent judgments.]

Table 4: Examples comparing LLM and human ratings. **Blue** highlights content most relevant to the inference judgment. **Green** indicates agreement between LLM and human (i.e., both deem the target triple inferable), while **Red** denotes disagreement (i.e., the LLM and human assign different ratings regarding inferability).

1090 E.3 Case Studies of Supporting Subgraph Inferability

1091 Certain triples appear successfully unlearned at the instance level, yet remain inferable when consider-
1092 ing their supporting subgraphs. Several representative examples in Tab. 5 highlight this phenomenon.

Example 1: (Alexander_Morten, playsFor, Wanderers_F.C.)

The triple (Alexander_Morten, playsFor, Wanderers_F.C.) (entropy: 0.437 before, 1.328 after) seems unlearned individually, but the evaluator LLM can still infer it through the related fact (Alexander_Morten, workat, Wanderers_F.C.) (entropy: 0.512 before, 0.644 after) present in the supporting subgraph, as the relationships playsFor and workat are semantically similar in the context of professional athletes.

Example 2: (Robyn_Miller, workat, Cyan_Worlds)

The triple (Robyn_Miller, workat, Cyan_Worlds) (entropy: 0.386 before, 1.201 after) demonstrates successful instance-level unlearning, yet remains inferable through the related fact (Robyn_Miller, edited, Cyan_Worlds) (entropy: 0.582 before, 0.626 after) in the supporting subgraph, as editorial contributions strongly suggest a working relationship with the company.

Example 3: (Oxford_Properties, owns, Brookfield_Place_(Toronto))

The triple (Oxford_Properties, owns, Brookfield_Place_(Toronto)) (entropy: 0.411 before, 1.205 after) appears unlearned when assessed individually, but can be inferred through a chain of ownership relationships in the supporting subgraph: (Oxford_Properties, owns, Brookfield_Office_Properties) (entropy: 0.349 before, 0.726 after) and (Brookfield_Office_Properties, owns, Brookfield_Place_(Toronto)) (entropy: 0.132 before, 0.361 after), establishing a corporate ownership hierarchy that reveals the target relationship.

Example 4: (Bridgewater_Township_New_Jersey, isLocatedIn, Somerset_County_New_Jersey)

The triple (Bridgewater_Township_New_Jersey, isLocatedIn, Somerset_County_New_Jersey) (entropy: 0.591 before, 1.477 after) shows instance-level unlearning success, but remains inferable through geographic proximity facts in the supporting subgraph: (Bridgewater_Township_New_Jersey, hasNeighbor, Bernardsville_New_Jersey) (entropy: 0.202 before, 0.389 after) and (Bernardsville_New_Jersey, isLocatedIn, Somerset_County_New_Jersey) (entropy: 0.455 before, 0.757 after), which together likely imply that Bridgewater Township is also located in Somerset County.

Table 5: Examples where unlearned triples remain inferable via supporting subgraphs.

1093 These examples illustrate a critical challenge in knowledge unlearning: even when direct knowledge
1094 of a specific triple is removed, the model may retain inferential paths through related information that
1095 remains in its knowledge base.

1096 F Additional Experimental Results

1097 F.1 In-Depth Analysis of Local Consistency

1098 Recall the definition of the utility metric Local Consistency (Loc) (Sec. 6.1), which quantifies the
1099 consistency of an LLM’s predictions on multiple-choice questions (Yes, No, and Unknown) regarding
1100 neighboring knowledge triples before and after unlearning. In this section, we conduct a more fine-
1101 grained analysis of how model predictions shift under the Loc metric, capturing any transitions the
1102 LLM may exhibit among the three predefined choices. Additionally, we observe that the unlearning
1103 process can sometimes impair the model’s ability to follow instructions, particularly under the
1104 QA-based format. To account for such cases, we introduce an additional Other category to denote
1105 predictions that fall outside the standard multiple-choice options. In Fig. 12 13 14 15 16 17 18 19, we
1106 present the corresponding confusion matrices under various settings to illustrate these dynamics. Our
1107 findings reveal several key observations. First, in the majority of cases, the LLM retains its utility on
1108 local knowledge triples. Second, we find that each unlearning method introduces changes in a distinct
1109 manner, i.e., transitions may occur from one valid category (e.g., Yes) to another (e.g., Unknown),
1110 reflecting shifts in the model’s belief. The patterns of these shifts can be different across different
1111 methods. Finally, in scenarios where the original utility metric Loc score is low (e.g., below 0.5 in
1112 Tab. 1), we observe that the model occasionally fails to adhere to the multiple-choice instruction

1113 format altogether. This result in predictions that fall into the Other category, especially for QA-based
 1114 unlearning.

