GraphNarrator: Generating Textual Explanations for Graph Neural Networks

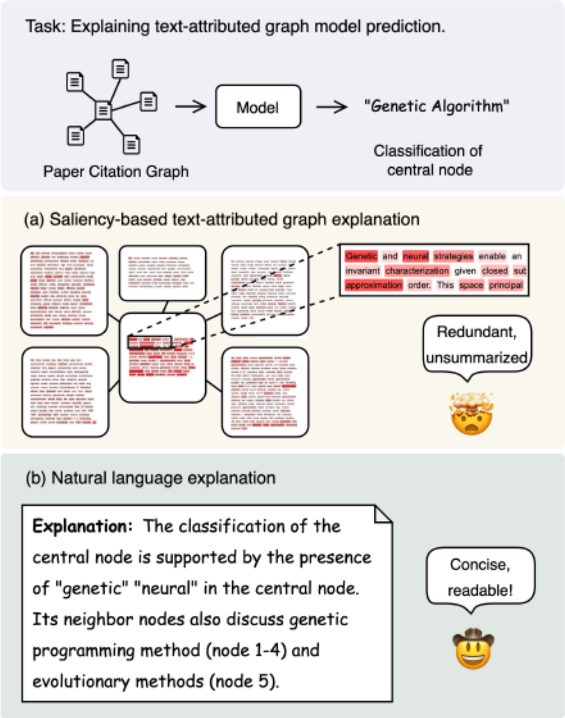
Bo Pan[*], **Zhen Xiong**[*], **Guanchen Wu**[*], **Zheng Zhang**, **Yuntong Hu**, **Yifei Zhang**, **Liang Zhao**[†]
Emory University, Atlanta, USA
{bo.pan, guanchen.wu, liang.zhao}@emory.edu, xiongzhen0711@gmail.com

Abstract

Graph representation learning has garnered significant attention due to its broad applications in various domains, such as recommendation systems and social network analysis. Despite advancements in graph learning methods, challenges still remain in explainability when graphs are associated with semantic features. In this paper, we present GraphNarrator, the first method designed to generate natural language explanations for Graph Neural Networks. GraphNarrator employs a generative language model that maps input-output pairs to explanations reflecting the model's decision-making process. To address the lack of ground truth explanations to train the model, we propose first generating pseudo-labels that capture the model's decisions from saliency-based explanations, then using Expert Iteration to iteratively train the pseudo-label generator based on training objectives on explanation quality. The high-quality pseudo-labels are finally utilized to train an end-to-end explanation generator model. Extensive experiments are conducted to demonstrate the effectiveness of GraphNarrator in producing faithful, concise, and human-preferred natural language explanations.

1 Introduction

Deep learning on graphs, especially Graph Neural Networks (GNNs) (Kipf and Welling, 2016; Hamilton et al., 2017), has become the standard and dominant technique for various graph-related tasks due to its expressive power in capturing structured features useful for prediction and representation learning. In many real-world scenarios, graph nodes and edges are accompanied by textual features, either inherently textual as in Text-Attributed Graphs (TAGs) (Zhou et al., 2019; Yang et al., 2021) or described using semantic terms and numerical expressions. For example, in e-commerce graphs of recommendation systems, products can be associated with textual descriptions; in molecular graphs, atoms and bonds can be described by textual tokens such as element names or properties. These textual descriptions offer rich semantic cues that can augment graph learning (He et al., 2023; Pan et al., 2024; Hu et al., 2024). Despite the dominant performance of current GNNs across a range of tasks, their internal decision processes remain largely opaque, which hampers their adoption in high-stakes applications. In this work, we aim to explore the right explanation modality for graphs with revolutionary tradeoff between conciseness, fidelity, and readability.


Task: Explaining text-attributed graph model prediction.
Paper Citation Graph
Model
"Genetic Algorithm"
Classification of central node
(a) Saliency-based text-attributed graph explanation
Genetic and neural strategies enable an invariant characterization given closed sut approximation order. This space principal
Redundant, unsummarized
(b) Natural language explanation
Concise, readable!

Explanation: The classification of the central node is supported by the presence of "genetic" "neural" in the central node. Its neighbor nodes also discuss genetic programming method (node 1-4) and evolutionary methods (node 5).

Figure 1: Illustration of saliency-based graph explanation and natural language graph explanation.

Explainability for graph learning have received a large amount of attention, with methods usually by providing node- or edge-level importance scores

[*] These authors contributed equally.
[†] Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 23–42
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

(Ying et al., 2019; Vu and Thai, 2020; Luo et al., 2020), but they lack the ability to explain the semantic information in graphs since the node- and edge-level importance scores cannot include any information of the text features. Some methods (Ying et al., 2019; Štrumbelj and Kononenko, 2014; Bach et al., 2015) can also provide node feature-level explanation, and thus can generate token importance scores for textual features on nodes. However, graph predictions are often made based on a subgraph with many nodes and their associated texts, such token importance-based explanations are hardly human-understandable as they are redundant and not integrated, as illustrated in Fig. 1 (a). The limitation and need for human understandability necessitates our research on natural language explanations for GNNs, which is expected to be summative, concise, and readable, as one example shown in Fig. 1 (b).

In this work, we present GraphNarrator, the first method to generate natural language explanations for GNNs. GraphNarrator is a model-agnostic method to learn a mapping from graph model input-output pairs to textual explanations. To generate textual explanations, since external language models have no knowledge of the model's internal decision-making process to generate high-quality explanations, it is necessary to fine-tune the model with high-quality explanation labels. In real-world scenarios, it is impractical to have annotated ground truth data for explaining model behaviors, therefore we propose graph explanation verbalization to prompt LLMs with saliency-based explanations to generate natural language explanation pseudo-labels. To continuously improve the quality of pseudo-labels, we propose three training objectives related to faithfulness and brevity, and iteratively fine-tune the pseudo-label generator model with these objectives with expert iteration. Finally, the generated pseudo-labels are used to train an end-to-end explainer model, which serves as our end-to-end explanation generation model.

2 Related Work

2.1 Explainability of Graph Neural Networks

GNNs have been widely adopted in fields such as social networks, molecular chemistry, and financial systems, yet their interpretability remains a significant challenge. Existing GNN explanation methods are generally categorized into instance-level and model-level approaches. Model-level methods seek to provide a broad understanding of GNN decision behaviors independent of specific inputs (Zhang et al., 2022; Yuan et al., 2020), while instance-level methods focus on explaining individual predictions by identifying important features that influence the model's decisions. These methods include gradients/features-based approaches (Baldassarre and Azizpour, 2019), perturbation-based methods (Ying et al., 2019; Luo et al., 2020), decomposition-based techniques (Baldassarre and Azizpour, 2019; Schwarzenberg et al., 2019) and surrogate model-based methods (Huang et al., 2022). While significant progress has been made in improving the explainability of GNNs, current instance-level methods are limited to providing feature importance-based explanations, which is difficult for humans to understand when the input comes to TAGs. To our best knowledge, no efforts have been made to generate natural language explanations of graph models.

2.2 Natural Language Explanation

The traditional explanation methods for NLP, such as feature importance and saliency maps (Lei et al., 2016; Yu et al., 2019), often fall short in providing human-interpretable insights, motivating the development of more intuitive approaches like natural language explanations (Cambria et al., 2023). These methods provide textual justifications for model predictions, aiming to bridge the gap between model decisions and human understanding (Camburu et al., 2018; Rajani et al., 2019; Narang et al., 2020). A notable example is self-explanation models, where models predict labels while simultaneously generating explanations in natural language (Wiegreffe et al., 2020; Zhang et al., 2024; Liu et al., 2023, 2024). With the advent of LLMs, Chain-of-Thought prompting (Wei et al., 2022) and zero-shot reasoning (Kojima et al., 2022) have enhanced their self-explanation abilities by generating coherent, step-by-step explanations. Additionally, LLMs' role as explainers for both their own predictions and other models' outputs has significantly expanded (Kroeger et al., 2023; Gat et al., 2023; Martens et al., 2023). Recent research also points out that LLM-based explanations do not necessarily faithfully reflect the internal model behaviours (Agarwal et al., 2024; Parcalabescu and Frank, 2024).

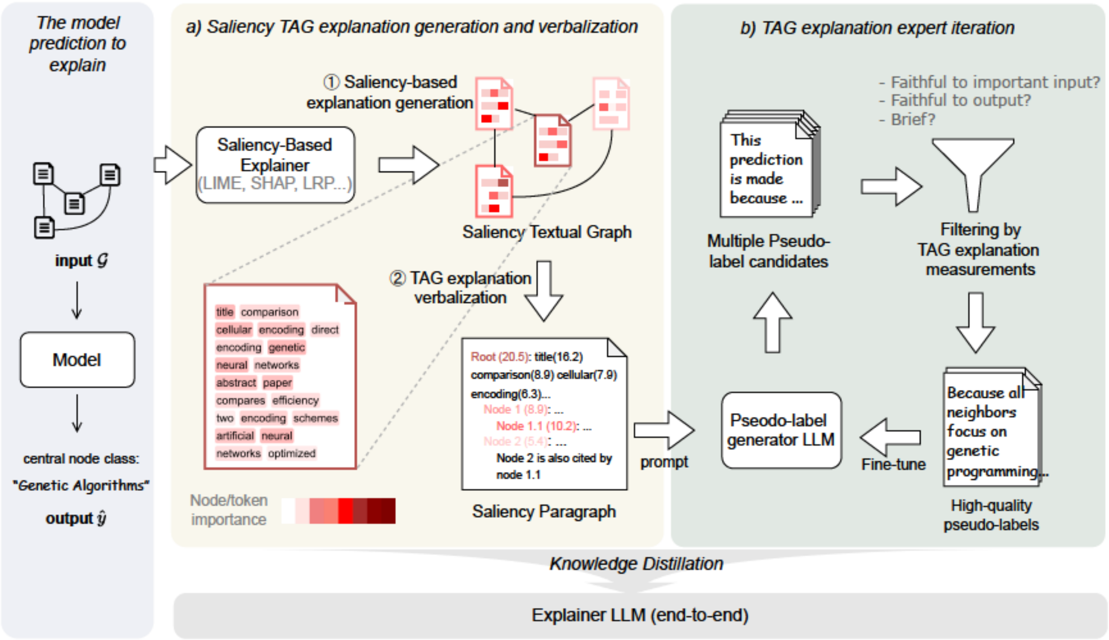

The model prediction to explain
input $\mathcal{G}$
Model
central node class: "Genetic Algorithms"
output $\hat{y}$
a) Saliency TAG explanation generation and verbalization
① Saliency-based explanation generation
Saliency-Based Explainer (LIME, SHAP, LRP...)
Saliency Textual Graph
② TAG explanation verbalization
title comparison cellular encoding direct encoding genetic neural networks abstract paper compares efficiency two encoding schemes artificial neural networks optimized
Root (20.5): title(16.2) comparison(8.9) cellular(7.9) encoding(6.3)...
Node 1 (8.9): ...
Node 1.1 (10.2): ...
Node 2 (5.4): ...
Node 2 is also cited by node 1.1
Saliency Paragraph
Node/token importance
prompt
b) TAG explanation expert iteration
This prediction is made because ...
Multiple Pseudo-label candidates
- Faithful to important input?
- Faithful to output?
- Brief?
Filtering by TAG explanation measurements
Because all neighbors focus on genetic programming...
High-quality pseudo-labels
Fine-tune
Pseudo-label generator LLM
Knowledge Distillation
Explainer LLM (end-to-end)

Figure 2: An illustration of GraphNarrator. A pseudo-label generator model is first trained to provide pseudo-labels, which are used for knowledge distillation to an LLM as an end-to-end explainer. (a) GraphNarrator first generates saliency-based graph explanations, then verbalizes them into a documented form (*Saliency Paragraph*) for easier understanding of LLMs, and feeds them to LLMs to generate initial natural language explanation pseudo-labels. (b) We propose the graph explanation expert iteration procedure to iteratively improve the pseudo-label generator LLM with three objectives related to faithfulness and brevity.

