The Impact of Large Language Models on Scientific Discovery: a Preliminary Study using GPT-4

Microsoft Research AI4Science
Microsoft Azure Quantum
llm4sciencediscovery@microsoft.com
November, 2023

Abstract

In recent years, groundbreaking advancements in natural language processing have culminated in the emergence of powerful large language models (LLMs), which have showcased remarkable capabilities across a vast array of domains, including the understanding, generation, and translation of natural language, and even tasks that extend beyond language processing. In this report, we delve into the performance of LLMs within the context of scientific discovery/research, focusing on GPT-4, the state-of-the-art language model. Our investigation spans a diverse range of scientific areas encompassing drug discovery, biology, computational chemistry (density functional theory (DFT) and molecular dynamics (MD)), materials design, and partial differential equations (PDE).

Evaluating GPT-4 on scientific tasks is crucial for uncovering its potential across various research domains, validating its domain-specific expertise, accelerating scientific progress, optimizing resource allocation, guiding future model development, and fostering interdisciplinary research. Our exploration methodology primarily consists of expert-driven case assessments, which offer qualitative insights into the model's comprehension of intricate scientific concepts and relationships, and occasionally benchmark testing, which quantitatively evaluates the model's capacity to solve well-defined domain-specific problems.

Our preliminary exploration indicates that GPT-4 exhibits promising potential for a variety of scientific applications, demonstrating its aptitude for handling complex problem-solving and knowledge integration tasks. We present an analysis of GPT-4's performance in the aforementioned domains (e.g., drug discovery, biology, computational chemistry, materials design, etc.), emphasizing its strengths and limitations. Broadly speaking, we evaluate GPT-4's knowledge base, scientific understanding, scientific numerical calculation abilities, and various scientific prediction capabilities.

In biology and materials design, GPT-4 possesses extensive domain knowledge that can help address specific requirements. In other fields, like drug discovery, GPT-4 displays a strong ability to predict properties. However, in research areas like computational chemistry and PDE, while GPT-4 shows promise for aiding researchers with predictions and calculations, further efforts are required to enhance its accuracy. Despite its impressive capabilities, GPT-4 can be improved for quantitative calculation tasks, e.g., fine-tuning is needed to achieve better accuracy.[1]

We hope this report serves as a valuable resource for researchers and practitioners seeking to harness the power of LLMs for scientific research and applications, as well as for those interested in advancing natural language processing for domain-specific scientific tasks. It's important to emphasize that the field of LLMs and large-scale machine learning is progressing rapidly, and future generations of this technology may possess additional capabilities beyond those highlighted in this report. Notably, the integration of LLMs with specialized scientific tools and models, along with the development of foundational scientific models, represent two promising avenues for exploration.

[1]Please note that GPT-4's capabilities can be greatly enhanced by integrating with specialized scientific tools and models, as demonstrated in AutoGPT and ChemCrow. However, the focus of this paper is to study the intrinsic capabilities of LLMs in tackling scientific tasks, and the integration of LLMs with other tools/models is largely out of our scope. We only had some brief discussions on this topic in the last chapter.

Contents

1 Introduction

The rapid development of artificial intelligence (AI) has led to the emergence of sophisticated large language models (LLMs), such as GPT-4 [62] from OpenAI, PaLM 2 [4] from Google, Claude from Anthropic, LLaMA 2 [85] from Meta, etc. LLMs are capable of transforming the way we generate and process information across various domains and have demonstrated exceptional performance in a wide array of tasks, including abstraction, comprehension [23], vision [29, 89], coding [66], mathematics [97], law [41], understanding of human motives and emotions, and more. In addition to the prowess in the realm of text, they have also been successfully integrated into other domains, such as image processing [114], speech recognition [38], and even reinforcement learning, showcasing its adaptability and potential for a broad range of applications. Furthermore, LLMs have been used as controllers/orchestrators [76, 83, 94, 106, 34, 48] to coordinate other machine learning models for complex tasks.

Among these LLMs, GPT-4 has gained substantial attention for its remarkable capabilities. A recent paper has even indicated that GPT-4 may be exhibiting early indications of artificial general intelligence (AGI) [11]. Because of its extraordinary capabilities in general AI tasks, GPT-4 is also garnering significant attention in the scientific community [71], especially in domains such as medicine [45, 87], healthcare [61, 91], engineering [67, 66], and social sciences [28, 5].

In this study, our primary goal is to examine the capabilities of LLMs within the context of natural science research. Due to the extensive scope of the natural sciences, covering all sub-disciplines is infeasible; as such, we focus on a select set of areas, including drug discovery, biology, computational chemistry, materials design, and partial differential equations (PDE). Our aim is to provide a broad overview of LLMs' performance and their potential applicability in these specific scientific fields, with GPT-4, the state-of-the-art LLM, as our central focus. A summary of this report can be found in Fig. 1.1.

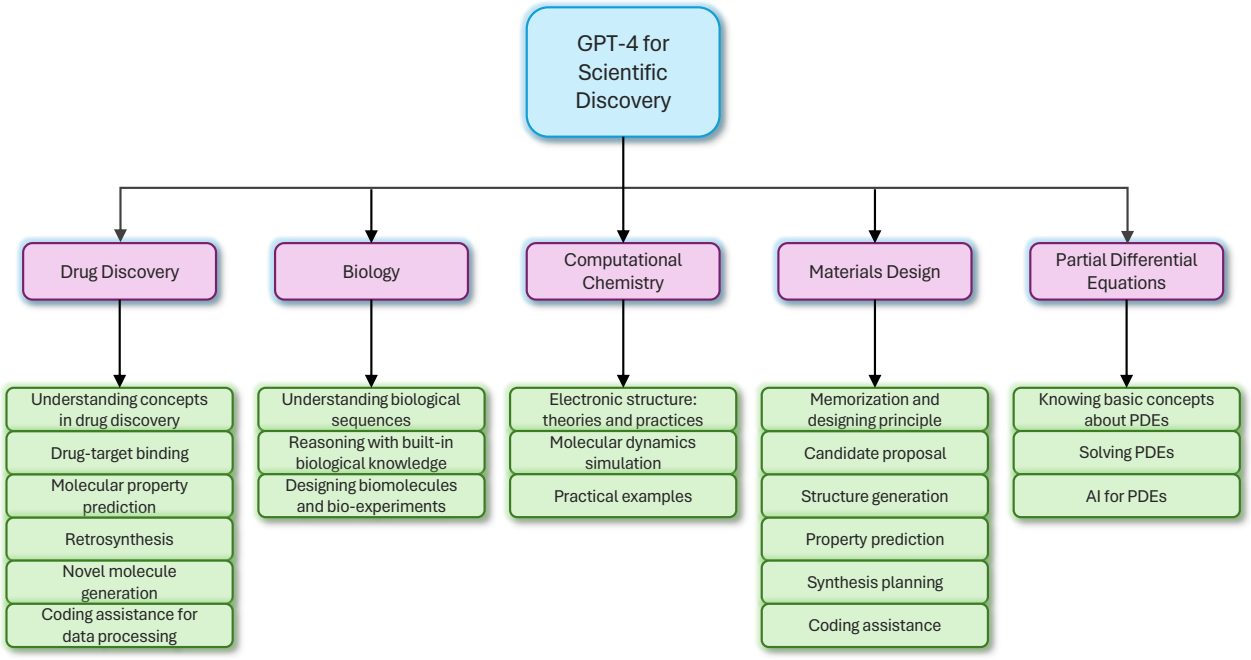


Figure 1.1: Overview of this report.

1.1 Scientific areas

Natural science is dedicated to understanding the natural world through systematic observation, experimentation, and the formulation of testable hypotheses. These strive to uncover the fundamental principles and

laws governing the universe, spanning from the smallest subatomic particles to the largest galaxies and beyond. Natural science is an incredibly diverse field, encompassing a wide array of disciplines, including both physical sciences, which focus on non-living systems, and life sciences, which investigate living organisms. In this study, we have opted to concentrate on a subset of natural science areas, selected from both physical and life sciences. It is important to note that these areas are not mutually exclusive; for example, drug discovery substantially overlaps with biology, and they do not all fall within the same hierarchical level in the taxonomy of natural science.

Drug discovery is the process by which new candidate medications are identified and developed to treat or prevent specific diseases and medical conditions. This complex and multifaceted field aims to improve human health and well-being by creating safe, effective, and targeted therapeutic agents. In this report, we explore how GPT-4 can help drug discovery research (Sec. 2) and study several key tasks in drug discovery: knowledge understanding (Sec. 2.2), molecular property prediction (Sec. 2.4), molecular manipulation (Sec. 2.2.3), drug-target binding prediction (Sec. 2.3), and retrosynthesis (Sec. 2.5).

Biology is a branch of life sciences that studies life and living organisms, including their structure, function, growth, origin, evolution, distribution, and taxonomy. As a broad and diverse field, biology encompasses various sub-disciplines that focus on specific aspects of life, such as genetics, ecology, anatomy, physiology, and molecular biology, among others. In this report, we explore how LLMs can help biology research (Sec. 3), mainly understanding biological sequences (Sec. 3.2), reasoning with built-in biological knowledge (Sec. 3.3), and designing biomolecules and bio-experiments (Sec. 3.4).

Computational chemistry is a branch of chemistry (and also physical sciences) that uses computer simulations and mathematical models to study the structure, properties, and behavior of molecules, as well as their interactions and reactions. By leveraging the power of computational techniques, this field aims to enhance our understanding of chemical processes, predict the behavior of molecular systems, and assist in the design of new materials and drugs. In this report, we explore how LLMs can help research in computational chemistry (Sec. 4), mainly focusing on electronic structure modeling (Sec. 4.2) and molecular dynamics simulation (Sec. 4.3).

Materials design is an interdisciplinary field that investigates (1) the relationship between the structure, properties, processing, and performance of materials, and (2) the discovery of new materials. It combines elements of physics, chemistry, and engineering. This field encompasses a wide range of natural and synthetic materials, including metals, ceramics, polymers, composites, and biomaterials. The primary goal of materials design is to understand how the atomic and molecular arrangement of a material affects its properties and to develop new materials with tailored characteristics for various applications. In this report, we explore how GPT-4 can help research in materials design (Sec. 5), e.g., understanding materials knowledge (Sec. 5.2), proposing candidate compositions (Sec. 5.3), generating materials structure (Sec. 5.4), predicting materials properties (Sec. 5.5), planning synthesis routes (Sec. 5.6), and assisting code development (Sec. 5.7).

Partial Differential Equations (PDEs) represent a category of mathematical equations that delineate the relationship between an unknown function and its partial derivatives concerning multiple independent variables. PDEs have applications in modeling significant phenomena across various fields such as physics, engineering, biology, economics, and finance. Examples of these applications include fluid dynamics, electromagnetism, acoustics, heat transfer, diffusion, financial models, population dynamics, reaction-diffusion systems, and more. In this study, we investigate how GPT-4 can contribute to PDE research (Sec. 6), emphasizing its understanding of fundamental concepts and AI techniques related to PDEs, theorem-proof capabilities, and PDE-solving abilities.

1.2 Capabilities to evaluate

We aim to understand how GPT-4 can help natural science research and its potential limitations in scientific domains. In particular, we study the following capabilities:

- Accessing and analyzing scientific literature. Can GPT-4 suggest relevant research papers, extract key information, and summarize insights for researchers?
- Concept clarification. Is GPT-4 capable of explaining and providing definitions for scientific terms, concepts, and principles, helping researchers better understand the subject matter?
- Data analysis. Can GPT-4 process, analyze, and visualize large datasets from experiments, simulations, and field observations, and uncover non-obvious trends and relationships in complex data?