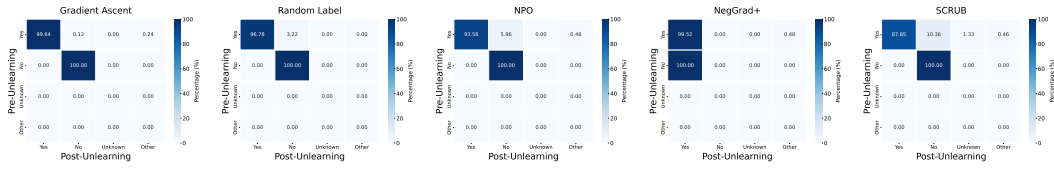


Figure 12: Confusion Matrix for Loc: LLaMA-QA-LoRA

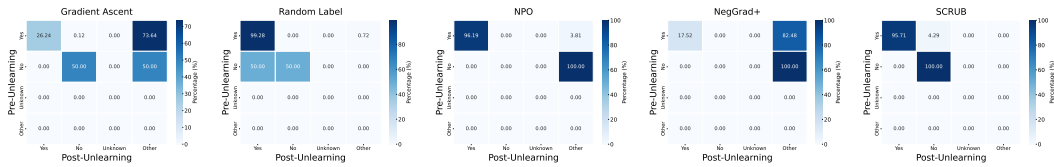


Figure 13: Confusion Matrix for Loc: LLaMA-QA-Full

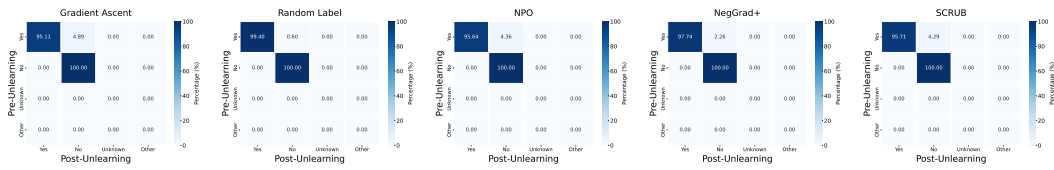


Figure 14: Confusion Matrix for Loc: LLaMA-Sentence-LoRA

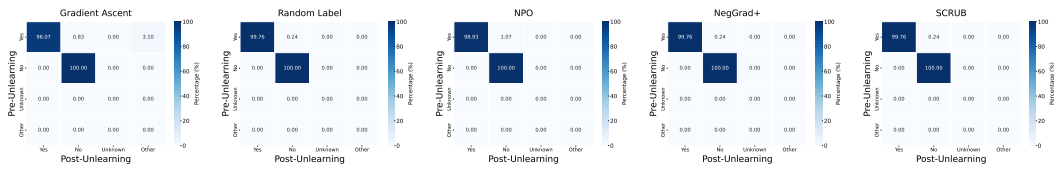


Figure 15: Confusion Matrix for Loc: LLaMA-Sentence-Full

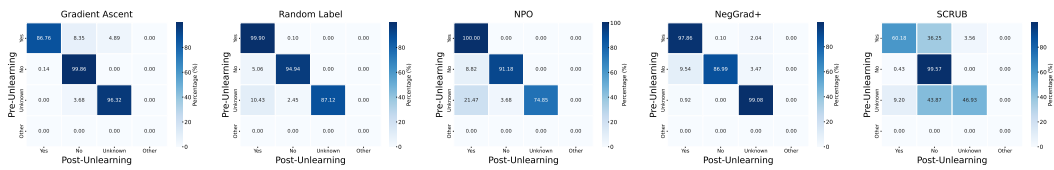


Figure 16: Confusion Matrix for Loc: Qwen-QA-LoRA

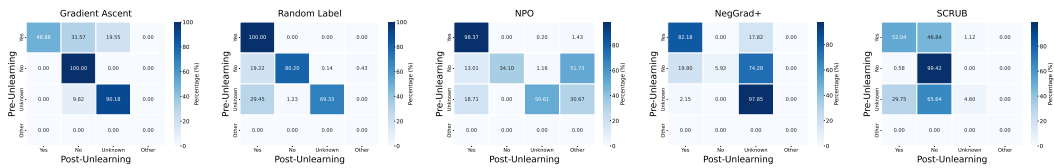


Figure 17: Confusion Matrix for Loc: Qwen-QA-Full

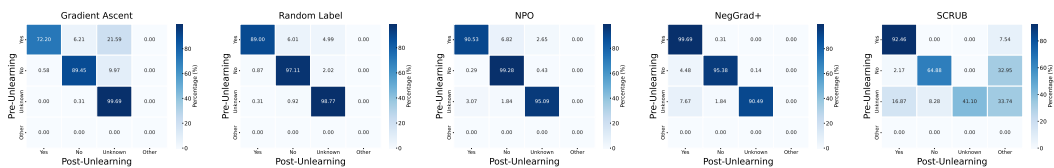


Figure 18: Confusion Matrix for Loc: Qwen-Sentence-LoRA

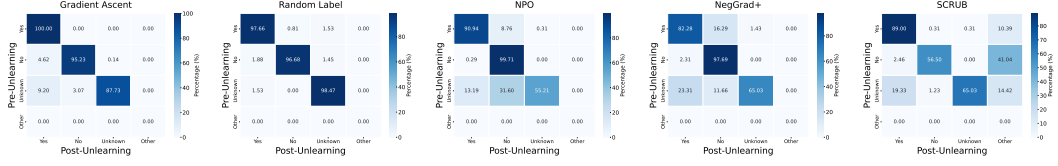


Figure 19: Confusion Matrix for Loc: Qwen-Sentence-Full

1115 E.2 Evaluating Shifts in LLM-Inferred Scores Pre- and Post-Unlearning

1116 In this section, we further analyze the distribution changes of discrete LLM ratings (ranging from 0
