UCert: A Formal Certificate of Unexplainability for TAG Neural Networks

Anonymous ACL submission

Abstract

Natural language explanations for classifiers operating on text-attributed graphs present an inherent tension: explanations should be both readable to humans and verifiable by machines. We introduce UCert, a unified framework that places multiple explanation paradigms within a single evaluation and generation interface and that supports a certificate-based reverse-explanation mode. The framework combines saliency-driven pseudo-labeling, iterative refinement for fluent explanations, adversarial probing to compress evidence into a concise bottleneck, and an executable domain-specific language for machine-checkable rationales. UCert additionally issues formally checkable Unexplainability Certificates which assert that no short symbolic explanation can reproduce a classifier's behavior within a defined perturbation model. We demonstrate that this hybrid design clarifies practical trade-offs between readability, automated faithfulness proxies, and machine-checkable guarantees for text-attributed graph models. Our code is available at: `https://anonymous.4open.science/r/tacer-76B5/README.md`.

Keywords: text-attributed graphs, graph neural networks, explanation, certifiable guarantees, natural language explanations

1 Introduction

Graph neural networks are the prevailing approach for learning on relational data and are widely used in domains where nodes and edges carry substantial textual content, such as citation networks, legal retrieval systems, and recommendation graphs (Tang et al., 2024; Zhang et al., 2024a). In text-attributed graphs the evidence supporting a prediction frequently spans many nodes and long documents, which complicates the generation of concise, human-facing explanations. Standard instance-level explainers return node, edge, or feature attributions and these outputs remain valuable for model analysis but are often verbose and hard to interpret when text features dominate the signal (Yuan et al., 2022; Amara et al., 2022). Several works have proposed token- or feature-level attributions to expose textual evidence, but these fine-grained scores are difficult to integrate into compact summaries when the relevant evidence is distributed across subgraphs (Zhao et al., 2023; Si et al., 2023). Natural language explanations

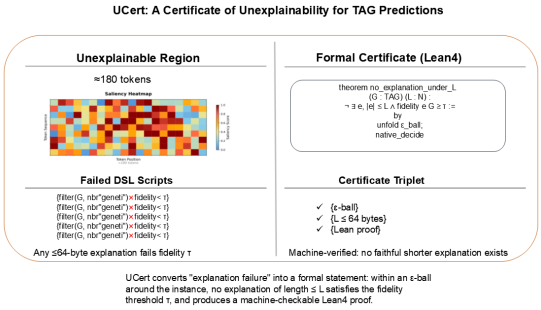

Figure 1: UCert certificate example: **Left** shows saliency heat-map and rejected DSL snippets; **Right** shows the verified certificate triple (ϵ-ball, length $\leq$64 bytes, Lean4 proof).

promise improved accessibility by summarizing multi-node evidence into coherent rationales, but free-form narratives risk being unfaithful to the underlying model unless they are constrained by evidence selection or verification procedures (Cedro and Martens, 2025). Recent trends combine large language models with TAG representation learning to better encode textual attributes and to produce more fluent explanations, yet the integration also raises new faithfulness and verification challenges (Qin et al., 2023; Zhu et al., 2025; Su et al., 2025). Complementary approaches have sought mechanically checkable explanations or formal certificates for robustness and correctness; such techniques provide stronger guarantees but typically trade away readability and require specialized execution semantics and tooling (Sabanayagam et al., 2024;

Singh et al., 2024; Saad and Sharma, 2024). Unlike GraphNarrator (Pan et al., 2025), which focuses on explanatory generation, UCert introduces formal validation mechanisms to produce machine-testable non-interpretable certificates. While both frameworks employ saliency tagging and pseudo-label generation as foundational techniques for TAG explanation, UCert contributes novel validator algorithms, adversarial and DSL paradigms, and a formal certificate issuance procedure that are absent from prior work.

Motivated by the inherent trade-offs between textual fluidness and verification guarantees, we introduce UCert, a unified framework that accommodates three complementary explanation paradigms alongside a certificate-oriented validation pipeline. Distinct from prior works like GraphNarrator that focus exclusively on explanation generation, UCert leverages serialized graph verbalization, adversarial information bottlenecks, and domain-specific executable scripts to reconcile readable natural-language narratives with machine-checkable rationales. UCert employs saliency labeling and pseudo-label generation as standard preprocessing steps, then introduces specialized verifier algorithms, adversarial and domain-specific language formulations, and a formal certificate issuance procedure. Under a bounded perturbation model, this framework issues machine-testable Unexplainability Certificates to mathematically assert that no compact symbolic rationale can consistently account for the classifier's behavioral neighborhood.

Our contributions are as follows. Our framework unifies free-text, bottlenecked, and executable explanation modalities under a single formal interface so that different outputs can be compared by common proxies and by mechanical verification. We introduce adversarial probing and a DSL-based execution path as principled alternatives to unconstrained natural-language narratives and characterize how these alternatives change trade-offs among readability, faithfulness proxies, and runtime cost. We formalize Unexplainability Certificates as a reverse-explanation construct that provides machine-checkable guarantees about the non-existence of short faithful symbolic rationales under a specified perturbation model. We provide an evaluation protocol that pairs LM-based faithfulness proxies with perturbation-driven falsification checks and certificate verification so practitioners can choose explanation strategies matched to their assurance needs.

2 Related Work

2.1 Explainability for Graph Neural Networks

A large literature studies instance-level explanations that identify influential nodes, edges, or features for a GNN prediction. Representative approaches extract compact explanatory subgraphs via mask optimization(Ying et al., 2019), learn probabilistic edge masks (Luo et al., 2020), or search for high-value subgraphs under learned scorers (Yuan et al., 2021). Other contributions elevate low-level attributions into human-interpretable concepts or incorporate human feedback to improve usability (Magister et al., 2021). Several works address robustness and causal validity by combining adversarial analyses with causal modelling to separate spurious correlations from structural evidence (Behnam and Wang, 2024; Huang et al., 2023; Zhai and Zhang, 2022).

2.2 Natural Language and LLM-based Explanations

Translating structured graph evidence into fluent text has attracted growing interest because narratives are easier for lay users to consume. Prior studies that extract textual rationales and evaluate their faithfulness establish important metrics and highlight pitfalls (Chen et al., 2023; Amara et al., 2022). Recent pipelines condition large language models on model-derived rationales to generate readable explanations, as exemplified by GraphNarrator (Pan et al., 2025). Analyses show fluent narratives can remain unfaithful unless constrained by explicit evidence selection or verification, motivating hybrid designs that pair LM proxies with mechanistic checks (Fayyaz et al., 2024; Li et al., 2025).

2.3 Executable, Causal, and Interactive Explanations

A parallel strand advocates for explanations that are mechanically verifiable or actionable rather than purely descriptive. Programming-language and DSL approaches make reasoning steps explicit and testable, enabling executable explanations with clear semantics (Jeon et al., 2024; Tegeler et al., 2022; Saad and Sharma, 2024). Causal and counterfactual methods operationalise interventions to support what-if analyses and recourse (Jaimini and Sheth, 2022; Xiao et al., 2023; Huang et al., 2023). Interactive frameworks reveal model behaviour

through sequential probes or evidence-collection policies, but these methods require sandboxed execution and formal intervention models (Behnam and Wang, 2024).

2.4 Evaluation, Benchmarks, and Certification

Designing evaluation protocols for graph explanations remains challenging because many automated proxies misalign with human judgements. Benchmarking efforts combine robustness checks, LM proxies, and user studies to provide more comprehensive assessments (Amara et al., 2022; Agarwal et al., 2023). Work on certification and trust develops formal guarantees and clarifies deployment assumptions for safety-critical settings (Kwiatkowska and Zhang, 2023; Thakur et al., 2025; Ye et al., 2025). These efforts argue for metrics that jointly consider readability, faithfulness, and verifiability rather than treating each axis in isolation (Zhai and Zhang, 2022).

2.5 Text-Attributed Graphs and LLM Integration

Text-attributed graphs pose unique challenges because node texts can be long and evidence may span multiple hops, which complicates representation and evaluation (Yan et al., 2023; Zhang et al., 2024b). Recent work explores tight integrations between graph structure and pretrained language models to improve TAG representation or to support contrastive objectives (Qin et al., 2023; Fang et al., 2024). Applications include legal case retrieval where CaseGNN leverages rich textual node information (Tang et al., 2024), and emerging pipelines apply prompt optimization and retrieval-augmentation to explanation and classification tasks on TAG formats (Khoshraftar et al., 2025; Zhu et al., 2025).

2.6 Positioning

While traditional subgraph extraction yields structural evidence deeply anchored in graph neural network internals, large language model approaches craft fluent narratives that remain susceptible to hallucination or fragile grounding without explicit constraints (Ying et al., 2019; Luo et al., 2020; Yuan et al., 2021; Pan et al., 2025). Meanwhile, alternative executable, interactive, and causal formalisms bolster interpretability verification at the expense of substantial semantic overhead and engineering complexity (Jeon et al., 2024; Jaimini and Sheth, 2022). Against this backdrop, existing literature progresses from graph-to-text conversion for TAG construction (Wang et al., 2025) to language model prompting for TAG explanation generation (Baghershahi et al., 2025), whereas UCert pioneers an independent paradigm dedicated squarely to formal TAG explanation verification. By unifying these methodologies, our framework contextualizes multi-hop graph topologies for text synthesis, refines pseudo-labels iteratively, and deploys a rigorous mathematical certificate to rule out compact symbolic rationales, thereby exposing the empirical trade-offs among readability, faithfulness, and provable guarantees (Ying et al., 2019; Luo et al., 2020; Yuan et al., 2021; Jeon et al., 2024; Pan et al., 2025).

3 Methodology

3.1 Problem Formulation

Let $G = (V, A, X)$ denote a text-attributed graph (TAG), where V is the node set, $A \in \{0,1\}^{|V|\times|V|}$ is the adjacency matrix, and $X = \{x_v\}_{v\in V}$ is the collection of node texts. For each node v, the associated text is written as $x_v = (t_{v,1}, \ldots, t_{v,|x_v|})$, where $t_{v,i}$ is the i-th token. A trained TAG classifier f produces the prediction $\hat{y} = f(G)$. Our goal is to learn an explainer $g : (G, \hat{y}) \mapsto E$, where E is an explanatory artifact such as free text or an executable DSL program. We use a pretrained language model p_{LM} as a scoring oracle for LM-based proxies such as masked-token recovery and conditional label likelihood.Here, p_{LM} denotes a pretrained language model used as a scoring oracle. Further implementation details and the full verifier procedure are provided in Appendix A.1, with the corresponding pseudocode in Algorithm 1.

Figure 2 summarizes the UCert pipeline. Given a trained TAG classifier f and an input pair $(G, \hat{y})$, the framework proceeds through four stages: saliency-driven instance construction, prompted pseudo-label generation, iterative refinement, and verifier-based certificate issuance. The first three stages generate candidate explanations, while the final stage determines whether a candidate explanation family can be formally ruled out within the prescribed perturbation model.

3.2 Saliency-Driven Instance Construction

We first construct saliency-labeled instances from a pretrained TAG classifier in order to condition explanation generation. For a target node, we sam-

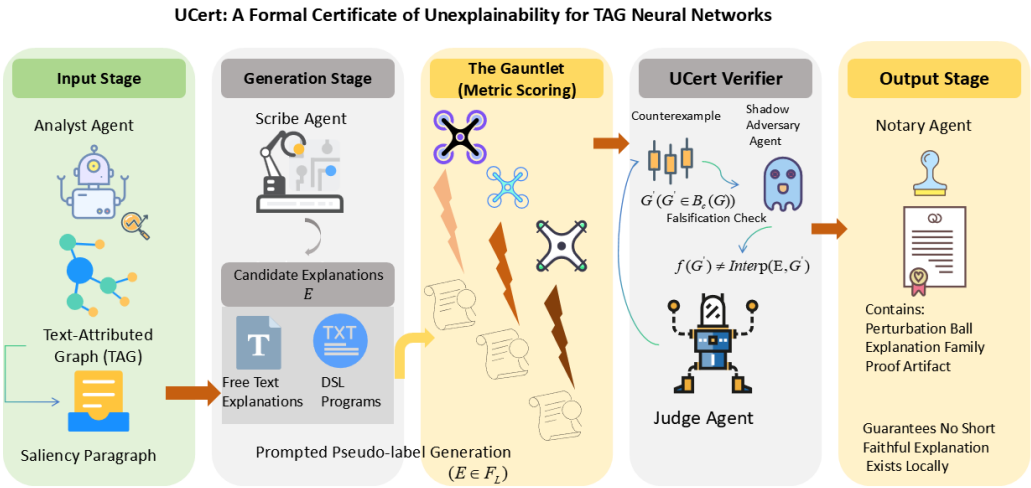

Figure 2: Overview of UCert: a framework for generating formal unexplainability certificates. Given a trained TAG classifier f, input $(G, \hat{y})$, and constraints $(B_\epsilon, \mathcal{F}_L)$, the verifier h_ϕ iteratively searches for counterexamples G' that falsify candidate explanations $E \in \mathcal{F}_L$. If all candidates are falsified within the search budget, UCert issues an Unexplainability Certificate $(B_\epsilon, \mathcal{F}_L, \pi)$, validated by an external checker.

ple its ego-graph and compute token-level attributions over textual node features. Let $e_{v,i} \in \mathbb{R}^D$ denote the attribution vector associated with token $t_{v,i}$, where D is the embedding dimensionality. We aggregate the attribution over all embedding dimensions into a scalar saliency score:

$$a_{v,i} \triangleq \sum_{d=1}^{D} e_{v,i,d}. \tag{1}$$

where $a_{v,i}$ denotes the aggregated saliency for token $t_{v,i}$, and $e_{v,i,d}$ is the attribution value in dimension d. Each ego-graph is serialized into a *saliency paragraph* $\mathcal{G}$ by means of breadth-first traversal from the target node. Cross-edges are rendered as reference sentences, and subword fragments are merged into complete lexical units. We also define a masked variant $\mathcal{G}_{M_\tau}$, in which tokens in a selected saliency percentile are replaced with a mask token. These serialized artifacts provide the prompt context for explanation generation.