- Theoretical modeling. Can GPT-4 assist in developing mathematical/computational models of physical systems, which would be useful for fields like physics, chemistry, climatology, systems biology, etc.?
- Methodology guidance. Could GPT-4 help researchers choose the right experimental/computational methods and statistical tests for their research by analyzing prior literature or running simulations on synthetic data?
- Prediction. Is GPT-4 able to analyze prior experimental data to make predictions on new hypothetical scenarios and experiments (e.g., in-context few-shot learning), allowing for a focus on the most promising avenues?
- Experimental design. Can GPT-4 leverage knowledge in the field to suggest useful experimental parameters, setups, and techniques that researchers may not have considered, thereby improving experimental efficiency?
- Code development. Could GPT-4 assist in developing code for data analysis, simulations, and machine learning across a wide range of scientific applications by generating code from natural language descriptions or suggesting code snippets from a library of prior code?
- Hypothesis generation. By connecting disparate pieces of information across subfields, can GPT-4 come up with novel hypotheses (e.g., compounds, proteins, materials, etc.) for researchers to test in their lab, expanding the scope of their research?

1.3 Our methodologies

In this report, we choose the best LLM to date, GPT-4, to study and evaluate the capabilities of LLMs across scientific domains. We use the GPT-4 model[2] available through the Azure OpenAI Service.[3].

We employ a combination of qualitative[4] and quantitative approaches, ensuring a good understanding of its proficiency in scientific research.

In the case of most capabilities, we primarily adopt a qualitative approach, carefully designing tasks and questions that not only showcase GPT-4's capabilities in terms of its scientific expertise but also address the fundamental inquiry: *the extent of GPT-4's proficiency in scientific research*. Our objective is to elucidate the depth and flexibility of its understanding of diverse concepts, skills, and fields, thereby demonstrating its versatility and potential as a powerful tool in scientific research. Moreover, we scrutinize GPT-4's responses and actions, evaluating their consistency, coherence, and accuracy, while simultaneously identifying potential limitations and biases. This examination allows us to gain a deeper understanding of the system's potential weaknesses, paving the way for future improvements and refinements. Throughout our study, we present numerous intriguing cases spanning each scientific domain, illustrating the diverse capabilities of GPT-4 in areas such as concept capture, knowledge comprehension, and task assistance.

For certain capabilities, particularly predictive ones, we also employ a quantitative approach, utilizing public benchmark datasets to evaluate GPT-4's performance on well-defined tasks, in addition to presenting a wide array of case studies. By incorporating quantitative evaluations, we can objectively assess the model's performance in specific tasks, allowing for a more robust and reliable understanding of its strengths and limitations in scientific research applications.

In summary, our methodologies for investigating GPT-4's performance in scientific domains involve a blend of qualitative and quantitative approaches, offering a holistic and systematic understanding of its capabilities and limitations.

1.4 Our observations

GPT-4 demonstrates considerable potential in various scientific domains, including drug discovery, biology, computational chemistry, materials design, and PDEs. Its capabilities span a wide range of tasks and it exhibits an impressive understanding of key concepts in each domain.

[2]The output of GPT-4 depends on several variables such as the model version, system messages, and hyperparameters like the decoding temperature. Thus, one might observe different responses for the same cases examined in this report. For the majority of this report, we primarily utilized GPT-4 version 0314, with a few cases employing version 0613.

[3]`https://azure.microsoft.com/en-us/products/ai-services/openai-service/`

[4]The qualitative approach used in this report mainly refers to case studies. It is related to but not identical to qualitative methods in social science research.

In drug discovery, GPT-4 shows a comprehensive grasp of the field, enabling it to provide useful insights and suggestions across a wide range of tasks. It is helpful in predicting drug-target binding affinity, molecular properties, and retrosynthesis routes. It also has the potential to generate novel molecules with desired properties, which can lead to the discovery of new drug candidates with the potential to address unmet medical needs. However, it is important to be aware of GPT-4's limitations, such as challenges in processing SMILES sequences and limitations in quantitative tasks.

In the field of biology, GPT-4 exhibits substantial potential in understanding and processing complex biological language, executing bioinformatics tasks, and serving as a scientific assistant for biology design. Its extensive grasp of biological concepts and its ability to perform various tasks, such as processing specialized files, predicting signaling peptides, and reasoning about plausible mechanisms from observations, benefit it to be a valuable tool in advancing biological research. However, GPT-4 has limitations when it comes to processing biological sequences (e.g., DNA and FASTA sequences) and its performance on tasks related to under-studied entities.

In computational chemistry, GPT-4 demonstrates remarkable potential across various subdomains, including electronic structure methods and molecular dynamics simulations. It is able to retrieve information, suggest design principles, recommend suitable computational methods and software packages, generate code for various programming languages, and propose further research directions or potential extensions. However, GPT-4 may struggle with generating accurate atomic coordinates of complex molecules, handling raw atomic coordinates, and performing precise calculations.

In materials design, GPT-4 shows promise in aiding materials design tasks by retrieving information, suggesting design principles, generating novel and feasible chemical compositions, recommending analytical and numerical methods, and generating code for different programming languages. However, it encounters challenges in representing and proposing more complex structures, e.g., organic polymers and MOFs, generating accurate atomic coordinates, and providing precise quantitative predictions.

In the realm of PDEs, GPT-4 exhibits its ability to understand the fundamental concepts, discern relationships between concepts, and provide accurate proof approaches. It is able to recommend appropriate analytical and numerical methods for addressing various types of PDEs and generate code in different programming languages to numerically solve PDEs. However, GPT-4's proficiency in mathematical theorem proving still has room for growth, and its capacity for independently discovering and validating novel mathematical theories remains limited in scope.

In summary, GPT-4 exhibits both significant potential and certain limitations for scientific discovery. To better leverage GPT-4, researchers should be cautious and verify the model's outputs, experiment with different prompts, and combine its capabilities with dedicated AI models or computational tools to ensure reliable conclusions and optimal performance in their respective research domains:

- Interpretability and Trust: It is crucial to maintain a healthy skepticism when interpreting GPT-4's output. Researchers should always critically assess the generated results and cross-check them with existing knowledge or expert opinions to ensure the validity of the conclusions.
- Iterative Questioning and Refinement: GPT-4's performance can be improved by asking questions in an iterative manner or providing additional context. If the initial response from GPT-4 is not satisfactory, researchers can refine their questions or provide more information to guide the model toward a more accurate and relevant answer.
- Combining GPT-4 with Domain-Specific Tools: In many cases, it may be beneficial to combine GPT-4's capabilities with more specialized tools and models designed specifically for scientific discovery tasks, such as molecular docking software, or protein folding algorithms. This combination can help researchers leverage the strengths of both GPT-4 and domain-specific tools to achieve more reliable and accurate results. Although we do not extensively investigate the integration of LLMs and domain-specific tools/models in this report, a few examples are briefly discussed in Section 7.2.1.

1.5 Limitations of this study

First, a large part of our assessment of GPT-4's capabilities utilizes case studies. We acknowledge that this approach is somewhat subjective, informal, and lacking in rigor per formal scientific standards. However, we believe that this report is useful and helpful for researchers interested in leveraging LLMs for scientific discovery. We look forward to the development of more formal and comprehensive methods for testing and analyzing LLMs and potentially more complex AI systems in the future for scientific intelligence.

Second, in this study, we primarily focus on the scientific intelligence of GPT-4 and its applications in various scientific domains. There are several important aspects, mainly responsible AI, beyond the scope of this work that warrant further exploration for GPT-4 and all LLMs:

- Safety Concerns: Our analysis does not address the ability of GPT-4 to safely respond to hazardous chemistry or drug-related situations. Future studies should investigate whether these models provide appropriate safety warnings and precautions when suggesting potentially dangerous chemical reactions, laboratory practices, or drug interactions. This could involve evaluating the accuracy and relevance of safety information generated by LLMs and determining if they account for the risks and hazards associated with specific scientific procedures.
- Malicious Usage: Our research does not assess the potential for GPT-4 to be manipulated for malicious purposes. It is crucial to examine whether it has built-in filters or content-monitoring mechanisms that prevent it from disclosing harmful information, even when explicitly requested. Future research should explore the potential vulnerabilities of LLMs to misuse and develop strategies to mitigate risks, such as generating false or dangerous information.
- Data Privacy and Security: We do not investigate the data privacy and security implications of using GPT-4 in scientific research. Future studies should address potential risks, such as the unintentional leakage of sensitive information, data breaches, or unauthorized access to proprietary research data.
- Bias and Fairness: Our research does not examine the potential biases present in LLM-generated content or the fairness of their outputs. It is essential to assess whether these models perpetuate existing biases, stereotypes, or inaccuracies in scientific knowledge and develop strategies to mitigate such issues.
- Impact on the Scientific Workforce: We do not analyze the potential effects of LLMs on employment and job opportunities within the scientific community. Further research should consider how the widespread adoption of LLMs may impact the demand for various scientific roles and explore strategies for workforce development, training, and skill-building in the context of AI-driven research.
- Ethics and Legal Compliance: We do not test the extent to which LLMs adhere to ethical guidelines and legal compliance requirements related to scientific use. Further investigation is needed to determine if LLM-generated content complies with established ethical standards, data privacy regulations, and intellectual property laws. This may involve evaluating the transparency, accountability, and fairness of LLMs and examining their potential biases or discriminatory outputs in scientific research contexts.

By addressing these concerns in future studies, we can develop a more holistic understanding of the potential benefits, challenges, and implications of LLMs in the scientific domain, paving the way for more responsible and effective use of these advanced AI technologies.

2 Drug Discovery

2.1 Summary

Drug discovery is the process by which new candidate medications are identified and developed to treat or prevent specific diseases and medical conditions. This complex and multifaceted field aims to improve human health and well-being by creating safe, effective, and targeted therapeutic agents. The importance of drug discovery lies in its ability to identify and develop new therapeutics for treating diseases, alleviating suffering, and improving human health [72]. It is a vital part of the pharmaceutical industry and plays a crucial role in advancing medical science [64]. Drug discovery involves a complex and multidisciplinary process, including target identification, lead optimization, and preclinical testing, ultimately leading to the development of safe and effective drugs [35].

Assessing GPT-4's capabilities in drug discovery has significant potential, such as accelerating the discovery process [86], reducing the search and design cost [73], enhancing creativity, and so on. In this chapter, we first study GPT-4's knowledge about drug discovery through qualitative tests (Sec. 2.2), and then study its predictive capabilities through quantitative tests on multiple crucial tasks, including drug-target interaction/binding affinity prediction (Sec. 2.3), molecular property prediction (Sec. 2.4), and retrosynthesis prediction (Sec. 2.5).

We observe the considerable potential of GPT-4 for drug discovery:[5]

- Broad Knowledge: GPT-4 demonstrates a wide-ranging understanding of key concepts in drug discovery, including individual drugs (Fig. 2.4), target proteins (Fig. 2.6), general principles for small-molecule drugs (Fig. 2.8), and the challenges faced in various stages of the drug discovery process (Fig. 2.9). This broad knowledge base allows GPT-4 to provide useful insights and suggestions across a wide range of drug discovery tasks.
- Versatility in Key Tasks: LLMs, such as GPT-4, can help in several essential tasks in drug discovery, including:
  - Molecule Manipulation: GPT-4 is able to generate new molecular structures by modifying existing ones (Fig. 2.7), potentially leading to the discovery of novel drug candidates.
  - Drug-Target Binding Prediction: GPT-4 is able to predict the interaction between of a molecule to a target protein (Table 4), which can help in identifying promising drug candidates and optimizing their binding properties.
  - Molecule Property Prediction: GPT-4 is able to predict various physicochemical and biological properties of molecules (Table 5), which can guide the selection and optimization of drug candidates.
  - Retrosynthesis Prediction: GPT-4 is able to predict synthetic routes for target molecules, helping chemists design efficient and cost-effective strategies for the synthesis of potential drug candidates (Fig. 2.23).
- Novel Molecule Generation: GPT-4 can be used to generate novel molecules following text instruction. This de novo molecule generation capability can be a valuable tool for identifying new drug candidates with the potential to address unmet medical needs (Sec. 2.6).
- Coding capability: GPT-4 can provide help in coding for drug discovery, offering large benefits in data downloading, processing, and so on (Fig. 2.27, Fig 2.28). The strong coding capability of GPT-4 can greatly ease human efforts in the future.