 1117 to 5) under our supporting subgraph-based unlearning framework across different settings, before and
 1118 after each unlearning method is applied. As shown in Fig. 20 21 22 23 24 25 26 27, we observe that
 1119 the QA-based format, which directly targets the question-answer pairs used for knowledge probing,
 1120 yields significantly higher unlearning effectiveness compared to the sentence-based format (albeit at
 1121 the cost of reduced utility, as demonstrated in Tab. 1).

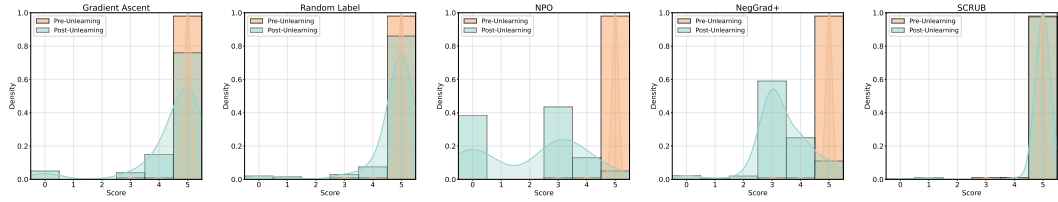


Figure 20: Shift in LLM Judge Scores Pre- and Post-Unlearning: LLaMA-QA-LoRA

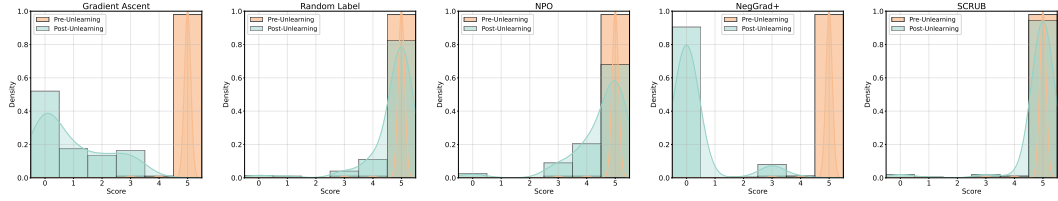


Figure 21: Shift in LLM Judge Scores Pre- and Post-Unlearning: LLaMA-QA-Full

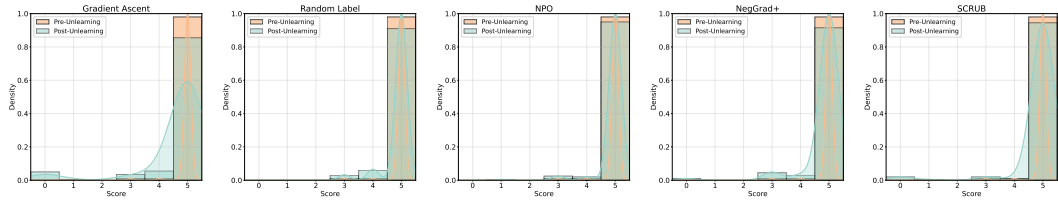


Figure 22: Shift in LLM Judge Scores Pre- and Post-Unlearning: LLaMA-Sentence-LoRA

