Harnessing Large Language Models as Post-hoc Correctors

Anonymous ACL submission

Abstract

As Machine Learning (ML) models grow in size and demand higher-quality training data, the expenses associated with re-training and fine-tuning these models are escalating rapidly. Inspired by recent impressive achievements of Large Language Models (LLMs) in different fields, this paper delves into the question: *can LLMs efficiently improve an ML's performance at a minimal cost?* We show that, through our proposed training-free framework LLMCORR, an LLM can work as a post-hoc *corrector* to propose *corrections* for the predictions of an arbitrary ML model. In particular, we form a contextual knowledge database by incorporating the dataset's label information and the ML model's predictions on the validation dataset. Leveraging the in-context learning capability of LLMs, we ask the LLM to summarise the instances in which the ML model makes mistakes and the correlation between primary predictions and true labels. Following this, the LLM can transfer its acquired knowledge to suggest corrections for the ML model's predictions. Our experimental results on the challenging molecular predictions show that LLMCORR improves the performance of a number of models by up to 39%[1].

1 Introduction

In recent decades, Machine Learning (ML) models have become increasingly prevalent in various real-world applications (Butler et al., 2018; Dixon et al., 2020; Zhong et al., 2023). As ML models grow in size and demand higher-quality training data, the expenses associated with re-training and fine-tuning these models to achieve superior performances are escalating rapidly (Devlin et al., 2018; He et al., 2021). Hence, there is an urgent need to develop effective, lightweight and practical solutions for users to improve their ML model's predictions.

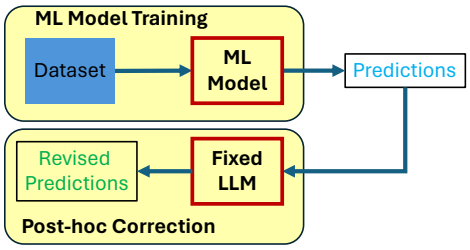

Figure 1: Harnessing LLMs as post-hoc *correctors*. A fixed LLM is leveraged to propose corrections to an arbitrary ML model's predictions without additional training or the need for additional datasets.

Large Language Models (LLMs) exhibit unprecedented capabilities in understanding and generating human-like text, making them invaluable across a wide range of Natural Language Processing (NLP) tasks, including machine translation (Hendy et al., 2023), commonsense reasoning (Krishna et al., 2023) and coding tasks (Bubeck et al., 2023). While LLMs have showcased their effectiveness across an array of NLP applications, the full extent of their potential in broader fields remains largely unexplored (Zhang et al., 2024). This paper delves into the essential research question: *can LLMs effectively improve an ML's performance at a minimal cost?*

To answer the research question, we propose a groundbreaking framework, LLMCORR, which extends the application scope of LLMs by positioning them as training-free post-hoc *correctors* (illustrated in Figure 1). A *fixed* LLM is leveraged to propose corrections to an arbitrary ML model's predictions without introducing additional training or the need for additional datasets.

At its core, LLMCORR consists of three main steps: (1) After completing the training of the ML model, we collect the dataset's available label information, along with the primary predictions made

[1]The code and models are available at https://anonymous.4open.science/r/LlmCorr.

by the ML model on the validation set, to construct a contextual knowledge database. We anticipate that the established database records the contextual knowledge about the types of instances for which the ML models produce accurate or inaccurate predictions, as well as the correlation between the primary predictions and the true labels. A recent breakthrough known as In-Context Learning (ICL) (Liu et al., 2023) has enhanced the adaptability of LLMs by enabling them to acquire contextual knowledge during inference, eliminating the need for extensive fine-tuning (Clark et al., 2020). (2) Consequently, given the target data with a primary prediction generated by the ML model, we extract the relevant contextual knowledge from the knowledge database to form a prompt. Given the input token constraints of current LLMs, transmitting all contextual information to one prompt for querying becomes impractical (Touvron et al., 2023; Achiam et al., 2023). To address this limitation, we introduce an embedding-based information retrieval approach, which can efficiently locate similar data likely to offer relevant insights with the target data. (3) Finally, we query the LLM using the created prompts for suggestions to refine the target data's primary prediction. To mitigate the knowledge hallucination (Huang et al., 2023), we implement a *self-correction* mechanism when the LLM demonstrates a tendency to make substantial modifications to the ML model's prediction.

The training-free nature of LLMCORR carries several natural advantages: *(i)* LLMCORR facilitates the straightforward application of LLMs, eliminating the necessity for expensive re-training and fine-tuning for an arbitrary ML model. *(ii)* One of the most obvious limitations of LLMs is their reliance on *unstructured* text (Guo et al., 2023), but LLMCORR can be adapted to arbitrary scenarios by incorporating different ML models.

To validate the effectiveness of LLMCORR, we deploy it in the face of structured molecule graph data within the domain of biology, *e.g.*, predicting the functionality of molecules. Through extensive experiments conducted on six real-world benchmark datasets covering diverse subjects, we empirically demonstrate that LLMCORR significantly elevates the quality of predictions across diverse ML models, achieving notable improvements, up to 39%. Furthermore, we conduct comprehensive follow-up ablation studies and analyses to validate the efficacy of LLMCORR's designs and elucidate the impact of key factors.

2 Related Work

Large Language Models. Traditional language models are typically trained on sequences of tokens, learning the likelihood of the next token dependent on the previous tokens (Vaswani et al., 2017). Recently, Brown et al. (2020) demonstrated that increasing the size of language models and the amount of training data can result in new capabilities, such as zero-shot generalisation, where models can perform text-based tasks without specific task-oriented training data. Consequently, Large Language Models (LLMs) have experienced exponential growth in both size and capability in recent years (Brown et al., 2020; Touvron et al., 2023; Achiam et al., 2023). A wide range of NLP applications have been reshaped by LLMs, including machine translation (Hendy et al., 2023), commonsense reasoning (Krishna et al., 2023) and coding tasks (Bubeck et al., 2023). In this work, we innovatively harness LLMs as post-hoc *correctors*, further extending the application scope of LLMs.

In-Context Learning. While the impressive performance and generalisation capabilities of language models have rendered them highly effective across various tasks (Wei et al., 2022a), they have also resulted in larger model parameters and increased computational costs for additional fine-tuning on new downstream tasks (Hu et al., 2021a). To address this challenge, recent research has introduced In-Context Learning (ICL), enabling LLMs to excel at new tasks by incorporating a few task samples directly into the prompt (Liu et al., 2023). Despite the success of these methods in improving LLM performance, their potential for correcting predictions made by ML models has not been thoroughly explored. This work investigates the utility of LLMs as post-hoc *correctors* to rectify incorrect predictions by leveraging their ICL abilities.

