Explaining Graph Neural Networks with Large Language Models: A Counterfactual Perspective for Molecular Property Prediction

Yinhan He[†], **Zaiyi Zheng**[†] **Patrick Soga**[†]
Yaochen Zhu[†] **Yushun Dong**[§] **Jundong Li**[†]

[†]University of Virginia, Charlottesville, VA, USA
[§]Florida State University, Tallahassee, FL, USA
{nee7ne, sjc4fq, zqe3cg, uqp4qh, jl6qk}@virginia.edu, yd24f@fsu.edu

Abstract

In recent years, Graph Neural Networks (GNNs) have become successful in molecular property prediction tasks such as toxicity analysis. However, due to the black-box nature of GNNs, their outputs can be concerning in high-stakes decision-making scenarios, e.g., drug discovery. Facing such an issue, Graph Counterfactual Explanation (GCE) has emerged as a promising approach to improve GNN transparency. However, current GCE methods usually fail to take domain-specific knowledge into consideration, which can result in outputs that are not easily comprehensible by humans. To address this challenge, we propose a novel GCE method, LLM-GCE, to unleash the power of large language models (LLMs) in explaining GNNs for molecular property prediction. Specifically, we utilize an autoencoder to generate the counterfactual graph topology from a set of counterfactual text pairs (CTPs) based on an input graph. Meanwhile, we also incorporate a CTP dynamic feedback module to mitigate LLM hallucination, which provides intermediate feedback derived from the generated counterfactuals as an attempt to give more faithful guidance. Extensive experiments demonstrate the superior performance of LLM-GCE. Our code is released on https://github.com/YinhanHe123/new_LLM4GNNExplanation.

1 Introduction

Molecular property prediction has attracted increasing attention in recent years, where Graph Neural Networks (GNNs) have achieved significant success in the related downstream tasks, such as drug discovery (Xiong et al., 2021) and toxicity analysis (Cremer et al., 2023). However, GNNs are typically considered as black-box models, making it difficult for users to understand how a given prediction is derived. Such a lack of explainability brings obstacles against their broader real-world applications to understand molecular properties.

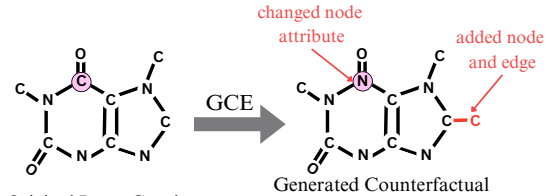

Figure 1: Example of graph counterfactual explanation. (a) the original input molecule graph; (b) an example counterfactual graph.

Facing such an issue, a series of approaches have been proposed to explain the predictions of GNNs, where graph counterfactual explanations (GCE) have become a prevalent approach in recent years (Ying et al., 2019; Lucic et al., 2022; Ma et al., 2022; Zhang et al., 2023a). Specifically, GCE aims to identify the minimum modification over a given graph, such that a trained GNN yields a desired prediction for the post-modified graph. Here, the graph with the identified modification is called the *counterfactual graph*, or simply counterfactual for short. The identified modifications may involve adding or removing nodes and/or edges, as well as altering the node/edge attributes. For instance, given an undesired (non-AIDS drug) molecule as in Fig. 1(a), a GCE method may generate modifications to produce a molecule as a "desired" graph (i.e., predicted as an AIDS-drug by the GNN model) as shown in Fig. 1(b).

However, existing GCE models have two significant limitations: *(i) Incomprehensible Counterfactual Optimization*. Most GCE models are optimized to generate counterfactuals either through heuristic methods, such as random walk (Huang et al., 2023b), or black-box deep learning methods (Ying et al., 2019; Bajaj et al., 2021; Lucic et al., 2022; Ma et al., 2022; Tan et al., 2022), which fail to involve any human-interpretable knowledge in optimizing the counterfactuals. *(ii) Lack of Domain Knowledge*. Most current GCE methods do

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 7079–7096
November 12-16, 2024

not consider any domain-specific knowledge (Ying et al., 2019; Bajaj et al., 2021; Lucic et al., 2022; Ma et al., 2022; Tan et al., 2022) thus the generated counterfactuals may not be realistic in real-world contexts. Continuing the example shown in Fig. 1, although the generated counterfactual Fig. 1(b) is classified as in the desired class, it is not chemically stable since it violates the valence bond theory (Lewis, 1933). To handle the above limitations, large language models (LLMs) (Radford et al., 2018; Wu et al., 2024) are ideal for addressing these limitations due to their ability to *(i)* generate comprehensible natural language texts, *(ii)* make the counterfactual optimization process human-interpretable, and *(iii)* leverage inherent domain knowledge from extensive pretraining to produce realistic counterfactuals. However, harnessing LLMs to improve counterfactual explanation generation faces challenges: *(i)* there exists a natural mismatch between texts (sequential data) and graph structures (Wang et al., 2024; Li et al., 2023); and *(ii)* LLMs may hallucinate, i.e., generate seemingly plausible while incorrect information (Huang et al., 2023a).

To handle these challenges, we propose a novel framework: LLM-GCE (**L**arge **L**anguage **M**odels guided **G**raph **C**ounterfactual **E**xplainer). Specifically, to mitigate the first challenge, instead of directly generating the counterfactual graphs by LLMs, we utilize a counterfactual autoencoder (CA) to construct counterfactual graph structures based on the the text pairs (TPs) and counterfactual text pairs (CTPs) given by LLMs. To tackle the second challenge, the hallucination, we design a CTP dynamic feedback module enlightened by Madaan et al. (2024) to update CTPs iteratively based on previously generated counterfactuals.

Our main contributions are summarized as follows: ***(i) Dataset Construction.*** We collect LLM-generated text pairs over five molecule datasets. They not only support our empirical evaluations but also facilitate future studies for researchers in this field. ***(ii) Algorithmic Design.*** We propose a novel LLM-GCE framework that learns to generate graph counterfactual explanations under the guidance of an LLM. LLM-GCE unlocks LLM's strong reasoning ability in GCE by addressing hallucinations and graph structure inference limitations. ***(iii) Experimental Evaluation.*** We conduct extensive experiments on multiple real-world datasets, validating the effectiveness of LLM in generating more feasible counterfactuals while providing a comprehensive optimization trajectory.

2 Preliminaries

In this section, we introduce the problem settings for GCE and the evaluation metrics used. We denote a molecule graph of m nodes (atoms) as $G = (\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{E})$, where $\boldsymbol{A} \in \{0,1\}^{m\times m}$ is the adjacency matrix, $\boldsymbol{X} \in \mathbb{R}+^{m\times d}$ is the node attribute (atom type) matrix (d is the dimension of the node attributes), and $\boldsymbol{E} \in \mathbb{R}+^{(m(m-1)/2)\times s}$ denotes the edge attribute (bond type) matrix where s is the number of edge attributes. Note that the real-world molecules have 3-D structures; we leave the GCE for 3-D molecule graphs for future work. Furthermore, to determine if a generated counterfactual is classified as desired, we assume that there exists a ground-truth Graph Neural Network (GT-GNN) represented as $\phi : \mathcal{D} \to \mathcal{Y}$, which provides label predictions for graphs in the input graph domain $\mathcal{D}$. This assumption is widely adopted in current literature (Ma et al., 2022; Mahajan et al., 2019). We define the problem of GCE:

Definition 1. ***(Graph Counterfactual Explanation).*** *Let $\phi : \mathcal{D} \to \mathcal{Y}$ be the GT-GNN, and let $G = (\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{E})$ be an input graph with $\phi(G) = 0$. The aim of graph counterfactual explanation is to find a model $f : \mathcal{D} \to \mathcal{D}$ which computes $f(G) = \hat{G}$ where $\hat{G}$ is a minimally perturbed version of G such that $\phi(\hat{G}) = 1$.*

Here, perturbations on the input graph G may include node/edge insertions and removals as well as changes to node/edge features. We refer to $\hat{G}$ as the counterfactual of the original graph G. For an input graph dataset $\mathcal{G}$ sampled from the input graph domain $\mathcal{D}$, we evaluate the performance of a GCE model with the following metrics: ***(i) Validity.*** Validity measures the fraction of the generated counterfactual graphs $f(G)$ for which $\phi(G) = 0$ and $\phi(f(G)) = 1$. Intuitively, it measures how many generated counterfactual graphs actually flip the GT-GNN's prediction of the non-perturbed graph. Accordingly, for convenience, let Valid($\mathcal{G}$) to denote the set $\{f(G) \mid \phi(G) = 0, \phi(f(G)) = 1\}$. We then define the validity metric of $\mathcal{G}$ as

$$\textbf{Validity}(\mathcal{G}) = \frac{|\text{Valid}(\mathcal{G})|}{\{G \in \mathcal{G} \mid \phi(G) = 0\}} \quad (1)$$