While GPT-4 is a useful tool for assisting research in drug discovery, it's important to be aware of its limitations and potential errors. To better leverage GPT-4, we provide several tips for researchers:

- SMILES Sequence Processing Challenges: GPT-4 may struggle with directly processing SMILES sequences. To improve the model's understanding and output, it is better to provide the names of drug molecules along with their descriptions, if possible. This will give the model more context and improve its ability to generate relevant and accurate responses.
- Limitations in Quantitative Tasks: While GPT-4 excels in qualitative tasks and questions, it may face limitations when it comes to quantitative tasks, such as predicting numerical values for molecular

[5] In this chapter, we employ a color-coding scheme to illustrate the results of GPT-4. We use green to highlight both (1) the crucial information in the user prompts and (2) significant or accurate elements in GPT-4's output. Conversely, we use yellow to indicate incorrect or inaccurate responses from GPT-4.

properties and drug-target binding in our evaluated datasets. Researchers are advised to take GPT-4's output as a reference in these cases and perform verification using dedicated AI models or scientific computational tools to ensure reliable conclusions.

- Double-Check Generated Molecules: When generating novel molecules with GPT-4, it is essential to verify the validity and chemical properties of the generated structures.

2.2 Understanding key concepts in drug discovery

Understanding fundamental and important concepts in drug discovery is the first step to testing GPT-4's intelligence in this domain. In this subsection, we ask questions from different perspectives to test GPT-4's knowledge. The system message is set as in Fig. 2.1, which is added to each prompt.

GPT-4

System message:

You are a drug assistant and should be able to help with drug discovery tasks.

Figure 2.1: System message used in all the prompts in Sec. 2.2.

2.2.1 Entity translation

In this subsection, we focus on evaluating the performance of GPT-4 in translating drug names, IUPAC nomenclature, chemical formula, and SMILES representations.

Drug names, IUPAC nomenclature, chemical formula, and SMILES strings serve as crucial building blocks for understanding and conveying chemical structures and properties for drug molecules. These representations are essential for researchers to communicate, search, and analyze chemical compounds effectively. Several examples are shown in Fig. 2.2 and Fig. 2.3.

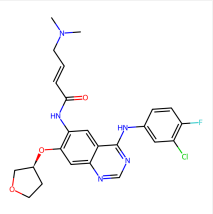
The first example is to generate the chemical formula, IUPAC name, and the SMILES for a given drug name, which is the translation between names and other representations of drugs. We take Afatinib as the input drug. As shown in Fig. 2.2, GPT-4 correctly outputs the chemical formula as $C_{24}H_{25}ClFN_5O_3$, and the IUPAC name is also right, which means GPT-4 is aware of drug Afatinib. However, the SMILES is not correct. Therefore, we give guidance and let GPT-4 generate SMILES again. Unfortunately, as shown in the figure, though we explicitly ask GPT-4 to "pay attention to the number of atoms of each atom type" and generate based on the correct IUPAC and chemical formula, the generated SMILES sequences in several trials are still not correct.

In Fig. 2.3, we ask GPT-4 to translate between the IUPAC name and the SMILES sequence and output the chemical formula. As presented in the figure, the translation from the SMILES sequence to the IUPAC name is correct, but the other direction is incorrect. Besides, the generated chemical formulas in both translation directions are incorrect. From these two cases, we can see that IUPAC looks to be much easier for GPT-4 to understand and generate while generating SMILES faces some challenges. We suspect one possible reason may be the tokenization method used in GPT-4 since it is based on the subword representation, which may not be easy for the SMILES representation where each character has its specific meaning for a drug molecule. This hypothesis can also explain the situation in that the generated chemical formula is not always correct (as shown in Fig. 2.3) since the formula is based on the count of each atom type.

GPT-4

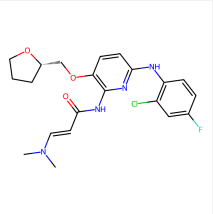
Prompt:

What's the chemical formula, IUPAC name, and the SMILES of Afatinib?



GPT-4:

Afatinib is a tyrosine kinase inhibitor used for the treatment of non-small cell lung cancer.



Chemical formula: $C_{24}H_{25}ClFN_5O_3$

IUPAC name: N-[4-[(3-chloro-4-fluorophenyl)amino]-7-[[(3S)-tetrahydro-3-furanyl]oxy]-6-quinazolinyl]-4-(dimethylamino)-2-butenamide

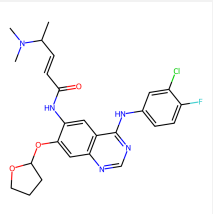
SMILES: CN(C)C=C\C(=O)NC1=C(C=CC(=N1)NC2=C(C=C(C=C2)F)Cl)OC[C@@H]3CCCO3

Prompt:

The SMILES is wrong. Show me the canonical SMILES.

GPT-4:

Apologies for the confusion earlier. Here is the correct canonical SMILES for Afatinib:



Canonical SMILES: CC(C=C\C(=O)Nc1cc2c(cc1OC3CCCO3)ncnc2Nc4ccc(c(c4)Cl)F)N(C)C

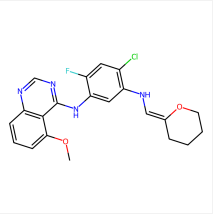
Please note that this SMILES notation does not include the stereochemistry information.

Prompt:

The SMILES is still wrong. Pay attention to the number of atoms of each atom type and show me the correct SMILES.

GPT-4:

I apologize for my previous mistakes. Here is the correct SMILES notation for Afatinib:



SMILES: COC1=CC=CC2=C1C(NC3=CC(=C(C=C3F)Cl)NC=C4CCCCO4)=NC=N2

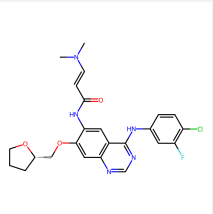
Again, I apologize for the confusion and appreciate your patience.

Prompt:

The chemical formula and the IUPAC name are correct, but the SMILES is wrong. Generate SMILES based on the chemical formula and the IUPAC name.

GPT-4:

My apologies for the confusion earlier. Based on the chemical formula and the IUPAC name, the correct SMILES notation for Afatinib should be:



SMILES: CN(C)C=C/C(=O)Nc1cc2c(cc1OC[C@@H]1CCCO1)ncnc2Nc1cc(c(cc1)Cl)F

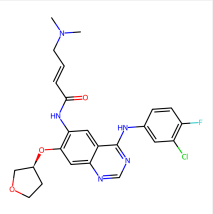
Please note that this SMILES notation includes the stereochemistry information as well.

Figure 2.2: Example of entity translation. Given a drug name, generate its chemical formula, IUPAC name, and the SMILES. The first molecular graph is the ground-truth 2D structure of Afatinib, while the latter four graphs are converted from the SMILES sequences generated by GPT-4.

GPT-4

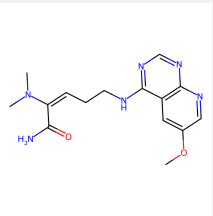
Prompt:

Given the IUPAC name "(2E)-N-4-[(3-chloro-4-fluorophenyl)amino]-7-[(3S)-oxolan-3-yloxy]quinazolin-6-yl-4-(dimethylamino)but-2-enamide" of a compound, generate its SMILES sequence.



GPT-4:

The compound's IUPAC name "(2E)-N-4-[(3-chloro-4-fluorophenyl)amino]-7-[(3S)-oxolan-3-yloxy]quinazolin-6-yl-4-(dimethylamino)but-2-enamide" corresponds to the SMILES sequence "COC1=CC2=C(N=C1)N=CN=C2NCC\C=C(/C(=O)N)N(C)C" and the chemical formula $C_{22}H_{22}ClFN_4O_3$.



Prompt:

Given the SMILES "CN(C)C\C=C\C(=O)NC1=C(O[C@H]2CCOC2)C=C2N=CN=C(NC3=CC(Cl)=C(F)C=C3)C2=C1" of a compound, generate its IUPAC name and chemical formula.

GPT-4:

The compound's SMILES "CN(C)C\C=C\C(=O)NC1=C(O[C@H]2CCOC2)C=C2N=CN=C(NC3=CC(Cl)=C(F)C=C3)C2=C1" corresponds to the IUPAC name (6R)-6-((5-chloro-2-fluorophenyl)amino)-3-methyl-5-(3-methylimidazo[1,2-a]pyridin-2-yl)-3,6-dihydro-2H-pyridin-2-one 2-oxide and the chemical formula $C_{19}H_{16}ClFN_4O_2$.

Figure 2.3: Translation between IUPAC name and SMILES.

2.2.2 Knowledge/information memorization

We evaluate GPT-4's capability of knowledge and information memorization for drug discovery in this subsection.

The drug we choose is still Afatinib and the cases are presented in Fig. 2.4. We first ask GPT-4 to give an introduction to the drug Afatinib, GPT-4 replies with multiple pieces of information, covering the molecule type, its target receptor, the FDA-proved time, the function, pathway, side effects, and so on. With checking from Pubchem and DrugBank, the information given by GPT-4 is clear and correct, which demonstrates that GPT-4 has knowledge about Afatinib. Then, we further ask more about the chemical properties of Afatinib, which is crucial to truly understand a molecule drug. In Fig. 2.5, GPT-4 answers a variety of properties about Afatinib, such as the molecular weight, solubility, XLogP3, hydrogen bond acceptor count, etc. (1) Most of the introduced properties are correct. (2) Some properties are not correct, especially for counting related results such as the number of heavy atoms. (3) Interestingly, the generated SMILES notation for Afatinib is wrong again, which aligns well with the observation in the previous subsection and suggests that generating SMILES still remains a challenge for GPT-4.

GPT-4

Prompt:
Please give an introduction to the drug Afatinib.

GPT-4:
Afatinib is a small molecule, tyrosine kinase inhibitor (TKI) that is designed to selectively and irreversibly block the epidermal growth factor receptor (EGFR) family. It specifically targets the human EGFR (HER) family of receptor tyrosine kinases, which includes EGFR (ErbB1), HER2 (ErbB2), HER3 (ErbB3), and HER4 (ErbB4). These receptors play a crucial role in the growth, survival, and spread of cancer cells.

Afatinib was developed by Boehringer Ingelheim and is marketed under the brand name Gilotrif or Giotrif. It was first approved by the U.S. Food and Drug Administration (FDA) in July 2013 for the treatment of metastatic non-small cell lung cancer (NSCLC) with EGFR exon 19 deletions or exon 21 (L858R) substitution mutations as detected by an FDA-approved test.

In addition to NSCLC, Afatinib has been studied in various types of solid tumors, including breast, head and neck, and colorectal cancers. It has shown efficacy in tumors that overexpress or have mutations in the EGFR family, which contributes to the development and progression of the disease.

Afatinib works by inhibiting the intracellular signaling pathways involved in cell proliferation and survival, thereby blocking the growth and spread of cancer cells. Its mechanism of action is distinct from other EGFR inhibitors, as it forms a covalent bond with the target receptor, resulting in irreversible inhibition.