Post-hoc Corrector for Machine Learning Models. Driven by the increasing prevalence of ML models in diverse real-world applications (Butler et al., 2018; Dixon et al., 2020; Zhong et al., 2023), academia and industry have invested significant efforts in enhancing ML effectiveness. However, the majority of existing research focuses on refining the design of the ML module. Meanwhile, as ML models grow in size and demand higher-quality training data, the expenses associated with re-training and fine-tuning these models are escalating rapidly (Devlin et al., 2018; He et al., 2021). Hence, there is an urgent need to develop effec-

tive, lightweight and practical solutions for users to improve their ML model's predictions adaptively. While some studies (Huang et al., 2021; Zhong et al., 2022) propose post-processing techniques to adjust ML model predictions on node-wise tasks of graph-structured data, their solutions often lack scalability across different types of data. This work introduces a novel and versatile post-hoc *corrector* framework applicable to an arbitrary ML model.

3 Preliminary and Problem

This paper aims to leverage LLMs as post-hoc *correctors* to enhance predictions made by an arbitrary ML model. Specifically, we focus on addressing challenging prediction tasks on structured molecule graph data within the field of biology. For instance, consider the supervised molecule property prediction task, where molecules can be represented using various formats such as *SMILES string* (Weininger, 1988) and *geometry structures* (Zhang et al., 2024) (as shown in Figure 8). However, a notable limitation of existing LLMs is their reliance on unstructured text, rendering them unable to incorporate essential geometry structures as input (Li et al., 2023; Guo et al., 2023). To overcome this limitation, Fatemi et al. (2023) propose encoding the graph structure into text descriptions. In this paper, as depicted in Figure 8, we extend this concept by encoding both the molecule's atom features and graph structure into textual *descriptions*.

Notion. Given a molecule, we formally represent it as a graph $\mathcal{G} = (S, G, D)$, where S, G and D denote the *SMILES string*, *geometry structures* and generated atom feature and graph structure *descriptions* of $\mathcal{G}$. $y \in \mathcal{Y}$ stands for the label for $\mathcal{G}$.

Problem Setup. Given a set of molecules $\mathcal{M} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m\}$, where $\mathcal{M}_{\mathcal{T}} \subset \mathcal{M}$ contains molecules with known labels y_v for all $\mathcal{G}_v \in \mathcal{M}_{\mathcal{T}}$. Our objective is to predict the unknown labels y_u for all $\mathcal{G}_u \in \mathcal{M}_{test}$, where $\mathcal{M}_{test} = \mathcal{M} \setminus \mathcal{M}_{\mathcal{T}}$. In addition, $\mathcal{M}_{\mathcal{T}}$ is split into two subsets: $\mathcal{M}_{train}$ and $\mathcal{M}_{val}$, where $\mathcal{M}_{train}$ is the training set and $\mathcal{M}_{val}$ works as the validation set.

ML Models. The conventional approach to tackle molecule property prediction tasks is employing ML models. Take the supervised molecule property prediction task as an example. The goal is to learn a mapping function $f_{ML} : \mathcal{M} \rightarrow \hat{\mathcal{Y}}$, by minimising loss function value $\min_{\Theta} \sum_{i=1}^{n} \mathcal{L}(\hat{\mathcal{Y}}^i_{train}, \mathcal{Y}^i_{train})$, where Θ represents the set of trainable parameters of f_{ML}. Subsequently, f_{ML} can be employed on test dataset $\mathcal{M}_{test}$ to generate predictions $\hat{\mathcal{Y}}_{test}$.

Leveraging LLMs as Post-hoc *Correctors*. In recent decades, significant efforts have been devoted to enhancing the effectiveness, robustness, and generalisation of advanced ML models (f_{ML}). However, the potential for improving the quality of ML model predictions after completing the training process remains largely unexplored. With the trend of ML models increasing in size and requiring higher-quality training data, the costs associated with re-training and fine-tuning ML models are rapidly escalating. This paper intends to explore the possibility of positioning LLMs (f_{LLM}) as post-hoc *correctors* to refine predictions of an arbitrary f_{ML}. Compared with re-training and fine-tuning a model, this work has outstanding advantages in terms of cost and versatility.

4 Methodology

In this section, we outline the workflow of LLMCorr, designed as a post-hoc *corrector* to refine predictions generated by any ML model. As illustrated in Figure 2, the key idea is to leverage the LLM's ICL ability to summarise the types of data in which the ML model makes mistakes and the correlation between primary predictions and true labels, thereby refining the ML's prediction on the test dataset. To achieve this goal, LLMCorr comprises three main steps: (1) Contextual knowledge database construction; (2) Contextual knowledge retrieval; (3) Prompt engineering and query.

4.1 Contextual Knowledge Database Construction

After completing the training of the ML model (f_{ML}) on the training set $\mathcal{M}_{train}$, we gather comprehensive data from both the training set $\mathcal{M}_{train}$ and the validation set $\mathcal{M}_{val}$, along with the primary predictions $\hat{\mathcal{Y}}_{val}$ made by f_{ML} on the validation set. Subsequently, we construct a contextual knowledge database $\mathcal{D}$ based on the collected data. Notably, the knowledge database $\mathcal{D}$ not only contains information regarding the original training and validation datasets but also provides insights into the types of molecules for which the ML models generate accurate or inaccurate predictions. Additionally, it captures the relationship between the initial prediction $\hat{\mathcal{Y}}_{val}$ and the true label $\mathcal{Y}_{val}$. We anticipate that this essential contextual knowledge will empower the LLM (f_{LLM}) to refine the predictions $\hat{\mathcal{Y}}_{test}$ made by f_{ML} on the test dataset $\mathcal{M}_{test}$.

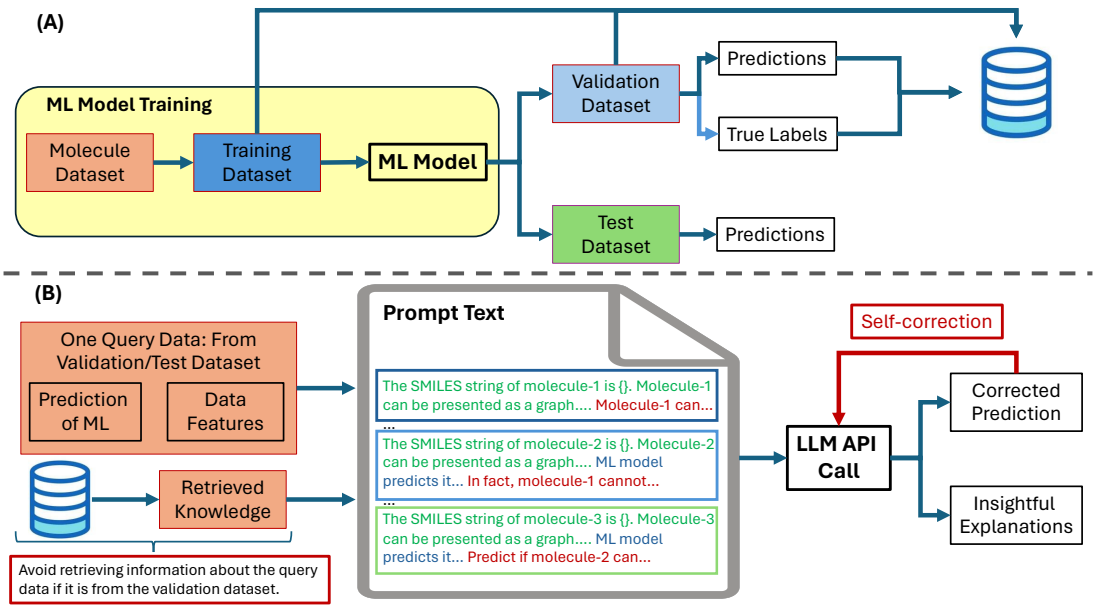

Figure 2: A high-level overview of LLMCORR, harnessing Large Language Models (LLMs) as post-hoc *correctors* to refine predictions made by an arbitrary Machine Learning (ML) model.