3 Problem Formulation

In this work, we delve into the task of explaining predictions for GNNs with natural languages. Formally, a Text-Attributed Graph (TAG) can be represented as $\mathcal{G} = (\mathcal{V}, A, \mathcal{X})$, where $\mathcal{V} = \{v_0, v_1, ..., v_{N-1}\}$ is a set of N nodes, $A \in \{0,1\}^{N \times N}$ is the adjacency matrix, and $\mathcal{X} = \{x_0, x_1, ..., x_{N-1}\}$ is the set of texts where $x_k = (t_{k,0}, t_{k,1}, ..., t_{k,n_k^{(t)}})$ is a sequence of tokens associated with node $v_k \in \mathcal{V}$. A TAG model f is a model that can make predictions on TAGs by $\hat{y} = f(\mathcal{G})$, where $\hat{y}$ is the model output.

Given a text-attributed graph $\mathcal{G}$ and a trained TAG model f, the goal is to learn a mapping $g : (\mathcal{G}, \hat{y}) \to E$ to generate a paragraph of text E to explain the decision-making process of $\hat{y} = f(\mathcal{G})$. The generated explanation E should faithfully explain the reason for model predictions and be friendly for humans to understand.

4 Proposed Method: GraphNarrator

The overall framework of GraphNarrator is illustrated in Figure 2. The high-level idea is we first train a *Pseudo-Label Generator LLM* to generate high-quality explanation labels, then the generated pseudo-labels are used to fine-tune an *Explainer LLM*, which serves as a post-hoc explainer, the product of our framework. Due to the lack of ground truth explanation labels, we first use a saliency-based explainer to attain *Saliency Textual Graph* explanation, then verbalize it to the form of a *Saliency Paragraph*, and use it as a hint of the important regions for the *Pseudo-Label Generator LLM* (Sec. 4.1). Then the *Pseudo-Label Generator LLM* iteratively self-improves via Expert Iteration based on our proposed three training objectives for quantifying explanation quality (Sec. 4.2). Finally, the generated high-quality pseudo-labels from the fine-tuned *Pseudo-Label Generator LLM* are used to train the end-to-end *Explainer LLM* (Sec. 4.3). The details of our proposed GraphNarrator framework will be introduced in the following.

4.1 Saliency-based Graph Explanation Generation and Verbalization

We first construct a *Saliency Paragraph* to pass the semantics and structural information of TAG data as well as the saliency information of the model decision, as a hint for the *Pseudo-Label Generator LLM*. As illustrated in Fig. 2 (a), specifically, a saliency-based explainer is first adopted to get the *Saliency Textual Graph* explanations, and then we verbalize the graph explanation into textual forms based on BFS and hierarchical organizing. Finally, we prompt them to the *pseudo-label generator LLM* to get the initial explanation of pseudo-label candidates. Such a procedure will be introduced in details as follows.

Saliency-based explanation generation. As shown in Fig. 2 (a), the saliency-based explanations are generated by a feature importance-based post-hoc explainer. They are represented in the form of the importance score of each node and token. An example is illustrated in Fig. 2 (a) as *Saliency Textual Graph*, where the red color from light to dark denotes the importance of nodes and tokens. Note that here the saliency-based explainer can be based on various methods that can generate post-hoc explanations for a TAG model, including widely used model-agnostic explanation methods including LRP, Input×Grad, Saliency, etc. This makes GraphNarrator a model-agnostic framework for explaining various TAG learning model architectures.

Saliency textual graph verbalization. Since the generated *Saliency Textual Graph* is graph-structured data, it is vital to transform it into a form that LLMs are easier to understand. Therefore, we propose saliency textual graph verbalization, to transform the saliency-based graphs explanation into a document-like *Saliency Paragraph* to pass the structural, semantic and feature importance information to LLMs.

In a TAG model prediction, the structure of a central node and its k-hop salient nodes can be represented as an ego graph, with the node itself as the root. Using Breadth-First Search (BFS), this ego graph can be decomposed into a tree structure. During the BFS process, we prune unimportant nodes by identifying nodes with no tokens whose saliency scores are higher than a threshold. Then we adopt a Pre-Order Traversal (first visit the root, then the left subtree, then the right subtree) to organize the tree structure into a hierarchical saliency paragraph, maintaining the hierarchical structure of the k-hop ego graph. In this Saliency Paragraph, each node's text is represented as a section, and its successor nodes are represented as subsections. Note that when converting the ego graph to a BFS tree, there can be a set of *cross-edges* that connect nodes in different branches. We verbalize them by adding reference sentences in the text of source nodes, pointing to the sections representing their respective destination nodes. This ensures that the saliency paragraph faithfully reflects the ego graph's structure.

When organizing the *Saliency Paragraph*, we attach the saliency scores of nodes and tokens to prompt the *Pseudo-Label Generator LLM* with the feature importance information. For tokens in nodes, we attach their importance score after the tokens to prompt with the importance information, in the form of `token(score)`, without perturbing the semantic order of tokens. An example of constructing the *Saliency Paragraph* is given in Figure 3.

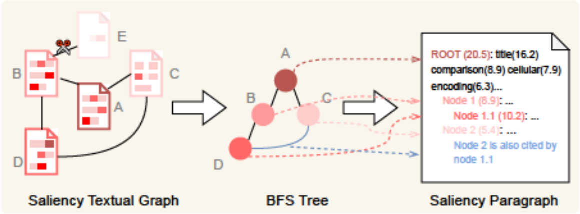


Figure 3: Illustration of graph explanation verbalization. The blue edge denotes a cross-edge.

4.2 TAG Explanation Expert Iteration for Explanation Self-Improvement

Initial explanations can be attained by simply passing the generated *Saliency Paragraph* to an LLM to generate more summarized words. However, such a zero-shot generation manner cannot ensure the explanation quality and may still lead to unfaithful or redundant textual explanations. To quantify the explanation quality, we propose Information-Theoretic TAG Explanation Measurements that quantify the text explanations' faithfulness and brevity. To effectively optimize such goals without ground truth labels, we propose TAG Explanation Expert Iteration framework that iteratively conducts *text explanation quality measuring*, *high-quality text explanation selecting*, and *text explanation updating*, as illustrated in Fig. 2 (b).

4.2.1 Information-Theoretic TAG Explanation Measurements

High quality model explanations should be faithful to the model decision process, and friendly for humans to understand. Therefore, we propose our TAG explanation measurements, quantify the explanation quality with faithfulness to important inputs, faithfulness to outputs, and brevity.

Faithfulness to important inputs. The explanation is expected to include enough necessary information about the rationale of the model prediction. Suppose there is a true rationale $\mathcal{R}$, then the faithfulness can be measured by the Pointwise Mutual Information (PMI) between the generated explanation E and $\mathcal{R}$, namely,

$$f_S = \text{PMI}(E, \mathcal{R}) = \log \frac{P(\mathcal{R}|E)}{P(\mathcal{R})} \tag{1}$$

However, estimating $P(\mathcal{R})$ and $P(\mathcal{R}|E)$ is intractable due to the big space of $\mathcal{R}$ to explore. Therefore, we formulate $\mathcal{R}$ as the subset of important nodes and tokens in the n the input graph $\mathcal{G}$ (the serialization of $\mathcal{G}$ can be found in Appendix B). We propose to mask the important tokens in nodes of $\mathcal{G}$, and turn this problem into a tractable masked token prediction problem with a pretrained language model. Therefore, $P(\mathcal{R})$ and $P(\mathcal{R}|E)$ can be estimated with $P(\mathcal{R}|\mathcal{G}_M)$ and $P(\mathcal{R}|\mathcal{G}_M, E)$, where $\mathcal{G}_M$ denotes the remaining part in S after masking $\mathcal{R}$ from $\mathcal{G}$. Moreover, we also need to consider the fact that not all the tokens in $\mathcal{R}$ are equally important, so we want to prioritize the faithfulness to those most important ones, though the threshold of being important is unknown. Therefore, we sample different thresholds τ in each iteration to give the model more flexibility to learn the importance. Then the above PMI can be written as

$$\begin{aligned} f_S =& \log \frac{P(\mathcal{R}|E)}{P(\mathcal{R})} \\ \approx& \int_0^1 P(\tau) \cdot \log \frac{P_{MLM}(\mathcal{R}_\tau|\mathcal{G}_{M_\tau}, E)}{P_{MLM}(\mathcal{R}_\tau|\mathcal{G}_{M_\tau})} \mathrm{d}\tau \end{aligned} \tag{2}$$

where τ denotes the ratio of tokens in $\mathcal{G}$ to be considered as important (e.g. $\tau = 0.1$ means $\mathcal{R}_\tau$ includes the tokens with top 10% high saliency scores in $\mathcal{G}$) , and $P(\tau)$ is the distribution of sampling τ, which can be implemented by any distribution that focuses on different thresholds (e.g. the uniform distribution from 0 to 0.3), $\mathcal{R}_\tau$ and $\mathcal{G}_{M_\tau}$ are the masked rationale $\mathcal{R}$ and the remaining text under the threshold τ (an illustration of constructing G_{M_τ} can be found in Appendix B).

Faithfulness to predictions. In addition to faithfulness to important inputs, we also encourage the generated explanations to be faithful to the outputs. Similarly, for faithfulness to predictions, we leverage the PMI between explanation E and the predicted label $\hat{y}$ as a measurement as

$$f_F = \text{PMI}(E, \hat{y}) = \log \frac{P(\hat{y}|E)}{P(\hat{y})} \tag{3}$$

where $\hat{y}$ denotes the textual form of the output prediction. The calculation of $P(\hat{y}|E)$ and $P(\hat{y})$ is also implemented with a pre-trained language model.

Brevity. Since the above objective of faithfulness encourages E to be informative, which may result in generating long and redundant explanations, which bring difficulty for humans understanding. Therefore, we also encourage the generated explanation E to be concise, as

$$f_B = \frac{|E|}{|\mathcal{G}|} \tag{4}$$

where $|E|$ and $|\mathcal{G}|$ denotes the length of E and $\mathcal{G}$.