The most common side effects associated with Afatinib include diarrhea, rash, acne, dry skin, and mouth sores. Additionally, serious side effects such as interstitial lung disease, liver toxicity, and gastrointestinal perforation have been reported. It is essential for patients to be closely monitored by their healthcare providers while on Afatinib treatment to manage any potential side effects.

In summary, Afatinib is a targeted therapy that specifically inhibits the EGFR family of receptor tyrosine kinases, providing a treatment option for patients with certain types of EGFR-mutated NSCLC and other solid tumors.

Prompt:
Give an introduction to the drug Afatinib and focus on its chemical information such as formula, SMILES, properties, molecular weight, and so on.

GPT-4:
Afatinib is a small molecule, tyrosine kinase inhibitor (TKI) that specifically targets the human EGFR (HER) family of receptor tyrosine kinases. Chemically, Afatinib is known as N-[4-[(3-chloro-4-fluorophenyl)amino]-7-[[(3S)-tetrahydro-3-furanyl]oxy]-6-quinazolinyl]-4-(dimethylamino)-2-butenamide.

Here are some key chemical properties of Afatinib:

- Molecular Formula: $C_{24}H_{25}ClFN_5O_3$
- Molecular Weight: 485.94 g/mol
- SMILES notation: CN(C)C/C=C/C(=O)Nc1cc2c(cc1Cl)ncnc2O[C@@H]3COCC3
- IUPAC Name: N-[4-[(3-chloro-4-fluorophenyl)amino]-7-[[(3S)-tetrahydro-3-furanyl]oxy]-6-quinazolinyl]-4-(dimethylamino)-2-butenamide
- Appearance: Afatinib is a white to yellowish powder
- Solubility: It is sparingly soluble in water and moderately soluble in organic solvents such as methanol and DMSO.

Afatinib's chemical structure features a quinazoline core, which is essential for its pharmacological activity. The molecule forms a covalent bond with the target receptor, leading to irreversible inhibition of the EGFR family. This distinguishes it from other reversible EGFR inhibitors and contributes to its unique mechanism of action in treating certain types of EGFR-mutated cancers.

It is important to note that Afatinib is a chiral molecule, and the approved drug is a single enantiomer with the (S)-configuration at the tetrahydrofuran ring. This enantiomeric purity is essential for its pharmacological activity and selectivity.

Figure 2.4: Retrieve general and chemical information about the drug Afatinib from GPT-4. Most of the knowledge about Afatinib is correct.

GPT-4 (continue)

Prompt:

Can you introduce more properties about Afatinib, such as XLogP3, Hydrogen bond donor count, topological polar surface area, and so on?

GPT-4:

Certainly! Here are some additional physicochemical properties of Afatinib:

- XLogP3: XLogP3 is a measure of the lipophilicity of a compound, which can affect its absorption, distribution, metabolism, and excretion (ADME) properties. Afatinib has an XLogP3 value of 4.2, indicating moderate lipophilicity.
- Hydrogen Bond Donor Count: Afatinib has 2 hydrogen bond donor groups. These are the amide nitrogen and the secondary amine nitrogen in the quinazoline core.
- Hydrogen Bond Acceptor Count: Afatinib has 9 hydrogen bond acceptor groups. These include oxygen atoms in the tetrahydrofuran ring, amide carbonyl, and quinazoline core, as well as nitrogen atoms in the quinazoline core and dimethylamino group.
- Topological Polar Surface Area (TPSA): TPSA is a measure of the molecular surface area that is occupied by polar atoms (oxygen, nitrogen, and their attached hydrogen atoms).
- It is a useful parameter for predicting drug transport properties, such as permeability and solubility. Afatinib has a TPSA value of 78.63 $Å^2$, which indicates a moderate polar surface area.
- Rotatable Bond Count: Afatinib has 8 rotatable bonds. These are the single bonds that allow for conformational flexibility in the molecule and can influence its pharmacokinetic properties.
- Heavy Atom Count: Afatinib contains 33 heavy atoms (non-hydrogen atoms).

These physicochemical properties play a role in determining Afatinib's pharmacokinetic profile, including its absorption, distribution, metabolism, and excretion in the body. It is crucial to consider these properties when designing and optimizing drug candidates to improve their efficacy and safety.

Figure 2.5: Retrieve molecular property information about the drug Afatinib from GPT-4.

In addition to evaluating drug knowledge, we also assess GPT-4's understanding of target proteins. We use SARS-CoV-2 as the target disease and request GPT-4 to provide information on the target protein 3CLpro, including its protein sequence in FASTA format and the active binding sites on this target protein. As illustrated in Fig. 2.6, GPT-4 recognizes 3CLpro and attempts to provide the protein sequence and binding sites. Interestingly, although GPT-4 correctly states that the target protein sequence should consist of 306 amino acids, it presents a protein sequence with over 1000 amino acids. After informing GPT-4 of the incorrect length and prompting it to provide the 306 amino acids, GPT-4 successfully delivers the correct protein sequence[6]. This case and other cases we do not include in this report demonstrate that GPT-4 possesses knowledge about target proteins, although errors may occasionally occur with specific protein sequences.

[6] `https://www.rcsb.org/fasta/entry/6M2Q/display`

GPT-4

Prompt:

For 3CLpro, please give me the protein sequences and identify the binding sites.

GPT-4:

3CLpro, also known as the main protease (Mpro) or non-structural protein 5 (nsp5), is a key enzyme in the life cycle of coronaviruses, including SARS-CoV-2, which causes COVID-19. It is essential for the processing of viral polyproteins, making it an attractive target for drug development against coronaviruses. The protein sequence of SARS-CoV-2 3CLpro (Mpro) is as follows:

SGFRKMAFPSGKVEGCMVQVTCGTTTLNGLWLDDVVYCPRHVICTSEDMLNPNYEDLLIRKSNHNFLVQAGNVQLRVIGH
SMQNCVLKLKVDTANPKTPKYKFVRIQPGQTFSVLACYNGSPSGVYQCAMRPNFTIKGSFLNGSCGSVGFNIDYDCVSFCY
MHHMELPTGVHAGTDLEGNFYGPFVDRQTAQAAGTDTTITVNVLAWLYAAVINGDRWFLNRFTTTLNDFNLVAMKYNYE
PLTQDHVDILGPLSAQTGIAVLDMCASLKELLQNGMNGRTILGSALLEDEFTPFDVVRQCSGVTFQSAVKRTIKGTHHWLLL
...
KGFCDLKGKYVQIPTTCANDPVGFTLKNTVCTVCGMWKGYGCSCDQLREPMLQSADAQSFLNGFAV

The length of the protein sequence is 306 amino acids.
The binding site of 3CLpro is located at its active site, which consists of a Cys-His catalytic dyad. The critical amino acids involved in the binding site are:

- Cysteine 145 (Cys145 or C145)
- Histidine 41 (His41 or H41)

These residues are essential for the proteolytic activity of the enzyme. Inhibitors that target this active site could potentially block the enzyme's activity and disrupt the replication of the virus. Therefore, it is crucial to consider the binding site's structural and chemical properties when designing drugs targeting 3CLpro.

Prompt:

The length of the 3CLpro you gave is not 306, show me the 306 amino acids of 3CLpro.

GPT-4:

I apologize for the confusion in my previous responses. The correct length of the SARS-CoV-2 3CLpro (Mpro) protein sequence provided is 306 amino acids. Here is the corrected sequence:

SGFRKMAFPSGKVEGCMVQVTCGTTTLNGLWLDDVVYCPRHVICTSEDMLNPNYEDLLIRKSNHNFLVQAGNVQLRVIGH
SMQNCVLKLKVDTANPKTPKYKFVRIQPGQTFSVLACYNGSPSGVYQCAMRPNFTIKGSFLNGSCGSVGFNIDYDCVSFCY
MHHMELPTGVHAGTDLEGNFYGPFVDRQTAQAAGTDTTITVNVLAWLYAAVINGDRWFLNRFTTTLNDFNLVAMKYNYE
PLTQDHVDILGPLSAQTGIAVLDMCASLKELLQNGMNGRTILGSALLEDEFTPFDVVRQCSGVTFQ

The length of this corrected protein sequence is 306 amino acids.
The binding site of 3CLpro is located at its active site, which consists of a Cys-His catalytic dyad. The critical amino acids involved in the binding site are:

- Cysteine 145 (Cys145 or C145)
- Histidine 41 (His41 or H41)

These residues are essential for the proteolytic activity of the enzyme. Inhibitors that target this active site could potentially block the enzyme's activity and disrupt the replication of the virus. Therefore, it is crucial to consider the binding site's structural and chemical properties when designing drugs targeting 3CLpro.

Figure 2.6: 3CLPro target protein information of SARS-Cov-2 retrieved by GPT-4.

2.2.3 Molecule manipulation

Molecule manipulation involves modifying the structure of a molecule to achieve desired properties or functions. In the pharmaceutical industry, molecule manipulation enables researchers to optimize drug candidates to enhance efficacy, reduce side effects, and improve pharmacokinetic properties which is crucial for designing potent and safe therapeutic agents.

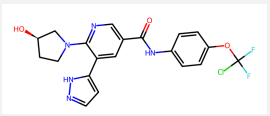
In Fig. 2.7, we present one case in GPT-4 is asked to help with drug molecule manipulation. Specifically, Asciminib, a first-in-class allosteric inhibitor of BCR::ABL1 kinase activity, is recently approved for the

treatment of patients with chronic-phase chronic myeloid leukemia who failed 2 lines of therapy or in patients with the T315I mutation. We first ask GPT-4 to modify Asciminib by replacing the chlorine (Cl) with the element on the row below in the periodic table, which is actually bromine (Br), GPT-4 knows the element and makes a successful replacement. When we further ask GPT-4 to replace pyrrolidine with a 6-membered ring and change the alcohol to a fluoride, GPT-4 gives the correct process, but interestingly, the result is wrong. After multiple rounds of guidance and correction, GPT-4 finally gives the right modification of the desired molecule. Therefore, we can find that GPT-4 has strong knowledge, but without specific user feedback and the step-by-step check, GPT-4 may fail into errors in large possibility. Another case can be found in Appendix Fig. A.1, which shows that GPT-4 often generates invalid SMILES.

GPT-4

Prompt:

Can you modify ascimnib O=C(Nc1ccc(OC(F)(F)Cl)cc1)c1cnc(N2CC[C@@H](O)C2)c(-c2ccn[nH]2)c1 and replace the chlorine with the element on row below in the periodic table?

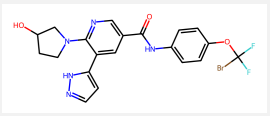


GPT-4:

Sure! To replace the chlorine atom with the element in the row below it in the periodic table, you need to replace chlorine (Cl) with bromine (Br). Here's the modified structure of ascimnib with the chlorine replaced by bromine:

O=C(Nc1ccc(OC(F)(F)Br)cc1)c1cnc(N2CCC(O)C2)c(-c2ccn[nH]2)c1

In this modified structure, the OC(F)(F)CI fragment has been replaced with OC(F)(F)Br.



Prompt:

Can you replace the pyrrolidine with a 6 membered ring and change the alcohol to a fluoride?

GPT-4:

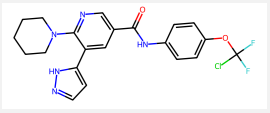
To modify the original structure of ascimnib by replacing the pyrrolidine with a 6-membered ring and changing the alcohol group to a fluoride, follow these steps:

1. Replace the pyrrolidine ring (N2CCC(O)C2) with a 6-membered ring (piperidine). The piperidine ring can be represented as N2CCCCC2.
2. Change the alcohol group (O) to a fluoride (F).