(ii) Proximity. Proximity measures the mean graph distance $d(\cdot|\cdot)$ between the original graphs G and their generated valid counterfactuals $f(G) = \hat{G}$ (see Appendix B.1.2). Low proximity indicates

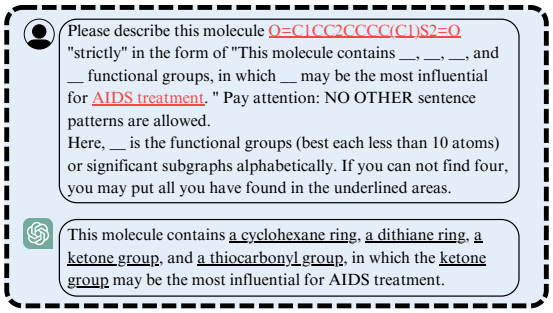

Figure 2: Prompt-answer pair in text pairs generation.

higher-quality counterfactual graphs since they should be made as similar as possible to the graphs they are explaining. The proximity of $\mathcal{G}$ is

$$\textbf{Proximity}(\mathcal{G}) = \frac{\sum_{\hat{G}\in \text{Valid}(\mathcal{G})} d(G, \hat{G})}{|\text{Valid}(\mathcal{G})|}. \quad (2)$$

In alignment with this, we also provide the validity and proximity results with the *feasibility check*, where we only calculate the two metrics to the set of *feasible* counterfactuals $\hat{G}$s, i.e., those counterfactuals that are chemically stable according to valence-bond theory (Lewis, 1933).

3 Data Construction

There exist multiple datasets of molecules paired with text descriptions (Qian et al., 2023; Fang et al., 2023; Zeng et al., 2023); however, most of these datasets only have text pairs describing the graph labeling information. To support the evaluation of GCE methods, a text pair should contain at least three aspects: graph structure information, graph label information, and the significant subgraphs that contribute the most to the graph's label. Generating satisfying text pairs for GCE is expensive since it requires the most advanced LLMs, such as GPT-4 and customized prompts, so we release five molecule datasets with our generated high-quality text pairs for the convenience of the community.

Construction Process. We construct five new text-paired graph datasets based on datasets commonly used in graph explanation (Abrate and Bonchi, 2021; Ying et al., 2019; Huang et al., 2023b), including AIDS and Mutagenicity from TUDataset (Morris et al., 2020); BBBP, ClinTox, and Tox21 from MoleculeNet (Ramsundar et al., 2019). In all datasets, each graph is a molecule with a binary label indicating a molecular property such as AIDS treatment effectiveness (see details in Appendix B.2.2). To generate the text-paired graphs, we take the following steps: (1) *Dataset Preprocessing.* We first convert all input molecular graphs to their SMILES representations (Weininger, 1988). We remove molecules that only have one atom or greater than 100 atoms. For Tox21, since the graph label distribution is heavily skewed (more than 95% of the labels are 0), we randomly select 600 zero-labeled graphs from the dataset. (2) *Text Pair Generation.* Using a custom prompt, we prompt GPT-4 with the SMILES representations and graph labeling semantics for each input graph and ask for a *description* of the molecule's graph structure, graph label, and subgraphs that are most responsible for its label. (3) *Data Post Processing.* For some graphs, the responses from GPT-4 would be erroneous (i.e., it does not identify the significant subgraphs/functional groups correctly). We fix this by reprompting until desired response is generated.

Prompt Design. We generate a text pair (TP) for each graph in a dataset with LLMs, incorporating the graph structure, label semantics, and significant subgraphs. Our prompts use the following template: "Please describe this graph strictly in the form of 'This graph contains __, __, __, and __ significant subgraphs, in which __ may be the most influential for `the_label_semantic`.' No other sentence patterns are allowed. If you can not find four, you may put all you have found in the underlined areas." The first half lists the significant subgraphs, revealing the graph's structural information. The second half provides the label semantics so the LLM determines the most significant subgraph for graph labeling.

4 Methodology

4.1 Model Overview

An overview of the LLM-GCE model is shown in Fig. 3. The proposed LLM-GCE has three modules: (1) *Contrastive Pretraining of Text Encoder.* We pretrain the text encoder with contrastive learning to align the embeddings of the GT-GNN and the text encoder. (2) *Training of the Counterfactual Autoencoder.* We design a counterfactual autoencoder composed of the pretrained text-encoder and a graph decoder, which is trained to recover the counterfactual topology. (3) *Dynamic Feedback of CTP Generation*. To tackle hallucination, we prompt the generated counterfactuals with the GT-GNN predictions back to the LLM as the dynamic feedback for further calibration.

4.2 Contrastive Text Encoder Pretraining

The first step of LLM-GCE is to pretrain the text encoder so that every TP's embedding aligns with

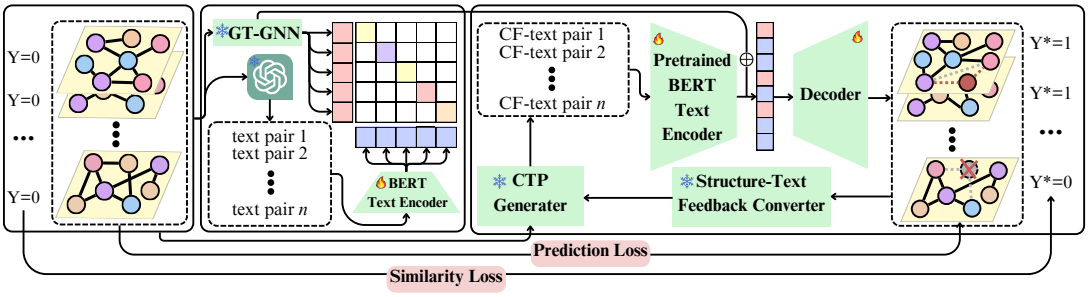

Figure 3: An overview of the proposed LLM-GCE model.

its corresponding graph embedding produced by GT-GNN (via a projection implemented by a multilayer perception (MLP) (Haykin, 1994)). We choose BERT (Kenton and Toutanova, 2019) as the text encoder due to its proven effectiveness in generating high-quality embeddings for natural language processing tasks. In the graph domain $\mathcal{D}$, we employ a contrastive learning strategy to align the text encoder's embeddings with the GT-GNN's embeddings. First, we sample from $\mathcal{D}$ a dataset $\mathcal{G} = \{G_i\}_{i=1}^{n}$ and their TPs. We then train the text encoder with batches of size N, which includes $N \times N$ possible graph-TP pairings. Next, we train an MLP to project the TP embeddings from the BERT (Kenton and Toutanova, 2019) text encoder into the embedding space of the GT-GNN and maximize the cosine similarity for the matching graph-TP embedding pairs while minimizing similarity for non-matching pairs. The contrastive loss is the symmetric cross-entropy given in Appendix C.1.

4.3 Training of the CA

After pretraining the text encoder, we generate counterfactuals from natural language. We first provide an overview of our counterfactual autoencoder's (CA) architecture. Then, we elaborate on each of its components, including CTP generation, the text encoder, the latent embedding combination, the graph decoder, and finally, we introduce the overall objective function for our LLM-GCE.

4.3.1 Architecture Overview.

We first generate a CTP, CTP_G, for each input graph G as a high-level instruction for GCE. Next, inspired by graph variational autoencoders (VGAEs) (Simonovsky and Komodakis, 2018), we design a CA with a text encoder and a graph decoder. The text encoder, pretrained as introduced in Section 4.2, maps the CTP_G to a probability distribution over a latent space, which is then decoded via an MLP to the counterfactual graph's adjacency matrix $\hat{\boldsymbol{A}}$ and node & edge attribute matrices $\hat{\boldsymbol{X}}$ & $\hat{\boldsymbol{E}}$. Training the CA maximizes the likelihood of $\hat{G}$ being a real counterfactual, i.e., $P(\hat{G}|G, \text{CTP}_G, Y^* = 1)$.

4.3.2 CTP Generation.

For each graph G from the dataset $\mathcal{G}$, we query the LLM for a CTP, a natural language sentence describing the potential counterfactual $\hat{G}$ of G. We use this CTP to instruct the generation of counterfactuals from the autoencoder. The CTP generation prompt is shown in Figure 4, which aims to perform functional group substitution to achieve a higher probability of generating a counterfactual.

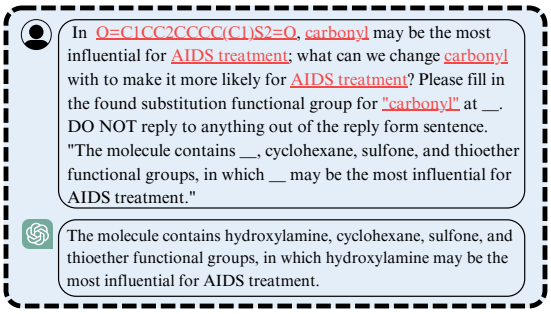

Figure 4: Illustration of the prompt for CTP generation. The red underlined phrases vary from graph to graph. The LLM strictly follows the input sentence pattern in the answer, which is the same as the one used for TPs.