4.2 Contextual Knowledge Retrieval

The effectiveness of the LLMCORR heavily relies on the richness and relevance of the information received by the LLM, as it determines the task-specific contextual knowledge available to the LLM. However, due to the input token constraints of existing LLMs, transmitting all contextual knowledge into the LLM is impractical (Touvron et al., 2023; Achiam et al., 2023). After constructing the contextual knowledge database $\mathcal{D}$, the next challenge is to retrieve relevant contextual knowledge for a given query data $\mathcal{Q}_u = (\mathcal{G}_u, \hat{y}_u)$ from either the validation set $\mathcal{M}_{val}$ or the test set $\mathcal{M}_{test}$. To address this limitation, we propose an *Embedding-based Information Retrieval* (EIR) approach. The EIR technique comprises two primary steps: (1) Utilising a text encoder (f_{Emb}) on available textual descriptions of molecule $\mathcal{G}_u$, we generate embedding vectors $f_{Emb} : (S, D) \rightarrow \mathbf{Z}$ for molecules from the knowledge database $\mathcal{D}$ and the query data $\mathcal{Q}_u$. (2) Calculating the similarity between the query data $\mathbf{Z}_u$ and molecules in the knowledge database based on the obtained embeddings $\mathbf{Z}_v, \forall \mathcal{G}_v \in \mathcal{D}$. Different selection strategies can be employed to retrieve various contextual knowledge based on the ranking. In this study, we retrieve the top-k similar data as the contextual knowledge. We delve into the influence of different retrieval selection strategies in Section 5.2.

Addressing Data Leakage Concerns. It is important to recognise that when the query data $\mathcal{Q}_u$ comes from the validation dataset $\mathcal{M}_{val}$, precautions are taken to prevent retrieving information about $\mathcal{Q}_u$ to avoid data leakage. Then, LLMCORR's performance on $\mathcal{M}_{val}$ can be assessed in a manner consistent with traditional ML pipeline, *e.g.*, hyper-parameters selection.

4.3 LLMCORR Prompt Engineering

The goal in prompt engineering is to find the correct way to formulate a question $\mathcal{Q}$ in such a way that an LLM (f_{LLM}) will return the corresponding answer A essentially represented as $A = f_{LLM}(\mathcal{Q})$. In this work, our goal is to provide the LLM with helpful and comprehensive contextual knowledge regarding molecules and the ML model's behaviours on the validation set so that it can make precise corrections to the ML model's predictions on the test dataset. A variety of approaches exist for modifying the f_{LLM} so that it could better perform downstream tasks such as fine-tuning (Clark et al., 2020) and LoRA (Hu et al., 2021a). However, these methods typically require access to the internals of the model and heavy computation capability, which can limit their applicability in many real-world scenarios. In this work, we are instead interested in the case where f_{LLM} and its parameters are fixed, and the system is available only for users in a black box setup where f_{LLM} only consumes and produces text. We believe this setting to be particularly valuable as the number of proprietary models available

```
LlmCorr = [
#Start - Instruction
{"role": "system",
"content":
    "You are an expert in biomedicine and chemistry, "
    "specializing in molecules.",
}, #End - Instruction
# Start - Knowledge from training dataset
{"role": "user",
"content":
    "The SMILES string of molecule-{ID} is {SMILES}. "
    "{description} "
    "Predict whether molecule-{ID} {task}."},
{"role": "assistant",
"content": "Molecule-{ID} {task}."},
# End - Knowledge from training dataset
# Start - Knowledge from validation dataset
{"role": "user",
"content":
    "The SMILES string of molecule-{ID} is {SMILES}. "
    "{description} "
    "The ML model predicts that molecule-{ID} {task} "
    "with a probability of {probability}. "
    "Predict whether molecule-{ID} {task}."},
{"role": "assistant",
"content": "Molecule-{ID} {task}."},
# End - Knowledge from validation dataset
# Start - Question
{"role": "user",
"content":
    "The SMILES string of molecule-{ID} is {SMILES}. "
    "{description} "
    "The ML model predicts that molecule-{ID} {task} "
    "with a probability of {probability}. "
    "Predict whether molecule-{ID} {task}. "
    "Answer this question in the format: "
    "Prediction: <True or False>; Probability: <number>; "
    "Explanation: <text>."}] # End - Question
```

Figure 3: LLMCORR prompt template. Multiple contextual knowledge from training and validation datasets can be included by expanding the template.

and their hardware demands increase.

LLMCORR Prompting. To this end, we propose a novel prompt that serves to position LLMs as post-hoc *correctors*, refining predictions made by an arbitrary ML model. Different from existing prompts that often work as *predictors* (Fatemi et al., 2023) or *explainers* (He et al., 2023), leveraging LLMs as *correctors* combining the strengths of LLMs' in context-based question answering with the ML model's proficiency in learning from specific datasets. Specifically, the proposed LLMCORR prompt template is illustrated in Figure 3, which mainly consists of following components:

1. *Instruction*: Provides general guidance to the LLM, clarifying its role in the conversation.
2. *Contextual knowledge from the training dataset*: Includes SMILES string, molecule atom feature and structure descriptions, and molecule label information of the training dataset.
3. *Contextual knowledge from the validation dataset*: Includes SMILES string, molecule atom feature and structure descriptions, molecule label information and the ML model's predictions of the validation dataset. This equips the LLM with insights into the ML model's error patterns, enhancing its ability to refine predictions on the test dataset.
4. *Question*: Tasks the LLM to refine the ML model's predictions for the query data, drawing on the provided contextual knowledge.

It's worth noting that by expanding the template's sections on contextual knowledge from the training and validation datasets, multiple instances of contextual knowledge from the knowledge database $\mathcal{D}$ can be included. Subsequently, we query the LLM with generated prompts to obtain an initial response concerning the refined prediction of the query data, along with probability values and explanations, offering significant interpretability.

```
self_correction = {"role": "user",
"content":
    "Your prediction differs from the ML model's prediction, "
    "and the {eval_metric} score of the ML model on the training "
    "dataset is {score}, on the validation dataset is {score}. "
    "Please conduct a self-check on your thought process. "
    "Examine your reasoning carefully, and search for additional "
    "relevant information to validate or refine your understanding. "
    "Provide your refined response in the format: "
    "Prediction: <True or False>; Probability: <number>; "
    "Explanation: <text>."}
```

Figure 4: *Self-correction* prompt template.