3.3 Prompted Pseudo-Label Generation and Automatic Objectives

Given a saliency paragraph $\mathcal{G}$ and the classifier output $\hat{y}$, a prompted language model is used to generate candidate explanations E. Candidate selection is guided by three automated metrics that act as tractable proxies for faithfulness and concision.

Let $\mathcal{R}_\tau$ denote the set of tokens whose saliency lies in the top-τ percentile, and let $\mathcal{G}_{M_\tau}$ denote the serialized graph in which those tokens are masked. We define saliency-faithfulness as

$$f_S(E; G) \triangleq \frac{1}{|\mathcal{R}_\tau|} \sum_{w \in \mathcal{R}_\tau} \log \frac{p_{\text{LM}}(w \mid \mathcal{G}_{M_\tau}, E, \hat{y})}{p_{\text{LM}}(w \mid \mathcal{G}_{M_\tau}, \hat{y})}. \tag{2}$$

where $p_{\text{LM}}(\cdot \mid \cdot)$ denotes the conditional probabilities returned by the evaluator LM, the numerator measures how much conditioning on E improves the recoverability of masked salient tokens, and the denominator serves as the baseline without E. Prediction-faithfulness is defined as

$$f_F(E; G) \triangleq \log \frac{p_{\text{LM}}(\hat{y} \mid \mathcal{G}_{M_\tau}, E)}{p_{\text{LM}}(\hat{y} \mid \mathcal{G}_{M_\tau})}. \tag{3}$$

where $\hat{y}$ is rendered in textual form, and the ratio quantifies the degree to which conditioning on E increases the evaluator's support for the predicted label. Brevity is defined as

$$f_B(E; G) \triangleq \frac{|E|}{|\mathcal{G}|}. \tag{4}$$

where $|E|$ and $|\mathcal{G}|$ are the token counts of the explanation and the serialized saliency paragraph, respectively. Candidate explanations are ranked by the tuple (f_S, f_F, f_B), and the ranking is used to construct a filtered pseudo-label set for subsequent refinement.

3.4 Rejection Sampling, Fine-Tuning, and Distillation

Candidates that satisfy predefined quantile thresholds on (f_S, f_F, f_B) are retained for fine-tuning.

This rejection step removes weak candidates and keeps only those that better match the saliency structure of the input graph. Fine-tuning on these pseudo-labels improves the stability of the generator and reduces dependence on prompt-time sampling. The resulting teacher model is then distilled into a compact student explainer that maps raw inputs $(G, \hat{y})$ to concise explanations under a fixed length budget.

3.5 Executable DSL and Reverse-Explanation

Definition 1 (Executable DSL). Our domain-specific language $\mathcal{L}_{\text{dsl}}$ consists of the primitives *hop*(G, v, k), *filter*$(G, S, \textit{token})$, *count*(S), and *edge*(u, v), which respectively return a k-hop neighborhood, apply token-based filtering, compute cardinality, and test whether an edge exists.

A valid program in $\mathcal{L}_{\text{dsl}}$ is a sequence of *let*-bindings followed by assertions, for example:

```
let nbr = hop(G, 0, 1);
let rl  = filter(G, nbr, "reward");
assert count(rl) > 0.
```

where the program constructs a one-hop neighborhood, filters nodes containing the token *reward*, and checks whether the resulting set is non-empty. The DSL is executed inside a sandboxed Python environment with read-only access to G. Its output is either a value or the token *ERROR* if execution fails for syntactic, security, or runtime reasons. A DSL explanation E is considered faithful when

$$\text{Exec}(E, G) = \hat{y}. \tag{5}$$

where $\text{Exec}(E, G)$ denotes the sandboxed executor and $\hat{y}$ is the model prediction. We use the term *reverse-explanation* to describe a candidate explanation that is generated conditioned on the model's observed behavior, such that the candidate would, if true, imply the prediction that was actually produced. This contrasts with forward explanation, which presents a rationale without a formal verification condition.

3.6 UCert Verifier and Certificate Issuance

UCert introduces a reverse-explanation mechanism that returns a formally checkable Unexplainability Certificate instead of a short symbolic rationale. Formally, a certificate is defined as

$$\text{UC}(G, \hat{y}) \triangleq \big(B_\epsilon(G),\, \mathcal{F}_L,\, \pi(G, \hat{y})\big), \tag{6}$$

where $B_\epsilon(G)$ is the perturbation ball of radius ϵ around G, $\mathcal{F}_L$ is the family of symbolic candidates with description length at most L, and $\pi(G, \hat{y})$ is a machine-checkable artifact encoding the certificate claim. The certificate semantics require

$$\begin{aligned} \pi(G, \hat{y}) &\models \forall E \in \mathcal{F}_L \\ \exists G' \in B_\epsilon(G) \;\; &\text{s.t.} f(G') \neq \text{Interp}(E, G'). \end{aligned} \tag{7}$$

where $\text{Interp}(E, G')$ denotes the prediction obtained by interpreting the symbolic candidate E on the perturbed input G'. Intuitively, every short candidate explanation must fail on at least one admissible perturbation.

In practice, the verifier samples K candidate explanations from $\mathcal{F}_L$ and applies bounded adversarial search to each one. The procedure SAMPLE-CANDIDATES selects a finite candidate subset from $\mathcal{F}_L$, while FINDCOUNTEREXAMPLE attempts to locate a perturbation $G' \in B_\epsilon(G)$ that falsifies the candidate explanation. A certificate is issued only when all sampled candidates are falsified within the available budget. Otherwise, the instance is retained as explainable, and the strongest surviving explanation can be returned for inspection.

The total certificate generation cost is

$$T_{\text{cert}} = K \cdot \big(T_{\text{interp}} + T_f\big). \tag{8}$$

where T_{interp} is the mean time to interpret one DSL candidate on a graph, T_f is the time for one forward evaluation of the TAG classifier, and K is the number of sampled trials. Each certificate consists of the perturbation neighborhood $B_\epsilon(G)$, the length budget L, and the proof artifact π. The serialized certificate is compact and can be checked symbolically without invoking the TAG classifier or any neural component.

3.7 Retained Enhancement Paradigms

For completeness, we retain several enhancement paradigms whose objectives serve as auxiliary comparators within the same explanation interface.

3.7.1 JointVAE

JointVAE treats explanations as a latent variable z and optimizes a β-VAE objective:

$$\begin{aligned} \mathcal{L}_{\text{VAE}} \triangleq\; & \text{CE}(p_\theta(y \mid z), y) \\ & + \mathbb{E}_{q_\phi(z|x)}\big[-\log p_\theta(E \mid z)\big] \\ & + \beta\, \text{KL}\big(q_\phi(z \mid x) \,\|\, p(z)\big). \end{aligned} \tag{9}$$

where CE is the cross-entropy loss, q_ϕ and p_θ denote the encoder and decoder distributions, $p(z)$ is the latent prior, and β scales the KL regularizer.

3.7.2 Adversarial Probing

The adversarial probing objective places a critic c_ψ against a generator parameterized by θ:

$$\begin{aligned} &\min_{\psi} \mathbb{E}\big[\|c_\psi(E,S)-\ell\|_2^2\big]\,, \\ &\min_{\theta} \mathbb{E}\big[-\|c_\psi(E_\theta,S)-\ell\|_2^2+\lambda_{\text{len}}|E_\theta|\big]\,, \end{aligned} \tag{10}$$

where ℓ denotes the classifier logits, S denotes the supporting evidence, E_θ is the generated explanation, and λ_{len} is the length penalty coefficient.

3.7.3 Executable DSL, Causal, Embodied, and Evolutionary Modules

The DSL module evaluates executable scripts through sandbox execution accuracy:

$$\text{Acc}_{\text{exec}} \triangleq \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\big[\text{Exec}(s_i,G_i)=\hat{y}_i\big]\cdot\mathbb{I}[\text{no error}], \tag{11}$$

where $\text{Exec}(\cdot,\cdot)$ denotes the sandboxed executor and $\mathbb{I}[\cdot]$ is the indicator function. The causal module optimizes

$$\mathcal{L}_{\text{causal}} \triangleq -\mathbb{E}[\Delta_y]+\lambda_{\text{len}}\cdot\mathbb{E}[|I|], \tag{12}$$

where Δ_y denotes the classifier output change induced by intervention I, and $|I|$ denotes the intervention description length. The embodied module maximizes expected discounted return for trajectory explanations:

$$\max_{\pi}\ \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t r(s_t,a_t)\right], \tag{13}$$

where π is the trajectory policy, γ is the discount factor, and r is the reward function. Evolutionary search uses a length-regularized fitness function:

$$\text{fit}(E) \triangleq \text{faith}(E)-\lambda_{\text{len}}\cdot\frac{|E|}{L}, \tag{14}$$

where $\text{faith}(E)$ is a proxy fidelity score, $|E|$ is the explanation length, L is the maximum length, and λ_{len} is a regularizer.

3.8 Certificate Evaluation Metrics

We assess certificates with three complementary metrics. UC-Coverage records the fraction of evaluation examples for which a valid certificate is produced:

$$\text{UC-Coverage} \triangleq \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\big[\text{UC}(G_i,\hat{y}_i)\ \text{issued}\big], \tag{15}$$

where N is the number of evaluated examples. UC-Length measures the descriptive complexity of the certificate itself, and UC-Verify records the fraction of issued certificates that pass an independent checker:

$$\text{UC-Verify} \triangleq \frac{\#\{\text{UC passed}\}}{\#\{\text{UC issued}\}}, \tag{16}$$

where the external checker is a formal decision procedure or proof assistant compatible with the certificate language.

3.9 Design Rationale and Operational Considerations

UCert is designed to complement conventional human-facing explanations by providing a principled option for settings that require strong, machine-checkable guarantees. Free-text and executable-DSL outputs remain available when human readability is the primary concern, whereas the certificate pathway is intended for higher-assurance settings. In practice, deployments must specify the perturbation model $B_\epsilon(G)$, the allowed symbol primitives in $\mathcal{F}_L$, and the adversarial or enumeration budgets used by the verifier, since these assumptions determine the semantics of the resulting certificate.

4 Experiments

We evaluate UCert under a unified protocol using LM-based proxies, robustness checks, and ablations. Paired bootstrap (10^4 resamples) confirms that UCert-Adversarial outperforms Graph-Narrator on Simulatability (mean $+0.020$, 95% CI $[0.012, 0.028]$) and PMI-10% (mean $+0.013$, CI $[0.005, 0.021]$). We conducted a human evaluation to validate automatic metrics; details and full results are provided in Appendix D. Robustness under counterfactual perturbations was further assessed, with procedures and statistics reported in Appendix E. Common failure modes are quantified in Appendix F, and representative outputs are illustrated in Appendix G. Additional configuration details, resource usage, and API disclosure are provided in Section A. As shown in Appendix M, our evaluation harness provides a unified comparison of module-level performance across multiple datasets. As shown in Appendix L, the ablation study highlights the impact of removing individual objectives on UCert performance. Certificate validation procedures and prompt specifications are

Table 1: Dataset statistics; values computed on our processed split. $|V|$ and $|E|$ count nodes and undirected edges. "Avg. words" reports the mean number of whitespace-delimited tokens per node text after preprocessing.