Combining these measurements, we are essentially doing a multi-objective optimization problem, where we maximize f_S, f_F, and minimize f_B, which serves as our overall objective and be optimized with the following TAG expert iteration framework.

4.2.2 TAG Explanation Expert Iteration

To effectively optimize the objectives to improve the *Pseudo-Label Generator LLM*, we propose a TAG explanation iterative training method based on Expert Iteration (Dong et al., 2023; Gulcehre et al., 2023), as shown in Fig. 2 (b). Specifically, the training is composed of a closed loop of *TAG explanation quality measuring*, *high-quality TAG explanation selecting*, and *TAG explanation updating*, as introduced in details as follows:

(1) TAG explanation quality measuring. Aligned with our training objectives, we calculate the scores f_S, f_F and f_B of the generated explanation pseudo-label E based on Eq. 2, Eq. 3 and Eq. 4.

(2) High-quality TAG explanation selection. Among all generated explanations, a subset of high-quality explanations is filtered from all candidates with customizable criteria to balance f_S, f_F, and f_B, such as weighted sum and top-k.

(3) TAG explanation updating. The selected high-quality explanations are used to fine-tune the *Pseudo-Label Generator LLM*. Then the model generates a new batch of explanation candidates, and it goes back to step (1).

Such three steps form a closed loop, allowing us to iteratively increase the performance of the model. Finally, we got the *Pseudo-Label Generator LLM* fine-tuned to generate faithful and brief explanations with input as the saliency paragraph.

4.3 End-to-End Explainer Training via Knowledge Distillation

The TAG Explanation Expert Iteration process gives us a pseudo-label generator model that experts in generating high-quality explanations based on saliency-based explanations. However, our goal is to have an end-to-end explainer model that can generate natural language explanations based on the raw input and its prediction. Therefore, after fine-tuning, we distill the whole pipeline to the *Explainer LLM*. The distillation is conducted by accumulating a dataset of filtered high-quality candidates during the expert iteration process, and using this dataset to fine-tune the *Explainer LLM*. The *Explainer LLM* serves as the product of our framework, an end-to-end post-hoc explainer for the model f.

5 Experiments

5.1 Experimental Setup

Datasets. We use three real-world TAG datasets, including two citation networks (Cora (Yang et al., 2016) and DBLP (Tang et al., 2008)), and one E-commerce co-purchasing network (Book-History (Yan et al., 2023)), to evaluate the performance of our method. More details of the datasets are given in Appendix E.

Compared Methods. To our best knowledge, no existing method are designed to generate natural language explanations for graph learning. The most relevant method is SMV (Feldhus et al., 2022), which verbalizes saliency map explanations for text classification models. To evaluate the effectiveness of GraphNarrator, we compare it with various most advanced LLMs to generate explanations in a zero-shot manner given the input subgraph and the model prediction. We benchmarked our method with the most advanced LLMs, including GPT-4o, GPT-3.5 turbo, LLaMA 3.1, and SMV method based on GPT-4o (denoted as SMV in the results).

Automatic Evaluation Metrics. Automatic evaluation is applied to evaluate the faithfulness and brevity of explanations. Following previous research (Padmakumar and He, 2021; Li et al., 2020), *Pointwise Mutual Information* (PMI) and *Simulatability* (Simul.) are used as indicators for faithfulness. PMI (Padmakumar and He, 2021; Chen et al., 2022; Colombo et al., 2022; Darrin et al., 2024) measures the mutual information between the generated explanations and the important regions in the input text. The top 10%, 20%, and 30% important tokens are used as references for the calculation of PMI, denoted as PMI-10%, PMI-20%, and PMI-30% in the result tables. *Simulatability* (Sushil et al., 2018; Sia et al., 2023; Li et al., 2020; Pruthi et al., 2022) measures the accuracy of the model prediction can be correctly inferred from the explanation. For *Brevity*, the average ratio of explanation length and input length is used as an indicator.

Human Evaluation Setting. We conduct a human annotation to investigate how human readers view the quality of explanations from different methods. For each dataset, we sample a split of 50 data points per method. We recruit three annotators with knowledge in computational linguistics (at least undergraduate level). Given the explanations, the annotators are asked to rate on a scale of 1-7 whether the explanations were (1) easy to understand, (2) insightful for the underlying decision process, (3) informative in preserving graph semantics, (4) informative in preserving graph structures.

Implementation Details. Our TAG model to explain is implemented with the commonly used Bert+GNN pipeline. We use bert-base-uncased as the text encoder and 2-layer SAGE as the GNN backbone. For each dataset we train the TAG model until converged with a learning rate of 1e-3 and batch size of 500. We used the GPT-4o-mini-2024-07-18 model with our prompt for TAG explanation for generating explanation pseudo label candidates (the details of prompts are given in Appendix G), applying one-shot learning for consistency. In the masked token prediction part, we utilized the gemma2-2b-it model to estimate the conditional probability distribution. We applied a balanced configuration for the three objectives (we select pseudo-label candidates whose three scores are all among the top 50% of all candidates generated in that iteration). Finally, we used the fine-tuned LLaMA-3.1-8b as the base student model for knowledge distillation using the LoRA technique. More implementation details are given in Appendix F.

5.2 Explanation Quality

Automatic Evaluation. We use automatic methods to evaluate the faithfulness and brevity of generated explanations. Our experimental results, shown in Table 1, demonstrate that GraphNarrator consistently performs well in generating high-quality explanations. Specifically, GraphNarrator shows an 8.2% average improvement over the second-

Dataset	Method	Metrics				
		Simul. (↑)	PMI-10% (↑)	PMI-20% (↑)	PMI-30% (↑)	Brevity (↓)
DBLP	LLaMA3.1 8B	0.63	0.139	0.109	0.077	0.394
	GPT-3.5 Turbo	0.71	0.136	**0.110**	0.084	0.403
	GPT-4o	0.82	0.142	0.101	0.085	0.385
	SMV	0.76	0.139	0.098	0.082	0.419
	GraphNarrator$_{\text{LLaMA3.1 8B}}$	**0.95**	**0.155**	0.108	**0.085**	**0.354**
Cora	LLaMA3.1 8B	0.78	0.335	0.278	0.199	0.600
	GPT-3.5 Turbo	0.83	0.340	0.281	0.213	0.318
	GPT-4o	0.95	0.414	0.284	0.225	0.357
	SMV	0.88	0.359	0.267	0.217	0.431
	GraphNarrator$_{\text{LLaMA3.1 8B}}$	**0.97**	**0.418**	**0.290**	**0.227**	**0.315**
Book-History	LLaMA3.1 8B	0.79	0.465	**0.390**	0.281	0.735
	GPT-3.5 Turbo	0.83	0.436	0.361	0.270	0.853
	GPT-4o	0.89	0.456	0.313	0.240	0.768
	SMV	0.87	0.441	0.320	0.257	0.836
	GraphNarrator$_{\text{LLaMA3.1 8B}}$	**0.96**	**0.533**	0.374	**0.291**	**0.506**

Table 1: **Automatic evaluation** of natural language explanations quality generated by different methods. Best results are bolded.

Dataset	Method	Metrics			
		EU	DMI	SI	SeI
DBLP	GPT-4o	4.2	4.5	3.6	3.6
	SMV	3.8	4.1	3.4	3.7
	GraphNarrator	**4.5**	**4.8**	**5.4**	**4.3**
Cora	GPT-4o	4.7	4.0	4.7	4.3
	SMV	4.2	**5.2**	3.2	3.6
	GraphNarrator	**4.8**	4.6	**5.2**	**5.0**
History	GPT-4o	4.9	4.6	3.9	3.9
	SMV	4.8	4.7	3.9	4.1
	GraphNarrator	**5.0**	**5.1**	**5.4**	**5.3**

Table 2: **Human evaluation**. Best results are bolded. **EU**: Easy to Understand. **DMI**: Insightful in explaining the decision-making process. **SI**: Structural Informative. **SeI**: Semantic Informative.

best performer in the PMI-10% metric over three datasets, highlighting its effectiveness at capturing the most important information for model decisions. In terms of simulatability, GraphNarrator outperforms all baseline methods by 8.6%, achieving a simulatability score of 0.95 across all three datasets, significantly higher than other methods, proving highly faithful to model predictions. For the brevity metric, GraphNarrator is 13.4% better than the second-best performer, effectively balancing conciseness and accuracy. Across all three datasets, GraphNarrator generates relatively compact explanations while maintaining high simulatability scores. These results demonstrate that GraphNarrator successfully navigates the inherent trade-offs among PMI, simulatability, and brevity. It consistently produces explanations that align closely with model predictions while remaining concise, further enhancing their interpretability. This balance highlights GraphNarrator's strength in delivering both faithful and interpretable explanations. The improvement on Cora dataset is relatively low due to the low data quality brought OCR-based collection.

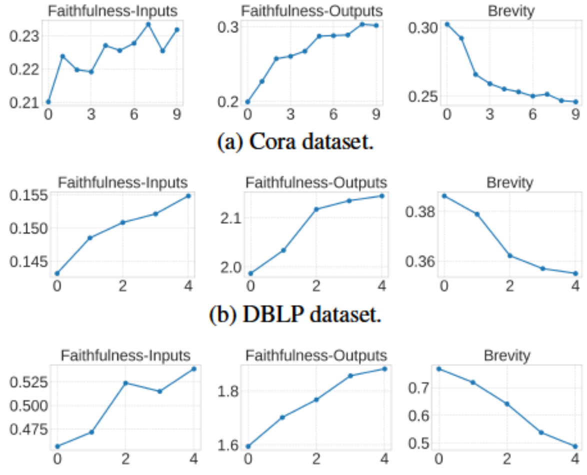
Human Evaluation. Results of human evaluation are given in Table 2. Results reveal our approach demonstrates strong performance across multiple dimensions of human evaluation. In particular, it achieves the best overall results in terms of ease of understanding, insightfulness in the explanation process, and the preservation of both semantic and structural information. When compared with GPT-4o, our method yields notable improvements—approximately 33.7% in structural informativeness and 23.9% in semantic informativeness. Moreover, its ease of understanding is on par with both GPT-4o and SMV, a benefit likely derived from the robust large language model backbone underlying our approach.

5.3 Pseudo-Label Self-Improvement

Effectiveness of TAG Explanation Expert Iteration. The training score curve in our iterative training (Expert Iteration) of the pseudo-label generator LLM is illustrated in Figure 4. With an increasing number of iterations, both faithfulness to important inputs and faithfulness to predictions exhibit a overall upward trend, while brevity shows a gradual decline. This iterative learning process underscores our TAG explanation expert iteration's effectiveness in improving the explanation quality. It is worth highlighting that during each itera-

Method	Metrics				
	Simul. (↑)	PMI-10% (↑)	PMI-20% (↑)	PMI-30% (↑)	Brevity (↓)
GraphNarrator	0.97	0.418	0.290	0.227	0.315
w/o f_S	0.98	0.407	0.298	0.213	0.304
w/o f_F	0.90	0.419	0.311	0.241	0.315
w/o f_B	0.96	0.432	0.327	0.239	0.361
w/o saliency	0.97	0.402	0.284	0.218	0.343

Table 3: Results of ablation study. w/o f_S, f_F, F_B denotes removing the corresponding objective for training. Cells highlighted represent metrics where the performance is **expected to drop** when the corresponding component is removed.