Self-correction Prompting. An inherent limitation of existing LLMs is their tendency to generate *hallucinations* producing content that deviates from real-world facts or user inputs (Ganguli et al., 2023; Huang et al., 2023). One promising solution is known as *self-correction*, where the LLM is prompted or guided to rectify errors in its own output (Pan et al., 2023). In this work, after obtaining corrected prediction $\tilde{y}_u$ from the LLM, we introduce a *self-correction* mechanism (prompt template is shown in Figure 4) if the LLM makes significant modifications on the primary prediction generated by the ML (for classification tasks - reversing the prediction label; for regression tasks - modifying the prediction value range by more than 20%). As demonstrated empirically in our experiments (see Section 5.2), this approach can prevent LLM from hallucinating incorrect corrections in many cases.

Extensibility. LLMCORR is designed as a post-hoc *corrector* framework that leverages the ICL capability of LLMs. As illustrated in Figure 2, LLMCORR is adaptable to an arbitrary ML model, showcasing its remarkable extensibility. With the trend of ML models increasing in size and requiring higher-quality training data, the costs associated with enhancing ML models by re-training and fine-tuning are rapidly escalating. LLMCORR emerges as a promising, general-purpose practical solution in the LLM era. LLMCORR is summarised in Algorithm 1 in Appendix B.

5 Are LLMs Post-hoc Correctors?

In this section, we evaluate the effectiveness of LLMs as post-hoc *correctors*. Our experimental analysis focuses on the challenging structured molecular graph property prediction tasks.

Dataset. We consider six widely used molecule datasets from the OGB benchmark (Hu et al., 2021b), including `ogbg-molbace`, `ogbg-molbbbp`, `ogbg-molhiv`, `ogbg-molesol`, `ogbg-molfreesolv` and `ogbg-mollipo`. Detailed descriptions are summarised in Appendix C.

ML Models. To investigate whether LLMCORR can effectively improve predictions across various types of ML models. We consider ML models of three different categories: (1) Language Model (LM) that only takes text information as inputs, *i.e.*, DeBERTa (He et al., 2021). (2) Graph Neural Networks (GNNs) that capture the molecule's geometry structure information. We consider two classic GNN variants, *i.e.*, GCN (Kipf and Welling, 2017) and GIN (Xu et al., 2019), and two SOTA GNN variants collected from the OGB leaderboards (Hu et al., 2021b), *i.e.*, HIG and PAS (Wei et al., 2021). (3) And we consider one recently released hybrid framework, TAPE (He et al., 2023), in our experiments. TAPE leverages LM and LLMs to capture textual information as features, which can be used to boost GNN performance. The implementation details are discussed in Appendix D.

LLMs. In this work, we are interested in where the LLM's parameters are fixed, and the system is available for users in a black box setup where the LLM only consumes and produces text. We believe this setting to be particularly valuable as most users would practically have access to LLMs. In this case, we consider GPT-3.5 and GPT-4 (Achiam et al., 2023) as LLMs in this work, and GPT-3.5 is the major LLM for most experiments. All responses are obtained by calling their official APIs. Because the generated *descriptions* following (Fatemi et al., 2023) have tons of tokens, easily over the LLM's input token constraints, hence we do not include descriptions in the LLMCORR prompt in this study.

5.1 Main Results

Observation 1: LLMCORR is a potent post-hoc corrector. Examining the molecule graph property prediction performance across six datasets in Table 1, it's evident that LLMCORR consistently delivers substantial enhancements over various ML models, with improvements reaching up to 39% in terms of RMSE. This consistent and notable performance underscores the effectiveness of LLMs within our framework LLMCORR, serving as proficient post-hoc *correctors* to refine the primary predictions generated by ML models.

Observation 2: The significance of geometric structure. Table 1 underscores the superiority of models incorporating geometric structure over others. This highlights the crucial role of geometric structure in accurately predicting a molecule's property. However, LLMCORR currently cannot directly incorporate geometric structure in the prompt due to limitations in the token count of generated descriptions over the LLM's constraints. Addressing this limitation is identified as a promising avenue for future exploration.

Observation 3: Enhanced assistance for lower-performing ML models. Furthermore, we observe that LLMCORR *provides more substantial assistance when the performance of ML models is lower.* This trend is noticeable across various datasets; for instance, LLMCORR boosts LM performance from 0.6163 to 0.6915 with a 12.2% improvement on the test dataset of `ogbg-molbace`. Even though the ultimate performance still falls short compared to GNN models, the magnitude of improvement is most significant.

Observation 4: GPT-4 underperforms compared to GPT-3.5. Table 2 displays the molecule graph property prediction performance and execution time for three datasets, comparing GPT-3.5 and GPT-4. Interestingly, we find that despite its larger training data and more complex fine-tuning process, GPT-4 exhibits inferior performance compared to GPT-3.5 in this study. We hypothesise that this discrepancy may be attributed to interventions such as reinforcement learning through human feedback.

Table 1: Molecule graph property prediction performance for the ogbg-molbace, ogbg-molbbbp, ogbg-molhiv, ogbg-molesol, ogbg-molfreesolv and ogbg-mollipo datasets. Classification tasks are evaluated on ROC-AUC (↑: higher is better), and regression tasks are evaluated on RMSE (↓: lower is better). The improvements of LLMCORR over the ML predictive models are reported below LLMCORR's performance.