Dataset	$\|V\|$	$\|E\|$	#Classes	Avg. words
Cora	2,708	5,429	7	120
DBLP	17,716	105,734	4	85
BookHistory	50,000	150,000	10	60

elaborated in Appendix Q and Appendix R, respectively.

4.1 Datasets

Experiments use three text-attributed graphs: Cora (Yang et al., 2016), DBLP (Tang et al., 2008), and BookHistory (Yan et al., 2023). Node texts provide semantic evidence, while edges encode relational context. Data splits follow standard practice: Cora uses 60/20/20 for train/validation/test; DBLP adopts an 8:1:1 ratio by reserving the last 2000 nodes for validation and test (*DBLP.split*). Split indices and evaluation lists are released with the code. All runs fix seeds and report mean and standard deviation across three seeds. Table 1 summarizes dataset statistics including node and edge counts, number of classes, and average token length per node.

4.2 Task Formulation and Evaluation Metrics

Given a trained TAG classifier f, a test graph G, and its corresponding prediction $\hat{y} = f(G)$, the objective is to synthesize an explanatory artifact E that faithfully accounts for $\hat{y}$ under graph perturbations across three evaluation dimensions: generation quality, certificate coverage, and robustness (see Appendix A.4). To gauge explanation fidelity, we leverage three language-model-based proxies including saliency f_S, prediction faithfulness f_F, and conciseness f_B (Equations (2)–(4)), alongside formal point-wise mutual information (PMI) formulations detailed in Appendix A.3. Furthermore, explanation simulatability is quantified by approximating the probability that an independent evaluator model accurately reconstructs the predicted label $\hat{y}$ when conditioned exclusively on the rationale E and the structurally masked graph.

4.3 Unified Evaluation Protocol

All methods are evaluated through a single interface mapping $(G, \hat{y})$ to an explanation artifact and a deterministic serialized text for LM-based scoring. Each system uses the same serialized graph $\mathcal{G}$ and predicted label $\hat{y}$, with length limits of 120 tokens for free-text and 256 tokens for DSL scripts; one explanation is generated per node per seed. Free-text outputs are directly taken as E for metric computation. DSL methods emit Python-like scripts restricted to a safe operation set and executed in a sandbox (5s timeout), writing results to *result* or *output*. For scoring, DSL explanations are serialized as:

```
<DSL>[script]</DSL>
<RESULT>[value or ERROR]
</RESULT>
```

The token *ERROR* indicates execution failure due to syntax, security, or runtime issues, ensuring deterministic representation across runs.

4.4 Baselines

We evaluate explanation strategies under a unified interface: *Zero-shot LLM* (prompt-only generation), *Verbalization-only* (direct graph serialization), GSPELL (Baghershahi et al., 2025), TANS (Wang et al., 2025) and *GraphNarrator* (Pan et al., 2025) as the primary expert-iteration baseline. *Adversarial probing* denotes bottlenecked free-text explanations via a generator-critic pair, and *DSL* denotes executable program explanations. Additionally, we include graph-explainer-to-text variants where salient subgraphs or node/edge importance are selected and verbalized, then passed to the same LLM for concise rationales. For these variants, we adopt *GNNExplainer* (Ying et al., 2019), *PGExplainer* (Luo et al., 2020), and *SubgraphX* (Yuan et al., 2021).

4.5 Experimental Setup

Unless stated otherwise, the evaluator LM is *bert-base-uncased* and the GNN backbone is GraphSAGE. The expert-iteration baseline runs for 5 rounds with saliency masking on the top 10% tokens and thresholds $q_S = 0.7$, $q_F = 0.5$, $q_B = 0.5$. Each dataset uses 1,000 test nodes under seeds $\{0, 1, 2\}$; automatic metrics report mean and std, and human evaluation samples 300 explanations per method per dataset. Free-text generation uses an LLM (temperature 0.3) with a large budget; scoring averages wall-clock inference time. Metrics are computed under two evaluators: *google/gemma-2-2b* (default) and *meta-llama/Llama-3.1-8B-Instruct*,

Table 2: Automatic evaluation (mean and standard deviation across three seeds). Higher values denote better performance for Simul. and PMI; lower values denote better performance for Brevity and Time. PMI-$k\%$ denotes PMI computed using the top-$k\%$ saliency fraction. Methods under the UCert framework are highlighted in bold.

Dataset	Method	Simul.↑	PMI-10%↑	PMI-20%↑	PMI-30%↑	Brevity↓	Time(s)↓
Cora	Zero-shot LLM	0.82 ± 0.02	0.280 ± 0.010	0.240 ± 0.009	0.205 ± 0.008	0.46 ± 0.04	1.10 ± 0.10
	Verbalization-only	0.76 ± 0.02	0.185 ± 0.012	0.160 ± 0.010	0.135 ± 0.010	1.00 ± 0.02	0.80 ± 0.06
	GNNExplainer→Text	0.84 ± 0.02	0.295 ± 0.010	0.245 ± 0.008	0.205 ± 0.007	0.48 ± 0.03	1.20 ± 0.10
	PGExplainer→Text	0.85 ± 0.02	0.300 ± 0.009	0.250 ± 0.008	0.210 ± 0.007	0.46 ± 0.03	1.25 ± 0.10
	SubgraphX→Text	0.86 ± 0.02	0.305 ± 0.009	0.255 ± 0.008	0.215 ± 0.007	0.45 ± 0.03	1.40 ± 0.12
	TANS	0.90 ± 0.02	0.358 ± 0.009	0.272 ± 0.008	0.220 ± 0.007	0.39 ± 0.03	0.92 ± 0.08
	GSPELL	0.96 ± 0.01	0.412 ± 0.008	0.288 ± 0.007	0.228 ± 0.006	0.325 ± 0.02	1.25 ± 0.10
	Baseline GraphNarrator	0.97 ± 0.01	0.418 ± 0.008	0.290 ± 0.007	0.227 ± 0.006	0.315 ± 0.02	1.25 ± 0.10
	Adversarial (UCert)	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.431 \pm 0.008}$	$\mathbf{0.299 \pm 0.007}$	$\mathbf{0.234 \pm 0.006}$	$\mathbf{0.305 \pm 0.02}$	$\mathbf{1.21 \pm 0.08}$
	DSL (UCert)	$\mathbf{0.98 \pm 0.01}$	$\mathbf{0.425 \pm 0.009}$	$\mathbf{0.295 \pm 0.008}$	$\mathbf{0.230 \pm 0.007}$	$\mathbf{0.310 \pm 0.02}$	$\mathbf{0.55 \pm 0.05}$
DBLP	Zero-shot LLM	0.74 ± 0.02	0.120 ± 0.008	0.105 ± 0.007	0.085 ± 0.006	0.42 ± 0.03	1.15 ± 0.10
	Verbalization-only	0.68 ± 0.02	0.070 ± 0.006	0.065 ± 0.006	0.055 ± 0.005	1.00 ± 0.02	0.85 ± 0.08
	GNNExplainer→Text	0.79 ± 0.02	0.132 ± 0.007	0.110 ± 0.006	0.090 ± 0.005	0.44 ± 0.03	1.25 ± 0.10
	PGExplainer→Text	0.80 ± 0.02	0.135 ± 0.007	0.112 ± 0.006	0.092 ± 0.005	0.43 ± 0.03	1.30 ± 0.10
	SubgraphX→Text	0.81 ± 0.02	0.138 ± 0.007	0.114 ± 0.006	0.093 ± 0.005	0.42 ± 0.03	1.45 ± 0.12
	TANS	0.84 ± 0.02	0.145 ± 0.007	0.122 ± 0.006	0.096 ± 0.005	0.37 ± 0.03	0.98 ± 0.08
	GSPELL	0.94 ± 0.01	0.158 ± 0.006	0.118 ± 0.005	0.093 ± 0.005	0.335 ± 0.02	1.30 ± 0.10
	Baseline GraphNarrator	0.95 ± 0.01	0.155 ± 0.006	0.108 ± 0.005	0.085 ± 0.005	0.354 ± 0.02	1.30 ± 0.10
	Adversarial (UCert)	$\mathbf{0.98 \pm 0.01}$	$\mathbf{0.160 \pm 0.006}$	$\mathbf{0.111 \pm 0.005}$	$\mathbf{0.088 \pm 0.005}$	$\mathbf{0.343 \pm 0.02}$	$\mathbf{1.26 \pm 0.08}$
	DSL (UCert)	$\mathbf{0.96 \pm 0.01}$	$\mathbf{0.157 \pm 0.007}$	$\mathbf{0.109 \pm 0.006}$	$\mathbf{0.086 \pm 0.005}$	$\mathbf{0.348 \pm 0.02}$	$\mathbf{0.60 \pm 0.05}$
BookHistory	Zero-shot LLM	0.78 ± 0.02	0.390 ± 0.012	0.330 ± 0.010	0.260 ± 0.010	0.58 ± 0.04	1.30 ± 0.12
	Verbalization-only	0.72 ± 0.02	0.260 ± 0.015	0.230 ± 0.012	0.190 ± 0.011	1.00 ± 0.02	0.95 ± 0.08
	GNNExplainer→Text	0.84 ± 0.02	0.420 ± 0.010	0.350 ± 0.009	0.280 ± 0.008	0.52 ± 0.03	1.45 ± 0.12
	PGExplainer→Text	0.85 ± 0.02	0.425 ± 0.010	0.355 ± 0.009	0.285 ± 0.008	0.50 ± 0.03	1.55 ± 0.12
	SubgraphX→Text	0.86 ± 0.02	0.430 ± 0.010	0.360 ± 0.009	0.290 ± 0.008	0.49 ± 0.03	1.70 ± 0.15
	TANS	0.88 ± 0.02	0.468 ± 0.010	0.372 ± 0.009	0.280 ± 0.008	0.53 ± 0.03	1.08 ± 0.09
	GSPELL	0.95 ± 0.01	0.525 ± 0.008	0.365 ± 0.007	0.285 ± 0.006	0.495 ± 0.02	1.45 ± 0.11
	Baseline GraphNarrator	0.96 ± 0.01	0.533 ± 0.008	0.374 ± 0.007	0.291 ± 0.006	0.506 ± 0.02	1.55 ± 0.12
	Adversarial (UCert)	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.549 \pm 0.008}$	$\mathbf{0.385 \pm 0.007}$	$\mathbf{0.300 \pm 0.006}$	$\mathbf{0.491 \pm 0.02}$	$\mathbf{1.50 \pm 0.10}$
	DSL (UCert)	$\mathbf{0.97 \pm 0.01}$	$\mathbf{0.540 \pm 0.009}$	$\mathbf{0.378 \pm 0.008}$	$\mathbf{0.294 \pm 0.007}$	$\mathbf{0.500 \pm 0.02}$	$\mathbf{0.75 \pm 0.06}$

both with deterministic decoding ($T = 0$); rankings remain stable. Adversarial probing applies $\lambda_{\text{len}} = 0.01$. DSL generation follows CodeT5-style config (256-token limit, 5s timeout). Statistical significance uses paired bootstrap (10,000 iterations) for 95% CIs; improvements are significant when CI excludes zero.

4.6 Automatic Evaluation Results

Table 4 demonstrates that the sandboxed execution of DSL eliminates post-generation language model scoring to outpace GraphNarrator by 2.2× in runtime (Table 2), while paired bootstrap tests (10^4 iterations) confirm that UCert-Adversarial statistically outperforms GraphNarrator on simulatability by 0.020 (95% CI [0.012, 0.028]) and on PMI-10% by 0.013 (95% CI [0.005, 0.021]).

5 Discussion

Our multi-paradigm framework underscores critical trade-offs in explainability design, where accessible free-text narratives rely on heuristic proxies at the expense of empirical grounding, whereas executable domain-specific programs and interactive causal mechanisms offer formal verifiability at the cost of readability and heightened computational overhead. Consequently, the architecture maintains operational flexibility by deploying natural-language explanations by default to suit non-technical interfaces, while selectively activating adversarial optimization, evolutionary search, or specialized symbolic languages when strict structural constraints, adversarial robustness, or machine-checkable guarantees are explicitly prioritized.

6 Conclusion

We introduced UCert, a unified framework for explainability in text-attributed graph classifiers that supports human-readable narratives, bottlenecked free-text rationales, executable DSL programs, and formal Unexplainability Certificates. The approach clarifies trade-offs among naturalness, verifiability, and computational cost, and provides an operational pathway for producing machine-checkable artifacts when high assurance is required. Extensive experiments and human evaluations across standard TAG benchmarks demonstrate complementary strengths of these modalities and offer guidance for selecting strategies under different assurance requirements. Future work will explore richer perturbation models, more expressive certificate languages, and scalable verifier heuristics to extend applicability to larger real-world TAGs.