4.3.3 Text Embedding Generation.

The text encoder maps our generated CTPs into a latent space for counterfactual structure reconstruction. The input to our text encoder is the CTP of each input graph, denoted as $\{\text{CTP}_{Gi}\}_{\{i=1\}}^{n}$, where $n = |\mathcal{G}|$ is the number of input graphs. Given CTP_G, the text encoder generates a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_G, \boldsymbol{\sigma}_G)$ in the latent space, where $\boldsymbol{\mu}_G \in \mathbb{R}^l$ and $\boldsymbol{\sigma}_G \in \mathbb{R}^l$ are the distribution's mean and variance. The output of the encoder is a latent embedding $\boldsymbol{z} \in \mathbb{R}^l$ sampled from $\mathcal{N}(\boldsymbol{\mu}_G, \boldsymbol{\sigma}_G)$. In practice, the text encoder is implemented with BERT (Kenton and Toutanova, 2019) and initialized with the pretrained parameters from Section 4.2. Recall that our contrastive pretraining objective encourages the latent text embedding of each TP_G to be close to its corresponding graph

latent embedding provided by the GT-GNN under a projection by an MLP. Therefore, although the exact graph structure, node, and edge attributes of the desired counterfactual $\hat{G}$ are not immediately accessible, the embedding of the CTP approximates $\hat{G}$'s GT-GNN embedding, which guides counterfactual decoding. Intuitively, the text encoder's embeddings contain the information of a high-level counterfactual generation instruction, while the GT-GNN's embeddings encode each counterfactual's graph structure and node & edge attributes. Next, we introduce how we combine the embedding of the text encoder with that of the GT-GNN.

4.3.4 Latent Embedding Combination.

The generated CTP and acquired latent embedding from the text encoder for an input graph G are still insufficient for counterfactual generation. This is for two reasons. (1) Each CTP, as introduced in Section 4.3.2, only contains information about the significant subgraphs (e.g., functional groups for molecule graphs) of the counterfactual $\hat{G}$ while the specific structure of the counterfactual is not described in detail. (2) While the pretraining process enhances the consistency of each text embedding with its corresponding GT-GNN embedding, the limited availability of a large amount of pretraining data prevents the model from achieving high pretraining accuracy. Therefore, to merge information from both the counterfactual text and graph embeddings, we update the final encoded embedding e_G by concatenating the text encoder embeddings $\boldsymbol{z}_G$ of $\{\text{CTP}_{Gi}\}_{\{i=1\}}^{n}$ with the GT-GNN embeddings $\boldsymbol{q}_G$ of the input graphs $\mathcal{G}$, written as $\boldsymbol{e}_G = \boldsymbol{z}_G \oplus \boldsymbol{q}_G$.

4.3.5 Graph Decoder.

In the graph decoder, the resulting latent embeddings $\{\boldsymbol{e}_{G_i}\}_{i=1}^{n}$ are used to reconstruct the adjacency matrix $\hat{\boldsymbol{A}}_G \in \mathbb{R}^{m\times m}$, node attribute matrix $\hat{\boldsymbol{X}}_G \in \mathbb{R}^{m\times d}$, and edge attribute matrix $\hat{\boldsymbol{E}}_G \in \mathbb{R}^{m(m-1)/2\times s}$ of the counterfactual graph $\hat{G}$ for each input graph G. The graph decoder is implemented as an MLP with a sigmoid activation, restricting its output range to $(0, 1)$. As a result, every entry of the generated matrices is a continuous probabilistic real number. However, a real graph's adjacency matrix is a $0, 1$-matrix, where $\hat{\boldsymbol{A}}_{ij} = 1$ indicates the presence of an edge between nodes i and j, and $\hat{\boldsymbol{A}}_{ij} = 0$ indicates the absence of an edge. Furthermore, each row of the node/edge attribute matrix is a one-hot code indicating the discrete node/edge type. To make the decoder compatible with these constraints, we discretize the generated adjacency matrix $\hat{\boldsymbol{A}}_{\hat{G}}$ by thresholding its probabilistic entries, setting entries to 1 if the corresponding value exceeds 0.5 and to 0 otherwise. Similarly, we generate the one-hot node and edge attribute matrices $\hat{\boldsymbol{X}}_{\hat{G}}$ and $\hat{\boldsymbol{E}}_{\hat{G}}$ by taking the one-hot row-wise argmax of each probabilistic matrix.

4.3.6 Objective Function.

As discussed in Section 4.3.1, the optimization target is to maximize the likelihood of the generated graph being a real counterfactual $\hat{G}$ conditioned on CTP_G and the desired label Y^*, denoted as $P(\hat{G}|G, \text{CTP}_G, Y^* = 1)$. We formalize this objective with the Kullback-Leibler (KL) divergence (Csiszár, 1975) of the distribution given by the encoder $Q(\boldsymbol{e}|G, \text{CTP}_G, Y^*)$ and the posterior distribution $P(\boldsymbol{e}|\hat{G}, \text{CTP}_G, Y^*)$. For simplicity, we write the condition $\{\text{CTP}_G, Y^*\}$ as T, and the divergence term $\text{KL}[Q(\boldsymbol{e}|G, T)||P(\boldsymbol{e}|\hat{G}, T)] = -\mathbb{E}_{\boldsymbol{e}\sim Q}[\log P(\boldsymbol{e}|\hat{G}, T) - \log Q(\boldsymbol{e}|G, T)]$. By $\log P(\boldsymbol{e}|\hat{G}, T) = \log P(\boldsymbol{e}|T) + \log P(\hat{G}|\boldsymbol{e}, T)$, we have the equation

$$\begin{aligned} &\log P(\hat{G}|\text{CTP}_G, Y^*) - \text{KL}[Q||P] \\ =&\mathbb{E}_{\boldsymbol{e}\sim Q}[\log P(\hat{G}|z, T)] - \text{KL}[Q||P], \end{aligned} \tag{3}$$

where $Q = Q(\boldsymbol{e}|G, T)$ and $P = P(\boldsymbol{e}|\hat{G}, T)$. In this equation, the first term in the left-hand side (LHS) is our optimization target, and the second term is a KL divergence, which is inaccessible since the posterior $P(\boldsymbol{e}|\hat{G})$ is intractable. However, the right-hand side is available for direct calculation. Therefore, we optimize the log-likelihood $P(\hat{G}|G, \text{CTP}_G, Y^* = 1)$ with its Evidence Lower Bound (ELBO) (Kingma and Welling, 2013):

$$\begin{aligned} \log P(\hat{G}|T) \geq &\mathbb{E}_{\boldsymbol{e}\sim Q}[\log P(\hat{G}|z, T)] \\ &- \text{KL}[Q(\boldsymbol{e}|G, T)||P(\boldsymbol{e}|T)]. \end{aligned} \tag{4}$$

However, due to the lack of the ground-truth counterfactual $\hat{G}$, we substitute the first term of RHS in Equ. (4) with two loss terms. (1) *Graph Distance Loss ($\mathcal{L}_{dist}$)* encourages small graph distances between G and its counterfactual $\hat{G}$. Formally,

$$\mathcal{L}_{\text{dist}} := \Sigma_{\{G\in\mathcal{G}\}} r(G, \hat{G}), \tag{5}$$

where $r(\cdot, \cdot)$ is the weighted sum of the distances between graph adjacency matrices $r_{\boldsymbol{A}}(\cdot, \cdot)$, node attribute matrices $r_{\boldsymbol{X}}(\cdot, \cdot)$, and edge attribute matrices $r_{\boldsymbol{E}}(\cdot, \cdot)$, i.e., $r(G, \hat{G}) = r_{\boldsymbol{A}}(\boldsymbol{A}_G, \boldsymbol{A}_{\hat{G}}) + r_{\boldsymbol{X}}(\boldsymbol{X}_G, \boldsymbol{X}_{\hat{G}}) + r_{\boldsymbol{E}}(\boldsymbol{E}_G, \boldsymbol{E}_{\hat{G}})$ (see Appendix B.1.2

for definitions). (2) *Counterfactual Prediction Loss ($\mathcal{L}_{pred}$)* is the log-likelihood that the generated counterfactual is classified as desired by GT-GNN:

$$\mathcal{L}_{\text{pred}} := -\log P_{\text{GT-GNN}}(\phi(\hat{G}) = Y^*). \quad (6)$$

In conclusion, our overall loss function is

$$\mathcal{L} = \alpha\mathcal{L}_{\text{dist}} + \beta\mathcal{L}_{\text{pred}} - \mathcal{L}_{\text{KL}}, \quad (7)$$

$$\mathcal{L}_{\text{KL}} = \text{KL}[Q(\boldsymbol{e}|G, \text{CTP}_G, Y^*)||P(\boldsymbol{e}|\text{CTP}_G, Y^*).$$

4.4 Dynamic Feedback of CTP Generation

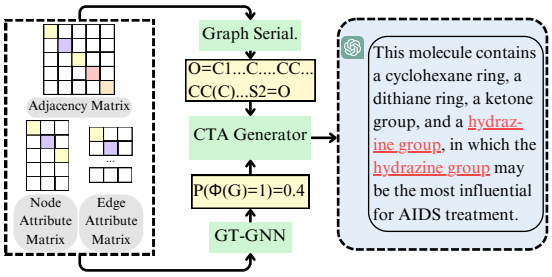

Figure 5: Illustration of the CTP's dynamic feedback. "Graph Serial." is short for "Graph serialization" which converts graphs to their SMILES representations.