	ogbg-molbace		ogbg-molbbbp		ogbg-molhiv		ogbg-molesol		ogbg-molfreesolv		ogbg-mollipo	
	ROC-AUC ↑						RMSE ↓					
	Valid	Test	Valid	Test	Valid	Test	Valid	Test	Valid	Test	Valid	Test
LM	0.5584	0.6163	0.9307	0.6727	0.5024	0.5037	2.1139	2.2549	6.6189	4.4532	1.2095	1.1066
$LM^{LlmCorr}$	0.6110	0.6915	0.9481	0.6897	0.6253	0.6154	1.4113	1.3747	5.7195	3.5595	1.0210	0.9468
	+9.4%	+12.2%	+1.9%	+2.5%	+24.5%	+22.2%	-33.2%	-39.0%	-13.6%	-20.1%	-15.6%	-14.4%
GCN	0.7879	0.7147	0.9582	0.6707	0.8461	0.7376	0.8538	1.0567	2.8275	2.5096	0.6985	0.7201
$GCN^{LlmCorr}$	0.8203	0.7718	0.9595	0.7045	0.8540	0.7529	0.7744	0.9108	2.0325	2.2102	0.6874	0.7043
	+4.1%	+8.0%	+0.0%	+5.0%	+0.9%	+2.1%	-9.3%	-13.8%	-28.1%	-11.9%	-1.6%	-2.2%
GIN	0.8042	0.7833	0.9611	0.6821	0.8406	0.7601	0.7685	0.9836	2.4141	2.2435	0.6503	0.7100
$GIN^{LlmCorr}$	0.8336	0.8214	0.9710	0.6982	0.8523	0.7822	0.7418	0.9137	2.1790	1.9219	0.6219	0.6995
	+3.7%	+4.9%	+1.0%	+2.4%	+1.4%	+2.9%	-3.5%	-7.1%	-9.7%	-14.3%	-4.4%	-1.5%
TAPE	0.7824	0.7410	0.9421	0.6994	0.8364	0.7514	0.8351	0.9872	2.8453	2.2134	0.6839	0.7168
$TAPE^{LlmCorr}$	0.8074	0.7788	0.9653	0.6996	0.8406	0.7693	0.7966	0.9605	2.6184	2.0470	0.6751	0.7074
	+3.2%	+5.1%	+2.5%	+0.0%	+0.5%	+2.4%	-4.6%	-2.7%	-8.0%	–7.5%	-1.3%	-1.3%
HIG	0.8213	0.8094	0.9730	0.6974	0.8400	0.8393	0.7756	0.9504	2.3590	2.2546	0.6130	0.7036
$HIG^{LlmCorr}$	0.8294	0.8135	0.9748	0.7074	0.8489	0.8447	0.7536	0.9322	2.3556	1.8799	0.6040	0.6920
	+1.0%	+0.5%	+0.2%	+1.4%	+1.1%	+0.6%	-2.8%	-1.9%	-0.1%	-16.6%	-1.5%	-1.6%
PAS	0.8199	0.7473	0.9403	0.6618	0.8273	0.8402	0.8791	1.0348	2.3500	2.3546	0.6715	0.7088
$PAS^{LlmCorr}$	0.8230	0.7920	0.9671	0.6842	0.8422	0.8490	0.8251	0.9859	2.1130	1.9320	0.6342	0.6897
	+0.4%	+6.0%	+2.9%	+3.4%	+1.8%	+1.0%	-6.1%	-4.7%	-10.1%	-17.9%	-5.6%	-2.7%

Table 2: Molecule graph property prediction performance and execution time for the ogbg-molbace, ogbg-molesol and ogbg-molfreesolv datasets, with different LLMs. Classification tasks are evaluated on ROC-AUC (↑: higher is better), and regression tasks are evaluated on RMSE (↓: lower is better).

	ogbg-molbace			ogbg-molesol			ogbg-molfreesolv		
	ROC-AUC ↑		Execution	RMSE ↓		Execution	RMSE ↓		Execution
	Valid	Test		Valid	Test		Valid	Test	
$GCN^{LlmCorr}_{GPT3.5}$	0.8203	0.7718	∼9.5 min	0.7744	0.9108	∼11.5 min	2.0325	2.2102	∼10.5 min
$GCN^{LlmCorr}_{GPT4}$	0.7910	0.7713	∼155 min	0.8953	1.0105	∼204min	6.5331	3.5777	∼107min
$GIN^{LlmCorr}_{GPT3.5}$	0.8336	0.8214	∼9.6 min	0.7418	0.9137	∼12.1 min	2.1790	1.9219	∼11.7 min
$GIN^{LlmCorr}_{GPT4}$	0.8022	0.7875	∼148 min	1.1384	0.9552	∼192min	7.4731	3.9611	∼112min

Table 3: Molecule graph property prediction performance for the ogbg-molbace, ogbg-molbbbp, ogbg-molhiv, ogbg-molesol, ogbg-molfreesolv and ogbg-mollipo datasets. Classification tasks are evaluated on ROC-AUC (↑: higher is better), and regression tasks are evaluated on RMSE (↓: lower is better).

	ogbg-molbace		ogbg-molbbbp		ogbg-molhiv		ogbg-molesol		ogbg-molfreesolv		ogbg-mollipo	
	ROC-AUC ↑						RMSE ↓					
	Valid	Test	Valid	Test	Valid	Test	Valid	Test	Valid	Test	Valid	Test
LLM_{IP}	0.5690	0.5756	0.4606	0.5399	0.5519	0.5892	2.6221	2.0422	6.1699	4.4421	1.9836	1.8411
LLM_{IPD}	0.4835	0.5534	0.4643	0.4664	0.4732	0.5693	3.7395	3.1721	8.1598	7.2877	2.6464	2.5046
LLM_{IE}	0.4769	0.5220	0.4463	0.5237	0.5487	0.5419	2.1055	2.5549	5.9059	4.3097	2.1044	1.9158
LLM_{IED}	0.5299	0.4761	0.4742	0.4091	0.5361	0.5512	3.9001	4.2289	7.4837	5.3689	2.4191	2.4219
LLM_{FS-1}	0.4822	0.5122	0.5955	0.4954	0.5229	0.5268	1.7699	2.8762	6.4785	4.7553	1.9810	1.8432
LLM_{FS-2}	0.4277	0.6090	0.6019	0.5075	0.5619	0.5731	1.9271	2.1020	5.5078	4.5606	1.9138	1.8118
LLM_{FS-3}	0.5405	0.5949	0.6000	0.5388	0.5475	0.5616	1.9548	1.9963	6.3753	4.7241	1.8291	1.7923
LLM_{FS-10}	0.4973	0.5160	0.5214	0.4740	0.6233	0.6114	1.4735	1.4661	5.9601	4.2810	1.5178	1.4493
LLM_{FS-30}	0.6110	0.6354	0.5164	0.5245	0.6251	0.6276	2.7207	2.3669	6.7362	4.6829	1.8060	1.4808
LLM_{FS-50}	0.5749	0.6027	0.4572	0.4682	0.5312	0.5843	2.7465	2.5133	6.3208	4.3760	1.8499	1.3644

Observation 5: LLMs exhibit limited competitiveness as predictors. Given LLMCORR's remarkable performance as *correctors*, another intriguing question arises: *can LLM generate accurate predictions directly?* To investigate, we conduct additional experiments where the LLM is tasked with directly predicting the molecule's property. For detailed findings, please refer to Appendix E due to space constraints. As shown in the results of Table 3, LLMs do not demonstrate competitive performance as predictors. This observation reinforces the efficacy of LLMCORR, which

leverages LLMs as post-hoc *correctors*.

5.2 Ablation Study

Table 4: Ablation study of LLMCORR on `ogbg-molbace` and `ogbg-molesol` with variants of contextual knowledge retrieval.