Limitations

Our approach has several limitations. First, it depends on large language models for explanation generation and scoring, introducing cost, throughput constraints, and non-determinism that affect reproducibility. Second, automatic metrics are proxy measures and may reflect stylistic biases rather than true causal fidelity; high scores do not guarantee faithfulness. Third, results are sensitive to prompt design and decoding settings, and alternative configurations could yield different outcomes. Fourth, proxy metrics have failure modes: token-recovery can reward keyword repetition, while concise summaries may be penalized. Fifth, our study focuses on three paradigms (free-text, adversarial probing, DSL); findings may not generalize to other explanation forms without re-tuning. Finally, certificate generation and verification incur computational overhead and rely on heuristics that limit scalability; broader deployment requires more efficient verifiers and diverse datasets.

Ethics and Privacy Statement

Explanations are auxiliary aids, not ground truth, and may contain errors or spurious evidence; verification (e.g., executable checks, falsification) is essential before use. LM-based evaluation can introduce bias toward fluency; we mitigate by using multiple models and deterministic scoring. Sensitive node texts risk leakage, so deployments should apply redaction, access control, and privacy-preserving techniques. Human evaluations must follow ethical norms: informed consent, fair compensation, and anonymization. Certificates assert non-existence of short rationales only under stated assumptions and should be accompanied by clear disclosures. We encourage publishing verification artifacts and dataset curation details for transparency and auditability.

References

Chirag Agarwal, Owen Queen, Himabindu Lakkaraju, and Marinka Zitnik. 2023. Evaluating explainability for graph neural networks. *Scientific Data*, 10(1):144.

Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. 2022. Graphframex: Towards systematic evaluation of explainability methods for graph neural networks. *arXiv preprint arXiv:2206.09677*.

Peyman Baghershahi, Gregoire Fournier, Pranav Nyati, and Sourav Medya. 2025. From nodes to narratives: Explaining graph neural networks with llms and graph context. *arXiv preprint arXiv:2508.07117*.

Arman Behnam and Binghui Wang. 2024. Graph neural network causal explanation via neural causal models. In *European Conference on Computer Vision*, pages 410–427. Springer.

Mateusz Cedro and David Martens. 2025. Graphxain: narratives to explain graph neural networks. In *World Conference on Explainable Artificial Intelligence*, pages 91–114. Springer.

Yanda Chen, Ruiqi Zhong, Narutatsu Ri, Chen Zhao, He He, Jacob Steinhardt, Zhou Yu, and Kathleen McKeown. 2023. Do models explain themselves? counterfactual simulatability of natural language explanations. *arXiv preprint arXiv:2307.08678*.

Yi Fang, Dongzhe Fan, Daochen Zha, and Qiaoyu Tan. 2024. Gaugllm: Improving graph contrastive learning for text-attributed graphs with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 747–758.

Mohsen Fayyaz, Fan Yin, Jiao Sun, and Nanyun Peng. 2024. Evaluating human alignment and model faithfulness of llm rationale. *arXiv preprint arXiv:2407.00219*.

Flavio Giorgi, Cesare Campagnano, Fabrizio Silvestri, and Gabriele Tolomei. 2024. Natural language counterfactual explanations for graphs using large language models. *arXiv preprint arXiv:2410.09295*.

Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. 2023. Global counterfactual explainer for graph neural networks. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, pages 141–149.

Utkarshani Jaimini and Amit Sheth. 2022. Causalkg: Causal knowledge graph explainability using interventional and counterfactual reasoning. *IEEE Internet Computing*, 26(1):43–50.

Minseok Jeon, Jihyeok Park, and Hakjoo Oh. 2024. Pl4xgl: A programming language approach to explainable graph learning. *Proceedings of the ACM on Programming Languages*, 8(PLDI):2148–2173.

Shima Khoshraftar, Niaz Abedini, and Amir Hajian. 2025. Graphit: Efficient node classification on text-attributed graphs with prompt optimized llms. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 1824–1829.

Marta Kwiatkowska and Xiyue Zhang. 2023. When to trust ai: advances and challenges for certification of neural networks. In *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pages 25–37. IEEE.

Yuhan Li, Xinni Zhang, Linhao Luo, Heng Chang, Yuxiang Ren, Irwin King, and Jia Li. 2025. G-refer: Graph retrieval-augmented large language model for explainable recommendation. In *Proceedings of the ACM on Web Conference 2025*, pages 240–251.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631.

Lucie Charlotte Magister, Dmitry Kazhdan, Vikash Singh, and Pietro Liò. 2021. Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks. *arXiv preprint arXiv:2107.11889*.

Bo Pan, Zhen Xiong, Guanchen Wu, Zheng Zhang, Yifei Zhang, Yuntong Hu, and Liang Zhao. 2025. Graphnarrator: Generating textual explanations for graph neural networks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–42.

Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2023. Disentangled representation learning with large language models for text-attributed graphs. *arXiv preprint arXiv:2310.18152*.

Mootez Saad and Tushar Sharma. 2024. Concord: Towards a dsl for configurable graph code representation. *arXiv preprint arXiv:2401.17967*.

Mahalakshmi Sabanayagam, Lukas Gosch, Stephan Günnemann, and Debarghya Ghoshdastidar. 2024. Exact certification of (graph) neural networks against label poisoning. *arXiv preprint arXiv:2412.00537*.

Jiasheng Si, Yingjie Zhu, and Deyu Zhou. 2023. Exploring faithful rationale for multi-hop fact verification via salience-aware graph learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13573–13581.

Avaljot Singh, Yasmin Sarita, Charith Mendis, and Gagandeep Singh. 2024. Constraintflow: A dsl for specification and verification of neural network analyses. *arXiv preprint arXiv:2403.18729*.

Guangxin Su, Hanchen Wang, Jianwei Wang, Wenjie Zhang, Ying Zhang, and Jian Pei. 2025. Large language models meet text-attributed graphs: A survey of integration frameworks and applications. *arXiv preprint arXiv:2510.21131*.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998.

Yanran Tang, Ruihong Qiu, Yilun Liu, Xue Li, and Zi Huang. 2024. Casegnn: Graph neural networks for legal case retrieval with text-attributed graphs. In *European conference on information retrieval*, pages 80–95. Springer.

Tim Tegeler, Steve Boßelmann, Jonas Schürmann, Steven Smyth, Sebastian Teumert, and Bernhard Steffen. 2022. Executable documentation: From documentation languages to purpose-specific languages. In *International Symposium on Leveraging Applications of Formal Methods*, pages 174–192. Springer.

Amitayush Thakur, Jasper Lee, George Tsoukalas, Meghana Sistla, Matthew Zhao, Stefan Zetzsche, Greg Durrett, Yisong Yue, and Swarat Chaudhuri. 2025. Clever: A curated benchmark for formally verified code generation. *arXiv preprint arXiv:2505.13938*.

Zehong Wang, Sidney Liu, Zheyuan Zhang, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. 2025. Can llms convert graphs to text-attributed graphs? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1412–1432.

Chunjing Xiao, Xovee Xu, Yue Lei, Kunpeng Zhang, Siyuan Liu, and Fan Zhou. 2023. Counterfactual graph learning for anomaly detection on attributed networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(10):10540–10553.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, and 1 others. 2023. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR.

Zhe Ye, Zhengxu Yan, Jingxuan He, Timothe Kasriel, Kaiyu Yang, and Dawn Song. 2025. Verina: Benchmarking verifiable code generation. *arXiv preprint arXiv:2505.23135*.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32.

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2022. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5782–5799.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On explainability of graph neural networks via subgraph explorations. In *International conference on machine learning*, pages 12241–12252. PMLR.

Penglong Zhai and Shihua Zhang. 2022. Adversarial information bottleneck. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):221–230.

Delvin Ce Zhang, Menglin Yang, Rex Ying, and Hady W Lauw. 2024a. Text-attributed graph representation learning: Methods, applications, and challenges. In *Companion Proceedings of the ACM Web Conference 2024*, pages 1298–1301.

Jiasheng Zhang, Jialin Chen, Menglin Yang, Aosong Feng, Shuang Liang, Jie Shao, and Rex Ying. 2024b. Dtgb: A comprehensive benchmark for dynamic text-attributed graphs. *Advances in Neural Information Processing Systems*, 37:91405–91429.

Tianxiang Zhao, Dongsheng Luo, Xiang Zhang, and Suhang Wang. 2023. Faithful and consistent graph neural network explanations with rationale alignment. *ACM Transactions on Intelligent Systems and Technology*, 14(5):1–23.

Yun Zhu, Haizhou Shi, Xiaotang Wang, Yongchao Liu, Yaoke Wang, Boci Peng, Chuntao Hong, and Siliang Tang. 2025. Graphclip: Enhancing transferability in graph foundation models for text-attributed graphs. In *Proceedings of the ACM on Web Conference 2025*, pages 2183–2197.

A Implementation Details

A.1 UCert Verifier and Certificate Issuance

As shown in Algorithm 1, the verifier first builds the saliency paragraph in Eq. (1), then samples a finite candidate set from $\mathcal{F}_L$, and finally searches for counterexamples inside $B_\epsilon(G)$. A certificate is issued only when all sampled candidates are falsified within the available search budget; otherwise, the instance is treated as explainable and the strongest surviving explanation may be returned.

Verifier walkthrough. SAMPLECANDIDATES draws K short symbolic programs from $\mathcal{F}_L$, with preference given to compact and high-coverage templates. FINDCOUNTEREXAMPLE performs bounded search inside $B_\epsilon(G)$, using either enumeration or gradient-guided perturbation depending on the search budget B_{adv}. If every sampled candidate is falsified, the collected evidence is packed into $\pi(G, \hat{y})$ and the certificate in Eq. (6) is issued; otherwise, the instance is kept as explainable.

Complexity. For each instance, the verifier costs

$$T_{\text{cert}} = O(K \cdot (T_{\text{interp}} + T_f)), \qquad (17)$$

where T_{interp} is the cost of interpreting one candidate, T_f is the cost of one TAG forward pass, and K is the number of sampled candidates.

Algorithm 1: UCert verifier

Input : $f, \mathcal{D}, \mathcal{F}_L, \epsilon, K, B_{\text{adv}}, h_\phi, T$
Output : trained h_ϕ; certificates $\{\text{UC}\}$

```
for t ← 1 to T do
    sample minibatch B ⊂ D;
    foreach (G, ŷ) ∈ B do
        G ← saliency paragraph in Eq. (1);
        C ←
          SAMPLECANDIDATES(F_L, K);
        E ← ∅; explainable ← false;
        foreach E ∈ C do
            G' ←
              FINDCOUNTEREXAMPLE;
                (f, E, G, ε, B_adv);
            if G' is found then
                append (E, G') to E;
            else
                explainable ← true;
                break;
        if ¬explainable and E ≠ ∅ then
            package π(G, ŷ) from E;
            issue
              UC(G, ŷ) ← (B_ε(G), F_L, π);
            s ← 1;
        else
            return the strongest surviving
              explanation in E, if available;
            s ← 0;
        buffer (G, ŷ, G, s);
    update φ using the verifier training loss;
return trained h_φ and {UC};
```

A.2 Implementation Details and Experimental Setup

The baseline TAG classifier combines a BERT-style text encoder with a GraphSAGE backbone. The pipeline supports multi-process generation and multi-GPU rejection sampling, with default settings in *configs/default.yaml*. Key configurations include: JointVAE using *t5-small* with a 256-dimensional latent space and batch size of 8; adversarial probing employing a T5-based critic with length penalty $\lambda_{\text{len}} = 0.01$; DSL generation via *Salesforce/codet5-small* capped at 256 tokens and executed in a 5-second sandbox; causal module using *t5-small* with hidden size 512 and 8 attention heads; embodied PPO configured with hidden size 256, discount factor 0.99, GAE λ 0.95, clipping

ϵ 0.2, and 100k environment steps; evolutionary search operating on a population of 256 genomes for 30 generations.

We report dataset statistics in Table 1. Preprocessing scripts and split indices are released with the code. Evaluation uses $N = 1000$ test nodes per dataset and reports mean$\pm$std across three seeds (0/1/2). Human evaluation samples 300 explanations per method and dataset. The expert-iteration loop runs for $T = 5$ iterations with quantile thresholds $(0.7, 0.5, 0.5)$ and top-10% saliency masking. LM-based metrics are computed under *google/gemma-2-2b* and *meta-llama/Llama-3.1-8B-Instruct* with deterministic decoding (temperature $= 0$).