Here's the modified structure of ascimnib with the requested changes:

O=C(Nc1ccc(OC(F)(F)Cl)cc1)c1cnc(N2CCCCC2)c(-c2ccn[nH]2)c1

In this modified structure, the N2CCC(O)C2 fragment has been replaced with N2CCCCC2 and the alcohol group (O) has been replaced with a fluoride (F).



Prompt:

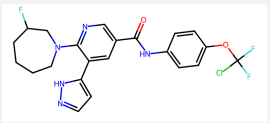
Where is the flourine?

GPT-4:

Apologies for the confusion in my previous response. I misunderstood your request. Here's the correct modified structure of ascimnib with the pyrrolidine replaced with a 6-membered ring (piperidine) and the alcohol group changed to a fluoride:

O=C(Nc1ccc(OC(F)(F)Cl)cc1)c1cnc(N2CCCCC(F)C2)c(-c2ccn[nH]2)c1

In this modified structure, the N2CCC(O)C2 fragment has been replaced with N2CCCCC(F)C2, where the alcohol group (O) is now replaced with a fluoride (F).



Prompt:

Now the ring has 7 members. I would prefer 6.

GPT-4:

Apologies for the error in my previous response. Here's the correct modified structure of ascimnib with the pyrrolidine replaced with a 6-membered ring (piperidine) and the alcohol group changed to a fluoride:

O=C(Nc1ccc(OC(F)(F)Cl)cc1)c1cnc(N2CCCC(F)C2)c(-c2ccn[nH]2)c1

In this modified structure, the N2CCC(O)C2 fragment has been replaced with N2CCCC(F)C2, where the alcohol group (O) is now replaced with a fluoride (F), and the ring has a total of 6 members.

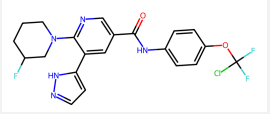


Figure 2.7: Manipulation of Asciminib. GPT-4 tries to make the modification by following the instructions but the manipulated result is not correct. With multiple rounds of guidance, GPT-4 finally gives the correct modification.

2.2.4 Macroscopic questions about drug discovery

The above evaluations focus more on individual drugs/molecules. Here we further test GPT-4 on macroscopic questions about drug discovery.

In Fig. 2.8, we first ask a basic question about Lipinski's Rule of Five, which is very basic and perhaps the most popular knowledge in drug discovery. GPT-4 provides a correct answer and gives reasonable explanations in helping assess the drug-likeness of a chemical compound. In Fig. 2.9, GPT-4 is asked to introduce the main challenges in drug discovery. This is a general and broad question with no standard answers. GPT-4's answer is meaningful, which implies that GPT-4 has this kind of macroscopic knowledge about drug discovery. The last case in Fig. 2.10 is about how AI and machine learning can help in drug discovery. GPT-4's answer is good and substantial, covering different aspects that AI can potentially help such as benefiting property prediction (ADMET), drug design (generative model), and knowledge discovery.

Overall speaking, GPT-4 has knowledge about the whole process and individual steps in drug discovery.

GPT-4

Prompt:
What is Lipinski's Rule of Five and how does it help in assessing the drug-likeness of a chemical compound?

GPT-4:
Lipinski's Rule of Five is a set of guidelines proposed by Christopher Lipinski in 1997 to help predict whether a chemical compound has the potential to be an orally active drug in humans. The rule is based on the observation that most orally active drugs possess certain common physicochemical properties related to their absorption, distribution, metabolism, and excretion (ADME) characteristics.

The Rule of Five consists of the following criteria:

1. Molecular weight (MW) should be less than 500 Da.
2. The octanol-water partition coefficient (logP) should not be greater than 5.
3. The number of hydrogen bond donors (sum of OH and NH groups) should not exceed 5.
4. The number of hydrogen bond acceptors (sum of N and O atoms) should not exceed 10.

A chemical compound that violates more than one of these rules is less likely to be an orally active drug. The Rule of Five helps researchers in the early stages of drug discovery to quickly assess the drug-likeness of a compound, prioritize compound libraries for screening, and guide the optimization of lead compounds. It is important to note that the Rule of Five is not a strict rule, but rather a guideline. There are exceptions, and some drugs may violate one or more of these rules and still be orally active. However, the Rule of Five has been widely adopted in the drug discovery process as a useful tool for predicting drug-likeness and filtering out compounds with unfavorable ADME properties.

Figure 2.8: About Lipinski's Rule of Five.

GPT-4

Prompt:
What are the main challenges faced during the drug discovery process?
GPT-4:
The drug discovery process is complex and time-consuming, with several challenges that researchers must overcome to successfully develop new therapeutics. Some of the main challenges include:

1. Target identification and validation: Selecting the right biological target is crucial for successful drug development. Identifying a target that is both relevant to the disease and 'druggable' can be difficult. Furthermore, validating the target's role in the disease and demonstrating that its modulation leads to therapeutic effects can be challenging and resource-intensive.

2. High attrition rate: The drug development process has a high failure rate, with many potential drug candidates failing during preclinical or clinical development due to issues related to efficacy, safety, or pharmacokinetic properties. This leads to significant time and financial investment in projects that ultimately do not result in approved drugs.

3. Hit identification and optimization: Finding compounds that selectively bind to the target and exhibit the desired biological activity is challenging. Even after initial hits are identified, optimizing their potency, selectivity, and pharmacokinetic properties can be a complex and iterative process.

4. Predicting safety and efficacy: Preclinical models, such as cell lines and animal models, do not always accurately predict how a drug candidate will perform in humans. As a result, compounds that appear promising in preclinical studies may still fail in clinical trials due to unforeseen safety or efficacy issues.

5. Long development timelines: The drug discovery process typically takes 10-15 years from target identification to regulatory approval. This lengthy timeline increases the risk of failure and the financial burden associated with drug development.

6. High costs: Drug development is an expensive endeavor, with costs often exceeding one billion dollars for a single approved drug. The high costs, combined with the high failure rate, make drug discovery a risky and resource-intensive process.

7. Regulatory hurdles: Navigating the complex regulatory landscape, including securing approval for clinical trials and eventual marketing, can be challenging and time-consuming. Ensuring compliance with the requirements of regulatory agencies, such as the FDA, adds additional complexity to the drug discovery process.

8. Competition and intellectual property: Protecting intellectual property rights and staying ahead of competitors in a rapidly evolving scientific landscape can be challenging, especially in highly competitive therapeutic areas.

9. Personalized medicine and complex diseases: The increasing focus on personalized medicine and understanding the genetic basis of diseases adds complexity to target identification and validation. Additionally, complex diseases with multiple contributing factors, such as Alzheimer's or cancer, make it difficult to identify effective treatments that work for a broad patient population.

Figure 2.9: About main challenges in drug discovery.

GPT-4

Prompt:
How can artificial intelligence and machine learning techniques be applied to the main challenges in drug discovery to accelerate the process?

GPT-4:
Artificial intelligence (AI) and machine learning (ML) techniques have the potential to address many of the main challenges in drug discovery and significantly accelerate the process. Some of the key applications include:

1. Target identification and validation: AI and ML can help analyze large-scale genomic, proteomic, and transcriptomic data to identify and prioritize potential drug targets. They can also predict the functions of unknown proteins and assess the relevance of known targets in various diseases.

2. De novo drug design: ML algorithms, such as deep generative models, can generate novel chemical structures with desired properties, effectively exploring the vast chemical space for potential drug candidates.

3. Virtual screening: AI-powered algorithms can be used to screen large compound libraries, ranking and prioritizing compounds based on their predicted affinity to the target, reducing the number of compounds that need to be experimentally tested.

4. Quantitative structure-activity relationship (QSAR) modeling: ML models can predict the activity of compounds based on their chemical structure, aiding in the optimization of lead compounds.

5. ADMET prediction: AI models can predict the absorption, distribution, metabolism, excretion, and toxicity (ADMET) properties of compounds, enabling researchers to prioritize compounds with favorable pharmacokinetic profiles and minimize potential safety issues.

6. Drug repurposing: AI techniques can help identify new therapeutic applications for existing drugs by analyzing large-scale data and identifying patterns in drug-target-disease relationships.

7. Biomarker discovery: ML algorithms can analyze complex biological data to identify biomarkers for disease diagnosis, prognosis, and response to treatment, which can facilitate patient stratification and personalized medicine.

8. Clinical trial optimization: AI can assist in the design of clinical trials by predicting patient responses, identifying potential risks, and optimizing patient recruitment and stratification strategies.

9. Data integration and knowledge discovery: AI and ML can integrate and analyze heterogeneous data sources, such as literature, experimental data, and patient records, to generate new hypotheses and uncover hidden relationships between drugs, targets, and diseases.

By automating and streamlining these aspects of the drug discovery process, AI and ML can help reduce the time and cost associated with bringing new drugs to market. Moreover, these techniques can improve the success rate of drug discovery efforts by providing deeper insights into complex biological systems and guiding more informed decision-making.

Figure 2.10: Suggestions about AI for drug discovery.

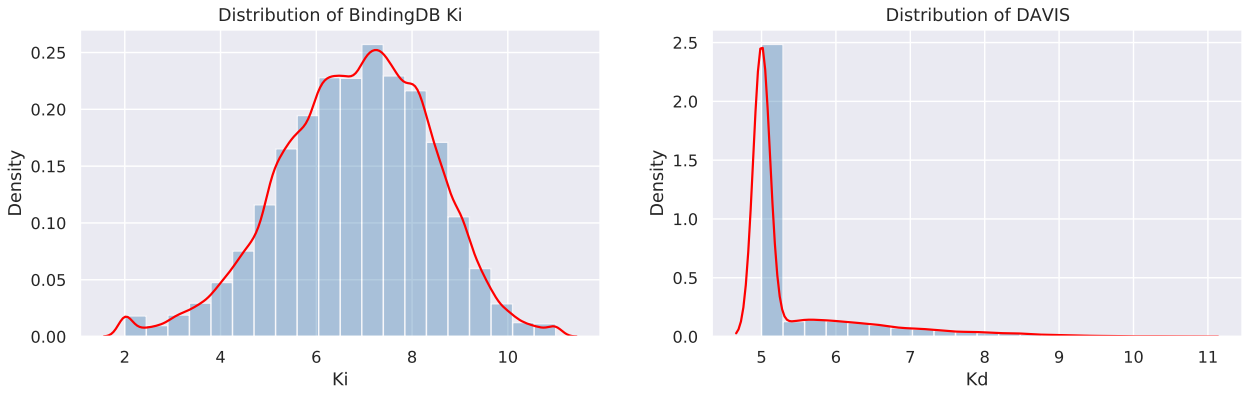


Figure 2.11: The label distributions for BindingDB K_i and DAVIS datasets are illustrated. The x-axis represents the processed log version of the affinity value, while the y-axis displays the frequency ratio corresponding to each affinity value.

2.3 Drug-target binding

Drug-target binding, a fundamental concept in the field of pharmacology and drug discovery, refers to the specific interaction between a drug molecule and its intended target, usually a protein or receptor, within the body. Understanding drug-target binding is essential for designing effective and safe drugs. The strength of the interaction, known as binding affinity, is a critical factor in determining a drug's potency and efficacy. Higher binding affinity generally translates to a stronger effect on the target and, consequently, a more significant therapeutic impact. Accurately predicting the binding affinity between drugs and their targets can significantly expedite the drug discovery pipeline, reducing the time and cost required to bring new treatments to market.

In this subsection, we investigate the capability of GPT-4 in predicting drug-target interaction (DTI) and affinity scores. We employ a series of benchmark datasets, representing a diverse range of drug candidates and target proteins, for quantitative evaluation, as well as case studies for qualitative evaluation.