(a) Cora dataset.

(b) DBLP dataset.

(c) Book-History dataset.

Figure 4: The change of three pseudo-label quality scores in the TAG explanation expert iteration process, w.r.t number of training iteration.

tion of the expert iteration process, we select only 50 high-quality samples. Despite this small sample number, our TAG explanation expert iteration consistently enhances faithfulness to both important inputs and predictions. This demonstrates that the process is highly efficient, as steady improvements in performance are achieved by using just a small and high-quality set of high-quality samples in each iteration. We also evaluated the quality of pseudo-labels to verify the effectiveness of the Expert Iteration pipeline, as given in Appendix C. A study of different pseudo-label selection strategies is given in Appendix D.1. The performance improvement of the explainer LLM after fine-tuning with generated high-quality pseudo-labels are given in Appendix D.2.

5.4 Ablation Study

In the ablation study, we removed f_S, f_F, f_B and Expert Iteration to test their corresponding effectiveness, as shown in Table 3. The results highlight the effectiveness of different components of the proposed framework. Removing the saliency optimization objective (w/o f_S) decreases PMI scores, proving its importance for faithfulness. Removing the fidelity objective (w/o f_F) results in lower simulatability, and removing the brevity optimization objective (w/o f_B). Notably, sometimes when one objective is removed, the other two often improve. This is because the three objectives inherently involve trade-offs—removing one allows the remaining two to be optimized within a larger space, free from the constraints imposed by the removed objective, naturally leading to better performance in those areas.

6 Conclusion

In this paper, we present GraphNarrator, a model-agnostic post-hoc explainer to generate natural language explanations for TAG learning models. GraphNarrator fine-tunes a generative language model as an explanation generator with pseudo-labels derived from saliency-based explanations. Through iterative self-training, we improve the quality of generated explanation pseudo-labels, ensuring the explanation generator can be trained with high-quality data. Our extensive experiments demonstrate the effectiveness of GraphNarrator.

7 Acknowledgments

This work was supported by the National Science Foundation (NSF) Grant No. 2414115, No. 2403312, No. 2007716, No. 2007976, No. 1942594, No. 1907805, NIH R01AG089806, and NIH R01CA297856.

Limitation

The backbone of GraphNarrator is based on LLMs, which may be more costly to do inference than saliency-based explanation methods. For long documents although KV-cache can bring significant improvements for inference, we still find for a few extremely large subgraphs, the inference time could exceed 2 mins.

References

Chirag Agarwal, Sree Harsha Tanneru, and Himabindu Lakkaraju. 2024. Faithfulness vs. plausibility: On the (un) reliability of explanations from large language models. *arXiv preprint arXiv:2402.04614*.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Federico Baldassarre and Hossein Azizpour. 2019. Explainability techniques for graph convolutional networks. In *International Conference on Machine Learning (ICML) Workshops, 2019 Workshop on Learning and Reasoning with Graph-Structured Representations*.

Erik Cambria, Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, and Navid Nobani. 2023. A survey on xai and natural language explanations. *Information Processing & Management*, 60(1):103111.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31.

Hanjie Chen, Faeze Brahman, Xiang Ren, Yangfeng Ji, Yejin Choi, and Swabha Swayamdipta. 2022. Rev: information-theoretic evaluation of free-text rationales. *arXiv preprint arXiv:2210.04982*.

Pierre Jean A Colombo, Chloé Clavel, and Pablo Piantanida. 2022. Infolm: A new metric to evaluate summarization & data2text generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 10554–10562.

Maxime Darrin, Philippe Formont, Jackie Chi Kit Cheung, and Pablo Piantanida. 2024. Cosmic: Mutual information for task-agnostic summarization evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12696–12717.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Nils Feldhus, Leonhard Hennig, Maximilian Dustin Nasert, Christopher Ebert, Robert Schwarzenberg, and Sebastian Möller. 2022. Saliency map verbalization: Comparing feature importance representations from model-free and instruction-based methods. *arXiv preprint arXiv:2210.07222*.

Yair Gat, Nitay Calderon, Amir Feder, Alexander Chapanin, Amit Sharma, and Roi Reichart. 2023. Faithful explanations of black-box nlp models using llm-generated counterfactuals. *arXiv preprint arXiv:2310.00603*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2023. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *arXiv preprint arXiv:2305.19523*.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506*.

Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. 2022. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):6968–6972.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Nicholas Kroeger, Dan Ley, Satyapriya Krishna, Chirag Agarwal, and Himabindu Lakkaraju. 2023. Are large language models post hoc explainers? *arXiv preprint arXiv:2310.05797*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*.

Jierui Li, Lemao Liu, Huayang Li, Guanlin Li, Guoping Huang, and Shuming Shi. 2020. Evaluating explanation methods for neural machine translation. *arXiv preprint arXiv:2005.01672*.

Wei Liu, Haozhao Wang, Jun Wang, Zhiying Deng, Yuankai Zhang, Cheng Wang, and Ruixuan Li. 2024. Enhancing the rationale-input alignment for self-explaining rationalization. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 2218–2230. IEEE.

Wei Liu, Jun Wang, Haozhao Wang, Ruixuan Li, Zhiying Deng, Yuankai Zhang, and Yang Qiu. 2023. D-separation for causal self-explanation. *Advances in Neural Information Processing Systems*, 36:43620–43633.

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.

David Martens, James Hinns, Camille Dams, Mark Vergouwen, and Theodoros Evgeniou. 2023. Tell me a story! narrative-driven xai with large language models. *arXiv preprint arXiv:2309.17057*.

Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. 2020. Wt5?! training text-to-text models to explain their predictions. *arXiv preprint arXiv:2004.14546*.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197.

Vishakh Padmakumar and He He. 2021. Unsupervised extractive summarization using pointwise mutual information. *arXiv preprint arXiv:2102.06272*.

Bo Pan, Zheng Zhang, Yifei Zhang, Yuntong Hu, and Liang Zhao. 2024. Distilling large language models for text-attributed graph learning. *arXiv preprint arXiv:2402.12022*.

Letitia Parcalabescu and Anette Frank. 2024. On measuring faithfulness or self-consistency of natural language explanations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6048–6089.

Danish Pruthi, Rachit Bansal, Bhuwan Dhingra, Livio Baldini Soares, Michael Collins, Zachary C Lipton, Graham Neubig, and William W Cohen. 2022. Evaluating explanations: How much do explanations from the teacher aid students? *Transactions of the Association for Computational Linguistics*, 10:359–375.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*.

Robert Schwarzenberg, Marc Hübner, David Harbecke, Christoph Alt, and Leonhard Hennig. 2019. Layerwise relevance visualization in convolutional text graph classifiers. *arXiv preprint arXiv:1909.10911*.

Suzanna Sia, Anton Belyy, Amjad Almahairi, Madian Khabsa, Luke Zettlemoyer, and Lambert Mathias. 2023. Logical satisfiability of counterfactuals for faithful explanations in nli. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9837–9845.

Erik Štrumbelj and Igor Kononenko. 2014. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41:647–665.

Madhumita Sushil, Simon Šuster, Kim Luyckx, and Walter Daelemans. 2018. Patient representation learning and interpretable evaluation using clinical notes. *Journal of biomedical informatics*, 84:103–113.

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.

Minh Vu and My T Thai. 2020. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems*, 33:12225–12235.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Sarah Wiegreffe, Ana Marasović, and Noah A Smith. 2020. Measuring association between labels and free-text rationales. *arXiv preprint arXiv:2010.12762*.

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

Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. 2021. Graphformers: Gnn-nested transformers for representation learning on textual graph. *Advances in Neural Information Processing Systems*, 34:28798–28810.

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.

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.

Mo Yu, Shiyu Chang, Yang Zhang, and Tommi S Jaakkola. 2019. Rethinking cooperative rationalization: Introspective extraction and complement control. *arXiv preprint arXiv:1910.13294*.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. 2020. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 430–438.

Yifei Zhang, Bo Pan, Chen Ling, Yuntong Hu, and Liang Zhao. 2024. Elad: Explanation-guided large language models active distillation. *arXiv preprint arXiv:2402.13098*.

Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. 2022. Protgnn: Towards self-explaining graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9127–9135.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. Gear: Graph-based evidence aggregating and reasoning for fact verification. *arXiv preprint arXiv:1908.01843*.

A Qualitative Evaluation

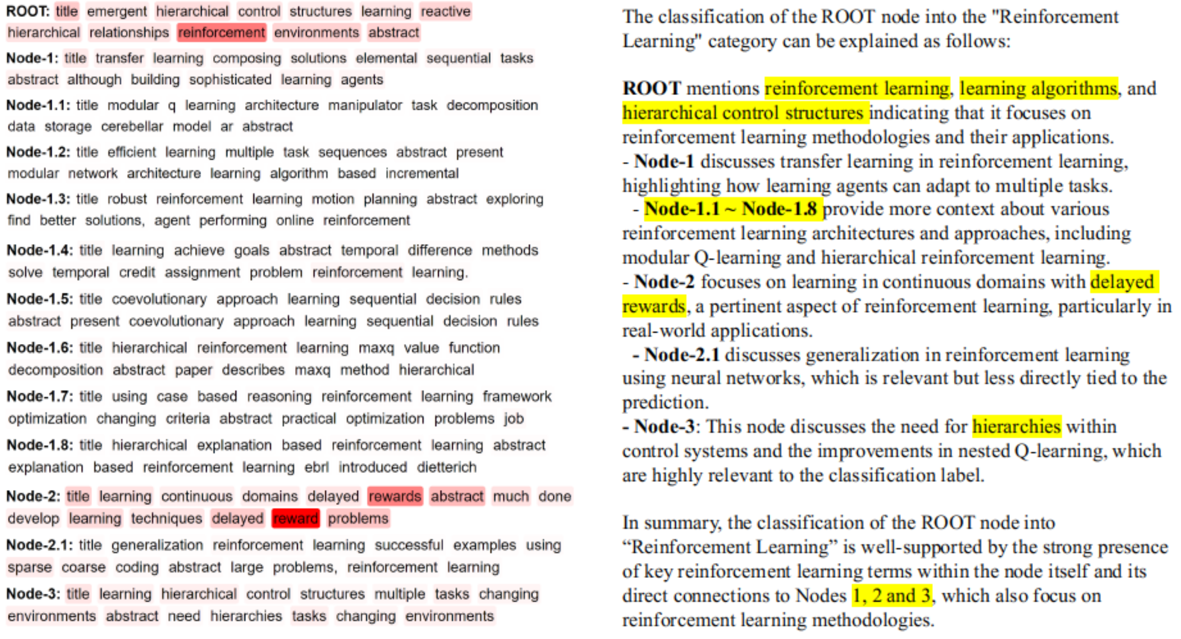
We show three examples of explanations generated by GraphNarrator in Fig. 5, Fig. 6, Fig. 7. As depicted in Figure 5a, we visualize the token importance of saliency-based explanation, where individual words within each node are highlighted with varying intensities of red to indicate their saliency scores. Darker red hues correspond to higher saliency scores, while lighter shades represent lower ones. In contrast, Figure 5b showcases the natural language explanations generated by GraphNarrator, with key terms such as "reinforcement learning" and "learning algorithm" emphasized in yellow. This demonstrates GraphNarrator's ability to not only capture the salient information identified in the saliency map but also present it in a more accessible and interpretable manner. Compared with the saliency-based explanations that merely highlight important words, GraphNarrator goes beyond synthesizing and abstracting content across nodes. For example, in the case of Node-1.1 through Node-1.8, GraphNarrator effectively integrates the most relevant information into a coherent explanation rather than simply reproducing the input. This showcases GraphNarrator's strength in generating explanations that are more informative and contextualized than the visual saliency approach. Similar properties of GraphNarrator-generated explanations can also be seen in Fig. 6, Fig. 7.