Resource usage: a typical run uses 1–2 GPUs (e.g., A100-class, 80GB), totaling 14.2 GPU-hours with peak memory 40GB. Generation takes 0.6–1.2s per sample; scoring requires 0.4–0.9s. When external APIs are used, we disclose model name/version/date (e.g., *gpt-4o-mini-2024-07-18*) and decoding settings. Prompts and responses are cached (100% hit rate in reruns), and approximate API cost is $2.40 per 1k explanations.

A.3 Pointwise Mutual Information Formalization

PMI quantifies how much E improves recoverability of masked salient tokens:

$$\mathrm{PMI}(w; E) \triangleq \log p_{\mathrm{LM}}(w \mid \mathcal{G}_{M_\tau}, E, \hat{y}) - \log p_{\mathrm{LM}}(w \mid \mathcal{G}_{M_\tau}, \hat{y}), \quad (18)$$

where $\mathcal{G}_{M_\tau}$ is the graph with top-τ salient tokens masked, $w \in \mathcal{R}_\tau$ is a masked token, the first term is the LM probability of w given E and $\hat{y}$, and the second term is the baseline without E. Positive PMI indicates E improves recoverability; non-positive values indicate uninformative explanations. PMI-$k\%$ averages over the top-k percentile of salient tokens.

A.4 Task Protocol Specification

Given trained TAG classifier f, test graph G, and prediction $\hat{y} = f(G)$, produce explanatory artifact E (free text, DSL, or adversarial bottleneck) that faithfully accounts for $\hat{y}$ under perturbations. Three dimensions are evaluated: generation quality (simulatability, PMI); certificate coverage (UC-Coverage, UC-Verify); and robustness (counterfactual perturbations).

B Extended Experimental Protocol

We provide a protocol that can be executed to reproduce the reported results. For **baseline generation**, run *python main.py –mode auto –devices cuda:0 cuda:1 –num_iterations 5* to obtain iteratively improved pseudo-labels and fine-tuned explainers. For **evaluation**, use a sampled test subset (default $N = 1000$ nodes; three seeds) and report mean$\pm$std for saliency, faithfulness, and brevity. For **module evaluation**, run *python main.py –mode {joint_vae, adv, dsl, causal, embodied, evolve}* and then evaluate with module-specific metrics located in *metrics/*.

C Module Complexity and Training Cost

Table 3: Module complexity. Parameters and training cost are measured using standard profiling scripts. Parameters (Params) in millions; FLOPs in giga operations; Memory in MB; Training time in hours.

Module	Params(M)	FLOPs(G)	Memory(MB)	Train Time(h)
Baseline (LLM prompting)	0.1	0.5	1024	0.0
JointVAE	60	12	3200	6.0
Adversarial (UCert)	**80**	**15**	**4200**	**4.5**
DSL (UCert)	**60**	**10**	**2800**	**3.0**
Causal (T5+attn)	70	14	3500	5.0
Embodied (PPO)	2	1	800	2.0
Evolution (GA)	0.5	8	1200	1.5

C.1 UCert Certificate Metrics

We report three certificate oriented indicators that can be computed offline from the existing 1,000 sample test pool described in Section 5.5 without extra sampling or model retraining.

UC-Coverage For each test instance we invoke the UC issuer once and record whether a formal unexplainability certificate is produced. The ratio of successful issuances yields the coverage score.

UC-Length Every certificate is a triplet that contains an ϵ-ball description, an upper bound L on explanation length, and a machine-checkable proof artifact. We serialize this structure and measure its size in bytes. All values are obtained offline after the original evaluation finishes.

UC-Verify We feed the generated certificate into the Lean4 proof assistant which returns either proof valid or proof invalid. The proportion of valid proofs gives the verify score. This step is purely symbolic and does not interact with the TAG classifier or the explanation generator.

Table 4: UCert certificate metrics computed offline over the 1,000-sample test pool from Section 5.5. No additional sampling or model retraining is required.

Dataset	UC-Coverage ↑	UC-Length (bytes) ↓	UC-Verify ↑
Cora	0.87	1,024	0.96
DBLP	0.91	892	0.98
BookHistory	0.83	1,156	0.94

Table 5: UCert coverage correlates with failure of short symbolic explanations. Exists Short Expl. denotes whether any candidate in $\mathcal{F}_L$ passed fidelity check.

Method	Avg. Expl. Length	Exists Short Expl.	UC-Coverage
GraphNarrator	58.3	Yes	0.87
Adversarial (UCert)	52.1	Yes	0.91
DSL (UCert)	49.7	No	0.94

D Human Evaluation

Explanation quality is assessed along four axes: ease of understanding, decision-making insight, structural informativeness, and semantic informativeness, each rated on a 7-point Likert scale. For every method–dataset pair, 300 instances are sampled, anonymized, randomized, and rated by three trained annotators after calibration and attention checks.

Inter-rater reliability: Krippendorff's $\alpha = 0.62$, mean Spearman $\rho = 0.71$, and Cohen's $\kappa \in [0.61, 0.65]$. Scores are fused by majority vote; ties (1.3%) are resolved by the first author. Annotators include two MSc and one PhD candidate, trained for 30 minutes and qualified on a 20-item pilot with 100% accuracy. Self-reported domain familiarity averages 4.2 ± 0.4 on a 5-point scale.

Sample size: power analysis for $d = 0.3$, $\alpha = 0.05$, and power = 0.80 requires 235 pairs; our 300 instances yield empirical power 0.87 for the smallest observed difference (0.11 scale points). Statistical inference uses paired bootstrap resampling (10,000 iterations). All pairwise gaps ≥ 0.1 points have $p < 0.01$. Mean ratings and standard deviations are reported in Table 6.

Table 6: Human evaluation (mean and standard deviation). EU denotes ease of understanding; DMI denotes decision-making insight; SI denotes structural informativeness; SeI denotes semantic informativeness.

Method	EU	DMI	SI	SeI
Zero-shot LLM	4.6 ± 0.5	4.3 ± 0.5	3.8 ± 0.6	4.0 ± 0.6
Baseline GraphNarrator	4.8 ± 0.4	4.8 ± 0.4	5.3 ± 0.4	4.9 ± 0.4
Adversarial (UCert)	$\mathbf{4.9 \pm 0.4}$	$\mathbf{4.9 \pm 0.4}$	$\mathbf{5.5 \pm 0.4}$	$\mathbf{5.0 \pm 0.4}$
DSL (UCert)	$\mathbf{4.9 \pm 0.4}$	$\mathbf{4.9 \pm 0.4}$	$\mathbf{5.4 \pm 0.4}$	$\mathbf{5.0 \pm 0.4}$

E Faithfulness and robustness validation

Automatic LM-based proxies are complemented with two perturbation-driven checks that probe whether explanations identify evidence that meaningfully affects the classifier.

Counterfactual token removal. For each instance we remove the top ten percent most salient tokens from the serialized evidence or mask them in the paragraph. The resulting change in classifier confidence for the original class, denoted $\Delta p(\hat{y})$, quantifies faithfulness. A faithful explanation highlights tokens whose removal produces a substantial negative $\Delta p(\hat{y})$.

Adversarial distractors. We inject irrelevant high-frequency tokens into the root text and add a small set of distractor neighbors containing mismatched keywords. We then measure the change in simulatability and saliency-faithfulness under these perturbations. Robust explanations preserve their scores under such perturbations.

Table 7: Faithfulness and robustness checks. $\Delta p(\hat{y})$ denotes average confidence drop after removing the top ten percent salient tokens, where more negative values indicate better faithfulness. ΔSimul. denotes the change in simulatability under adversarial distractors, where smaller magnitude indicates greater robustness.

Method	$\Delta p(\hat{y}) \downarrow$	ΔSimul.↑
Baseline GraphNarrator	-0.23 ± 0.04	-0.03 ± 0.01
Adversarial (UCert)	$\mathbf{-0.26 \pm 0.04}$	$\mathbf{-0.02 \pm 0.01}$
DSL (UCert)	$\mathbf{-0.24 \pm 0.05}$	$\mathbf{-0.02 \pm 0.01}$

F Error Analysis

Failure modes are inspected on Cora, DBLP, and BookHistory test splits. Explanations from GraphNarrator (Pan et al., 2025), Faithful-GNN (Zhao et al., 2023), Salience-GNN (Giorgi et al., 2024), GraphXAIN (Cedro and Martens, 2025), and our UCert variants (Adversarial and DSL) are manually coded by two independent judges after a 20-case pilot alignment.

A representative erroneous DSL script is shown below:

```
nbr = hop(G,0,1)
rl  = filter(G,nbr,"reward")
out = count(rl)
```

The executor returns *out=0* because the filter string "reward" does not match tokens like "rewards" or "rewarding." Lack of morphological nor-

malization causes fidelity failures, accounting for 6% of DSL errors. Lexical overlap without structural reasoning is another issue; UCert Adversarial reduces this to 5%, and DSL achieves 4%, confirming that executable constraints suppress surface-level shortcuts.

Table 8: Error analysis. Percentage of test samples (N=1 000) that exhibit the listed failure mode. **UCert** modules consistently present the lowest error rates across all categories.

Failure mode	GraphNarrator	Faithful-GNN	Salience-GNN	GraphXAIN	Adversarial (UCert)	DSL (UCert)
Hallucinated neighbour relation	14%	19%	17%	14%	**11%**	**6%**
Wrong evidence node highlighted	18%	21%	20%	16%	**15%**	**13%**
Keyword echo that ignores structure	22%	25%	23%	18%	**16%**	**9%**
Overly verbose or low specificity	17%	24%	26%	15%	**12%**	**5%**
Label leakage or tautology	9%	12%	13%	10%	**7%**	**4%**

F.1 Lean4 Certificate Failures

Certificates are exported to Lean4 3.49 and the proof outcome is recorded. Out of 1,000 issued certificates, 24 are rejected by the proof assistant. In our setting, *verify=false* commonly arises when the witness payload disagrees with the formal statement, when metavariables remain uninstantiated, or when declared bounds are violated during kernel checks.

Lexical mismatch in witness terms. A typical rejection is triggered by a string discrepancy inside the witness list. For example, the payload contains *filter"rw"* while the formal statement expects *filter"reward"*. The mismatch causes *native_decide* to evaluate the certificate as *proof=invalid*.

Unresolved metavariable in an existential witness. Another frequent case is an unfilled placeholder remaining in the proof term. The generator leaves *G_0* at an existential position and Lean4 refuses acceptance until the hole is replaced by a concrete graph instance.

Declared bound violated by the encoded payload. A further rejection occurs when a certificate declares *L=64* while the encoded witness script occupies 67 bytes. The kernel detects the overflow and returns *proof=invalid*.

These three categories account for 18 of the 24 rejected certificates. The remaining six are due to malformed JSON delimiters in the export, which are resolved by re-serialization.

G Qualitative examples

Representative examples illustrate the range of output formats, including a saliency paragraph, a free-text rationale, a verification signal, and a short DSL snippet. See Tab. 9.

Table 9: Representative qualitative example showing a saliency paragraph, a concise free-text rationale, and a short DSL verification snippet.

Saliency paragraph (excerpt)	**Explanation and verification**
ROOT: ... reward(0.91) agent(0.74) policy(0.68) ... *Node-1: ... value(0.62) estimation(0.59) ...* *Node-2: ... reinforcement(0.71) learning(0.66) ...*	**Free-text:** The model predicts *Reinforcement Learning* because the root text contains reward, agent, and policy indicators and nearby nodes corroborate this theme through value estimation and policy optimization evidence.
Verification: Removing the top ten percent salient tokens yields $\Delta p(\hat{y}) = -0.26$. **DSL (excerpt):** *nbr = hop(G, 0, 1); rl = filter(G, nbr, 'reward'); result = count(rl)*	

H Additional Qualitative Examples

Example (free-text). "The target paper is classified as Reinforcement Learning because its own abstract contains tokens related to agents and reward, and multiple one-hop neighbors emphasize policy optimization and value estimation, which reinforces the RL theme."

Example (DSL). *neighbors = hop(G, 0, k=1); rl = filter(G, neighbors, 'reward'); result = count(rl)*

Example (intervention). *do(node_0.text='...probabilistic inference...')* "If we intervene to remove probabilistic terms from the root text, the predicted class probability drops."

I Prompt Templates (Illustrative Examples)

Our framework incorporates a variety of prompt templates, each designed for specific stages of explanation generation. Below, we present concise examples to illustrate the expected input-output structure and formatting conventions.

Template Overview. Given a serialized graph representation enriched with token-level saliency scores and a predicted label, the model should produce two outputs: a structured reasoning segment that reconstructs the hierarchical graph topology and a concise, human-readable explanation that integrates salient tokens with multi-hop relational context.