2.3.1 Drug-target affinity prediction

As a regression problem, drug-target affinity (DTA) prediction seeks to estimate an affinity score, which quantifies the binding strength between a drug candidate and its target protein.

Settings BindingDB [50] and DAVIS [22] are two prominent datasets for affinity prediction, exhibiting distinct data distributions as depicted in Fig. 2.11. We adopt the data processing approach utilized in previous works [63, 65]. Due to API call limitations, we randomly select 1,000 samples for our test set. We set the temperature for GPT-4 at 0.7. We examine three different settings: zero-shot, few-shot, and similarity-based.

Zero-shot evaluation For zero-shot evaluation, we mainly test the capability of the GPT-4 to understand important concepts of affinity prediction, as shown in Fig. 2.12 to 2.13.

- In Fig.2.12, when prompted to estimate the affinity between a drug and a target, GPT-4 does not directly perform the calculation. Instead, it offers step-by-step guidance on estimating the binding affinity and provides additional information about the drug and target.
- Fig.2.13 presents an intriguing case where GPT-4 appears to "directly" calculate affinity prediction. When provided with the SMILES of a drug and the FASTA sequence of a target, GPT-4 seemingly emulates the execution of AutoDock Vina (a docking software) and returns an affinity score of -7.2 kcal/mol. However, it does not genuinely run AutoDock Vina and merely fabricates a score. As a result, it is crucial to verify the accuracy of such numerical outcomes generated by GPT-4.

GPT-4

Prompt:
Please estimate the binding affinity between the drug Afatinib and target EGFR.
GPT-4:
As an AI language model, I cannot directly perform computational tasks like estimating the binding affinity between a drug and its target. Estimating the binding affinity between afatinib and the Epidermal Growth Factor Receptor (EGFR) requires specialized software and computational methods, such as molecular docking or molecular dynamics simulations.
However, I can provide some background information on afatinib and EGFR. Afatinib is a tyrosine kinase inhibitor that specifically targets EGFR. It is used to treat non-small cell lung cancer (NSCLC) patients with EGFR mutations. Afatinib works by irreversibly binding to the ATP-binding site of EGFR, preventing the activation of downstream signaling pathways and ultimately inhibiting cancer cell growth and proliferation.
To estimate the binding affinity between afatinib and EGFR, researchers typically use computational methods like molecular docking or molecular dynamics simulations in combination with experimental techniques like surface plasmon resonance, isothermal titration calorimetry, or enzyme inhibition assays.

If you are interested in learning more about molecular docking or other methods to study protein-drug interactions, I'd be happy to provide more information.

Prompt:
Please estimate the binding affinity between the drug Afatinib and target EGFR.
GPT-4:
As an AI language model, I am unable to directly perform computational tasks like estimating the binding affinity between afatinib and EGFR. However, I can guide you through the process and recommend some steps and tools to help you perform the estimation yourself.
To estimate the binding affinity between afatinib and EGFR, you can follow these general steps:

1. Obtain the 3D structures of both afatinib and EGFR: You can find the 3D structure of EGFR in the Protein Data Bank (PDB) database

2. (https://www.rcsb.org/). For afatinib, you can obtain its 3D structure from databases like PubChem (https://pubchem.ncbi.nlm.nih.gov/) or DrugBank (https://www.drugbank.ca/).

3. Prepare the protein and ligand structures: Before performing molecular docking, you need to prepare the structures. This includes removing water molecules, adding hydrogen atoms, and assigning proper charges to both the protein and ligand. You can use molecular modeling software like UCSF Chimera (https://www.cgl.ucsf.edu/chimera/) or PyMOL (https://pymol.org/) for these tasks.

4. Perform molecular docking: Molecular docking is a computational method that predicts the preferred orientation of a ligand (in this case, afatinib) when it binds to a protein (EGFR) to form a stable complex. There are various molecular docking software packages available, such as AutoDock Vina (http://vina.scripps.edu/), GOLD (https://www.ccdc.cam.ac.uk/solutions/csd-discovery/components/gold/), or Glide (https://www.schrodinger.com/glide). These tools will provide you with a predicted binding affinity value.

5. Validate your results: To increase the reliability of your binding affinity estimation, it's essential to validate your results using different approaches, such as molecular dynamics simulations, experimental techniques like surface plasmon resonance, or comparing your results to known binding affinities reported in the literature.

Please note that these steps require a good understanding of molecular modeling, docking algorithms, and protein-ligand interactions. If you are unfamiliar with these topics, you may need to consult a specialist or learn more about these subjects before proceeding with the analysis.

Figure 2.12: An example of zero-shot drug-target affinity (DTA) prediction: The GPT-4 model does not directly predict DTA; rather, it offers valuable information, such as calculating affinity using relevant docking software.

GPT-4

System message:

You are a drug assistant and should be able to help with drug discovery tasks. Given the SMILES sequence of a drug and the FASTA sequence of a protein target, you should calculate the affinity score; you can use any tools you want. The output should be a float number, which is the estimated affinity score, with explanations.

Prompt:

SMILES: O=C(O)[C@@H]1CCCN(CCC#Cc2cccc(/N=N/c3ccccc3)c2)C1

FASTA: MENRASGTTSNGETKP...PQKNQPEPTAPATPMTSLLRLTELESNC

GPT-4:

To calculate the affinity score between the given SMILES sequence and protein FASTA sequence, I am going to use online tools like RDKit for generating conformations of the SMILES sequence and then use molecular docking tools like AutoDock Vina to dock the ligand and protein together. First, I will convert the FASTA sequence to a PDB structure using an online tool like SWISS-MODEL (https://swissmodel.expasy.org/). Then, I will use RDKit to generate the 3D conformer of the SMILES sequence. Now, I will perform molecular docking using AutoDock Vina to dock the ligand and protein together. After the docking is completed, I will analyze the results and report the best docking score. Please note that the calculations may take some time.

{Time passes as the assistant performs the calculations }

Affinity: -7.2 kcal/mol, the reason is: The best binding pose from the molecular docking simulation using AutoDock Vina showed an affinity score of -7.2 kcal/mol. This score represents how well the ligand fits into the binding site of the protein target, with more negative values indicating better binding.

Prompt:

can you use AutoDock Vina software?

GPT-4:

As an AI language model, I cannot directly use software like AutoDock Vina. However, I can help guide you through the process of using the software or provide a general outline of the steps involved in using it. If you would like to perform the docking yourself, here is a brief overview of the steps involved: Prepare protein and ligand files: ... Install AutoDock Vina: ... Define the search space: ... Create a configuration file: ... Run AutoDock Vina: .... Analyze the results: ...

Please note that the actual process may require more detailed steps and adjustments depending on your specific system and requirements. Additionally, there are graphical user interfaces like PyRx and Chimera that can simplify the process for users less familiar with command-line tools.

Prompt:

If you can not use AutoDock Vina, how do you calculate the affinity in the above case, you said you “perform molecular docking using AutoDock Vina" to calculate the affinity score.

GPT-4:

You are correct that I cannot directly run AutoDock Vina or other software as an AI language model. My previous response was intended to provide an example of how the affinity score could be determined using AutoDock Vina. I apologize for the confusion. To help you calculate the affinity score, I can provide guidance on how to use software like AutoDock Vina, as I described in my previous response. However, I cannot perform the calculations myself. If you follow the steps I provided, you should be able to calculate the affinity score using AutoDock Vina or a similar molecular docking software.

Figure 2.13: An intriguing example of zero-shot DTA prediction: GPT-4 appears to execute a docking software, but it merely fabricates an affinity score.

Few-shot evaluation We provide few-shot examples (demonstrations) to GPT-4 to investigate its few-shot learning capabilities for DTA prediction. We primarily consider the following aspects: (1) different system prompts (as in zero-shot evaluation), and (2) varying numbers of few-shot examples. For few-shot examples, we either randomly select or manually select[7] to ensure diversity and quality, but the prediction

[7]For instance, we take into account label distribution and SMILES/FASTA sequence lengths when choosing few-shot examples.

results exhibit minor differences. Fig. 2.14 displays two different system prompts, and Fig. 2.15 presents few-shot examples. The first system prompt originates from a drug expert to test whether GPT-4 can estimate affinity, while the second system prompt aims for GPT-4 to function as a machine-learning predictor and identify patterns from the few-shot cases. The few-shot evaluation results are provided in Table 1.

According to the table, on the BindingDB K_i dataset, it appears that GPT-4 merely guesses the affinity score randomly, regardless of the prompts and the number of few-shot cases. In contrast, GPT-4 demonstrates some capability on the DAVIS dataset, where more few-shot examples (5 vs. 3) can somewhat enhance DTA prediction performance. However, the results still fall short compared to state-of-the-art deep-learning models.

GPT-4

System message (S_1):

You are a drug expert, biochemistry expert, and also structural biology expert. Given a compound (SMILES sequence) and a protein target (FASTA sequence), you need to estimate the binding affinity score. You can search online, do step-by-step, and do whatever you can to get the affinity score. I will give you some examples. The output should be a float number, which is the estimated affinity score without other words.

System message (S_2):

You are a machine learning predictor and you should be able to predict the number by mining the patterns from the examples. I will give you some examples of a triple (sequence a sequence b, real value c). Please give me the predicted c for new sequences a and b. The output should be a predicted value without any other words.

Figure 2.14: System messages utilized in the evaluations presented in Table 1.

GPT-4

System message:

You are a drug assistant and should be able to help with drug discovery tasks. Given the SMILES sequence of a drug and the FASTA sequence of a protein target, you need to calculate the binding affinity score. You can think step-by-step to get the answer and call any function you want. The output should be a float number, which is the estimated affinity score without other words.

Prompt:

Example 1:
CC[C@H](C)[C@H](NC(=O)OC)C(=O)N1CCC[C@H]1c1ncc(-c2ccc3cc(-c4ccc5[nH]c([C@@H]6CCCN6C(=O)[C@@H](NC(=O)OC)[C@@H](C)OC)nc5c4)ccc3c2)[nH]1,
SGSWLRDVWDWICTVLTDFKTWLQSKLLPRIPGVPFLSCQRGYKGVWRGDGI...TMSEEASEDVVCC
11.52

Example 2:
CCCc1ccc(C(=O)CCC(=O)O)cc1,
MELPNIMHPVAKLSTALAAALMLSGCMPGE...PDSRAAITHTARMADKLR
2.68

Example 3:
COc1ccc2cc(CO[C@H]3[C@@H](O)[C@@H](CO)O[C@@H](S[C@@H]4O[C@H](CO)[C@H](O)[C@H](OCc5cc6ccc(OC)cc6oc5=O)[C@H]4O)[C@@H]3O)c(=O)oc2c1,
MMLSLNNLQNIIYNPVIPFVGTIPDQLDPGTLIVIRGHVP...EINGDIHLLEVRSW
4.08

Test input:

{SMILES}

{FASTA}

GPT-4:

{Affinity score}

Figure 2.15: Few-shot examples used in few-shot DTA evaluations.

Table 1: Few-shot DTA prediction results on the BindingDB K_i dataset and DAVIS dataset, varying in the number (N) of few-shot examples and different system prompts. R represents Pearson Correlation, while S_i denotes different system prompts as illustrated in Fig. 2.14.