The CTP generator acts as a commander, giving high-level instructions (CTPs) for counterfactual graph generation. The CA is the executor, implementing the instruction to create a specific counterfactual graph. However, CTPs can be inaccurate due to LLM hallucination (Zhang et al., 2023b) and limited graph decoding ability. To remedy this, a CTP dynamic feedback scheme is designed to calibrate CTP with the CA-generated counterfactuals.

An illustration of the scheme is shown in Fig. 5. We first convert the counterfactual molecule generated by the CA into its SMILES representation. Then, we combine it with the GT-GNN's probability that the graph is a valid counterfactual into a specific prompt asking for a new, calibrated CTP for the original graph G. This concludes a single iteration of dynamic feedback, which we treat as a hyperparameter. Each iteration of dynamic feedback can be seen as an indirect reasoning step, forcing the model to reflect on its past outputs and the label information from the GT-GNN in order to produce more truthful CTPs. A similar calibration approach is verified effective in Dhuliawala et al. (2023); Madaan et al. (2024)

5 Experiments

In this section, we evaluate LLM-GCE with extensive experiments on five real-world datasets. Our experiments aim to answer the following research questions (RQs): **RQ1**: How does LLM-GCE perform w.r.t. validity and proximity compared with state-of-the-art baselines? **RQ2**: How does each component of LLM-GCE affect its overall performance? **RQ3:** What insights can LLM-GCE provide given its counterfactual explanation results?

5.1 Experimental Setup

5.1.1 Datasets.

Our experiments utilize five real-world datasets (AIDS, Mutagenicity, BBBP, ClinTox, Tox21). Among them, AIDS and Mutagenicity are from TUDataset (Morris et al., 2020), while BBBP, SIDER and Tox21 are from MoleculeNet (Ramsundar et al., 2019). In these datasets, graphs represent chemical compounds with nodes as atoms and edges as bonds. They are labeled based on relevance to properties such as blood-brain barrier penetration, mutagenicity, HIV activity, side effects resource, and toxicological activity. We associate each graph with a text pair by the procedure in Section 3. For details, see Appendix B.2.2.

5.1.2 Baselines

We adopt the following state-of-the-art baselines. (1) **GNNExplainer** (Ying et al., 2019) is a graph factual explanation (GFE) model. GFE is a graph explanation strategy slightly different from GCE, and so we adjust it for GCE by revising its loss function. See Appendix B.2.1 for details. (2) **CF-GNNExplainer** (Lucic et al., 2022) is targeted at node-level GCE. We adapt it for graph-level GCE by changing the input from ego-graphs to whole graphs and changing the supervisory signal from node labels to graph labels. (3) **CLEAR** (Ma et al., 2022) is a generative GCE model that is based on graph variational autoencoders (Simonovsky and Komodakis, 2018). We adapt it by removing its causality-specific component for a fair comparison. (4) **RegExplainer** (Zhang et al., 2023a) is a GCE model for graph regression that can be directly adapted to graph classification. We evaluate those GCE baselines based on Appendix B.1.2.

5.1.3 GNN Classifier

Before training LLM-GCE, we first train a two-layer Graph Convolutional Network (GCN) slightly modified to incorporate edge attributes (see Appendix B.1.1) for the five datasets to serve as the GT-GNN. The GT-GNN has a node embedding dimension of 32, a maximum pooling layer,

Table 1: The performance of different GCE methods. The best results are in bold, and the runner-up results are underlined. 'n/a' refers to unavailable proximity scores since there is no valid counterfactual graph.

			GNNExplainer	CF-GNNExplainer	CLEAR	RegExplainer	LLM-GCE
AIDS	**Validity**	w. Feas.	0.25 ± 0.00	0.25 ± 0.00	$\mathbf{2.56 \pm 3.10}$	0.25 ± 0.00	0.25 ± 0.27
		w/o Feas.	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
	Proximity	w. Feas.	$\mathbf{2.00 \pm 0.00}$	n/a	109.82 ± 1.18	7.28 ± 0.00	5.86 ± 5.62
		w/o Feas.	$\mathbf{4.27 \pm 0.00}$	n/a	109.57 ± 0.96	15.11 ± 0.00	86.37 ± 2.65
Mutagenicity	**Validity**	w. Feas.	0.00 ± 0.00	0.00 ± 0.00	4.89 ± 6.92	0.00 ± 0.00	$\mathbf{13.52 \pm 9.93}$
		w/o Feas.	65.96 ± 0.00	50.17 ± 2.32	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
	Proximity	w. Feas.	n/a	n/a	406.42 ± 0.00	n/a	$\mathbf{1.75 \pm 2.48}$
		w/o Feas.	27.71 ± 0.078	$\mathbf{8.77 \pm 0.48}$	409.52 ± 1.11	24.91 ± 0.00	56.31 ± 2.57
BBBP	**Validity**	w. Feas.	0.74 ± 0.00	0.00 ± 0.00	9.56 ± 8.34	0.74 ± 0.00	$\mathbf{38.25 \pm 10.25}$
		w/o Feas.	22.79 ± 0.00	34.38 ± 4.42	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
	Proximity	w. Feas.	16.06 ± 0.00	n/a	141.85 ± 0.71	18.25 ± 0.00	$\mathbf{11.68 \pm 0.14}$
		w/o Feas.	17.26 ± 0.00	$\mathbf{6.84 \pm 0.76}$	142.00 ± 0.11	27.45 ± 0.00	55.98 ± 5.79
ClinTox	**Validity**	w. Feas.	0.00 ± 0.00	0.00 ± 0.00	1.33 ± 1.89	0.00 ± 0.00	$\mathbf{30.86 \pm 26.41}$
		w/o Feas.	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
	Proximity	w. Feas.	n/a	n/a	50.10 ± 72.27	n/a	$\mathbf{0.95 \pm 0.93}$
		w/o Feas.	$\mathbf{16.61 \pm 0.00}$	n/a	150.50 ± 0.97	28.74 ± 0.00	42.89 ± 3.72
Tox21	**Validity**	w. Feas.	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{3.70 \pm 4.01}$
		w/o Feas.	0.21 ± 0.00	0.00 ± 0.00	$\mathbf{100.00 \pm 0.00}$	0.00 ± 0.00	$\mathbf{100.00 \pm 0.00}$
	Proximity	w. Feas.	n/a	n/a	n/a	n/a	$\mathbf{5.93 \pm 4.91}$
		w/o Feas.	$\mathbf{18.28 \pm 0.00}$	n/a	190.70 ± 1.72	n/a	77.34 ± 4.28

Table 2: GT-GNN accuracy (%) on adopted datasets.

	AIDS	Muta.	BBBP	ClinTox	Tox21
Training	0.995	0.781	0.887	0.749	0.970
Validation	0.996	0.722	0.852	0.782	0.972
Testing	0.994	0.762	0.850	0.783	0.974

and a fully connected layer for graph classification. We set the edge embedding dimension d to 1. The model is trained with the Adam optimizer (Kingma and Ba, 2014) with a learning rate 1e-3 for 500 epochs. The train/validate/test split is 50%/25%/25%. Accuracies are shown in Table 2.

5.1.4 Explainer Settings.

We detail the experimental settings for pretraining (the second part in Fig. 3) and training (the third part in Fig. 3) stages of our pipeline. The list of all the prompts used are in Appendix B.1.3.

Pretraining. We use GPT-4 as the TP generator. The BERT text encoder is implemented through Huggingface (Wolf et al., 2020). We use the AdamW optimizer (Loshchilov and Hutter, 2018) for pretraining BERT over 100 epochs with a learning rate of 0.01.

Training. We choose GPT-3.5-Turbo as the CTP generator, where feedback is performed three iterations in every epoch. Finetuning hyperparameters are the same as those used in pretraining.