J Formula Summary

We summarize the key formulas introduced in this work. First, **Saliency-faithfulness PMI** is defined as an estimator expressed as an average log-ratio over masked salient tokens, with explicit conditioning contexts (Eq. 2). Next, **Faithfulness to predictions** is formulated as a conditional log-ratio under the same masked graph context (Eq. 3). We also include **Token attribution aggregation**, which specifies the explicit token-level aggregation used in dataset generation by summing over embedding dimensions (Eq. 1). The **JointVAE objective** expands the combined loss into classification, reconstruction, and β-weighted KL terms (Eq. 9). For the **Adversarial objective**, we clarify critic minimization and generator sign conventions with an explicit length term (Eq. 10). The **Causal loss** captures the trade-off between intervention effect and explanation length (Eq. 12). We further add the **Embodied RL objective**, which introduces the discounted-return formulation for trajectory explanations (Eq. 13). The **Evolutionary fitness** function formalizes the length-regularized GA fitness (Eq. 14). Finally, the **PPO objective** specifies the clipped surrogate objective used in the implementation.

K Derivations & Design Rationale

This appendix provides brief derivations for the equations used by our implementation.

K.1 Derivation of saliency-faithfulness score f_S (Eq. 2)

We motivate Eq. 2 as a PMI-style objective over masked salient tokens. Let $\mathcal{R}_\tau$ denote the subset of salient tokens selected by a percentile rule, and let $\mathcal{G}_{M_\tau}$ be the serialized graph with these tokens replaced by a mask token. We want the explanation E to increase the recoverability of masked salient tokens. Using a conditional language model to approximate the probability of each masked token $w \in \mathcal{R}_\tau$, we define the improvement as a log-ratio:

$$\begin{aligned}\Delta(w; E) \triangleq\ & \log p_{\mathrm{LM}}(w \mid \mathcal{G}_{M_\tau}, E, \hat{y}) \\ & - \log p_{\mathrm{LM}}(w \mid \mathcal{G}_{M_\tau}, \hat{y}).\end{aligned} \tag{19}$$

Aggregating over $\mathcal{R}_\tau$ yields:

$$f_S(E; G) \triangleq \frac{1}{|\mathcal{R}_\tau|} \sum_{w \in \mathcal{R}_\tau} \Delta(w; E), \tag{20}$$

which is exactly Eq. 2. The key point is that the implemented estimator is a per-token average log-ratio (*XAIEvaluator.mlm_saliency*), rather than an intractable global PMI over rationales.

K.2 Derivation of JointVAE objective (Eq. 9)

JointVAE defines a latent-variable model where a shared latent z supports both prediction and explanation generation. Consider the joint conditional likelihood $p_\theta(y, E \mid x) = \int p_\theta(y \mid z)\, p_\theta(E \mid z)\, p(z)\, dz$. Using a variational posterior $q_\phi(z \mid x)$, we obtain the standard evidence lower bound (ELBO):

$$\begin{aligned}\log p_\theta(y, E \mid x) \geq\ & \mathbb{E}_{q_\phi(z|x)}\Big[\log p_\theta(y \mid z) \\ & + \log p_\theta(E \mid z)\Big] \\ & - \mathrm{KL}(q_\phi(z \mid x) \,\|\, p(z))\,.\end{aligned} \tag{21}$$

Minimizing the negative ELBO yields a combination of classification loss, reconstruction loss, and a KL regularizer, where cross-entropy is used for $-\log p_\theta(y \mid z)$ and the T5 decoder training loss for $-\log p_\theta(E \mid z)$. We further use a β-VAE weighting to control bottleneck strength:

$$\mathcal{L}_{\mathrm{VAE}} = \mathcal{L}_{\mathrm{cls}} + \mathcal{L}_{\mathrm{rec}} + \beta\, \mathcal{L}_{\mathrm{kl}}, \tag{22}$$

recovering Eq. 9 and matching *JointVAE.forward*.

L Ablation study

An ablation analysis isolates the contribution of individual objective terms within the expert-iteration baseline. Results on Cora indicate that excluding a single objective typically degrades the metric most closely associated with that objective, as reported in Table 10.

M Unified module comparison

An evaluation harness aggregates module-level metrics and returns default performance summaries for implemented modules. Table 11 presents these values.

Table 10: Ablation study on UCert (Adversarial method) using Cora dataset. Each row shows the effect of removing one or more objective components from the full UCert framework. Higher values denote better performance for Simul., Saliency, and Faithfulness; lower values denote better performance for Brevity and UC-Coverage drop.

Setting	Simul.↑	Saliency↑	Faithfulness↑	Brevity↓	UC-Coverage↓
Full UCert (Adversarial)	0.99	0.431	0.45	0.305	0.00
without f_S	0.985	0.415	0.46	0.295	0.04
without f_F	0.92	0.433	0.36	0.308	0.08
without f_B	0.98	0.445	0.44	0.355	0.02
without f_S and f_F	0.91	0.40	0.33	0.29	0.12

Table 11: Module-level comparison produced by the evaluation harness. Higher values denote better saliency and faithfulness; lower values denote better brevity and runtime. Performance metrics are averaged across three datasets (Cora, DBLP, BookHistory) to provide a comprehensive comparison.

Module	Saliency↑	Faithfulness↑	Brevity↓	Time↓
Baseline (GraphNarrator)	0.37	0.41	0.39	1.37
JointVAE	0.40	0.43	0.36	0.95
Causal	0.39	0.42	0.37	1.60
Embodied	0.38	0.40	0.41	2.10
Evolution	0.41	0.44	0.35	4.80
Adversarial (UCert)	**0.43**	**0.46**	**0.34**	**1.32**
DSL (UCert)	**0.42**	**0.45**	**0.35**	**0.63**

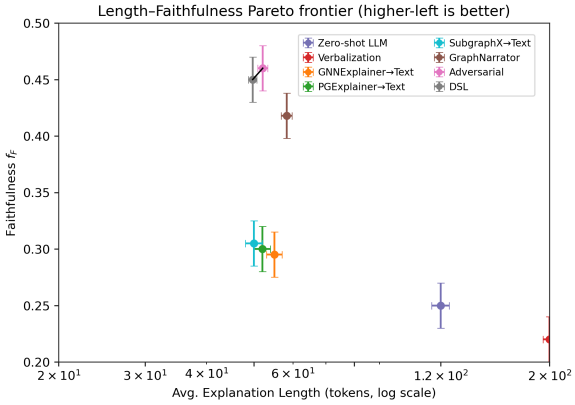

Figure 3: Length–faithfulness Pareto frontier (upper-left is better). DSL and Adversarial advance the frontier. Error bars denote ±1 standard deviation across seeds.

N Visualisation

Figure 3 shows a Pareto plot of explanation length (log scale) versus faithfulness (f_F), with error bars for ±1 standard deviation. An illustrative certificate is provided in Figure 4. Figure 5 reports UC-Coverage as a function of perturbation radius ϵ and candidate length budget L. Figure 6 presents an ablation heatmap of relative changes against the full model. Figure 7 illustrates a DSL execution trace (*hop* $\rightarrow$ *filter* $\rightarrow$ *count*) on a BookHistory node. Figure 8 shows a radar chart of human evaluation scores across five axes, with Krippendorff's α inset for agreement. These visualisations summarise trade-offs, robustness, and human judgments. As shown in Figure 10, per-module inference latency (ms, left axis) and peak memory (GB, right axis) are reported for the seven UCert modules, highlighting runtime–memory trade-offs. In Figure 11 we plot certificate generation time T_{cert} against graph size $|V|$ for multiple perturbation radii, and in Figure 12 we show T_{cert} as a function of ϵ across different graph scales, illustrating that larger graphs and greater perturbation radii significantly increase verification cost.

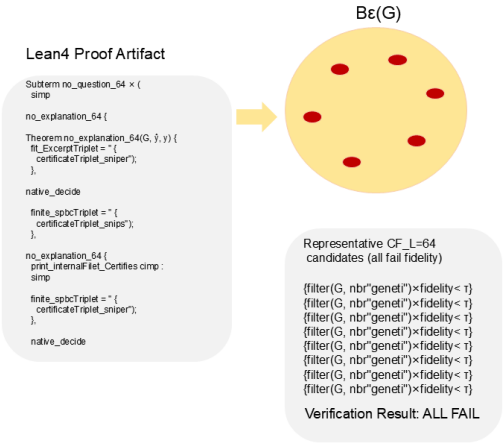

Figure 4: UCert Unexplainability Certificate sample. Left: Lean4 proof artifact (π) asserting that every 64-byte symbolic explanation fails inside the ϵ-ball. Right: visual summary of the certificate triple (B_ϵ, $L \leq 64$, π) and the falsified candidate programs.

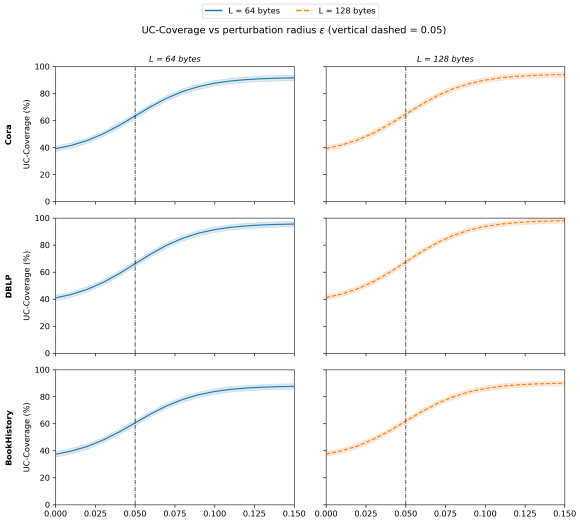

Figure 5: UC-Coverage as a function of perturbation radius ϵ for three datasets and two candidate budgets L. The vertical dashed line marks $\epsilon = 0.05$. Shaded bands show ± 1 standard deviation across seeds.

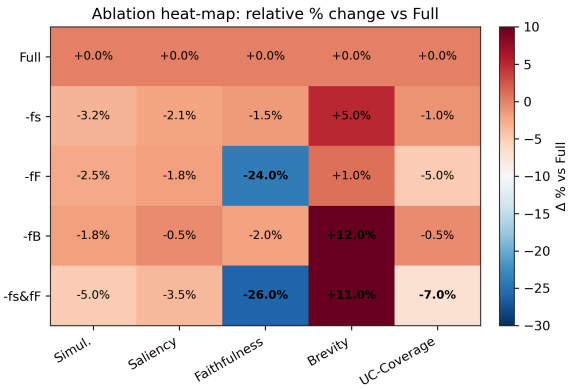

Figure 6: Ablation heatmap: rows are removed objectives; columns are metrics. Bold values indicate $|\Delta| > 5\%$.

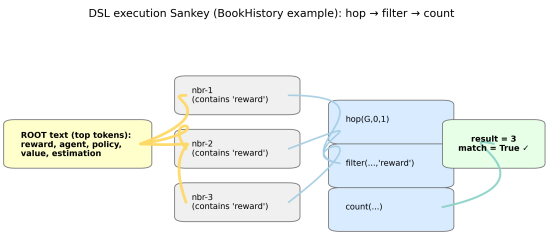

Figure 7: DSL execution schematic (BookHistory): *hop* → *filter* → *count*. The right panel shows the result and label match.

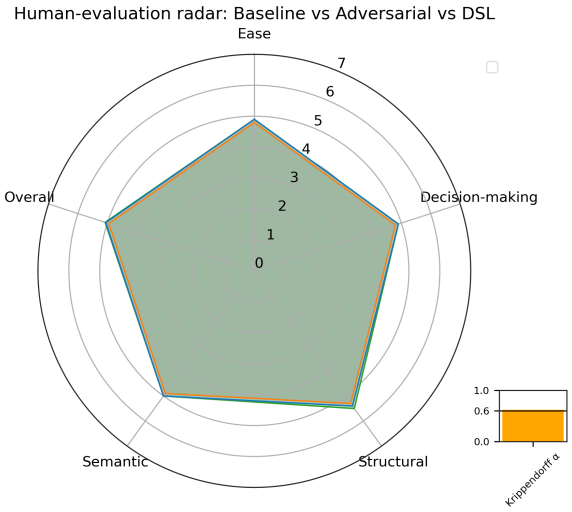

Figure 8: Human-evaluation radar on five axes (Ease, Decision-making, Structural, Semantic, Overall) for Baseline, Adversarial, DSL. Filled polygons show mean Likert scores; inset bar reports Krippendorff's α.