	`ogbg-molbace`	`ogbg-molesol`
	ROC-AUC ↑	RMSE ↓
$\text{GIN}^{\text{LlmCorr}}$	0.8214	0.9137
w/ Jump	0.7799	1.0696
w/ Random	0.7868	1.1116
$\text{LM}^{\text{LlmCorr}}$	0.6915	1.3747
w/ Jump	1.4633	1.9781
w/ Random	1.9517	2.0615

Variants of contextual knowledge retrieval. Within the EIR of LLMCORR, the selection of top-k data from the knowledge database following similarity calculations is a critical step. This ablation study explores alternative approaches such as *Jump* and *Random*. In *Random*, k data are randomly selected from the knowledge database, disregarding similarity rankings. On the other hand, *Jump* selects k evenly spaced indices, ensuring diversity in the selected data. Results from Table 4 suggest that selecting top-k data yields optimal results, with *Jump* outperforming *Random*. We posit that LLMs benefit from closely relevant knowledge to generate effective corrections.

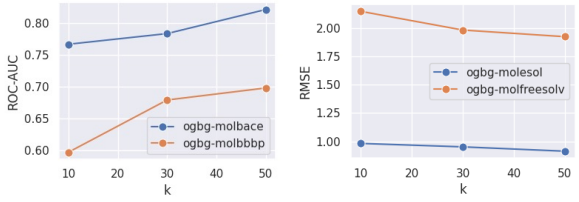

Figure 5: Ablation study of LLMCORR with a different number of contextual knowledge data (k) on `ogbg-molbace` and `ogbg-molesol` datasets.

Impact of k. In the EIR, the parameter k dictates the number of knowledge instances sampled from the database to construct LLMCORR's prompt, thus influencing the knowledge presented to the LLM. It is observed (Figure 5) that larger k values correlate with improved performance, underscoring the significance of comprehensive knowledge to guide LLMs for enhanced performance.

Effect of f_{Emb}. Another ablation study concerning the EIT process examines the influence of different f_{Emb} functions. Figure 6 suggests that larger

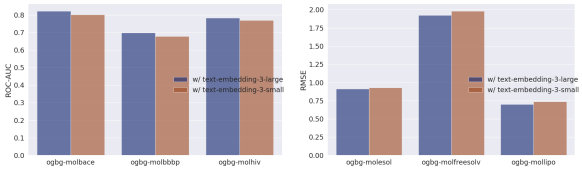

Figure 6: Ablation study of LLMCORR on six datasets with different f_{Emb}.

f_{Emb} values yield superior performance on benchmark testing, aiding LLMCORR in achieving better results. This is attributed to accurate semantic embeddings facilitating the identification of relevant instances during the EIT process, reinforcing the importance of selecting top-k relevant knowledge instances.

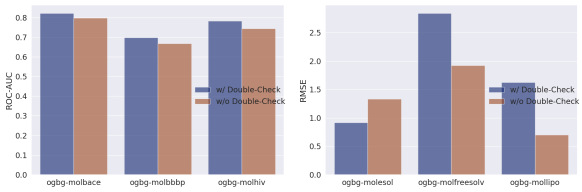

Figure 7: Ablation study of LLMCORR on six datasets w/ and w/o self-correction.

Impact of *self-correction*. Upon completion of LLMCORR's inference process, the LLM is tasked with self-correction if major modifications to primary predictions are made. Figure 7 illustrates LLMCORR's performance on the test dataset across six datasets, revealing instances where the self-correction component leads to uncertain impacts. This phenomenon is attributed to the LLM becoming hesitant and cautious after the questioning. Designing a more effective self-correction prompt emerges as an intriguing area for future investigation.

6 Concluding Discussion

We have introduced a novel framework, LLMCORR, a training-free, lightweight, yet effective approach, harnessing the in-context learning capabilities of LLMs to improve the predictions of arbitrary ML models. Through this simple and versatile approach, we have demonstrated significant improvements over a number of ML models on different challenging tasks. As LLMs continue to improve in performance and in-context learning capabilities, LLMCORR stands to directly benefit from these advancements.

7 Limitations and Ethic Statement

Limitations. While LLMCORR demonstrates simplicity and effectiveness in improving the predictions of an arbitrary ML model, our verification was confined to structured molecular graph property prediction tasks. Further extensive empirical investigations across diverse domains are warranted to establish its generalisability. Additionally, considering the purported enhanced ICL capabilities of GPT-4 on various benchmark tasks (OpenAI, 2023), it is noteworthy that our findings (as discussed in Section 5.1 and illustrated in Table 2) reveal GPT-4's underperformance compared to the GPT-3.5 model. This discrepancy merits further exploration to elucidate the underlying reasons. Moreover, while LLMCORR's prompt template accommodates the insertion of molecule atom features and geometry structure descriptions, limitations stemming from the LLM's input token constraints prevented their inclusion in the prompt. Lastly, while our approach incorporates contextual knowledge into the prompt, its utility is constrained by several factors, including limited flexibility. For example, further leveraging different techniques, *e.g.*, RAG (Lewis et al., 2020), to involve more contextual knowledge into the LLM is also a fruitful direction. Further enhancements in this regard are warranted to maximise LLMCORR's effectiveness.

Ethic Statement. Our proposed framework, LLMCORR, is designed as a post-hoc *corrector* aims at improving the prediction of an arbitrary ML model. However, given the emergent in-context learning ability within LLMs, which typically consist of billions of parameters, the accessibility of computational resources may inadvertently introduce disparities in the utilisation of these methods. Research groups with limited access to computational resources will be handicapped, while resourceful groups will be able to investigate and advance the future directions of this research. Throughout our work, we did not utilise any private or sensitive information. However, it's essential to note that if any private information were to be inadvertently exposed to an LLM during internal pertaining and fine-tuning stages, LLMCORR does not offer any privacy filtration mechanism. Therefore, there exists the potential for privacy concerns associated with the underlying model to manifest through the output provided by LLMCORR.

References

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. *CoRR*, abs/2307.09288.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. 2023. Graph of thoughts: Solving elaborate problems with large language models. *CoRR*, abs/2308.09687.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *CoRR*, abs/2303.12712.

Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. 2018. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *CoRR*, abs/2002.05867.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Matthew F Dixon, Igor Halperin, and Paul Bilokon. 2020. *Machine learning in finance*, volume 1170. Springer.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a graph: Encoding graphs for large language models. *CoRR*, abs/2310.04560.

Deep Ganguli, Amanda Askell, Nicholas Schiefer, Thomas Liao, Kamilė Lukošiūtė, Anna Chen, Anna Goldie, Azalia Mirhoseini, Catherine Olsson, Danny Hernandez, et al. 2023. The capacity for moral self-correction in large language models. *CoRR*, abs/2302.07459.

Jiayan Guo, Lun Du, and Hengyu Liu. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *CoRR*, abs/2305.15066.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *CoRR*, abs/2111.09543.

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. *CoRR*, abs/2305.19523.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *CoRR*, abs/2302.09210.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021a. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. 2021b. OGB-LSC: A large-scale challenge for machine learning on graphs. In *Proceedings of the 2021 Annual Conference on Neural Information Processing Systems (NeurIPS)*.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 22118–22133.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *CoRR*, abs/2311.05232.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R. Benson. 2021. Combining label propagation and simple models out-performs graph neural networks. In *Proceedings of the 2021 International Conference on Learning Representations (ICLR)*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 2017 International Conference on Learning Representations (ICLR)*.