5.2 RQ1: Performance of Different Methods

We evaluate our LLM-GCE framework on five real-world datasets and compare its validity and proximity performance against state-of-the-art baselines, with and without a feasibility (Feas.) check (Table 1) as determined by checking for stability under valence theory with RDKit (RDKit, online). We have the following observations: (1) From the perspective of validity, LLM-GCE achieves comparable validity with other baselines without the chemical feasibility check. However, with the feasibility check, our model achieves the highest validity among almost all baselines across all datasets. (2) From the perspective of proximity, LLM-GCE achieves the lowest proximity among chemically feasible counterfactuals among almost all baselines. This means that our model can find valid counterfactuals that are not only feasible but also have a minimal graph distance from their corresponding input graph. (3) Based on these observations, LLM-GCE achieves satisfying graph counterfactual explanation performance, especially when chemical feasibility is considered. This reveals the effectiveness of LLM's pretrained knowledge and reasoning abilities in GCE. For details on the efficiency of LLM-GCE, see Appendix D.2.

5.3 RQ2: Ablation Study

We experiment with three ablated variants of LLM-GCE: **LLM-GCE-NP**: Without pretraining our BERT text-encoder, we directly train the counterfactual autoencoder with a dynamic CTP feedback module. **LLM-GCE-NT**: We freeze the counterfactual autoencoder in LLM-GCE, i.e., the counterfactual autoencoder is not optimized between two dynamic CTP feedback iterations. **LLM-GCE-NF**: We remove the CTP dynamic feedback module and

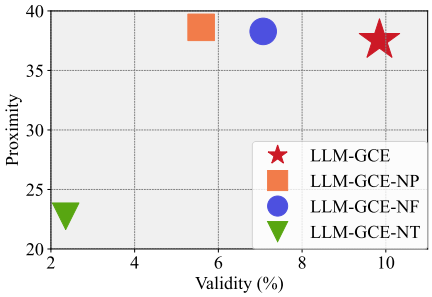

Figure 6: Ablation study on BBBP. -NP: No Pretraining. -NF: No Feedback. -NT: Bert Autoencoder Frozen.

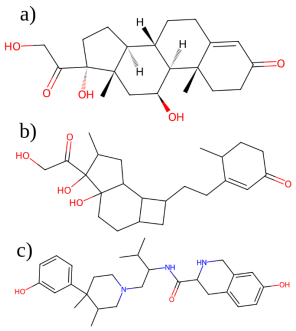

Figure 7: Generated molecules by LLM-GCE and GNNExplainer. a) from ClinTox, b) by LLM-GCE, c) by GNNExplainer. Proximity of b) is 16.52, c) is 24.40.

generate graph counterfactuals only from the autoencoder and the initial CTP. From Fig. 6, we make the following observations. (1) LLM-GCE-NP: Refraining from pretraining decreases model validity and increases proximity. The small decline in performance is possibly due to insufficient data for contrastive learning. (2) LLM-GCE-NT: Validity dramatically degrades when the autoencoder is frozen. Proximity also decreases, possibly because a frozen autoencoder can only generate a small number of counterfactuals that are less diverse. (3) LLM-GCE-NF: Removing the feedback module significantly reduces validity and slightly increases proximity, revealing the importance of dynamic feedback for GCE. We have similar observations on other datasets. For more ablation studies, see Appendix D.5. Additionally, we substitute the text encoder of the CA and CTP generator to other language models, see the results in Appendix D.4.

5.4 RQ3: Case Study

We highlight two main advantages of LLM-GCE: (1) *More Faithful Counterfactuals.* The generated counterfactuals by LLM-GCE have consistently lower proximity across datasets than that of the baseline methods. We show a counterfactual generated by our LLM-GCE compared with that by GNNExplainer in Fig. 7. Molecule a) is from the ClinTox dataset, b) is generated by LLM-GCE, and c) by GNN Explainer. LLM-GCE's output better preserves the original molecule's structural integrity than GNNExplainer. (2) *More Feasible Counterfactuals.* With the LLMs' extensive domain knowledge, our LLM-GCE generates more counterfactuals satisfying valence bond theory, while those given by baselines do not satisfy this theory. Only CLEAR generates a high proportion of feasible ones, but they reveal no chemical insight as most are disconnected carbon atoms, misleadingly passing the feasibility check (Weininger, 1988). For more case studies, see Appendix D.3.

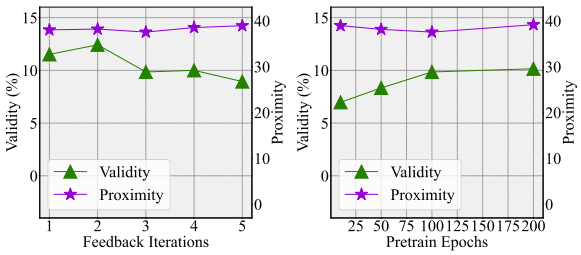

Figure 8: Parameter Analysis on BBBP. (a) Validity and proximity w.r.t. CTP feedback iterations. (b) Validity and proximity w.r.t. text encoder pretrain epochs.

5.5 RQ4: Parameter Analysis

We study how dynamic CTP feedback iterations and text encoder pretrain epochs affect LLM-GCE's performance on BBBP. We test feedback iterations from 1 to 5 and pretraining epochs in {10, 50, 100, 200}. We observe from Fig. 8 that (1) validity improvement saturates after two feedback iterations while proximity remains nearly unchanged. While a small number of feedback iterations can enhance performance, it is possible that simply increasing rounds worsens hallucinations. Also, (2) increasing pretraining epochs to around 100 boosts validity. At the same time, proximity initially improves up to 100 epochs, but then gradually increases. This trend suggests a correlation between validity and proximity regarding pretraining epochs, with the ideal count being approximately 100. Similar findings are observed on other datasets. For more results, see Appendix D.5.

6 Conclusion

In this work, we explore the ability of LLMs in guiding the GCE for molecule properties prediction. Specifically, we propose a novel model called LLM-GCE, comprised of a contrastive pre-training module, a counterfactual autoencoder, and a dynamic feedback module. Extensive experiments validate the superior performance of LLM-GCE.

7 Acknowledgement

This work is supported in part by the National Science Foundation under grants (IIS-2006844, IIS-2144209, IIS-2223769, CNS-2154962, BCS-2228534, and CMMI-2411248), Office of Naval Research under grant (N000142412636), the Commonwealth Cyber Initiative Awards under grants (VV-1Q24-011, VV-1Q24-011), and the research gift funding from Netflix and Snap.

8 Limitations

Firstly, the effectiveness of LLM-GCE relies heavily on the quality and relevance of the pretraining data used for the large language models. If the pretraining data is biased or lacks sufficient coverage of the target domain, it may lead to less accurate or relevant counterfactuals being generated. Ensuring the LLMs are trained on high-quality, domain-specific data is crucial for optimal performance.

Additionally, the computational cost associated with using large language models can be a drawback. Training and inference with LLMs are more time-consuming and resource-intensive than other graph counterfactual explanation methods. Although we have shown that the time execution time is comparable for our methods with other baselines on the adopted datasets, LLM-GCE can be time consuming when dealing with extremely large-scale graphs, such as some big proteins.

Furthermore, while the experiments demonstrate the effectiveness of LLM-GCE on several real-world datasets, the evaluation is still limited to specific domains, such as molecular property prediction. The generalizability of the proposed framework to other types of graphs and application areas remains to be investigated. Further research is needed to assess the performance and adaptability of LLM-GCE across a wider range of graph structures and problem domains.

Lastly, the potential for hallucinations or inconsistencies in the generated counterfactuals remains a solid challenge. Although the dynamic feedback module aims to mitigate this issue, there may still be cases where the LLMs produce counterfactuals that are not entirely faithful to the original graph or the desired properties.

9 Ethics Statement

In this work, we propose a novel LLM-guided graph counterfactual explanation method that utilizes the strong reasoning ability of LLMs. We do not anticipate any ethical issues that should be specifically highlighted in this paper.

References

Carlo Abrate and Francesco Bonchi. 2021. Counterfactual graphs for explainable classification of brain networks. In *SIGKDD*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv*.

Mohit Bajaj, Lingyang Chu, Zi Yu Xue, Jian Pei, Lanjun Wang, Peter Cho-Ho Lam, and Yong Zhang. 2021. Robust counterfactual explanations on graph neural networks. *NeurIPS*.

Julian Cremer, Leonardo Medrano Sandonas, Alexandre Tkatchenko, Djork-Arné Clevert, and Gianni De Fabritiis. 2023. Equivariant graph neural networks for toxicity prediction. *Chemical Research in Toxicology*.

Imre Csiszár. 1975. I-divergence geometry of probability distributions and minimization problems. *Ann. Probab.*

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv*.