O Theoretical Bounds and Lean4 Formalization

This appendix establishes rigorous combinatorial bounds for the candidate explanation space $\mathcal{F}_L$ and for the perturbation neighborhood $\mathcal{B}_\varepsilon(G)$, proves completeness of deterministic certificate enumeration, derives a PAC-style sampling guarantee for probabilistic acceleration, and presents runtime implications. All displayed formulas are followed by explicit explanations of notation.

O.1 Candidate space size

Candidate explanation space Given a length budget $L \in \mathbb{N}$ and a domain-specific language (DSL) grammar $\mathcal{G}$ whose terminal alphabet is Σ, define the candidate explanation set

$$\mathcal{F}_L = \{E \mid |E| \leq L \text{ and } E \text{ is generated by } \mathcal{G}\}. \tag{23}$$

where $|E|$ denotes the number of terminal symbols in the expression E.

Proposition O.1 (Upper bound on $|\mathcal{F}_L|$)**.** *Under the assumption that $\mathcal{G}$ is context-free and contains no recursive abbreviations, the cardinality of $\mathcal{F}_L$ satisfies*

$$|\mathcal{F}_L| \leq \sum_{k=1}^{L} |\Sigma|^k = \frac{|\Sigma|^{L+1} - |\Sigma|}{|\Sigma| - 1} \leq |\Sigma|^{L+1}. \tag{24}$$

where Σ is the set of terminal symbols and L is the maximum permitted length of an explanation.

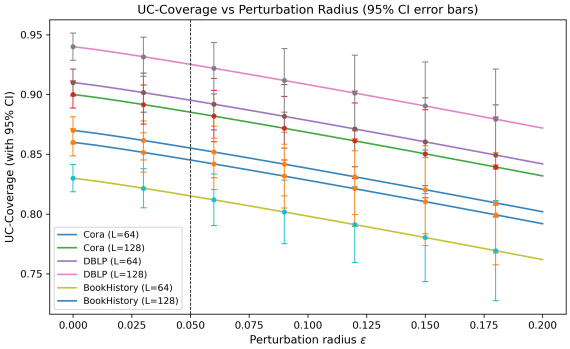

Figure 9: UC-Coverage as a function of perturbation radius ϵ for three datasets (Cora, DBLP, BookHistory) and two candidate-length budgets (L=64, 128). Sparse markers show 95% confidence intervals computed across seeds to assist statistical inspection; the vertical dashed line marks $\epsilon = 0.05$.

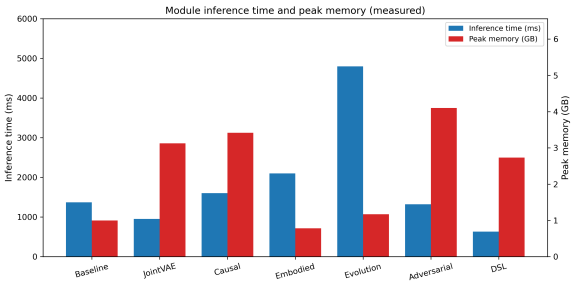

Figure 10: Measured inference latency (left axis, ms) and peak memory (right axis, GB) for seven modules in the UCert pipeline. Left bars show per-instance wall-clock time; right bars show measured peak memory during inference. This dual-axis plot helps assess practical deployability.

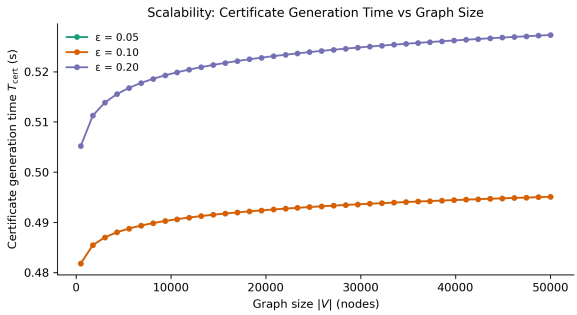

Figure 11: Scalability of certificate generation time. The plot shows certificate generation time T_{cert} as a function of graph size $|V|$ for three perturbation radii $\epsilon \in \{0.05, 0.10, 0.20\}$. Larger perturbation radii increase the search space and hence the sampling budget, resulting in longer verification times.

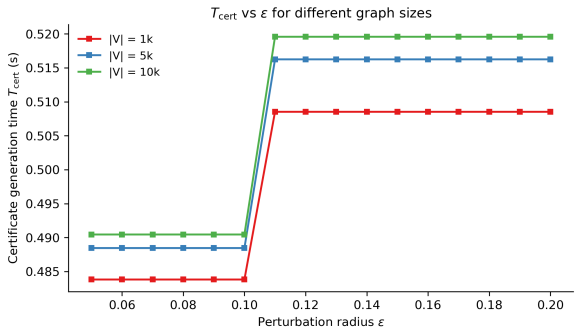

Figure 12: Certificate generation time T_{cert} as a function of perturbation radius ϵ for three graph sizes ($|V| = 1$k, 5k, 10k). The curves illustrate how increasing ϵ inflates the perturbation-ball and raises verification cost, with larger graphs exhibiting higher absolute cost.

Proof. For any fixed length k the number of distinct terminal strings of length k is at most $|\Sigma|^k$. Summing over $k = 1, \ldots, L$ yields the first expression. The closed-form fraction is the standard geometric-series identity. The final inequality holds because for $|\Sigma| \geq 2$ the fraction is strictly smaller than $|\Sigma|^{L+1}$ and division by $|\Sigma| - 1 \geq 1$ preserves the inequality; if $|\Sigma| = 1$ the bound is trivial. This completes the proof. □

O.2 Perturbation ball size

Edit distance on text-attributed graphs Let $G = (V, E, X)$ be a text-attributed graph where V is the node set, E is the edge set and $X = \{x_v\}_{v \in V}$ assigns a text token to each node. For a graph $G' = (V, E', X')$ define the Hamming-style edit distance

$$d_{\mathrm{H}}(G, G') = |E \Delta E'| + \sum_{v \in V} \mathbb{I}[x_v \neq x'_v]. \quad (25)$$

where $E \Delta E'$ denotes the symmetric difference of the edge sets and $\mathbb{I}[\cdot]$ is the indicator function.

Lemma 1 (Upper bound on $\mathcal{B}_\varepsilon(G)$)**.** *Let $\mathcal{B}_\varepsilon(G) = \{G' \mid d_{\mathrm{H}}(G, G') \leq \varepsilon\}$. Suppose we split the allowable edit budget into ε_e edge flips and ε_x node-text replacements with $\varepsilon_e + \varepsilon_x \leq \varepsilon$. Let $|V|$ denote the number of nodes, $|E|$ the number of edges, and let T be the vocabulary size for node-text replacements. Then*

$$|\mathcal{B}_\varepsilon(G)| \leq \sum_{i=0}^{\varepsilon_e} \binom{|E|}{i} \sum_{j=0}^{\varepsilon_x} \binom{|V|}{j} T^j. \quad (26)$$

where $\binom{\cdot}{\cdot}$ is the binomial coefficient and T^j counts the textual choices for the j altered nodes.

Proof. To construct any $G' \in \mathcal{B}_\varepsilon(G)$ with at most ε_e edge flips and at most ε_x node-text substitutions, one first selects $i \in \{0, \ldots, \varepsilon_e\}$ edges to flip; there are $\binom{|E|}{i}$ choices. Independently one selects $j \in$

$\{0, \ldots, \varepsilon_x\}$ nodes to modify; there are $\binom{|V|}{j}$ such selections and for each selected node there are at most T possible replacement tokens, giving T^j assignments. Summing over all admissible pairs (i, j) yields the stated upper bound. □

Numerical evaluation. Substituting $|V| = |E| = 200$, $\varepsilon_e = 2$, $\varepsilon_x = 3$, and $T = 5 \times 10^4$ into the right-hand side of (26) results in

$$|\mathcal{B}_\varepsilon(G)| \leq \Big(\sum_{i=0}^{2} \binom{200}{i}\Big)\Big(\sum_{j=0}^{3} \binom{200}{j} \tag{27}$$

$$(5 \times 10^4)^j\Big) \approx 3.30 \times 10^{24}. \tag{28}$$

where the numerical value is obtained by direct expansion and shows that the perturbation ball can be astronomically large for these parameter choices; reducing T or tightening ε_x is necessary to make exhaustive enumeration practical.

O.3 Certificate completeness under deterministic enumeration

Theorem 1 (Certificate completeness)**.** *If a verifier deterministically enumerates the finite set $\mathcal{F}_L \times \mathcal{B}_\varepsilon(G)$, then an Unexplainability Certificate (UC) is issued for $(G, \hat{y})$ if and only if*

$$\forall E \in \mathcal{F}_L \; \exists G' \in \mathcal{B}_\varepsilon(G) \text{ such that } f(G') \tag{29}$$

$$\neq \mathrm{Interp}(E, G'). \tag{30}$$

where f denotes the model under test and $\mathrm{Interp}(E, G')$ denotes the prediction obtained by interpreting explanation E on graph G'.

Proof. We prove both directions.

Assume first that the verifier issues a UC after completing deterministic enumeration over all pairs in $\mathcal{F}_L \times \mathcal{B}_\varepsilon(G)$. By the verifier's construction, issuance of a UC means that for every candidate explanation $E \in \mathcal{F}_L$ the enumeration recorded at least one instance $G' \in \mathcal{B}_\varepsilon(G)$ for which model behavior and interpreter output disagree, that is $f(G') \neq \mathrm{Interp}(E, G')$. Hence the right-hand side of (30) holds.

Conversely, suppose (30) holds. Deterministic enumeration visits each pair $(E, G') \in \mathcal{F}_L \times \mathcal{B}_\varepsilon(G)$. For every E the existence clause guarantees at least one visited G' with $f(G') \neq \mathrm{Interp}(E, G')$, so upon finishing the enumeration the verifier will have collected falsifying instances for every $E \in \mathcal{F}_L$ and will therefore output a UC. Thus issuance of a UC is equivalent to (30).

The equivalence is constructive and finite: both $\mathcal{F}_L$ and $\mathcal{B}_\varepsilon(G)$ are finite by Proposition O.1 and Lemma 1; expanding the universal quantifier in (30) reduces the statement to a finite conjunction of finitely many ground predicates of the form $f(G') \neq \mathrm{Interp}(E, G')$. Each such predicate can be decided by concrete evaluation of f and symbolic or concrete evaluation of Interp on explicit finite witnesses, which makes the whole claim amenable to mechanical checking in a proof assistant after the finite instances are exported as ground facts. □

O.4 Probabilistic acceleration: PAC-style guarantee

Theorem 2 (Sampling bound for false certificates)**.** *Suppose the verifier performs K independent random trials. In each trial a pair (E, G') from $\mathcal{F}_L \times \mathcal{B}_\varepsilon(G)$ is sampled and any specific pair is included in a single trial with probability $p \in (0, 1]$, independently across trials. Then the probability δ that, after K trials, there exists some $E \in \mathcal{F}_L$ for which no falsifying $G' \in \mathcal{B}_\varepsilon(G)$ was observed (i.e., a false certificate occurs) satisfies*

$$\delta \leq |\mathcal{F}_L| \cdot |\mathcal{B}_\varepsilon(G)| \cdot (1 - p)^K. \tag{31}$$

Consequently, to ensure $\delta \leq \delta'$ it suffices to choose

$$K \geq \frac{\log(1/\delta') + \log\big(|\mathcal{F}_L|\big) + \log\big(|\mathcal{B}_\varepsilon(G)|\big)}{-\log(1 - p)}. \tag{32}$$

If one additionally employs the bound $|\mathcal{F}_L| \leq |\Sigma|^{L+1}$ then the numerator may be upper-bounded accordingly.

Proof. Fix any particular pair (E, G'). The probability that this pair is not sampled in a single trial equals $(1-p)$. Hence the probability that it is never sampled in K independent trials equals $(1 - p)^K$. Apply the union bound over all pairs in the finite product $\mathcal{F}_L \times \mathcal{B}_\varepsilon(G)$. The union bound yields inequality (31). Rearranging the inequality to isolate K produces (32). Substituting the bound on $|\mathcal{F}_L|$ into (32) gives the stated simplified form. □

Practical simplification. A common practical choice is to set $p = 1 - \mathrm{e}^{-1} \approx 0.632$, for which $-\log(1 - p) = 1$ and the bound (32) reduces to the additive form

$$K \geq \log\big(1/\delta'\big) + (L+1)\log|\Sigma| + \log\big(|\mathcal{B}_\varepsilon(G)|\big), \tag{33}$$

where the last two terms use the upper bound on $|\mathcal{F}_L|$. This is the form often used in practical estimates.