B Serialization of $\mathcal{G}$ and construction of masked token prediction task

Serialization of $\mathcal{G}$. To serialize the text-attributed graph $\mathcal{G}$, we use similar method as our saliency TAG verbalization method, but not include any saliency score information. Therefore, a TAG $\mathcal{G}$ is serialized by first conducting a BFS search from the root node, then using pre-order traversal to organize each node into a section in the serialized document, and adding the "cross-edges" to include the edges that connect nodes from different branches. An example is given in Fig. 8.

Masked token prediction with pre-trained language model: Given a masked token prediction task, we calculate $\log P(Y|X)$, where X and Y are two arbitrary token sequences, by decomposing it into token-level predictions. Specifically,

$$\log P(Y|X) = \sum_{i=1}^{|Y|} \log P(y_i|X, y_0, \ldots, y_{i-1})$$



ROOT: title emergent hierarchical control structures learning reactive hierarchical relationships reinforcement environments abstract

Node-1: title transfer learning composing solutions elemental sequential tasks abstract although building sophisticated learning agents

Node-1.1: title modular q learning architecture manipulator task decomposition data storage cerebellar model ar abstract

Node-1.2: title efficient learning multiple task sequences abstract present modular network architecture learning algorithm based incremental

Node-1.3: title robust reinforcement learning motion planning abstract exploring find better solutions, agent performing online reinforcement

Node-1.4: title learning achieve goals abstract temporal difference methods solve temporal credit assignment problem reinforcement learning.

Node-1.5: title coevolutionary approach learning sequential decision rules abstract present coevolutionary approach learning sequential decision rules

Node-1.6: title hierarchical reinforcement learning maxq value function decomposition abstract paper describes maxq method hierarchical

Node-1.7: title using case based reasoning reinforcement learning framework optimization changing criteria abstract practical optimization problems job

Node-1.8: title hierarchical explanation based reinforcement learning abstract explanation based reinforcement learning ebrl introduced dietterich

Node-2: title learning continuous domains delayed rewards abstract much done develop learning techniques delayed reward problems

Node-2.1: title generalization reinforcement learning successful examples using sparse coarse coding abstract large problems, reinforcement learning

Node-3: title learning hierarchical control structures multiple tasks changing environments abstract need hierarchies tasks changing environments

(a) Saliency-Based Explanation

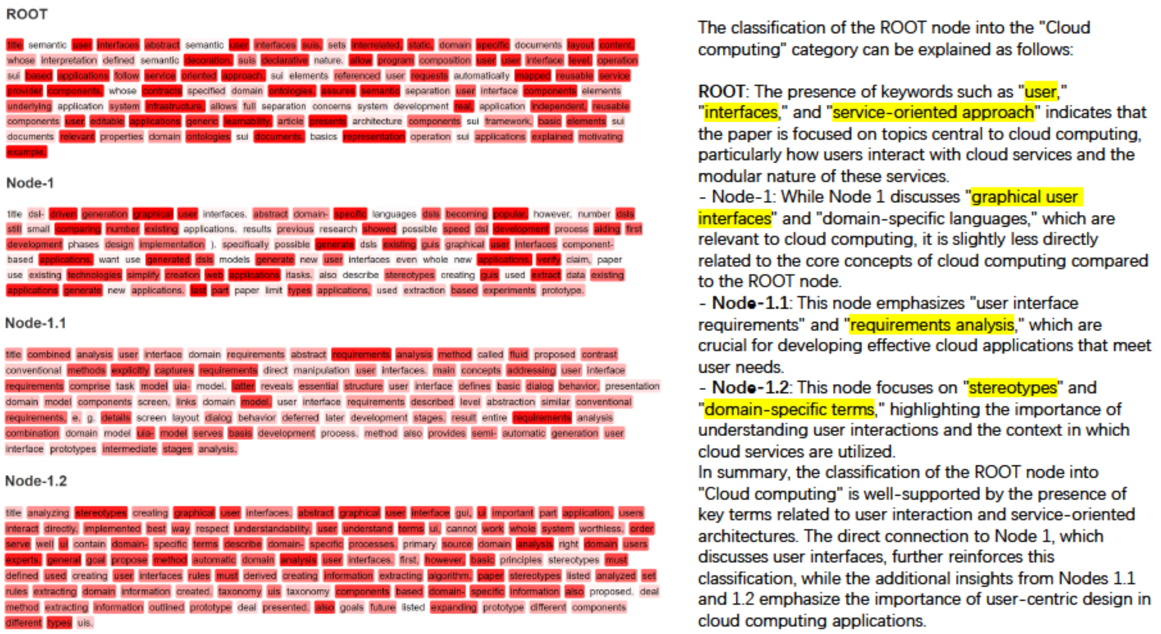
The classification of the ROOT node into the "Reinforcement Learning" category can be explained as follows:

ROOT mentions reinforcement learning, learning algorithms, and hierarchical control structures indicating that it focuses on reinforcement learning methodologies and their applications.
- **Node-1** discusses transfer learning in reinforcement learning, highlighting how learning agents can adapt to multiple tasks.
  - **Node-1.1 ~ Node-1.8** provide more context about various reinforcement learning architectures and approaches, including modular Q-learning and hierarchical reinforcement learning.
- **Node-2** focuses on learning in continuous domains with delayed rewards, a pertinent aspect of reinforcement learning, particularly in real-world applications.
  - **Node-2.1** discusses generalization in reinforcement learning using neural networks, which is relevant but less directly tied to the prediction.
- **Node-3**: This node discusses the need for hierarchies within control systems and the improvements in nested Q-learning, which are highly relevant to the classification label.

In summary, the classification of the ROOT node into "Reinforcement Learning" is well-supported by the strong presence of key reinforcement learning terms within the node itself and its direct connections to Nodes 1, 2 and 3, which also focus on reinforcement learning methodologies.

(b) GraphNarrator

Figure 5: Visualization of a saliency-based explanation and a corresponding natural language explanation generated by GraphNarrator. In (a), red words indicate important terms, with darker red showing higher importance. We only visualized first 15 words in each paper due to the space limitation. In (b), yellow highlights reference high-saliency areas and emphasize that the explanation summarized key information.



ROOT

title semantic user interfaces abstract semantic user interfaces suis, sets interrelated, static, domain specific documents layout content, whose interpretation defined semantic decoration, suis declarative nature, allow program composition user user interface level, operation sui based applications follow service oriented approach. sui elements referenced user requests automatically mapped reusable service provider components, whose contracts specified domain ontologies, assures semantic separation user interface components elements underlying application system infrastructure, allows full separation concerns system development real, application independent, reusable components user editable applications generic learnability, article presents architecture components sui framework, basic elements sui documents relevant properties domain ontologies sui documents. basics representation operation sui applications explained motivating example.

Node-1

title dsl- driven generation graphical user interfaces. abstract domain- specific languages dsls becoming popular, however, number dsls still small comparing number existing applications. results previous research showed possible speed dsl development process aiding first development phases design implementation). specifically possible generate dsls existing guis graphical user interfaces component- based applications. want use generated dsls models generate new user interfaces even whole new applications. verify claim, paper use existing technologies simplify creation web applications itasks. also describe stereotypes creating guis used extract data existing applications generate new applications. last part paper limit types applications, used extraction based experiments prototype.

Node-1.1

title combined analysis user interface domain requirements abstract requirements analysis method called fluid proposed contrast conventional methods explicitly captures requirements direct manipulation user interfaces. main concepts addressing user interface requirements comprise task model uia- model. latter reveals essential structure user interface defines basic dialog behavior, presentation domain model components screen, links domain model. user interface requirements described level abstraction similar conventional requirements, e. g. details screen layout dialog behavior deferred later development stages. result entire requirements analysis combination domain model uia- model serves basis development process. method also provides semi- automatic generation user interface prototypes intermediate stages analysis.

Node-1.2

title analyzing stereotypes creating graphical user interfaces. abstract graphical user interface gui, ui important part application, users interact directly. implemented best way respect understandability. user understand terms ui, cannot work whole system worthless. order serve well ui contain domain- specific terms describe domain- specific processes. primary source domain analysis right domain users experts. general goal propose method automatic domain analysis user interfaces. first, however, basic principles stereotypes must defined used creating user interfaces rules must derived creating information extracting algorithm. paper stereotypes listed analyzed set rules extracting domain information created. taxonomy uis taxonomy components based domain- specific information also proposed. deal method extracting information outlined prototype deal presented. also goals future listed expanding prototype different components different types uis.

(a) Saliency-Map

The classification of the ROOT node into the "Cloud computing" category can be explained as follows:

ROOT: The presence of keywords such as "user," "interfaces," and "service-oriented approach" indicates that the paper is focused on topics central to cloud computing, particularly how users interact with cloud services and the modular nature of these services.
- Node-1: While Node 1 discusses "graphical user interfaces" and "domain-specific languages," which are relevant to cloud computing, it is slightly less directly related to the core concepts of cloud computing compared to the ROOT node.
- **Nod•-1.1**: This node emphasizes "user interface requirements" and "requirements analysis," which are crucial for developing effective cloud applications that meet user needs.
- **Nod•-1.2**: This node focuses on "stereotypes" and "domain-specific terms," highlighting the importance of understanding user interactions and the context in which cloud services are utilized.

In summary, the classification of the ROOT node into "Cloud computing" is well-supported by the presence of key terms related to user interaction and service-oriented architectures. The direct connection to Node 1, which discusses user interfaces, further reinforces this classification, while the additional insights from Nodes 1.1 and 1.2 emphasize the importance of user-centric design in cloud computing applications.