Yin Fang, Xiaozhuan Liang, Ningyu Zhang, Kangwei Liu, Rui Huang, Zhuo Chen, Xiaohui Fan, and Huajun Chen. 2023. Mol-instructions: A large-scale biomolecular instruction dataset for large language models. *arXiv*.

Simon Haykin. 1994. *Neural networks: a comprehensive foundation*.

Lei Huang, Weijiang Yu, et al. 2023a. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv*.

Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. 2023b. Global counterfactual explainer for graph neural networks. In *WWW*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv*.

Gilbert N Lewis. 1933. The chemical bond. *J. Chem. Phys.*

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv*.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2023. A survey of graph meets large language model: Progress and future directions. *arXiv*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *ICLR*.

Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. 2022. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *AISTATS*.

Jing Ma, Ruocheng Guo, Saumitra Mishra, Aidong Zhang, and Jundong Li. 2022. Clear: Generative counterfactual explanations on graphs. In *NeurIPS*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *NeurIPS*.

Divyat Mahajan, Chenhao Tan, and Amit Sharma. 2019. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv*.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*.

Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. 2023. A survey on graph counterfactual explanations: Definitions, methods, evaluation, and research challenges. *ACM Comput. Surv.*

Chen Qian, Huayi Tang, Zhirui Yang, Hong Liang, and Yong Liu. 2023. Can large language models empower molecular property prediction? *arXiv*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and et. al. Sutskever, Ilya. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. 2019. *Deep Learning for the Life Sciences*. O'Reilly Media.

RDKit, online. RDKit: Open-source cheminformatics. http://www.rdkit.org. [Online; accessed 11-April-2013].

Murray Shanahan. 2024. Talking about large language models. *CACM*.

Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *ICANN*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *NeurIPS*, 27.

Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. 2022. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *WWW*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NeurIPS*.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024. Can language models solve graph problems in natural language? *NeurIPS*.

David Weininger. 1988. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput.*

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, and etc. Moi, Anthony. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*.

Haoran Wu, Wei Chen, Shuang Xu, and Bo Xu. 2021. Counterfactual supporting facts extraction for explainable medical record based diagnosis with graph network. In *NAACL*.

Xuansheng Wu, Haiyan Zhao, Yaochen Zhu, Yucheng Shi, Fan Yang, Tianming Liu, Xiaoming Zhai, Wenlin Yao, Jundong Li, Mengnan Du, et al. 2024. Usable xai: 10 strategies towards exploiting explainability in the llm era. *arXiv*.

Jiacheng Xiong, Zhaoping Xiong, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. 2021. Graph neural networks for automated de novo drug design. *Drug discovery today*.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv*.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *NeuriPS*.

Zheni Zeng, Bangchen Yin, Shipeng Wang, Jiarui Liu, Cheng Yang, Haishen Yao, Xingzhi Sun, Maosong Sun, Guotong Xie, and Zhiyuan Liu. 2023. Interactive molecular discovery with natural language. *arXiv*.

Jiaxing Zhang, Zhuomin Chen, Hao Mei, Dongsheng Luo, and Hua Wei. 2023a. Regexplainer: Generating explanations for graph neural networks in regression task. *arXiv*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023b. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv*.

A Potential Risks

One risk of our LLM-GCE is the misuse or misinterpretation of the generated counterfactual explanations. If the explanations are not carefully validated or if the users lack the necessary domain knowledge, they may make incorrect decisions based on the provided counterfactuals. This could lead to adverse consequences, especially in sensitive domains such as healthcare or finance. Moreover, the reliance on large language models raises concerns about the perpetuation of biases present in the pretraining data. If the LLMs exhibit biases, these may be propagated through the generated counterfactuals, potentially leading to unfair or discriminatory explanations.

B Reproducibility

In this section, we provide more details of model implementation of our LLM-GCE and experiment setup of our evaluation results.

B.1 Details of the Model Implementation

B.1.1 GNN as the Prediction Model

We train a two-layer Graph Convolutional Network (GCN) on molecular graphs across all five datasets as GT-GNN with a slight change to incorporate various edge types. Specifically, we calculate the edge embeddings for each type of edge separately and construct the enhanced adjacency matrix $\hat{A}$ used as $\hat{A}_{ij} = A_{ij} * E_{ij}$ for all $1 \leq i, j \leq m$, where m is the number of nodes in the graph. $A \in [0, 1]^{m \times m}$ denotes the original adjacency matrix that allows the elements to be decimal numbers, $E \in \mathbb{R}^{m \times m \times d}$ represents the embedding of each edge (with a dimension of d). The GNN has a node embedding dimension of 32, a maximum pooling layer, and a fully connected layer for graph classification. We set the edge embedding dimension d as 1. The model is trained with an adaptive moment estimation optimizer (Adam) (Kingma and Ba, 2014) with a learning rate of 1e-3 for 500 epochs. The train/validate/test split is 50%/25%/25%. The accuracy of the GNNs is shown in Table 2

B.1.2 Graph Distance Calculation

We approximate graph edit distance with

$$\begin{aligned} d(g, g^{cf}) =& \lambda_1 \cdot d_{\boldsymbol{A}}(\boldsymbol{A}_g, \boldsymbol{A}_{g^{cf}}) \\ &+ \lambda_2 \cdot d_{\boldsymbol{X}}(\boldsymbol{X}_g, \boldsymbol{X}_{g^{cf}}) \\ &+ \lambda_3 \cdot d_{\boldsymbol{E}}(\boldsymbol{E}_g, \boldsymbol{E}_{g^{cf}}). \end{aligned} \tag{8}$$

Here, $d_{\boldsymbol{A}}(A_1, A_2) = ||A_1 \odot (1 - A_2)||_2$, $d_{\boldsymbol{X}}$ and $d_{\boldsymbol{E}}$ are calculated with l_2 pairwise distance. As the computation of $d_{\boldsymbol{A}}$ is time-consuming, we simplify this value as cross-entropy with logits in the training of the counterfactual autoencoder. In all other situations, such as the evaluation of GlobalGCE and all baselines, we adopt the definition with the dot product. We set the $\alpha = 10, \beta = \gamma = 1$ to emphasize the counterfactual's structural change.

B.1.3 Prompts

TA Query: Please describe this molecule: *{molecule data}* Your generated response is a text description STRICTLY in the form of: "This molecule contains __, __, __, and __ functional groups, in which __ may be the most influential for *{dataset description}*." NO OTHER sentence patterns are allowed. Here, __ is the functional groups (best each less than 10 atoms) or significant subgraphs alphabetically. If you can not find 4 functional groups as significant subgraphs, you may just put all you have found in the __ areas).
CTA Query: In *{smiles}*, *{key component}* may be the most influential for *{dataset description}*; what can we change *{key component}* to *{increase/decrease}* the likelihood of it being *{molecule description}*? Please find the best substitution functional group for *{key component}* that can replace the "__" in the last sentence (shown below within " "). DO NOT reply with more than 3 words. Reply ONLY the substitution function group. *"{text pairs to be revised}"*.
Feedback: The generated counterfactual is *{SMILES}*. The probability of it being *{molecule description}* is *{true prob}*. Please adjust ONE of the functional groups in the last sentence (shown below within " ") to *{increase/decrease}* the likelihood of the generated counterfactual being *{molecule description}*. ONLY the functional group names in the sentence may be changed. Reply ONLY in the format (old functional group) : (new functional group). *"{original text pairs}"*.

B.2 Details of Experiment Setup

B.2.1 Baseline Settings

GNNExplainer: For each graph, GNNExplainer (Ying et al., 2019) outputs an edge mask that estimates the importance of different edges in model prediction. We adapt this model to counterfactual generation by changing the prediction loss term from the GNN prediction value on the entry of the original classified class to the entry of the de-

sired counterfactual class. We set a threshold of 0.5 and remove edges with edge mask weights smaller than the threshold. The perturbation on node features cannot be designed as straightforwardly as the perturbation on the graph structure; thus, we do not perturb graph node features in GNNExplainer.

CF-GNNExplainer: CF-GNNExplainer (Lucic et al., 2022) is originally proposed for node classification tasks, focusing on the perturbations on the graph structure. Originally, for each explainee node, it takes its neighborhood subgraph as input. To apply it on graph classification tasks, we use the whole graph as the neighborhood subgraph and assign the graph label as the label for all nodes in the graph. We set the number of iterations to generate counterfactuals as 500.

CLEAR: CLEAR (Ma et al., 2022) is a generative model that produces counterfactuals of all the counterfactuals simultaneously while preserving causality. This model generates both a graph adjacency matrix and a graph node feature matrix perturbation, which allows all forms of graph edition such as node/edge addition/deletion/feature perturbation. We adapt it by removing the causality component for fair comparison. We train the model with 500 epochs, other hyperparameters are in default according to Ma et al. (2022).