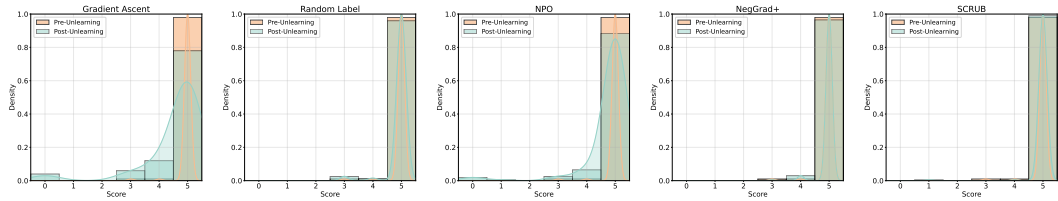


Figure 23: Shift in LLM Judge Scores Pre- and Post-Unlearning: LLaMA-Sentence-Full

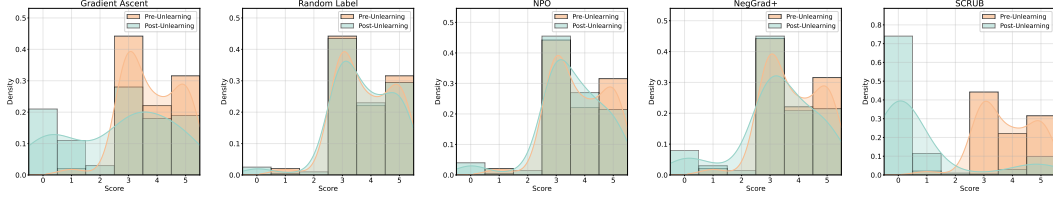


Figure 24: Shift in LLM Judge Scores Pre- and Post-Unlearning: Qwen-QA-LoRA

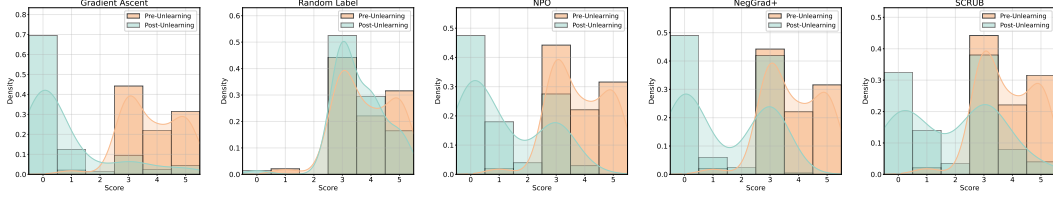


Figure 25: Shift in LLM Judge Scores Pre- and Post-Unlearning: Qwen-QA-Full

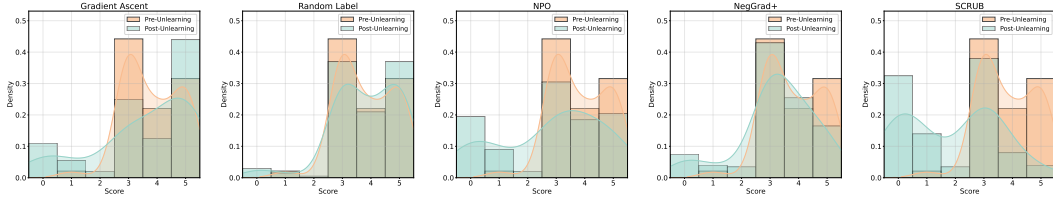


Figure 26: Shift in LLM Judge Scores Pre- and Post-Unlearning: Qwen-Sentence-LoRA