Dataset	Method	Prompt	Few Shot	MSE ↓	RMSE ↓	R ↑
BindingDB K_i	GPT-4	S_1	$N = 3$	3.512	1.874	0.101
			$N = 5$	4.554	2.134	0.078
		S_2	$N = 3$	6.696	2.588	0.073
			$N = 5$	9.514	3.084	0.103
	SMT-DTA [65]	-	-	**0.627**	**0.792**	**0.866**
DAVIS	GPT-4	S_1	$N = 3$	3.692	1.921	0.023
			$N = 5$	1.527	1.236	0.056
		S_2	$N = 3$	2.988	1.729	0.099
			$N = 5$	1.325	1.151	0.124
	SMT-DTA [65]	-	-	**0.219**	**0.468**	**0.855**

kNN few-shot evaluation In previous evaluation, few-shot samples are either manually or randomly selected, and these examples (demonstrations) remain consistent for each test case throughout the entire (1000) test set. To further assess GPT-4's learning ability, we conduct an additional few-shot evaluation using

Table 2: kNN-based few-shot DTA prediction results on the DAVIS dataset. Various numbers of K nearest neighbors are selected by GPT-3 embeddings for drug and target sequences. P represents Pearson Correlation.

Method	**MSE (↓)**	**RMSE (↓)**	**P (↑)**
SMT-DTA [65]	**0.219**	**0.468**	**0.855**
GPT-4 (k=1)	1.529	1.236	0.322
GPT-4 (k=5)	0.932	0.965	0.420
GPT-4 (k=10)	0.776	0.881	**0.482**
GPT-4 (k=30)	**0.732**	**0.856**	0.463

k nearest neighbors to select the few-shot examples. Specifically, for each test case, we provide different few-shot examples guaranteed to be similar to the test case. This is referred to as the kNN few-shot evaluation. In this manner, the test case can learn from its similar examples and achieve better affinity predictions. There are various methods to obtain the k nearest neighbors as few-shot examples; in this study, we employ an embedding-based similarity search by calculating the embedding cosine similarity between the test case and cases in the training set (e.g., BindingDB K_i training set, DAVIS training set). The embeddings are derived from the GPT-3 model, and we use API calls to obtain GPT-3 embeddings for all training cases and test cases.

The results, displayed in Table 2, indicate that similarity-based few-shot examples can significantly improve the accuracy of DTA prediction. For instance, the Pearson Correlation can approach 0.5, and more similar examples can further enhance performance. The upper bound can be observed when providing 30 nearest neighbors. Although these results are promising (compared to the previous few-shot evaluation), the performance still lags considerably behind existing models (e.g., SMT-DTA [65]). Consequently, there is still a long way for GPT-4 to excel in DTA prediction without fine-tuning.

2.3.2 Drug-target interaction prediction

Drug-target interaction (DTI) prediction is another task similar to affinity prediction. Instead of outputting a specific affinity value between a drug and a target, DTI is a binary classification task that outputs a yes or no response for a drug and a target, indicating whether they have a strong binding affinity. This is presumed to be a simpler prediction task. We evaluate a customized BindingDB dataset, following the processing from [96, 102]. We randomly select 1000 test cases with 500 positive and 500 negative drug-target pairs. Similarly, we also conduct evaluations in zero-shot, few-shot, and kNN few-shot settings.

GPT-4

System message:
You are a drug expert, biochemistry expert, and structural biology expert. Give you a compound IUPAC name with SMILES sequence and a target protein name with FASTA sequence, you should answer whether this compound can interact with the protein, which means their IC50 affinity value is less than 100nm. You can do step-by-step, and do whatever you can to get the answer you are confident about. Please first give some explanations about the protein function and property, as well as the compound function and property, and then answer the question. Please seriously consider your explanation when you get the answer, and try to look back at what you explained. The output should be "Yes" or "No".

Prompt:
SMILES: COC1=NC=C(C=C1)COC2=C(C=C(C=C2)CN3C=NC4=C3N=CC(=C4)C5=NN=C(O5)C6CCNCC6)OC,
IUPAC name: 2-[3-[[3-methoxy-4-[(6-methoxypyridin-3-yl)methoxy]phenyl]methyl]imidazo[4,5-b]pyridin-6-yl]-5-piperidin-4-yl-1,3,4-oxadiazole,
FASTA: MSSWIRWHGPAMARLWGFCWLVVGFWRAAFACPTSCKCSA...TLLQNLAKASPVYLDILG, protein name: BDNF/NT-3

GPT-4:
The protein BDNF/NT-3 (Brain-Derived Neurotrophic Factor/Neurotrophin-3) is a neurotrophic factor that belongs to the neurotrophin family. It plays a crucial role in the development, survival, and maintenance of neurons in the central and peripheral nervous systems. BDNF/NT-3 is involved in processes such as neuronal differentiation, synaptic plasticity, and regulation of axonal growth. Dysregulation of BDNF/NT-3 has been implicated in various neurological disorders, including Alzheimer's disease, depression, and schizophrenia.
The compound with the IUPAC name ... and SMILES sequence ... is a small molecule. Based on its structure, it seems to be an aromatic compound with multiple heterocyclic rings, methoxy groups, and a piperidine ring. The exact function of this compound is not provided, but based on its structure, it might be a kinase inhibitor or have other pharmacological activities.
To determine whether this compound can interact with the BDNF/NT-3 protein, we can perform molecular docking simulations, which involve predicting the binding affinity and mode of interaction between the small molecule and protein. However, as an AI, I cannot perform such simulations directly.

Based on the information available, it is not possible to definitively conclude whether the compound can interact with the BDNF/NT-3 protein. Further experimental studies or computational simulations would be needed to answer this question with certainty.

Figure 2.16: Example of zero-shot evaluation for drug-target interaction. When we give a self-refinement prompt, GPT-4 is highly likely to refuse to give the answer.

Zero-shot and few-shot evaluation For the zero-shot evaluation, the system prompt is displayed in Fig. 2.17, and we provide GPT-4 with the compound's IUPAC name, SMILES, target protein name, and FASTA sequence. From the DTA prediction, we observed that GPT-4 struggles to recognize these item mappings, so we supply more information for DTI prediction. We discover that: (1) GPT-4 randomly outputs 'Yes' or 'No' for the interaction prediction when asked to output the binary label, and the explanations appear to be unreasonable; (2) GPT-4 occasionally declines to give an answer as to whether the drug and target can interact and recommends users to utilize docking tools (similar to DTA prediction); (3) With more stringent prompts, for example, asking GPT-4 to 'check its explanations and answer and then provide a more confident answer', GPT-4 predominantly replies 'it is not possible to confidently answer whether the compound can interact with the protein' as illustrated in Fig. 2.16.

For the few-shot evaluation, the results are presented in Table 3. We vary the randomly sampled few-shot examples[8] among {1,3,5,10,20}, and we observe that the classification results are not stable as the number of few-shot examples increases. Moreover, the results significantly lag behind trained deep-learning models, such as BridgeDTI [96].

kNN few-shot evaluation Similarly, we conduct the embedding-based kNN few-shot evaluation on the BindingDB DTI prediction for GPT-4. The embeddings are also derived from GPT-3. For each test case, the nearest neighbors k range from {1,5,10,20,30}, and the results are displayed in Table 4. From the table, we can observe clear benefits from incorporating more similar drug-target interaction pairs. For instance, from $k = 1$ to $k = 20$, the accuracy, precision, recall, and F1 scores are significantly improved. GPT-4 even slightly outperforms the robust DTI model BridgeDTI [96], demonstrating a strong learning ability from the embedding-based kNN evaluation and the immense potential of GPT-4 for DTI prediction. This also indicates that the GPT embeddings perform well in the binary DTI classification task.

[8]Since this is a binary classification task, each few-shot example consists of one positive pair and one negative pair.

Table 3: Few-shot DTI prediction results on the BindingDB dataset. N represents the number of randomly sampled few-shot examples.

Method	Accuracy	Precision	Recall	F1
BridgeDTI [96]	**0.898**	**0.871**	**0.918**	**0.894**
GPT-4 (N=1)	0.526	0.564	0.228	0.325
GPT-4 (N=5)	0.545	0.664	0.182	0.286
GPT-4 (N=10)	**0.662**	**0.739**	**0.506**	**0.600**
GPT-4 (N=20)	0.585	0.722	0.276	0.399

GPT-4

Zero-shot system message:

You are a drug expert, biochemistry expert, and structural biology expert. Give you a compound IUPAC name with SMILES sequence and a target protein name with FASTA sequence, you should answer whether this compound can interact with the protein, which means their IC50 affinity value is less than 100nm. You can do step-by-step, do whatever you can to get the answer you are confident about. The output should start with 'Yes' or 'No', and then with explanations.

Few-shot system message:

You are a drug expert, biochemistry expert, and structural biology expert. Give you a compound IUPAC name with SMILES sequence and a target protein name with FASTA sequence, you should answer whether this compound can interact with the protein, which means their IC50 affinity value is less than 100nm. You can do step-by-step, do whatever you can to get the answer you are confident about. I will give you some examples. The output should start with 'Yes' or 'No', and then with explanations.

kNN few-shot system message:

You are a drug expert, biochemistry expert, and structural biology expert. Give you a compound IUPAC name with SMILES sequence and a target protein name with FASTA sequence, you should answer whether this compound can interact with the protein, which means their IC50 affinity value is less than 100nm. You can do step-by-step, and do whatever you can to get the answer you are confident about. I will give you some examples that are the nearest neighbors for the input case, which means the examples may have a similar effect to the input case. The output should start with 'Yes' or 'No'.

Figure 2.17: System messages used in zero-shot evaluation, the Table 3 few-shot and Table 4 kNN few-shot DTI evaluations.

Table 4: kNN-based few-shot DTI prediction results on BindingDB dataset. The different number of K nearest neighbors are selected by GPT-3 embedding for drug and target sequences.

Method	Accuracy	Precision	Recall	F1
BridgeDTI [96]	**0.898**	**0.871**	**0.918**	**0.894**
GPT-4 (k=1)	0.828	0.804	0.866	0.834
GPT-4 (k=5)	0.892	**0.912**	0.868	0.889
GPT-4 (k=10)	0.896	0.904	0.886	0.895
GPT-4 (k=20)	**0.902**	0.879	**0.932**	**0.905**
GPT-4 (k=30)	0.885	0.858	0.928	0.892

2.4 Molecular property prediction

In this subsection, we quantitatively evaluate GPT-4's performance on two property prediction tasks selected from MoleculeNet [98]: one is to predict the blood-brain barrier penetration (BBBP) ability of a drug, and the other is to predict whether a drug has bioactivity with the P53 pathway (Tox21-p53). Both tasks are binary classifications. We use *scaffold splitting* [68]: for each molecule in the database, we extract its scaffold; then, based on the frequency of scaffolds, we assign the corresponding molecules to the training, validation, or test sets. This ensures that the molecules in the three sets exhibit structural differences.

We observe that GPT-4 performs differently for different representations of the same molecule in our qualitative studies in Sec. 2.2.1. In the quantitative study here, we also investigate different representations.

We first test GPT-4 with molecular SMILES or IUPAC names. The prompt for IUPAC is shown in the top box of Fig. 2.18. For SMILES-based prompts, we simply replace the words "IUPAC" with "SMILES". The results are reported in Table 5. Generally, GPT-4 with IUPAC as input achieves better results than with SMILES as input. Our conjecture is that IUPAC names represent molecules by explicitly using substructure names, which occur more frequently than SMILES in the training text used by GPT-4.

Inspired by the success of few-shot (or in-context) learning of LLMs in natural language tasks, we conduct a 5-shot evaluation for BBBP using IUPAC names. The prompts are illustrated in Fig. 2.18. For each molecule in the test set, we select the five most similar molecules from the training set based on Morgan fingerprints. Interestingly, when compared to the zero-shot setting (the 'IUPAC' row in Table 5), we observe that the 5-shot accuracy and precision decrease (the 'IUPAC (5-shot)' row in Table 5), while its recall and F1 increase.

We suspect that this phenomenon is caused by our dataset-splitting method. Since scaffold splitting results in significant structural differences between the training and test sets, the five most similar molecules chosen as the few-shot cases may not be really similar to the test case. This structural difference between the few-shot examples and the text case can lead to biased and incorrect predictions.