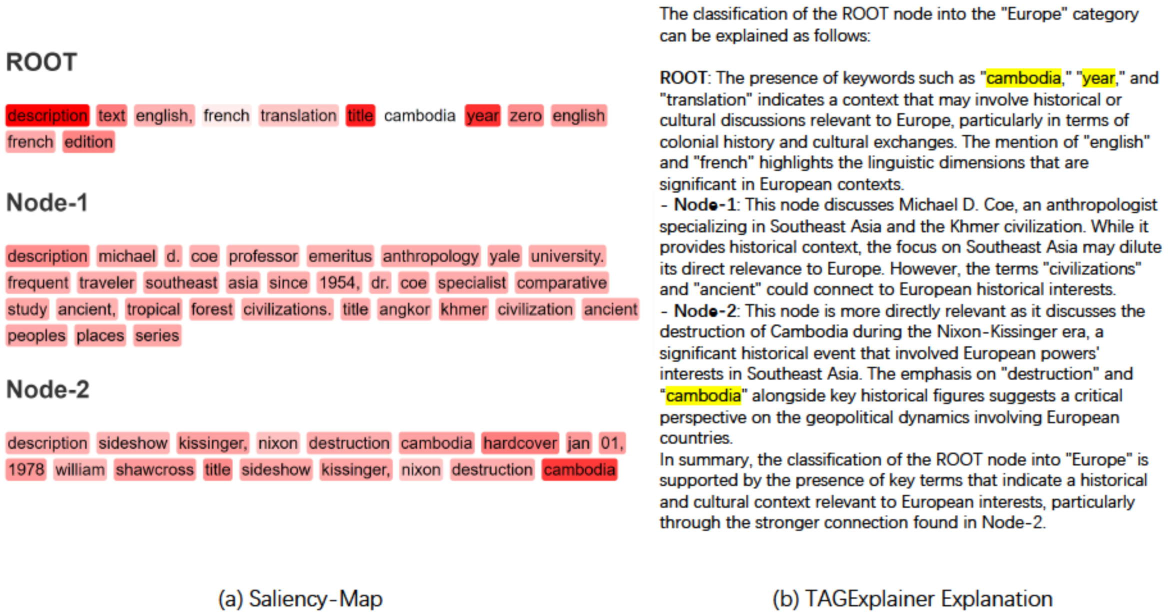
(b) TAGExplainer Explanation

Figure 6: Example from DBLP

To compute $P(y_i|X, y0, \ldots, y_{i-1})$, we first concatenate X with the partial sequence $[y_0, \ldots, y_{i-1}]$ as input to gemma, then extract the probability of token y_i from gemma's output logits.

C Pseudo-label quality

To showcase the effectiveness of our TAG Expert Iteration on improving pseudo label quality, we also conduct human evaluations on the pseudo labels used to train the Explainer LLM. The evaluation targets are same as the human evaluation on the



ROOT

description text english, french translation title cambodia year zero english french edition

Node-1

description michael d. coe professor emeritus anthropology yale university. frequent traveler southeast asia since 1954, dr. coe specialist comparative study ancient, tropical forest civilizations. title angkor khmer civilization ancient peoples places series

Node-2

description sideshow kissinger, nixon destruction cambodia hardcover jan 01, 1978 william shawcross title sideshow kissinger, nixon destruction cambodia

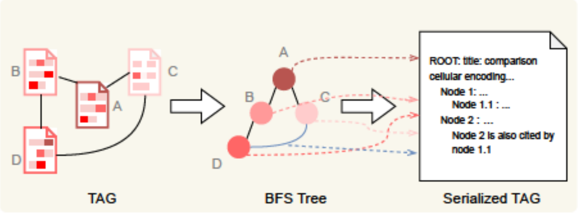
The classification of the ROOT node into the "Europe" category can be explained as follows:

ROOT: The presence of keywords such as "cambodia," "year," and "translation" indicates a context that may involve historical or cultural discussions relevant to Europe, particularly in terms of colonial history and cultural exchanges. The mention of "english" and "french" highlights the linguistic dimensions that are significant in European contexts.
- **Node-1**: This node discusses Michael D. Coe, an anthropologist specializing in Southeast Asia and the Khmer civilization. While it provides historical context, the focus on Southeast Asia may dilute its direct relevance to Europe. However, the terms "civilizations" and "ancient" could connect to European historical interests.
- **Node-2**: This node is more directly relevant as it discusses the destruction of Cambodia during the Nixon-Kissinger era, a significant historical event that involved European powers' interests in Southeast Asia. The emphasis on "destruction" and "cambodia" alongside key historical figures suggests a critical perspective on the geopolitical dynamics involving European countries.
In summary, the classification of the ROOT node into "Europe" is supported by the presence of key terms that indicate a historical and cultural context relevant to European interests, particularly through the stronger connection found in Node-2.

(a) Saliency-Map

(b) TAGExplainer Explanation

Figure 7: Example from Book-History



ROOT: title: comparison cellular encoding...
Node 1: ...
Node 1.1 : ...
Node 2 : ...
Node 2 is also cited by node 1.1

Figure 8: Illustration of text-attributed graph serialization.

results, stated in the experimental setting section. Results are given in Table 4. Results show that the quality of pseudo labels after the Expert Iteration is significantly better than zero-shot explanations from gpt-4o.

Dataset	Method	Metrics			
		EU	DMI	SI	SeI
DBLP	GPT-4o	4.2	4.5	3.6	3.6
	Pseudo-Label	**5.2**	**5.7**	**5.7**	**4.6**
Cora	GPT-4o	4.7	4.0	4.3	4.7
	Pseudo-Label	**4.8**	**4.8**	**5.2**	**5.0**
Book-History	GPT-4o	4.9	4.6	3.9	3.9
	Pseudo-Label	**5.0**	**5.1**	**5.4**	**5.4**

Table 4: Evaluation of pseudo-label quality.

D Additional Experiment Results

D.1 Experiments on different candidate selection criteria

To provide an initial validation of the effectiveness of the optimization process for the explanation generator, we conducted experiments using three extreme selection criteria: (a) a selection strategy that prioritizes only faithfulness to important inputs, (b) a strategy focusing exclusively on faithfulness to predictions, and (c) a strategy considering solely brevity. Our results (as shown in Figure 10) indicate that, under each of these conditions, the corresponding metric was significantly improved. These findings suggest that the proposed framework has the capacity to selectively enhance the performance of the explanation generator with respect to specific evaluation metrics, demonstrating its adaptability and targeted optimization potential.

D.2 The performance gain of Explainer LLM after knowledge distillation

We tested the performance of vanilla LLaMA3.1 8B model and the distilled version, which leads to our GraphNarrator, among three different datasets. The results demonstrated in table 5 shows that the distillation process indeed promoted the quality of explanation in terms of most PMI, Simulatability and Brevity metrics.

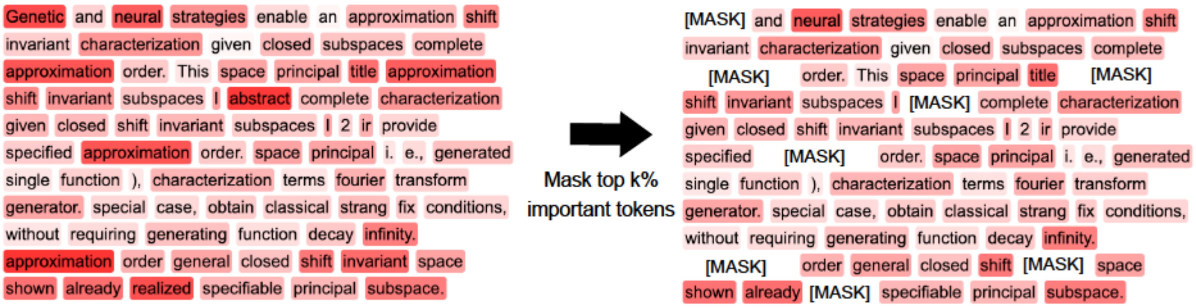
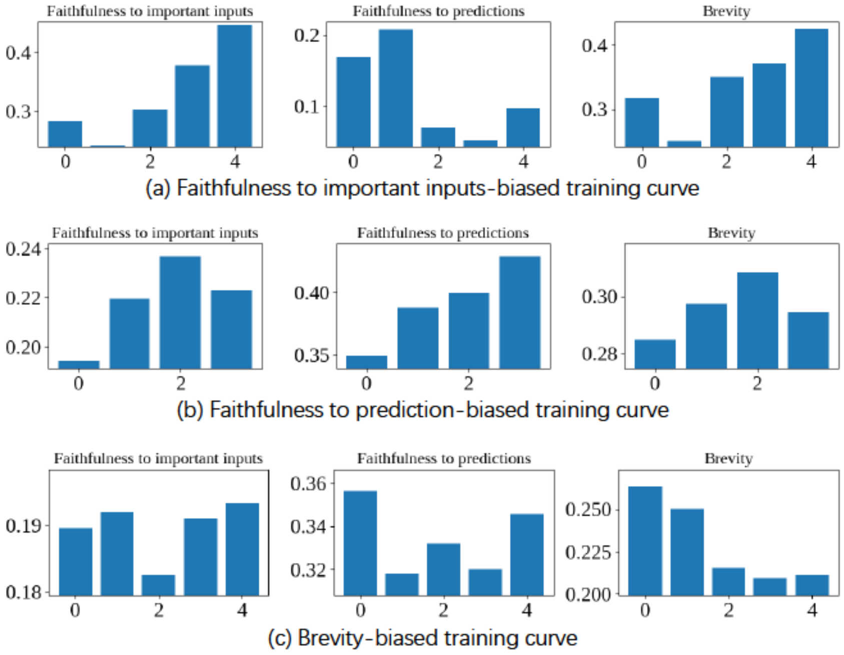

Genetic and neural strategies enable an approximation shift invariant characterization given closed subspaces complete approximation order. This space principal title approximation shift invariant subspaces I abstract complete characterization given closed shift invariant subspaces I 2 ir provide specified approximation order. space principal i. e., generated single function), characterization terms fourier transform generator. special case, obtain classical strang fix conditions, without requiring generating function decay infinity. approximation order general closed shift invariant space shown already realized specifiable principal subspace.
Mask top k% important tokens
[MASK] and neural strategies enable an approximation shift invariant characterization given closed subspaces complete [MASK] order. This space principal title [MASK] shift invariant subspaces I [MASK] complete characterization given closed shift invariant subspaces I 2 ir provide specified [MASK] order. space principal i. e., generated single function), characterization terms fourier transform generator. special case, obtain classical strang fix conditions, without requiring generating function decay infinity. [MASK] order general closed shift [MASK] space shown already [MASK] specifiable principal subspace.

Figure 9: Illustration of masking important tokens prediction



(a) Faithfulness to important inputs-biased training curve

(b) Faithfulness to prediction-biased training curve

(c) Brevity-biased training curve

Figure 10: experiment with different customized selecting criteria

E Dataset Details

We conduct experiments on 3 datasets, the basic statistics are shown in Table 6.