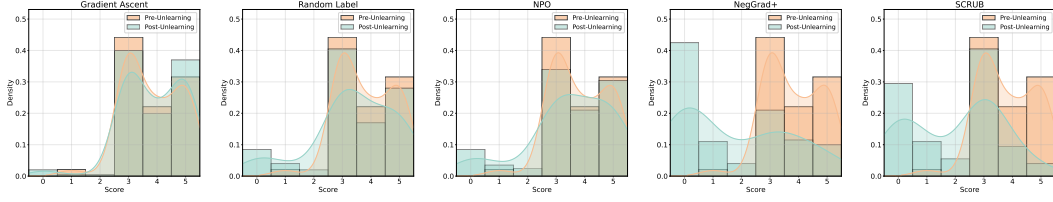


Figure 27: Shift in LLM Judge Scores Pre- and Post-Unlearning: Qwen-Sentence-Full

1122 F.3 Additional Results on the Impact of Confidence Scores on Unlearning Effectiveness

1123 In this section, we present additional results on how confidence scores within the supporting subgraph
 1124 affect unlearning outcomes across different settings. As shown in Fig. 28 29, gradually decreasing
 1125 the entropy threshold u^* , i.e., retaining only higher-confidence knowledge, systematically filters out
 1126 weaker supporting inferences. This leads to a significant increase in unlearning effectiveness. These
 1127 results highlight that low-confidence knowledge triples play a crucial role in evaluating unlearning,
 1128 and omitting them can substantially overestimate its effectiveness.

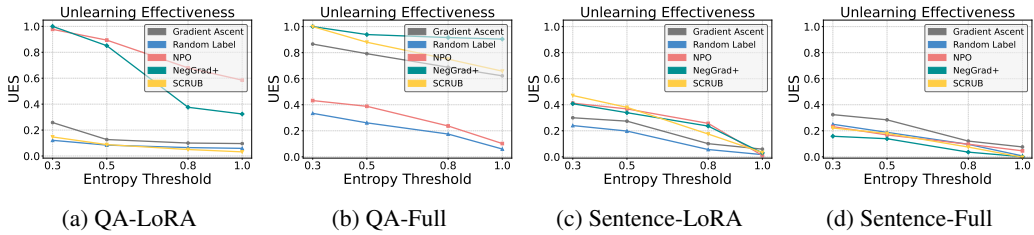


Figure 28: Impact of Confidence Scores on Unlearning Effectiveness: LLaMA Model

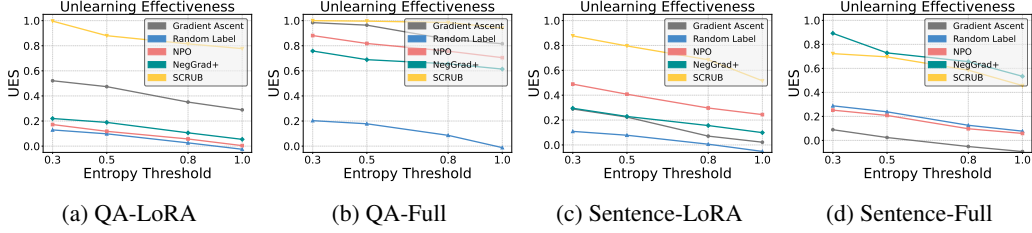


Figure 29: Impact of Confidence Scores on Unlearning Effectiveness: Qwen Model

1129 F.4 Additional Results on Performance Across Unlearning Epochs

1130 In Fig. 30, we present additional results analyzing the impact of unlearning epochs under the LLaMA3
 1131 full-model unlearning setting, comparing both sentence-based and QA-based formats. Across all
 1132 configurations, the unlearning effectiveness measured by our supporting subgraph evaluation remains
 1133 consistently lower than that of traditional instance-level evaluations. Notably, in the QA-based
 1134 format, unlearning directly targets the question-answer pairs used for probing factual knowledge. As
 1135 unlearning progresses, this often leads to severe degradation of model utility, resulting in unlearning
 1136 effectiveness approaching 1, indicating that both the target triples and their associated subgraphs
 1137 have been extensively corrupted. In contrast, under the sentence-based format (Left), the unlearning
 1138 process tends to be more conservative in its impact on model utility in most cases, causing less
 1139 degradation compared to the QA-based setting (Right). Consequently, we observe that unlearning
 1140 effectiveness, particularly under our evaluation framework, remains low across most epochs. These
 1141 findings highlight the limitations of current unlearning methods and underscore the need for more
 1142 robust approaches capable of effectively removing knowledge without compromising general model
 1143 performance.