Satyapriya Krishna, Jiaqi Ma, Dylan Slack, Asma Ghandeharioun, Sameer Singh, and Himabindu Lakkaraju. 2023. Post hoc explanations of language models can improve language models. *CoRR*, abs/2305.11426.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 2020 Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 9459–9474.

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. *CoRR*, abs/2311.12399.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys (CSUR)*, 55(9):1–35.

David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. 2019. Chembl: towards direct deposition of bioassay data. *Nucleic Acids Research*, 47(D1):D930–D940.

OpenAI. 2023. Gpt-4.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *CoRR*, abs/2308.03188.

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. *CoRR*, abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 2017 Annual Conference on Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *CoRR*, abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 2022 Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 24824–24837.

Lanning Wei, Huan Zhao, Quanming Yao, and Zhiqiang He. 2021. Pooling architecture search for graph classification. In *Proceedings of the 2021 ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2091–2100. ACM.

David Weininger. 1988. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *Proceedings of the 2019 International Conference on Learning Representations (ICLR)*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *CoRR*, abs/2305.10601.

Qiang Zhang, Keyang Ding, Tianwen Lyv, Xinda Wang, Qingyu Yin, Yiwen Zhang, Jing Yu, Yuhao Wang, Xiaotong Li, Zhuoyi Xiang, et al. 2024. Scientific large language models: A survey on biological & chemical domains. *CoRR*, abs/2401.14656.

Zhiqiang Zhong, Anastasia Barkova, and Davide Mottin. 2023. Knowledge-augmented graph machine learning for drug discovery: A survey from precision to interpretability. *CoRR abs/2302.08261*.

Zhiqiang Zhong, Sergei Ivanov, and Jun Pang. 2022. Simplifying node classification on heterophilous graphs with compatible label propagation. *Transactions on Machine Learning Research (TMLR)*.

A Illustration of Molecule Representations

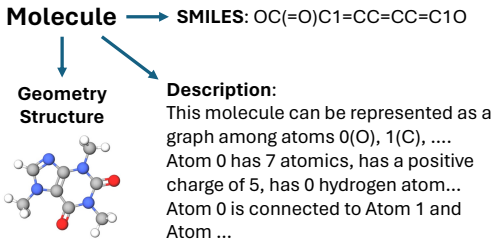

Figure 8: A molecule can be represented in different forms, *e.g.*, SMILES string, text description and geometry structure.

Molecules can be represented using various formats such as *SMILES string* (Weininger, 1988) and *geometry structures* (Zhang et al., 2024) (as shown in Figure 8). However, a notable limitation of existing LLMs is their reliance on unstructured text, rendering them unable to incorporate essential geometry structures as input (Li et al., 2023; Guo et al., 2023). To overcome this limitation, Fatemi et al. (2023) propose encoding the graph structure into text descriptions. In this paper, as depicted in Figure 8, we extend this concept by encoding both the molecule's atom features and graph structure into textual *descriptions*.

B Algorithm

Algorithm 1: LLMCORR

Input: Dataset $\mathcal{M} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m\}$, ML model f_{ML}, LLM f_{LLM}

Output: Refined predictions $\tilde{\mathcal{Y}}$

Complete training of $f_{ML} : \mathcal{M} \rightarrow \hat{\mathcal{Y}}$ by $\min_\Theta \sum_{i=1}^{n} \mathcal{L}(\hat{\mathcal{Y}}^i_{train}, \mathcal{Y}^i_{train})$;

Construct a contextual knowledge database $\mathcal{D} = \{\mathcal{M}_{train}, \mathcal{M}_{val}, \hat{\mathcal{Y}}_{val}\}$;

for $\mathcal{G}_u \in \{\mathcal{M}_{val} \cup \mathcal{M}_{test}\}$ **do**

  $\hat{y}_u = f_{ML}(\mathcal{G}_u)$

  $\mathcal{Q}_u = (\mathcal{G}_u, \hat{y}_u)$

  Create a prompt P_u using $\mathcal{Q}_u$ and retrieved contextual knowledge $\mathcal{D}_u \subset \mathcal{D}$

  Query the LLM and contain the refined prediction $\tilde{y}_u = f_{LLM}(P_u)$

end

We outline the process of LLMCORR in Algorithm 1. Given a dataset $\mathcal{M}$, an ML model f_{ML}, a

LLM f_{LLM}. After completing the training of the ML model (f_{ML}) on the training set $\mathcal{M}_{train}$ (line 1), we construct a contextual knowledge database $\mathcal{D}$ by incorporating the dataset's label information and the ML model's prediction on the validation dataset $\mathcal{M}_{val}$ (line 2). Given a query data $\mathcal{G}_u$, we create a prompt P_u using its primary prediction generated by f_{ML} and relevant contextual knowledge $\mathcal{D}_u$ (line 3-6). Finally, we query the LLM (f_{LLM}) to obtain the refined prediction $\tilde{y}_u$ (line 7).

C Dataset Description

We consider six benchmark molecule property prediction datasets that are common within ML research, which are summarised in Table 5.

1. `ogbg-molbace`. The `ogbg-molbace` dataset provides quantitative (IC_{50}) and qualitative (binary label) binding results for a set of inhibitors of human b-secretase 1 (BACE-1). All data are experimental values reported in the scientific literature over the past decade, some with detailed crystal structures available. MoleculeNet (Wu et al., 2018) merged a collection of 1,522 compounds with their 2D structures and binary labels, built as a classification task.
2. `ogbg-molbbbp`. The Blood–Brain Barrier Penetration (BBBP) dataset comes from scientific studies on the modelling and prediction of barrier permeability. As a membrane separating circulating blood and brain extracellular fluid, the blood–brain barrier blocks most drugs, hormones and neurotransmitters. Thus penetration of the barrier forms a long-standing issue in the development of drugs targeting the central nervous system. This dataset includes binary labels for over 2,039 compounds on their permeability properties. Scaffold splitting is also recommended for this well-defined target.
3. `ogbg-molhiv`. The HIV dataset was introduced by the Drug Therapeutics Program (DTP) AIDS Antiviral Screen, which tested the ability to inhibit HIV replication for 41,127 compounds. Screening results were evaluated and placed into three categories: confirmed inactive (CI), confirmed active (CA) and confirmed moderately active (CM). We further combine the latter two labels, making it a classification task between inactive (CI) and active (CA and CM). As we are more interested in discovering new categories of HIV inhibitors, scaffold splitting is recommended for this dataset.
4. `ogbg-molesol`. ESOL is a small dataset consisting of water solubility data for 1,128 compounds. The dataset has been used to train models that estimate solubility directly from chemical structures (as encoded in SMILES strings). Note that these structures don't include 3D coordinates, since solubility is a property of a molecule and not of its particular conformers.
5. `ogbg-molfreesolv`. The Free Solvation Database (FreeSolv) provides experimental and calculated hydration-free energy of small molecules in water. A subset of the compounds in the dataset is also used in the SAMPL blind prediction challenge. The calculated values are derived from alchemical free energy calculations using molecular dynamics simulations. We include the experimental values in the benchmark collection and use calculated values for comparison.
6. `ogbg-mollipo`. Lipophilicity is an important feature of drug molecules that affects both membrane permeability and solubility. This dataset, curated from the ChEMBL database (Mendez et al., 2019), provides experimental results of the octanol/water distribution coefficient (log D at pH 7.4) of 4200 compounds.