Cora is a network that contains computer science research papers, where each node represents a paper, and each edge represents one paper and cites the other one. Nodes in the Cora dataset are classified into seven categories: Case_Based, Genetic_Algorithms, Neural_Networks, Probabilistic_Methods, Reinforcement_Learning, Rule_Learning, and Theory.

DBLP dataset is a large-scale network of academic research papers, where each node represents a paper and each edge indicates a citation between two papers. Similar to the Cora dataset, which focuses on computer science research, DBLP covers a broader range of fields of study with an emphasis on computer science and related disciplines. Papers in the DBLP dataset are classified into various fields of study based on their topics. From the DBLP dataset, we extracted the top 30 most frequently occurring fields of study, along with their corresponding papers. Some of these categories include cluster analysis, cloud computing, computer science, the internet, wireless sensor networks, artificial neural networks, population, control theory, image segmentation, humanities, and image processing. These categories reflect the diverse range of research areas covered in the DBLP dataset.

Book-History dataset, extracted from the Amazon dataset (Ni et al., 2019), comprises items labeled under the second-level category "History." In this dataset, each node represents a book, and edges between nodes indicate frequent co-purchases or co-views of the books. The books in the Book-History dataset are classified into 12 distinct categories: Africa, Americas, Ancient Civilizations,

Dataset	Method	Metrics				
		PMI-10% (↑)	PMI-20% (↑)	PMI-30% (↑)	Simul. (↑)	Brevity (↓)
Cora	LLaMA3.1 8B	0.335	0.278	0.199	0.78	0.600
	GraphNarrator	**0.418**	**0.290**	**0.227**	**0.97**	**0.315**
DBLP	LLaMA3.1 8B	0.139	**0.109**	0.077	0.63	0.394
	GraphNarrator	**0.155**	0.108	**0.085**	**0.95**	**0.354**
Book-History	LLaMA3.1 8B	0.465	**0.390**	0.281	0.79	0.735
	GraphNarrator	**0.533**	0.374	**0.291**	**0.96**	**0.506**

Table 5: The performance of student model before (LLaMA3.1 8B) and after (our GraphNarrator) distillation. Better results are bolded.

	# Nodes	# Edges	# Categories
Cora	2,708	5,429	7
DBLP	110,757	655,766	30
Book-History	41,551	358,574	12

Table 6: Dataset Overview

Arctic & Antarctica, Asia, Australia & Oceania, Europe, Historical Study & Educational Resources, Middle East, Military, Russia, and World.

F Implementation Details

We first masked the last 5% of tokens (mostly stop words and punctuations without explicit semantical contribution to the downstream tasks) based on their importance scores to form a reduced saliency-based explanation as input. We then utilized the candidate explanation generator GPT-4o-mini-2024-07-18, prompting it with a carefully designed template (see Appendix G for details) and employing a one-shot learning technique to ensure consistency in the format and style of the generated explanations. For scoring and rejection sampling, we used the gemma2-2b-it model as the masked language model for the masked token prediction task to estimate the conditional probability distribution mentioned in information-theoretic objectives. In Equation 3, we have masked all label-related information in condition E to prevent answer leakage. During the rejection sampling phase, we found that a balanced configuration among all three objectives introduced in 4.2.1, i.e., $\lambda_S : \lambda_F : \lambda_B = 1 : 1 : 1$, provided stable and balanced performance across the three evaluation metrics (more customized criteria are included Appendix D). We applied Expert Iteration fine-tuning by selecting 50 high-quality samples during each loop via rejection sampling. These samples were used to fine-tune the model using OpenAI API with default learning rate and batch size for 3 epochs. The final model obtained from the optimization loop served as the teacher model. We then performed knowledge distillation using the fine-tuned LLaMA-3.1-8b as the base student model, employing the LoRA technique (rank r=16 and alpha=16) for efficient fine-tuning. We minimized the cross-entropy loss between the student outputs and the teacher outputs, which resulted in our final GraphNarrator model.

For expert iteration, each iteration involves generating three scores via a Gemma-2B model deployed on a single NVIDIA H100 GPU, taking 30 minutes per iteration. Fine-tuning and inference with GPT-4o via OpenAI API costs of $5 per iteration, with training of 10, 5, and 5 iterations across the three datasets. Finally, knowledge distillation via LoRA fine-tuning of a LLaMA-3.1 8B model requires 20 minutes per dataset on an NVIDIA A6000 GPU.

G Prompt Details

In our experiments, we utilize two types of prompts: one in which each token in the input is accompanied by a corresponding saliency score, and another without saliency scores. The difference between the two prompts is whether the words in the verbalized graph are accompanied by their corresponding importance scores in brackets or not.The teacher models require the inclusion of saliency scores, as they function as candidate explanation generators. The presence of saliency scores enables them to generate more accurate explanations by highlighting important tokens. In contrast, the student models do not use saliency scores; their task is to output the reasoning process of the black-box model based solely on the TAG and prediction. The student models are designed to align directly with the teacher models' outputs, ensuring consistency without requiring saliency information. All the prompts we used are given from here, due to the limited space.

G.1 w/o Saliency Prompt

HumanMessage: “The following verbalized graph contains important words in the text of each node. These words contribute to the classification of Node 0 into one of the seven possible categories ([’Case Based’, ’Genetic Algorithms’, ’Neural Networks’, ’Probabilistic Methods’, ’Reinforcement Learning’, ’Rule Learning’, ’Theory’]).
Generate a concise, human-readable explanation that justifies the classification result of Node 0 by identifying and explaining the relevant inner-node features (i.e., keywords) and inter-node relationships (i.e., graph structure). The explanation should focus on how these factors contribute to the classification label.

Example

Verbalized Graph

`<verbalized-graph>`
ROOT: title experiments real time decision algorithms abstract real time decision algorithms class incremental resource bounded horvitz, 89 anytime dean, 93 algorithms evaluating influence diagrams. present test domain real time decision algorithms, results experiments several real time decision algorithms domain. results demonstrate high performance two algorithms, decision evaluation variant incremental probabilisitic inference dambrosio, 93 variant algorithm suggested goldszmidt, goldszmidt, 95], pk reduced. discuss implications experimental results explore broader applicability algorithms.
Node-1: title learning policies partially observable environments scaling abstract partially observable markov decision processes pomdp model decision problems agent tries maximize reward face limited noisy sensor feedback.
Node-1.1: title formal framework speedup learning problems solutions abstract speedup learning seeks improve computational efficiency problem solving experience. paper, develop formal framework learning efficient problem solving random problems solutions.
Node-1.2: title acting uncertainty discrete bayesian models mobile robot navigation abstract discrete bayesian models used model uncertainty mobile robot navigation, question actions chosen remains largely unexplored.
Node-1.3: title incremental methods computing bounds partially observable markov decision processes abstract partially observable markov decision processes pomdps allow one model complex dynamic decision control problems include action outcome uncertainty imperfect observabil ity.
Node-1.4: title learning sorting decision trees pomdps abstract pomdps general models sequential decisions actions observations probabilistic. many problems interest formulated pomdps, yet use pomdps limited lack effective algorithms. recently started change number problems robot navigation planning beginning formulated solved pomdps.
“Node-1.5: title approximating optimal policies partially observable stochastic domains abstract problem making optimal decisions uncertain conditions central artificial intelligence. state world known times, world modeled markov decision process mdp).
Node-1.6: title efficient dynamic programming updates partially observable markov decision processes abstract examine problem performing exact dynamic programming updates partially observable markov decision processes pomdps computational complexity viewpoint.
Node-2: title efficient inference bayes networks combinatorial optimization problem abstract number exact algorithms developed perform probabilistic inference bayesian belief networks recent years.
Node-2.1: title sensitivities alternative conditional probabilities bayesian belief networks abstract show alternative way representing bayesian belief network sensitivities probability distributions. representation equivalent traditional representation conditional probabilities, makes dependencies nodes apparent intuitively easy understand.
Node-2.2: title algebraic techniques efficient inference bayesian networks abstract number exact algorithms developed perform probabilistic inference bayesian belief networks recent years. algorithms use graph theoretic techniques analyze exploit network topology.
Node-2.3: title interpretation complex scenes using bayesian networks abstract object recognition systems, interactions objects scene ignored best interpretation considered set hypothesized objects matches greatest number image features.
Node-2.4: title case based probability factoring bayesian belief networks abstract bayesian network inference formulated combinatorial optimization problem, concerning computation optimal factoring distribution represented net. since determination optimal factoring computationally hard problem, heuristic greedy strategies able find approximations optimal factoring usually adopted. present

paper investigate alternative approach based combination genetic algorithms ga case based reasoning cbr).
`</verbalized-graph>`” “ ### Classification Label Probabilistic Methods

Reasoning

0. Graph Structure Reconstruction:

In the provided verbalized graph, The ROOT node (first line) is the target for classification.
Single-digit indexed nodes are direct neighbors of ROOT.
Double-digit indexed nodes are:
- Two hops away from ROOT
- Direct children of their parent node

More digits indexed nodes follow the same principle as described above.

Thus, the graph structure of this verbalized graph is:

- ROOT
  - Node-1
    - Node-1.1
    - Node-1.2
    - Node-1.3
    - Node-1.4
    - Node-1.5
    - Node-1.6
  - Node-2
    - Node-2.1
    - Node-2.2
    - Node-2.3
    - Node-2.4

1. Word-Level Evaluation:
Detect important terms for the classification label.
Quantitatively, the importance (saliency) scores behind each word in the verbalized graph are calculated by a post-hoc XAI (explainable AI) algorithm.
Semantically, keywords synonymous with or closely related to label, ’Probabilistic Methods’ (such as ’probabilistic’, ’probability’, and ’inference’), are semantically influential in determining the classification because they directly indicate the node’s relevance to the ’Probabilistic Methods’ category.
2. Graph-Level Aggregation:
Node 1 does not contain as many directly relevant keywords. Yet, it still discusses POMDPs, which are probabilistic models, indicating that it contributes to the classification in a less significant way.
Direct neighbors like Node 2, which contains the keyword ’probabilistic’, have a substantial influence on ROOT node classification due to their close proximity to the ROOT node.
Indirect neighbors, Nodes-2.1 to Node-2.4, connected through Node 2, contain keywords like ’probabilistic’ and ’probability’, which further reinforce the classification of ROOT.
Meanwhile, the indirect neighbors connected through Node 1, Node-1.1 to Node-1.6, are less relevant because they do not include important keywords directly related to ’Probabilistic Methods’.

Free-Text Explanation “‘markdown The classification of ROOT node into the "Probabilistic Methods" category can be explained as follows:

ROOT: The presence of keywords like "probabilistic" and "inference" within ROOT node suggests that the paper is focused on topics central to probabilistic methods, such as performing probabilistic inference and evaluating influence diagrams.
- Node-1: While Node 1 focuses on POMDPs, which are also probabilistic models, it is slightly less directly related to the probabilistic inference methods emphasized in ROOT node.
- Node-1.1 Node-1.6: they are less relevant to probabilistic methods and thus contribute less to the classification.
- Node-2: This node discusses "probabilistic inference" and "Bayesian networks," both of which are foundational to probabilistic methods. The strong thematic connection between ROOT node and Node 2 reinforces the classification.
- Node-2.1 Node-2.4: they also discuss probabilistic topics, further supporting the classification.