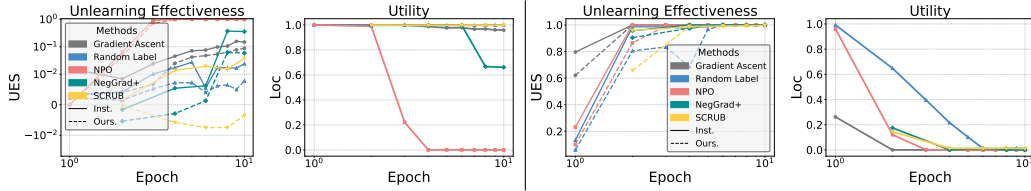


Figure 30: Impact of unlearning epochs on LLaMA3 full-model unlearning. **Left:** Unlearning effectiveness (measured by both instance-level and our proposed evaluation) and corresponding utility (measured by locality) under sentence-based format. **Right:** Same analysis for QA format.

1144 G Mapping Between Entropy Threshold u^* and Yes Token Probability

1145 In this section, we clarify how the entropy-based filtering criterion used for constructing supporting
 1146 subgraphs relates to the model’s predicted probability for the Yes token in a multiple-choice question
 1147 format (Yes, No, Unknown). Specifically, we consider a prediction to reflect retained knowledge if
 1148 the model’s output satisfies two conditions: (i) the Yes token is the argmax among the three options,
 1149 and (ii) the entropy computed over the distribution of these three options is below a threshold u^*
 1150 (assuming the model adheres to instructions). To make this relationship more interpretable, Tab. 6
 1151 presents the corresponding range of Yes token probabilities for different values of entropy under the
 1152 argmax constraint. This mapping provides an intuitive understanding of how the entropy threshold u^*
 1153 translates to the model’s confidence in answering Yes. To derive these intervals, we fix the entropy
 1154 value u^* and solve for the range of valid Yes probabilities p under the constraint that Yes is the
 1155 argmax. The upper bound occurs when the remaining two options (No and Unknown) share equal
 1156 probability, i.e., $u^* = H(p, \frac{1-p}{2}, \frac{1-p}{2})$ with $p \geq \frac{1-p}{2}$, where H denote entropy. The lower bound
 1157 corresponds to the case where one of the non-Yes options takes probability $1-p$ and the other is
 1158 zero, i.e., $u^* = H(p, 1-p, 0)$ with $p \geq 1-p$.

Entropy	0.10	0.15	0.20	0.25	0.30	0.35
Yes Prob. Range	[0.987, 0.989]	[0.978, 0.982]	[0.969, 0.974]	[0.958, 0.966]	[0.947, 0.957]	[0.934, 0.947]
Entropy	0.40	0.45	0.50	0.55	0.60	0.65
Yes Prob. Range	[0.921, 0.937]	[0.906, 0.927]	[0.890, 0.916]	[0.873, 0.905]	[0.854, 0.892]	[0.833, 0.880]
Entropy	0.70	0.75	0.80	0.85	0.90	0.95
Yes Prob. Range	[0.811, 0.867]	[0.785, 0.853]	[0.757, 0.838]	[0.724, 0.823]	[0.684, 0.807]	[0.631, 0.791]
Entropy	1.00					
Yes Prob. Range	[0.500, 0.773]					

Table 6: Feasible Yes token probability ranges corresponding to entropy thresholds u^* . Ranges are computed under the assumption that Yes is the most likely token.

1159 H Additional Details and Illustrations for YAGO3-10

1160 **Complete List of Relations in YAGO3-10:** actedIn, wasBornIn, hasGender, hasAcademicAdvisor,
1161 hasChild, hasCitizenship, hasDeathPlace, hasEmployer, hasGivenName, hasInstrument, hasLanguage,
1162 hasLegalResidence, hasMember, hasName, hasNationality, hasOccupation, hasOfficialLanguage,
1163 hasPlaceOfBirth, hasPlaceOfDeath, hasSpouse, hasSurname, hasTitle, holdsPoliticalPosition, isAffil-
1164 iatedTo, isConnectedTo, isKnownFor, isLocatedIn, isMarriedTo, isPoliticianOf, livesIn, playsFor,
1165 produced, studiedAt, wasBornOnDate, wasCreatedOnDate, wasDestroyedOnDate, worksAt.

1166 Using the complete set of relations from YAGO3-10, we present illustrative examples of potential
1167 (non-)deterministic inference patterns in the knowledge graph, as summarized in Tab. 7.

Rule & Explanation

Rule: $(a, hasSpouse, b) \Rightarrow (b, hasSpouse, a)$

Explanation: This rule captures the symmetry inherent in the marital relationship. Given that marriage is a bidirectional legal and social contract, if entity a is married to entity b , it logically follows that b is married to a .