D Implementation

Implementation. We implement ML predictive models following their available official implementations. For instance, we adopt the available code of variant GNN models on the OGB benchmark leaderboards, *e.g.*, GCN[2], GIN[3], HIG [4] and PAS [5]. About DeBERTa, we adopt its official implementation [6] and incorporate it within the pipeline of TAPE [7]. For the LLMs, we simply call the API provided by OpenAI with default hyper-parameter settings. We empirically tried with some combinations of recommended important hyper-parameters, *e.g.*, temperature and top_P, yet did not observe significant improvement. To realise the embedding-based information retrieval for LLMCORR, we

[2] https://github.com/snap-stanford/ogb/tree/master/examples/graphproppred/mol

[3] https://github.com/snap-stanford/ogb/tree/master/examples/graphproppred/mol

[4] https://github.com/TencentYoutuResearch/HIG-GraphClassification

[5] https://github.com/LARS-research/PAS-OGB

[6] https://huggingface.co/microsoft/deberta-v3-base

[7] https://github.com/XiaoxinHe/TAPE

Table 5: Statistics summary of datasets used in our empirical study and splits from benchmark (Wu et al., 2018; Hu et al., 2020).

Dataset	#Graphs	Avg. #Nodes	Avg. #Edges	#Train	#Valid	#Test	Task Type
ogbg-molbace (Wu et al., 2018)	1,513	34.1	73.7	1,210	151	152	Binary class.
ogbg-molbbbp (Wu et al., 2018)	2,039	24.1	51.9	1,631	204	204	Binary class.
ogbg-molhiv (Wu et al., 2018; Hu et al., 2020)	41,127	25.5	27.5	32,901	4,113	4,113	Binary class.
ogbg-molesol (Wu et al., 2018)	1,128	13.3	27.4	902	113	113	Regression
ogbg-molfreesolv (Wu et al., 2018)	642	8.7	16.8	513	64	65	Regression
ogbg-mollipo (Wu et al., 2018)	4,200	27.0	59.0	3,360	420	420	Regression

adopt two capable embedding models (f_{Emb}) provided by OpenAI [8], *e.g.*, *text-embedding-3-large* and *text-embedding-3-small*. In this work, we mainly adopt *text-embedding-3-large* for better empirical performance. We perform careful discussions about the influence of different variants in Section 5.2.

E Are LLMs Predictors?

Following the thorough demonstration of LLM-CORR's efficacy as a post-hoc corrector in Section 5.1, a fundamental question emerged: does LLMCORR's remarkable performance stem from the LLM's ability to comprehend and rectify the ML model's predictions, or does it possess an inherent capability to predict molecule properties? To answer this question, undertake another series of empirical investigations. Specifically, we devise *predictor* prompts that task LLMs with directly predicting molecule properties, devoid of any information regarding the predictions made by the ML model. In the following sections, we will present our designed prompts and demonstrate the experimental results.

```
IP = [Instruction,
{"role": "user",
"content":
    "Predict whether the molecule with the SMILES string {SMILES} "
    "{task}. {description} "
    "Answer this question in the format: "
    "Prediction: <True or False>."}]
IE = [Instruction,
{"role": "user",
"content":
    "Predict whether the molecule with the SMILES string {SMILES} "
    "{task}. {description} "
    "Answer this question in the format: "
    "Prediction: <True or False>; Explanation: <text>."}]
```

Figure 9: Zero-shot prompt templates.

[8] https://platform.openai.com/docs/models/embeddings

```
FS = [Instruction,
# Start - Knowledge from training dataset
{"role": "user",
"content":
    "The SMILES string of molecule-{ID} is {SMILES}. "
    "{description} "
    "Predict whether molecule-{ID} {task}.",
},
{"role": "assistant",
"content": "Molecule-{ID} {task}."},
# End - Knowledge from training dataset
{"role": "user",
"content":
    "The SMILES string of molecule-{ID} is {SMILES}. "
    "{description} "
    "Predict whether molecule-{ID} {task}."
    "Answer this question in the format: "
    "Prediction: <True or False>; Explanation: <text>."}]
```

Figure 10: Few-shot prompt template. Multiple contextual knowledge can be included by expanding the template.

E.1 Predictor Prompt Engineering

Zero-shot Prompting. The first set of prompts (**IP**, **IE**) simply provides the LLM with molecule and task descriptions and asks it to generate the desired output with a desired format without any prior training or knowledge on the task, as illustrated in Figure 9. The only guidance we provide to the LLM is instruction, which tells about a little background context. Particularly, **IP** only asks the LLM to provide predictions, while **IE** further asks for explanations, which may ask the LLM to clarify the thought process in explanation generation and provide helpful evidence to help users understand the given prediction. In addition, if we fill out the *description* of **IP** and **IE**, which derives **IPD** and **IPD** prompts.

Few-shot Prompting. The second kind of prompt (**FS**) that we propose provides the LLM with a small number of examples of the task, along with the desired outputs (Brown et al., 2020). The model then learns from these examples to perform the task on new inputs. This approach can be categorised as a simple in-context learning (ICL) technique, An example prompt template is shown in Figure 10.

FS-k indicates k contextual knowledge instances are included in the prompt. In this work, we do not discuss the **FSD** prompts since the generated descriptions have tons of tokens, which will easily go over the LLM's input constraints.

We note there are also some popular recent ICL techniques, *e.g.*, Chain-of-thought (CoT) (Wei et al., 2022b), Tree-of-thought (ToT) (Yao et al., 2023), Graph-of-thought (GoT) (Besta et al., 2023) and Retrieval Augmented Generation (RaG) (Lewis et al., 2020), which are theoretically available to support complicated tasks and include large knowledge context. However, our initial experiments showed that methods, *e.g.*, CoT, ToT and GoT, perform much worse for molecule property prediction tasks due to the significant difficulties in designing proper chain thoughts without solid expertise. RaG implementations that we tested are unstable and slow with query, and they fall short of the relatively simpler **FS**'s performance. We argue it is caused by the unqualified information retrieval system, and we will discuss it in the future work discussion section.

E.2 Results - LLMs work as Predictors

From the results of Table 3, we can observe that the LLM can generate predictions about the molecule's property. However, LLM's performances are not significantly competitive compared with the ML models' performance. Hence, we argue existing LLMs are not competitive predictors and employing LLMs as effective predictors is still an open challenge.