RegExplainer: RegExplainer (Zhang et al., 2023a) explains graph regression models with information bottleneck theory to help solve the distribution shift problem. Although the original model is tailored for graph regression tasks, it can be directly applied to the graph classification tasks. We implement the RegExplainer with GNNExplainer as the base explainer model in Algorithm 2 of the original paper (Zhang et al., 2023a) and train the model for 500 epochs. Other hyperparameters are tuned for the best performance in each dataset.

B.2.2 Datasets

We utilize five datasets from TUDataset (Morris et al., 2020) and MoleculeNet (Ramsundar et al., 2019). All of them are real-world molecule datasets. The meta-data of these datasets are presented in Table 3.

AIDS: The AIDS dataset is designed for the study of chemical compounds' effectiveness against HIV, focusing on the identification of potential inhibitors. It contains molecular structures with binary labels indicating the activity. AIDS plays a crucial role in facilitating drug discovery and predictive modeling efforts aimed at finding new treatments for HIV.

Mutagenicity: The Mutagenicity dataset is aimed at predicting the mutagenic potential of chemical compounds, which is vital for assessing chemical safety and drug development. It features chemical structures alongside binary labels indicating mutagenic (1) or non-mutagenic (0) effects, making it a useful dataset for computational toxicology.

BBBP: The BBBP (Blood-Brain Barrier Penetration) dataset focuses on identifying compounds' ability to cross the blood-brain barrier, which is crucial for CNS drug design. It includes molecular structures and binary labels indicating the penetrability. The BBBP dataset is a key resource for predictive modeling in drug discovery.

ClinTox: The ClinTox dataset offers insights into chemical compounds' clinical toxicity and FDA approval status, essential for evaluating human health impacts and drug development potential. It includes chemical structures with binary labels for toxicity and FDA approval, serving as a key resource in computational pharmaceutical research.

Tox21: The Tox21 dataset is designed for the prediction of chemicals' toxicity, contributing to environmental health and safety assessments. It includes chemical compounds' structures and binary labels for various toxicity endpoints, aiding in the identification of potentially hazardous substances. Tox21 supports the development of computational models for toxicity prediction.

Table 3: Dataset metadata. "Av." stands for average. "N.C." stands for the number of node classes.

dataset	Max Nodes	Av.Nodes	Av.Edges	#graphs
AIDS	95	15.69	16.20	2000
Muta.	96	16.59	17.45	2000
Tox21	132	17.18	17.74	2000
BBBP	132	24.05	25.94	2000
ClinTox	121	27.74	29.40	478

C Extended Elaboration on LLM-GCE

C.1 Contrastive Pretraining for Text Encoder

We provide a detailed illustration of contrastive pretraining of our text encoder, as shown in the middle part of Fig. 3 in the main paper.

C.1.1 Intuition

Ideally, the text pairs possess the graph structure, labeling, and significant subgraph information, allowing us to utilize them solely as model input for counterfactual generation. However, experiments show that the graph structural information embedded in the text pairs is insufficient in producing

satisfying counterfactuals. Therefore, we regularize the text embedding of TPs with fixed graph embeddings of the GT-GNN, which enhances the graph structural information of text embeddings. This method allows the information contained in the text pairs about significant subgraphs for GCE to be effectively embedded. We may consider this pretraining process to produce the encoded model embedding to be the perturbation of the GT-GNN graph embedding with the additional significant subgraph information given by LLM text forms.

Specifically, given a graph G_i, we generate the corresponding text attribute as the text pair TP_i of G_i. Our goal is to maximize the alignment between G_i and TP_i for all G_i in the dataset, $\max_\phi P(G_i, TP_i)$, where $P(G_i, TP_j)$ denotes the probability score for a text pair. Formally, we find

$$\max_{\phi} \text{sim}(\text{GT-GNN}(G_i), \phi(TP_i)).$$

where $\text{sim}(\cdot)$ is the cosine similarity function and ϕ represents the parameters of the BERT text encoder.

C.1.2 Contrastive Pretraining

During pretraining, each batch consists of B graph-TP pairs. Within each batch, the positive and negative samples are

- Positive samples: the original pairs in the batch $\{(G_{i_k}, TP_{i_k}) \mid 0 \leq k \leq B\}$.
- Negative samples: dissimilar (graph, text) pairs $\{(G_{i_k}, TP_{i_l}) \mid 0 \leq k, l \leq B, k \neq l\}$.

We optimize to increase the alignment of positive pairs and decrease the alignment of negative pairs. Thus, following Radford et al. (2021), we design the contrastive pretraining loss as the symmetric cross-entropy

$$\mathcal{L}_{\text{contr}} = \frac{1}{2}(CE_r(M_{(G,TP)}, L) + CE_c(M_{(G,TP)}, L)),$$

where the $CE_r(M_{(G,TP)}, L)$, $CE_c(M_{(G,TP)}, L)$ are the row-wise and column-wise cross-entropy for the graphs and text-pairs, respectively. Here, the $M_{(G,TP)}$ represents the similarity matrix of the graph set and their TP set, L is the label vector $[0, 1, .., B-1]$ (n is the number of graphs).

D Supplementary Experiments

D.1 Generating Counterfactuals Directly from LLM

We conduct experiments on directly generating counterfactual graphs with GPT3.5 and GPT 4.

Table 4: Validity and Proximity results for generating counterfactual graphs directly from GPT-4.

	Validity	Proximity
AIDS	0	na
Mutagenicity	0	na
BBBP	0	na
ClinTox	0	na
Tox21	0	na

For GPT-3.5, despite multiple trials with various prompts, the LLM denies the request with "I am sorry, I can not help with that."

For GPT-4, after applying some warnings such as "Please only output...", "Do not output...", and "Under no circumstances are you to ... else ..." to regularize the LLM output to generate standard SMILES representations, we acquire some molecule SMILES as answers. Specifically, our utilized prompt is: "Minimally edit *{SMILES}* to be a *{desired graph description}* and output its SMILES representation only. Please only output one SMILES molecule without brackets and quotation marks. Do not output anything besides the SMILES. Under no circumstance are you to output anything else lest your experiments fail."

The results of validity and proximity of the generaed counterfactuals are shown in Table 4, from which we observe that the current most advanced large language models, GPT-4, cannot generate valid counterfactuals, i.e., the ones that can be predicted by GT-GNN as in the desired class.

D.2 Model Efficiency

We measure the time efficiency of our LLM-GCE with comparison with that of other state-of-the-art GCE models on AIDS and ClinTox. We train all those models with 250 epochs and the results are shown in Table 5. The reported results represent the average of five separate experiments. All of the experiments were conducted on a single Nvidia RTX A6000 serially on a server equipped with 512GB RAM and dual AMD EPYC 7543 32-core CPUs. According to the table, our LLM-GCE takes approximately twice as long to execute compared

Table 5: Run time for baselines and LLM-GCE (s).

	AIDS	ClinTox
GNNExplainer	1112.28	80.13
CF-GNNExplainer	2878.85	215.40
RegExplainer	3457.97	201.26
CLEAR	1061.05	250.24
LLM-GCE	2023.70	597.03

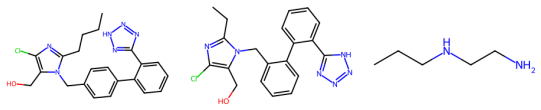

Figure 9: Comparison between the original molecule (left), our generated counterfactual (center), and CF-GNNExplainer's generated counterfactual (right).

to other baselines on the ClinTox dataset. However, on the AIDS dataset, the execution time of LLM-GCE is similar to that of other baselines. In conclusion, while the integration of the BERT text encoder and the LLM-implemented CPT feedback module increases the computational complexity and training time of our LLM-GCE model, the resulting execution time remains acceptable, making it a viable approach for practical applications.

D.3 Case Studies

We strengthen our claims regarding LLM-GCE's more feasible counterfactuals with another example from BBBP in Figure 9 where we compare CF-GNNExplainer with LLM-GCE on molecule 273 from BBBP. LLM-GCE is successfully able to produce a counterfactual with minimal changes to the original input, compared to CF-GNNExplainer, which removes a large portion of the original molecule. Further, LLM-GCE's output has a superior proximity of 19.26 versus the performance of CF-GNNExplainer, which is 24.76.

In addition, we inspect the invalid counterfactuals generated from the baseline outputs and compare them with the output from LLM-GCE. For example, for molecule 450 from BBBP, CLEAR produce `[AsH3].[As]#B[AsH]#C112(=[AsH])#[As](=[As]1)=[AsH]=2` while the ground-truth is `CSC1OC(C)(C)OC1=O`. In contrast, LLM-GCE produces `CSC1OC(C)(C)OC1=O`, which is chemically stable under valence-bond theory, while CLEAR struggles to produce a SMILES string which is chemically feasible.