Rule: $(a, playsFor, b) \wedge (b, isLocatedIn, c) \Rightarrow (a, livesIn, c)$

Explanation: This inference relies on a probabilistic assumption: professional athletes commonly reside in the same city or country where their affiliated teams are based. While not universally valid due to cases such as commuting or temporary contracts, this pattern holds in the majority of real-world scenarios.

Rule: $(a, wasBornIn, b) \Rightarrow (a, hasPlaceOfBirth, b)$

Explanation: This is a deterministic alias pattern, where both relations *wasBornIn* and *hasPlaceOfBirth* denote the same biographical fact. The two terms are semantically interchangeable in most ontological frameworks.

Rule: $(a, hasNationality, b) \Rightarrow (a, hasCitizenship, b)$

Explanation: This is a high-probability rule rooted in sociopolitical conventions. Although nationality and citizenship may differ in legal terms, they are often used interchangeably in knowledge bases. Most individuals who identify with a nationality also hold legal citizenship of the corresponding state.

Rule: $(a, studiedAt, b) \wedge (b, isLocatedIn, c) \Rightarrow (a, hasLegalResidence, c)$

Explanation: This rule models a moderate-probability correlation: enrollment at an educational institution often necessitates or implies legal residence in the institution’s geographical location, due to immigration and residency regulations applicable to students.

Rule: $(a, hasEmployer, b) \wedge (b, isLocatedIn, c) \Rightarrow (a, livesIn, c)$

Explanation: A high-probability inference, reflecting the assumption that employees generally reside in proximity to their workplace. This assumption may be weakened in the presence of remote work or multi-location employers but holds in standard employment contexts.

Rule: $(a, worksAt, b) \wedge (b, isLocatedIn, c) \wedge (c, hasOfficialLanguage, d) \Rightarrow (a, hasLanguage, d)$

Explanation: This multi-hop rule captures the linguistic environment of a worker. Individuals employed in a region are likely to acquire or use the region’s official language, particularly in professional or social interactions. The inference strength varies by linguistic diversity and integration policies.

Rule: $(a, hasDeathPlace, b) \Rightarrow (a, hasPlaceOfDeath, b)$

Explanation: Another deterministic alias. Both relations refer to the geographical location where an individual passed away. They are synonymous in the context of biographical datasets and can be used interchangeably in logical reasoning.

Table 7: Illustrative examples of inference patterns in YAGO3-10.

1168 I Limitations and Broader impacts

1169 In this paper, we have proposed a novel knowledge unlearning evaluation framework designed to
 1170 capture complex factual dependencies and the inherent uncertainty reflected by LLM confidence.
 1171 Our aim is to enable more comprehensive and realistic assessments of knowledge unlearning effi-
 1172 cacy. Nonetheless, several important limitations remain. First, the inference capabilities of both
 1173 LLM-based and human evaluators have inherent limitations. Both types of judges may overlook
 1174 subtle or complex inference patterns, and real-world adversaries could potentially surpass these
 1175 evaluators in inference power. Second, the process of constructing supporting subgraphs may be
 1176 incomplete. Our subgraph extraction approach might not encompass all relevant facts necessary
 1177 for accurate inference, and the reference knowledge graphs used in our evaluation may themselves
 1178 lack certain entities or relationships that are implicitly encoded within the target LLM. Furthermore,
 1179 our knowledge probing strategy for supporting subgraph construction might not be optimal, and
 1180 we acknowledge that certain knowledge extracted can be inaccurate. Third, our current evaluation
 1181 framework primarily focuses on relational factual knowledge and relies on externally constructed
 1182 reference knowledge graphs to guide the subgraph extraction process. Given that many pretrained
 1183 LLMs utilize undisclosed or proprietary pretraining corpora, discrepancies could arise between
 1184 these reference graphs and the model’s actual internal knowledge representations, further limiting
 1185 evaluation accuracy. Beyond methodological limitations, we also emphasize the ethical implications
 1186 associated with knowledge unlearning evaluations. Specifically, we advocate for the responsible use
 1187 of compliant, publicly accessible, or properly authorized data when conducting such studies. By
 1188 acknowledging these limitations and ethical considerations, we hope our work fosters more rigorous
 1189 and ethically responsible practices in the research, development, and deployment of machine learning
 1190 technologies.