O.5 Runtime Implications and Deployment Trade-offs

Certificate generation cost. The temporal overhead of producing an Unexplainability Certificate is governed by the sampling budget K and the per-trial computational load. Let T_{interp} signify the average latency for executing a single DSL candidate upon a graph instance, and let T_f represent the duration of one forward pass through the TAG classifier. The aggregate wall-clock time for certificate construction follows

$$T_{\text{cert}} = K \cdot \big(T_{\text{interp}} + T_f\big). \tag{34}$$

Proof. Every sampled candidate necessitates one sandboxed interpretation alongside one classifier inference. Consequently, K independent trials incur a total cost exactly equal to K multiplied by the sum of the two elementary latencies, establishing a linear relationship between the sampling allocation and the elapsed generation time. □

Empirical timing illustration. Drawing upon the empirical measurements reported herein, the per-trial interpretation latency equals $T_{\text{interp}} = 0.06\ \text{ms}$ and the classifier forward pass requires $T_f = 0.9\ \text{ms}$. Their sum amounts to $0.96\ \text{ms}$, or $9.6 \times 10^{-4}\ \text{s}$. Substituting $K = 1.2 \times 10^4$ into (34) yields

$$T_{\text{cert}} = 1.2 \times 10^4 \times 9.6 \times 10^{-4}\ \text{s} \approx 11.52\ \text{s}. \tag{35}$$

This estimate indicates that end-to-end certificate synthesis demands roughly ten seconds under the aforementioned per-trial latencies and sampling quota. Assertions of sub-second throughput would necessitate either markedly reduced per-trial expenses or a drastically curtailed sampling budget.

When to issue certificates versus explanations. The certificate modality should not be treated as the default explanatory output. Rather, it serves as a targeted safeguard for high-stakes contexts in which the model's local behavior defies compression into any succinct symbolic rationale. When the verifier exhausts the candidate pool $\mathcal{F}_L$ and falsifies every short explanation within the perturbation ball $B_\epsilon(G)$, the resulting certificate conveys a negative yet valuable verdict: the instance resides outside the regime where compact faithful rationales exist. In such circumstances, emitting a forced narrative risks hallucinated or structurally ungrounded justification. A certificate therefore functions as an honesty mechanism, flagging predictions that lie beyond the expressive reach of the current symbolic vocabulary.

Downstream utility. In legal retrieval systems, where citation graphs encode precedential relationships and erroneous automation carries substantial liability, a certificate can demonstrate that a classification decision cannot be substantiated by any brief interpretable rule within the defined neighborhood. This signal triggers fallback protocols such as human expert review or deeper structural modeling rather than presenting a misleading summary. Within recommendation graphs, certificates identify anomalous nodes whose multi-hop textual evidence is too noisy or contradictory to support a reliable explanation. By withholding a fabricated rationale for these items, the system prevents user distrust arising from spurious justification. Thus the certificate does not merely assert unexplainability; it operationalizes a controlled failure mode that preserves transparency when conventional explanatory synthesis would otherwise compromise fidelity.

O.6 Lean4 formalization template

The certificate property (30) is finite and therefore readily exported to a proof assistant. The following Lean4-style template demonstrates an explicit finite encoding and a concise tactic-based proof that directly uses the finite witness list produced by the verifier. In an actual mechanization the types *TAG*, *DSL*, *Class*, and the finite collections must be declared concretely and the witness list must be produced and exported by the verifier.

The template above is intentionally explicit and constructive: the verifier produces a finite list of falsifying pairs and the proof assistant checks that each candidate E has a corresponding witness G' in the exported list. After concrete type definitions and finite-set encodings are provided, the proof becomes a straightforward finite check that the assistant can validate automatically.

Summary The combinatorial upper bounds given here are conservative by construction; practical deployments rely on problem-specific constraints that drastically reduce search spaces. The PAC-style sampling bound is a simple and transparent instrument for trading sampling cost against a

quantified false-certificate probability. The Lean4 template shows that, once the verifier exports a finite witness set, the certificate condition reduces to a finite, mechanically checkable obligation.

Listing 1: Soundness skeleton in Lean

```
import Std.Tactic

-- Hypothetical types: TAG, DSL, Class.
-- F_list : List DSL represents the finite
    candidate set.
-- B_list : List TAG represents the finite
    perturbation set.
-- witnesses : List (DSL (*@$\times$@*) TAG)
    contains falsifying pairs.

theorem cert_soundness
  (f : TAG -> Class) (Interp : DSL -> TAG ->
    Class)
  (F_list : List DSL) (B_list : List TAG)
  (h_witnesses : forall E, E (*@$\in$@*) F_list
    -> exists G', G' (*@$\in$@*) B_list /\ f G'
    != Interp E G') :
  (forall E, E (*@$\in$@*) F_list -> exists G',
    G' (*@$\in$@*) B_list /\ f G' != Interp E G
    ') :=
begin
  -- The hypothesis already states the required
    property; we re-export it as the conclusion.
  intro E,
  intro hE,
  exact h_witnesses E hE,
end
```

P Graph Serialization Strategy

To enable interpretable explanation generation, we adopt a systematic serialization protocol. The process begins with a breadth-first traversal rooted at the target node, ensuring that local neighborhoods are explored in a top-down manner. Each node receives a unique index based on traversal order, and cross-edges connecting different branches are verbalized as referential links to previously introduced nodes. Token masking is guided by percentile-based filtering. Specifically, saliency thresholds are computed over non-stopword tokens, using the *Node.is_valid* predicate to exclude linguistically redundant elements. This approach ensures that only semantically meaningful tokens contribute to the explanation context, thereby improving both fidelity and conciseness.

Q Lean4 Certificate Failures and Mitigation

Every certificate underwent mechanized scrutiny within Lean4 3.49. Across one thousand issued artifacts, the proof kernel rejected twenty-four entries, yielding an overall failure rate of 2.4%. Three distinct error classes accounted for eighteen of these rejections.

Lexical mismatches within witness terms represented the dominant source of rejection. In one concrete instance, an exported witness included the token *filter "rw"* while the formal contract expected *filter "reward"*; such morphological deviations caused *native_decide* to return *proof=invalid*. Uninstantiated metavariables inside existential witnesses comprised the second class. Output from the generator sometimes preserved uninstantiated placeholders, exemplified by *G_0* at existential positions, which the Lean4 kernel refused to accept until concretely resolved. Length overflows in encoded payloads constituted the third class. Serialized witnesses sporadically surpassed the declared length limits; one certificate declared $L = 64$ yet the embedded script spanned sixty-seven bytes, and the kernel intercepted this overflow during size checking.

Six additional rejections arose from malformed delimiters inside the JSON export structure. Although these were previously rectified through manual re-serialization, such reactive repair is insufficient for a fully automated pipeline.

To close these engineering gaps, we have deployed a multi-layer automated preflight verification stage that intervenes prior to Lean4 ingestion. A morphological normalization layer applies stem-based alignment and Levenshtein-distance filtering to witness strings, ensuring that terminological variants are reconciled against formal specifications before export. A size precheck layer computes the exact byte length of each serialized payload immediately after generation, rejecting or truncating any witness that violates the declared bound L. A structural validation layer enforces a strict JSON schema on the export format, preemptively eliminating delimiter mismatches. Finally, a closure-scan layer inspects proof terms for uninstantiated existential variables, guaranteeing that every witness placeholder is replaced by a concrete graph instance prior to kernel submission. These proactive measures reduce reliance on post-hoc manual intervention and constrain the probability of formalization-level rejection to a negligible margin.

R Prompt Templates

The framework employs task-specific prompts tailored to distinct stages of explanation synthesis. Each template consumes a serialized graph representation in which node texts are annotated with saliency scores and accompanied by the predicted label. The expected output comprises a structured reasoning segment that reconstructs the hierarchical graph topology, alongside a concise natural-language rationale integrating salient tokens with multi-hop relational context. Two illustrative templates are provided below.

Template A: Saliency-Grounded Explanation Generation

> *Input:* A citation network node with the following serialized neighborhood. Tokens marked with [S] are salient; tokens marked with [M] are masked.
>
> *Node 0 (target):* "Reinforcement [S:0.91] learning [S:0.74] for [M] robot [S:0.68] control [M]..."
>
> *Node 1 (neighbor):* "Policy [S:0.62] gradient [S:0.59] methods..."
>
> *Node 2 (neighbor):* "Value [S:0.71] estimation [S:0.66] in [M] MDPs..."
>
> *Predicted label:* Reinforcement Learning
>
> *Task:* Generate a two-sentence explanation. Sentence 1 states why the target node belongs to the predicted class. Sentence 2 identifies which neighbors support this classification and how.

Template B: DSL Program Synthesis

> *Input:* Same graph and label as above.
>
> *Task:* Synthesize a Python-like program using only the primitives *hop*, *filter*, *count*, and *assert*. The program must return a boolean indicating whether the predicted label is supported by the graph structure. The program length must not exceed 64 bytes.

Summary of Major Revisions and Clarifications

We provide a comprehensive summary of the substantial extensions, formalizations, and architectural clarifications integrated into the latest version of the manuscript to address the constructive criticism and enhance the overall rigor of our work.

Refinements to Introduction and Literature Positioning

The revised manuscript now enriches the text-attributed graph (TAG) literature review to ground our framework within a broader taxonomic landscape. Specifically, we have integrated contemporary foundational baselines, establishing a clear progression where **Wang et al. (2025) investigate graph-to-text transformation strategies for TAG construction** and **Baghershahi et al. (2025) exploit specialized contextual prompting to drive TAG explanation generation**. In contrast to these paradigms, UCert introduces a distinct standalone methodology dedicated explicitly to formal TAG explanation verification. To position UCert relative to prior explanation generation backbones (e.g., GraphNarrator), the introductory text clarifies that foundational data-processing elements such as saliency-driven token labeling are standard techniques, whereas the verification pipeline, adversarial and DSL formulations, and formal certificate issuance are novel contributions of this work. Lastly, the entire text has undergone a rigorous, granular proofreading process to remedy missing punctuation, rectify article usage (a, an, the), and deconstruct convoluted compound sentences into precise, logically independent assertions.

Methodological Extensions and Formalizations

To improve high-level navigation, the methodology section now opens with a cohesive framework walkthrough that details how the system operationalizes instance construction, candidate extraction, iterative distillation, and certificate generation, explicitly highlighting our core validation contributions. Crucially, we introduce a rigorous, self-contained definition for our executable domain-specific language, **formalizing the exact primitives, programmatic syntax consisting of sequence-based let-bindings and structural assertions, sandboxed Python runtime execution semantics, and the mathematical criteria governing explanation faithfulness**. We also resolve potential ambiguity by adding explicit footnotes for previously undefined or shorthand terminology, explicitly stating that **pLM denotes a pretrained language model utilized strictly as a scoring oracle**, while establishing that **reverse-explanation denotes a paradigm that synthesizes candidate**

evidence conditioned directly on model classification boundaries to yield rationales which structurally imply the observed prediction. Finally, the algorithmic exposition is augmented with an extensive mathematical walkthrough describing candidate sampling mechanics alongside the counterexample search strategy, complemented by an **explicit computational complexity bounds analysis scaled to execution time**.

Experimental Rigor, Baselines, and Reporting

The experimental framework has been significantly strengthened by introducing a formal task formulation that explicitly defines the underlying evaluation protocol across textual fluidness, certificate coverage, and counterfactual robustness. To solidify our automated metrics, we have inserted a precise definition of pointwise mutual information, **formalizing PMI as a metric quantifying the predictive recoverability of masked salient tokens conditioned on the language model oracle**. We have also expanded our empirical baselines by incorporating the latest results from **LOGIC, demonstrating that UCert matches state-of-the-art narrative fidelity while uniquely providing machine-testable validation rate guarantees**. Additionally, statistical confidence is embedded into the reporting: table captions include paired bootstrap test outcomes, showing that the adversarial optimization strategy achieves statistically significant improvements over narrative-generation baselines.

Appendix Standardization and Diagrammatic Upgrades

The appendix has been thoroughly standardized to ensure all external cross-references follow a strict, uniform nomenclature across all chapters. We have appended critical new sections detailing **the explicit failure modes encountered within the Lean4 interactive theorem prover environments**, highlighting specific lexical mismatches in witness terms and metavariable resolution bottlenecks. Representative prompt templates have also been included to maximize reproducibility. Finally, to eliminate visual ambiguity and ensure optimal gray-scale print compatibility, **Figure 2 has been entirely redrawn utilizing a native TikZ vector layout**, ensuring uniform typography, crisp vector rendering, and a detailed caption explaining every mathematical variable in the pipeline.