In summary, the classification of ROOT node into "Probabilistic Methods" is well-supported by the strong presence of key probabilistic terms within the node itself and its direct connection to Node 2, which also focuses on probabilistic inference. Indirect neighbors through Node 2 further reinforce this classification.”

“## Task

Verbalized Graph

`<verbalized-graph>{document}`
`</verbalized-graph>`

Classification Label

`{label}`

Reasoning

Free-Text Explanation

(P.S.: 1. make sure to complete both the reasoning section and then Free-Text Explanation section with the same structure as exemplified above.
2. make good use of the importance (saliency) score behind each word as your guidance to generate the better explanation. However, it is not necessary to directly quote the saliency score.
*3. use the **whole** graph structure you constructed during reasoning for the format of the explanation. Indents and node indexes are necessary, which represent the hierarchy of the graph.)*

G.2 w/ Saliency Prompt

HumanMessage: "The following verbalized graph contains important words in the text of each nodes. These words (each with corresponding importance score in the bracket) contributes to the classification of Node 0 into one of the seven possible categories (['Case Based', 'Genetic Algorithms', 'Neural Networks', 'Probabilistic Methods', 'Reinforcement Learning', 'Rule Learning', 'Theory']).
Generate a concise, human-readable explanation that justifies the classification result of Node 0 by identifying and explaining the relevant inner-node features (i.e., keywords) and inter-node relationships (i.e., graph structure). The explanation should focus on how these factors contribute to the classification label.

Example

Verbalized Graph

`<verbalized-graph>`

ROOT: title(9.13) experiments(7.56) real(2.52) time(2.41) decision(5.20) algorithms(7.18) abstract(12.01) real(3.17) time(2.82) decision(5.46) algorithms(10.39) class(4.34) incremental(2.60) resource(4.50) bounded(5.79) horvitz,(2.67) 89(4.58) anytime(6.66) dean,(4.92) 93(5.03) algorithms(7.94) evaluating(4.75) influence(7.70) diagrams.

Node-1: title(12.47) learning(12.87) policies(9.77) partially(3.11) observable(2.82) environments(5.58) scaling(9.39) abstract(10.80) partially(4.42) observable(2.62) markov(4.50) decision(5.75) processes(4.53) pomdp(9.69) model(11.47) decision(7.63) problems(7.18) agent(12.00) tries(3.13) maximize(3.05) reward(6.03) face(2.13) limited(2.17) noisy(8.96) sensor(6.27) feedback. " "Node-1.1: title(0.95) formal(0.36) framework(0.41) speedup(0.35) learning(0.41) problems(0.48) solutions(0.48) abstract(1.14) speedup(0.33) learning(0.61) seeks(0.57) improve(0.27) computational(0.50) efficiency(0.35) problem(0.37) solving(0.41) experience.(0.57) paper,(0.70) develop(0.53) formal(0.40) framework(0.38) learning(0.37) efficient(0.40) problem(0.34) solving(0.32) random(0.54) problems(0.32) solutions.

Node-1.2: title(2.30) acting(0.98) uncertainty(2.31) discrete(1.03) bayesian(1.13) models(0.94) mobile(1.03) robot(1.75) navigation(1.12) abstract(2.80) discrete(1.18) bayesian(0.97) models(0.81) used(0.66) model(0.56) uncertainty(1.79) mobile(0.77) robot(1.55) navigation,(0.66) question(0.64) actions(1.17) chosen(0.67) remains(0.72) largely(0.56) unexplored.

Node-1.3: title(1.50) incremental(0.64) methods(0.50) computing(0.59) bounds(1.09) partially(0.24) observable(0.21) markov(0.31) decision(0.64) processes(0.38) abstract(0.97) partially(0.25) observable(0.21) markov(0.32) decision(0.58) processes(0.36) pomdps(0.21) allow(0.54) one(0.36) model(0.47) complex(0.38) dynamic(0.60) decision(0.76) control(0.38) problems(0.53) include(0.31) action(0.82) outcome(0.55) uncertainty(0.61) imperfect(0.54) observabil(0.22) ity.(0.36)

Node-1.4: title(0.98) learning(1.07) sorting(1.63) decision(1.56) trees(2.00) pomdps(1.04) abstract(1.34) pomdps(1.10) general(0.42) models(0.59) sequential(0.99) decisions(0.93) actions(0.63) observations(1.27) probabilistic.(0.59) many(0.37) problems(1.06) interest(0.66) formulated(0.98) pomdps. "

"Node-1.5: title(0.90) approximating(0.42) optimal(0.69) policies(0.65) partially(1.13) observable(0.41) stochastic(0.50) domains(0.63) abstract(1.25) problem(0.51) making(0.22) optimal(0.40) decisions(0.42) uncertain(1.01) conditions(0.80) central(0.56) artificial(0.82) intelligence.

Node-1.6: title(1.58) efficient(1.08) dynamic(0.83) programming(1.15) updates(2.24) partially(0.70) observable(0.55) markov(0.87) decision(1.19) processes(0.86) abstract(1.67) examine(0.99) problem(0.72) performing(0.50) exact(0.70) dynamic(0.57) programming(0.75) updates(1.48) partially(1.26) observable(0.58) markov(0.78) decision(1.58) processes(0.78) pomdps(1.18) computational(1.04) complexity(0.75) viewpoint.

Node-2: title(14.46) efficient(7.56) inference(7.77) bayes(5.83) networks(10.92) combinatorial(4.43) optimization(7.56) problem(7.08) abstract(20.68)

number(4.43) exact(10.23) algorithms(14.38) developed(3.37) perform(4.58) probabilistic(5.11) inference(6.22) bayesian(9.36) belief(43.68) networks(17.76) recent(7.91) years.
Node-2.1: title(0.96) sensitivities(0.32) alternative(0.73) conditional(0.51) probabilities(0.29) bayesian(0.53) belief(1.27) networks(0.93) abstract(1.29) show(0.82) alternative(0.44) way(0.32) representing(0.60) bayesian(0.48) belief(1.94) network(1.51) sensitivities(0.25) probability(0.69) distributions.”

“Node-2.2: title(1.81) algebraic(1.21) techniques(0.51) efficient(0.73) inference(0.70) bayesian(0.55) networks(1.04) abstract(1.57) number(0.37) exact(0.77) algorithms(1.75) developed(0.34) perform(0.43) probabilistic(0.35) inference(0.59) bayesian(0.50) belief(1.91) networks(1.06) recent(1.20) years.
Node-2.3: title(1.43) interpretation(0.76) complex(0.42) scenes(0.80) using(0.46) bayesian(0.87) networks(0.89) abstract(1.12) object(0.85) recognition(1.42) systems,(0.46) interactions(0.45) objects(0.43) scene(0.56) ignored(0.38) best(0.26) interpretation(0.69) considered(0.24) set(0.22) hypothesized(0.20) objects(0.39) matches(0.23) greatest(0.26) number(0.20) image(0.50) features.
Node-2.4: title(1.04) case(0.45) based(0.40) probability(0.95) factoring(0.36) bayesian(0.53) belief(1.33) networks(1.03) abstract(1.04) bayesian(0.90) network(0.97) inference(0.87) formulated(0.55) combinatorial(0.30) optimization(0.73) problem,(0.41) concerning(0.65) computation(0.55) optimal(0.54) factoring(0.40) distribution(1.54) represented(0.70) net.
</verbalized-graph>

Classification Label Probabilistic Methods

Reasoning

0. Graph Structure Reconstruction:

In the provided verbalized graph, The ROOT node (first line) is the target for classification.
Single-digit indexed nodes are direct neighbors of ROOT.
Double-digit indexed nodes are:
- Two hops away from ROOT
- Direct children of their parent node

More digits indexed nodes follow the same principle as described above.”

“Thus, the graph structure of this verbalized graph is:

- ROOT
  - Node-1
    - Node-1.1
    - Node-1.2
    - Node-1.3
    - Node-1.4
    - Node-1.5
    - Node-1.6
  - Node-2
    - Node-2.1
    - Node-2.2
    - Node-2.3
    - Node-2.4

1. Word-Level Evaluation:

Detect important terms for the classification label.
Quantitatively, the importance (saliency) scores behind each word in the verbalized graph are calculated by a post-hoc XAI (explainable AI) algorithm.
Semantically, keywords synonymous with or closely related to label, ’Probabilistic Methods’ (such as ’probabilistic’, ’probability’, and ’inference’), are semantically influential in determining the classification because they directly indicate the node’s relevance to the ’Probabilistic Methods’ category.

2. Graph-Level Aggregation:

Node 1 does not contain as many directly relevant keywords. Yet, it still discusses POMDPs, which are probabilistic models, indicating that it contributes to the classification in a less significant way.
Direct neighbors like Node 2, which contains the keyword ’probabilistic’, have a substantial influence on ROOT node classification due to their close proximity to the ROOT node.
Indirect neighbors, Nodes-2.1 to Node-2.4, connected through Node 2, contain keywords like ’probabilistic’ and ’probability’, which further reinforce the classification of ROOT.
Meanwhile, the indirect neighbors connected through Node 1, Node-1.1 to Node-1.6, are less relevant because they do not include important keywords directly related to ’Probabilistic Methods’.

Free-Text Explanation “‘markdown The classification of ROOT node into the "Probabilistic Methods" category can be explained as follows:

ROOT: The presence of keywords like "probabilistic" and "inference" within ROOT node

suggests that the paper is focused on topics central to probabilistic methods, such as performing probabilistic inference and evaluating influence diagrams.
- Node-1: While Node 1 focuses on POMDPs, which are also probabilistic models, it is slightly less directly related to the probabilistic inference methods emphasized in ROOT node.
- Node-1.1 Node-1.6: they are less relevant to probabilistic methods and thus contribute less to the classification.
- Node-2: This node discusses "probabilistic inference" and "Bayesian networks," both of which are foundational to probabilistic methods. The strong thematic connection between ROOT node and Node 2 reinforces the classification.
- Node-2.1 Node-2.4: they also discuss probabilistic topics, further supporting the classification.

In summary, the classification of ROOT node into "Probabilistic Methods" is well-supported by the strong presence of key probabilistic terms within the node itself and its direct connection to Node 2, which also focuses on probabilistic inference. Indirect neighbors through Node 2 further reinforce this classification."

"## Task

Verbalized Graph

`<verbalized-graph>{document}`
`</verbalized-graph>`

Classification Label

`{label}`

Reasoning

Free-Text Explanation

(P.S.: 1. make sure to complete both the reasoning section and then Free-Text Explanation section with the same structure as exemplified above.
2. make good use of the importance (saliency) score behind each word as your guidance to generate the better explanation. However, it is not necessary to directly quote the saliency score.
*3. use the **whole** graph structure you constructed during reasoning for the format of the explanation. Indents and node indexes are necessary, which represent the hierarchy of the graph.)"*