Furthermore, we consider a case where LLM-GCE fails to generate a perfect counterfactual but still shows improvement over CF-GNNExplainer. For molecule 1737 from Tox21 with the SMILES string `c1ccc2cc(CC3=NCCN3)ccc2c1`, LLM-GCE produces `OC1SNCCCNCCNCCNCCN`, while CF-GNNExplainer generates `C=CC.C=CC.CCC.O=CO`. Although LLM-GCE's output is not a valid counterfactual and includes hallucinated sulfur and oxygen atoms, it still demonstrates some improvements over CF-GNNExplainer, which is because LLM-GCE's counterfactual incorporates nitrogen and avoids hallucinating double bonds.

D.4 Performance Regarding Various LLMs

There are two applications of LLMs in LLM-CGE: *(i) counterfactual autoencoder* utilize a language model (which can be LLMs such as LLaMa given sufficient computational resources) as the encoder to embed the input graph into latent space of the autoencoder; and *(ii) CTP generator* generates CTP for GCE optimization in each iteration.

D.4.1 Different counterfactual auto-encoder

We conduct extensive experiments regarding different language models as autoencoders on the ClinTox and AIDS datasets. The language models are employed through the Huggingface library. We present the results in Table 6 and Table 7. In both datasets, our LLM-GCE achieves the best GCE performance, whereas Deberta and Electra performed poorly. Specifically, our LLM-GCE achieves validity of 12.04% while the other two methods achieve almost zero validity under feasiblity check.

D.5 Parameter sensitivity of α and β

In Fig. 10 and Fig. 11, we show the sensitivity of our method with varying choices for α (the weight applied to the loss term $\mathcal{L}_{dist}$) and β (the weight applied to the loss term $\mathcal{L}_{pred}$) on datasets AIDS and ClinTox, with $\frac{\beta}{\alpha}$ scaling from 0.2 to 5. Both figures demonstrate that the validity and proximity performance of LLM-GCE are largely inversely related as expected: high validity corresponds to low proximity, and low validity corresponds to high proximity. Intuitively, graphs with large perturbations are less likely to be feasible, given that the input graph is a ground-truth molecule. We conjecture that this relationship is not visible when ignoring chemical feasibility since the model is free to generate whatever graphs it needs to achieve high validity, leading to high proximity as well as high validity. We find that the best validity and proximity performance achieves simultaneously at around $\alpha/\beta \in [0.5, 1.0]$ for both datasets. We recommend that one adopts a ratio in this range. We also have similar observations on other datasets.

Table 6: Performance with different encoders for AIDS

	Validity	Proximity	Validity w/o Feas.	Proximity w/o Feas.
LLM-GCE (ours)	0.05 ± 0.1	0.6 ± 3.2	100.0 ± 0.0	82.30 ± 13.37
microsoft/deberta-base	0.0 ± 0.0	n/a	100.0 ± 0.0	83.95 ± 5.2
google/electra-base-discriminator	0.0 ± 0.0	n/a	100.0 ± 0.0	85.29 ± 6.1

Table 7: Performance with different encoders for ClinTox

	Validity	Proximity	Validity w/o Feas.	Proximity w/o Feas.
LLM-GCE (ours)	12.04 ± 7.15	0.82 ± 0.94	100.0 ± 0.0	40.93 ± 3.12
microsoft/deberta-base	0.19 ± 0.38	1.89 ± 5.78	100.0 ± 0.0	112.00 ± 12.55
google/electra-base-discriminator	0.0 ± 0.0	n/a	100.0 ± 0.0	89.51 ± 5.52

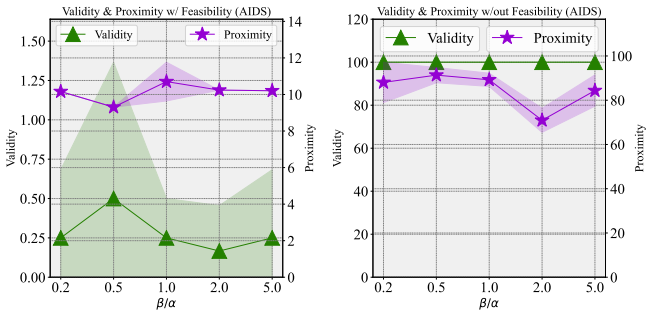

Figure 10: Performance with varying β/α ratios for AIDS

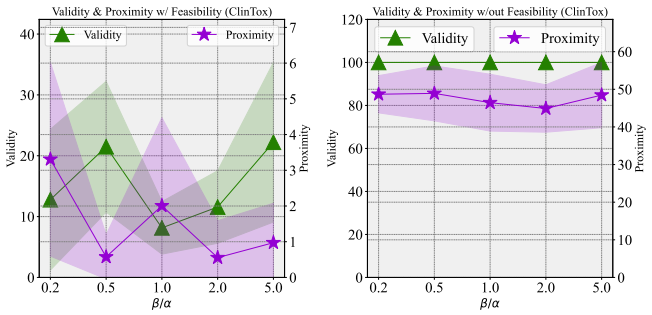

Figure 11: Performance with varying β/α ratios for ClinTox

E Related Works

E.1 Large Language Models

Since the advent of BERT (Kenton and Toutanova, 2019) and other transformer-based *pretrained language models* (PLMs), the research community has made a concerted effort to significantly enhance performance by scaling up these models. *Large language models* (LLMs) refer to PLMs with billions (or more) of parameters (Shanahan, 2024), which are trained on large-scale corpus and able to solve general purpose tasks.

Currently, there are some generative large language models such as BART (Lewis et al., 2019) that follow the encoder-decoder architecture as presented in the original Transformer paper (Vaswani et al., 2017). Benefiting from the excellent ability of the encoder to understand contextual content, these types of models are adept at sequence-to-sequence (seq2seq) tasks (Sutskever et al., 2014) such as translation. However, in text generation tasks, the input sequence might not directly match a specific output text, such as creating a story from a topic. Another type of LLMs predominantly employ the Transformer's decoder component, classifying them as decoder-only models. Studies have shown these models excel in text generation by leveraging output text as context, particularly enhancing unsupervised learning tasks (Radford et al., 2019). Currently, models such as GPT-4 (Achiam et al., 2023), LLaMA 2 (Touvron et al., 2023) and many other LLMs (Xu et al., 2023), primarily adopt a decoder-only architecture.

E.2 GNN Counterfactual Explanation

GCE problem has become popular among the research community, and several works have been proposed in recent years (Prado-Romero et al., 2023; Ying et al., 2019; Bajaj et al., 2021; Lucic et al., 2022; Ma et al., 2022; Tan et al., 2022; Huang et al., 2023b). Among them, GNNExplainer (Ying et al., 2019) aims to find the counterfactual by maximizing the mutual information between the GNN prediction and the distribution of possible subgraphs.

However, GNNExplainer is not robust to input noise. To address this problem, the RGCExplainer (Bajaj et al., 2021) generates robust counterfactuals by removing edges such that the remaining graph is just out of the decision boundary. The explanation is robust because the decision boundary is in GNN's last-layer feature space, where the features are naturally robust under perturbations. Similarly, CF-GNNExplainer (Lucic et al., 2022) and CF^2 (Tan et al., 2022) also generate counterfactuals by removing edges. Some generated counterfactuals may violate causality, Ma et al. (2022) propose a generative model, named CLEAR, to generate causally feasible counterfactuals. Besides, there are also some methods particularly designed for a certain domain, such as biomedical and chemistry (Abrate and Bonchi, 2021; Wu et al., 2021). Recently, (Huang et al., 2023b) propose the first global-level GCE model, GCFExplainer, which formulates global GCE as finding a small set of representative graph counterfactuals. However, these models overlook the incorporation of domain knowledge in GCE and provide explanations that cannot be easily understood by humans.

F Future Work

The proposed LLM-GCE framework shows encouraging results in generating graph counterfactual explanations for molecular property prediction. Here, we propose several areas that future research could explore to further enhance the capabilities and applicability of LLM-GCE.

One direction is to investigate the generalizability of LLM-GCE to other domains beyond molecular graphs, such as social networks or biological networks. This would help assess the framework's versatility and potential for broader impact.

Another avenue for future work is to extend LLM-GCE to incorporate 3D molecular structures, as the current framework focuses on 2D molecular graphs. Considering the crucial role of 3D structure in determining molecular properties, this extension could lead to more accurate and informative counterfactual explanations.

Additionally, efficiency is another important aspect to consider. Future research could explore techniques to reduce the computational cost and improve the scalability of LLM-GCE, such as knowledge distillation or model compression. This would make the framework more accessible and practical for real-world applications.

Finally, to further assess the interpretability and usefulness of the generated counterfactual explanations, future work could involve human evaluation and user studies. These studies would provide valuable insights for improving the LLM-GCE framework and making it more user-friendly for domain experts since more realistic metrics are adopted.