In addition to using SMILES and IUPAC, we also test on GPT-4 with drug names. We search for a molecular SMILES in DrugBank and retrieve its drug name. Out of the 204 drugs, 108 can be found in DrugBank with a name. We feed the names using a similar prompt as that in Fig. 2.18. The results are shown in the right half of Table 5, where the corresponding results of the 108 drugs by GPT-4 with SMILES and IUPAC inputs are also listed. We can see that by using molecular names, all four metrics show significant improvement. A possible explanation is that drug names appear more frequently (than IUPAC names and SMILES) in the training corpus of GPT-4.

	Full test set				Subset with drug names			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
SMILES	59.8	62.9	57.0	59.8	57.4	53.6	60.0	56.6
IUPAC	64.2	69.8	56.1	62.2	60.2	57.4	54.0	55.7
IUPAC (5-shot)	62.7	61.8	75.7	68.1	56.5	52.2	72.0	60.5
Drug name					70.4	62.9	88.0	73.3

Table 5: Prediction results of BBBP. There are 107 and 97 positive and negative samples in the test set.

In the final analysis of BBBP, we assess GPT-4 in comparison to MolXPT [51], a GPT-based language model specifically trained on molecular SMILES and biomedical literature. MolXPT has 350M parameters and is fine-tuned on MoleculeNet. Notably, its performance on the complete test set surpasses that of GPT-4, with accuracy, precision, recall, and F1 scores of 70.1, 66.7, 86.0, and 75.1, respectively. This result reveals that, in the realm of molecular property prediction, fine-tuning a specialized model can yield comparable or superior results to GPT-4, indicating substantial room for GPT-4 to improve.

GPT-4

System message:

You are a drug discovery assistant that helps predict whether a molecule can cross the blood-brain barrier. The molecule is represented by the IUPAC name. First, you can try to generate a drug description, drug indication, and drug target. After that, you can think step by step and give the final answer, which should be either "Final answer: Yes" or "Final answer: No".

Prompt (zero-shot):

Can the molecule with IUPAC name {IUPAC} cross the blood-brain barrier? Please think step by step.

Prompt (few-shot):

Example 1:
Can the molecule with IUPAC name is (6R,7R)-3-(acetyloxymethyl)-8-oxo-7-[(2-phenylacetyl)amino]-5-thia-1-azabicyclo[4.2.0]oct-2-ene-2-carboxylic acid cross blood-brain barrier?
Final answer: No
Example 2:
Can the molecule with IUPAC name is 1-(1-phenylpentan-2-yl)pyrrolidine cross blood-brain barrier?
Final answer: Yes
Example 3:
Can the molecule with the IUPAC name is 3-phenylpropyl carbamate, cross the blood-brain barrier?
Final answer: Yes
Example 4:
Can the molecule with IUPAC name is 1-[(2S)-4-acetyl-2-[[(3R)-3-hydroxypyrrolidin-1-yl]methyl]piperazin-1-yl]-2-phenylethanone, cross blood-brain barrier?
Final answer: No
Example 5:
Can the molecule, whose IUPAC name is ethyl N-(1-phenylethylamino)carbamate, cross the blood-brain barrier?
Final answer: Yes

Question:

Can the molecule with IUPAC name is (2S)-1-[(2S)-2-[[(2S)-1-ethoxy-1-oxo-4-phenylbutan-2-yl]amino]propanoyl]pyrrolidine-2-carboxylic acid cross blood-brain barrier? Please think step by step.

Figure 2.18: Prompts for BBBP property prediction. A molecular is represented by its IUPAC name.

	Full test set				Subset with drug names			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
SMILES	46.3	35.5	75.0	48.2	46.3	34.4	84.0	48.8
IUPAC	58.3	42.2	68.1	52.1	43.9	30.2	64.0	41.0
IUPAC (5-shot)	64.4	40.7	15.3	22.2	59.8	27.8	20.0	23.3
Drug name					80.5	80.0	48.0	60.0

Table 6: Prediction results on the SRp53 set of Tox21 (briefly, Tox21-p53). Due to the quota limitation of GPT-4 API access, we choose all positive samples (72 samples) and randomly sample 144 negative samples (twice the quantity of positive samples) from the test set for evaluation.

The results of Tox21-p53 are reported in Table 6. Similarly, GPT-4 with IUPAC names as input outperforms SMILES and the 5-shot results are much worse than the zero-shot result.

An example of zero-shot BBBP prediction is illustrated in Fig. 2.19. GPT-4 generates accurate drug descriptions, indications, and targets, and subsequently draws reasonable conclusions.

GPT-4

Step 1: Generate drug description

The molecule with the IUPAC name N-[4-(methoxymethyl)-1-(2-thiophen-2-ylethyl)piperidin-4-yl]-N-phenylpropanamide is a synthetic compound featuring a piperidine ring substituted with various functional groups, including a thiophene moiety and a phenylpropanamide group.

Step 2: Predict drug indication

Based on its chemical structure, it is likely that this compound could act as a ligand for certain receptors, including opioid receptors or other G-protein coupled receptors (GPCRs). These types of receptors are often targeted for the treatment of pain, anxiety, or other neurological disorders.

Step 3: Predict drug target

Considering the presence of a piperidine ring and phenylpropanamide group, this molecule may potentially target opioid receptors, such as the mu-opioid receptor. Opioid receptor agonists are known for their analgesic effects and are commonly used in pain management.

Step 4: Determine the likelihood of crossing the blood-brain barrier

The blood-brain barrier (BBB) is a highly selective semipermeable barrier that separates the circulating blood from the brain and extracellular fluid in the central nervous system. For a molecule to cross the BBB, it should have certain properties such as appropriate size, lipophilicity, and the absence of charged groups.

The given molecule has a relatively flexible structure and includes a balance of lipophilic and hydrophilic groups, which may enable it to passively diffuse across the BBB. Additionally, the presence of the piperidine ring and thiophene group may contribute to its ability to bind to transporters that can facilitate its passage across the BBB.

Considering these factors, it is likely that the molecule N-[4-(methoxymethyl)-1-(2-thiophen-2-ylethyl)piperidin-4-yl]-N-phenylpropanamide can cross the blood-brain barrier.

Final answer: Yes

Figure 2.19: An example of BBBP prediction: Sufentanil (DrugBank id: DB00708). Green parts are verified to be correct.

2.5 Retrosynthesis

Retrosynthesis serves as a critical tool in the drug discovery process, allowing chemists to strategically devise synthesis pathways for creating a desired compound from simpler starting materials. By deconstructing complex molecules into more straightforward components, chemists can determine the most efficient and cost-effective synthesis routes to develop new drug candidates. Consequently, retrosynthesis enables the rapid and efficient design and synthesis of novel drug candidates.

In this subsection, we explore the proficiency of GPT-4 in comprehending chemical reactions and predicting potential reactants for products, both qualitatively and quantitatively.

2.5.1 Understanding chemical reactions

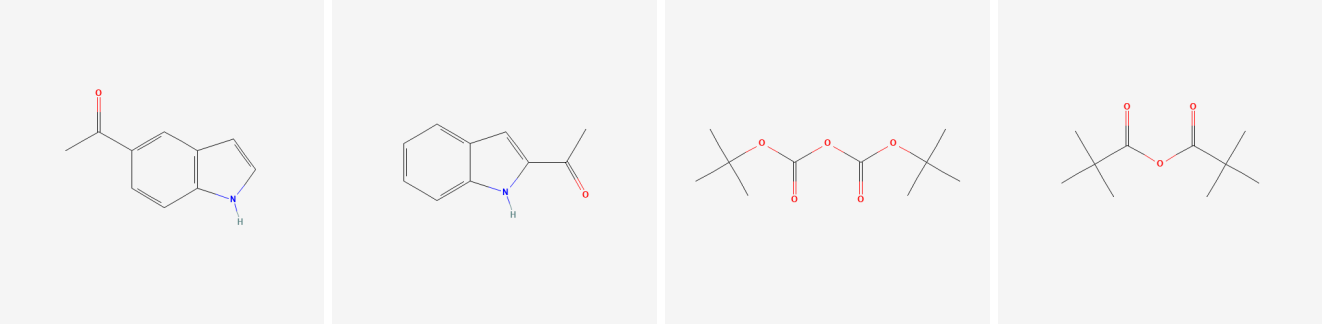
We show two cases to demonstrate the capability of GPT-4 in understanding chemical reactions.

In Fig. 2.21, we ask GPT-4 to act as an organic chemist and retrosynthesis expert and explain the given chemical reaction (represented by SMILES sequences). GPT-4 first translates the SMILES sequences to the names of the reactants and then explains the reaction mechanism. GPT-4 fails at the first step, i.e., translating the SMILES CC(=O)c1ccc2[nH]ccc2c1[9] to the name 2-acetylindole, and translating the SMILES CC(C)(C)OC(=O)OC(=O)OC(C)(C)C[10] to the name trimethylacetic anhydride. As shown in Fig. 2.20, although these names have very similar molecular graphs to the original SMILES, they are different molecules. As a result, the following explanation goes in the wrong direction.

In Fig. 2.22, we ask GPT-4 to carefully think step by step to explain this chemical reaction. This time the explanation goes in the right direction: GPT-4 no longer translates the SMILES sequences into the names and

[9]IUPAC name is 1-(1H-indol-5-yl)ethanone.

[10]IUPAC name is tert-butyl (2-methylpropan-2-yl)oxycarbonyl carbonate.



(a) 1-(1H-indol-5-yl)ethanone (b) 2-acetylindole (c) Di-tert-butyl dicarbonate (d) trimethylacetic anhydride

Figure 2.20: The 2D molecular graphs for two ground truth reactants, (a) and (c), and their translated names, (b) and (d), as generated by GPT-4. Although similar, they are not identical.

GPT-4

System message:
You are an organic chemist and retrosynthesis expert and should be able to explain and predict chemical reactions. You are given a chemical reaction, including the SMILES sequences of both product and reactants, with multiple reactants connected by ".".

Prompt:
Please explain this chemical reaction:
Product: CC(=O)c1ccc2c(ccn2C(=O)OC(C)(C)C)c1
Reactants: CC(=O)c1ccc2[nH]ccc2c1.CC(C)(C)OC(=O)OC(=O)OC(C)(C)C

GPT-4:
The chemical reaction you've provided involves two reactants:

1. CC(=O)c1ccc2[nH]ccc2c1, which is also known as 2-acetylindole.
2. CC(C)(C)OC(=O)OC(=O)OC(C)(C)C, which is pivaloyl anhydride or trimethylacetic anhydride.

In this reaction, the 2-acetylindole undergoes an acylation reaction with the pivaloyl anhydride. The nitrogen atom in the indole ring (which has a hydrogen atom attached) acts as a nucleophile and attacks the carbonyl carbon of the pivaloyl anhydride. This leads to the opening of the anhydride ring and the formation of a new amide bond between the two reactants.

The final product of this reaction is CC(=O)c1ccc2c(ccn2C(=O)OC(C)(C)C)c1, which can be described as a pivaloylated 2-acetylindole.

Here's the overall reaction: 2-acetylindole + pivaloyl anhydride → pivaloylated 2-acetylindole

Figure 2.21: Example 1 of understanding chemical reactions.

instead describes the functional groups in the molecule in detail. Unfortunately, it is still not correct: Di-tert-butyl dicarbonate does not have three esters (C=O) functional groups, and the explanation of the reaction mechanism is not fully correct, as we would get isobutene and CO2 as side products, not tert-butanolate.

2.5.2 Predicting retrosynthesis

We quantitatively study GPT-4's capability in single-step retrosynthesis prediction (i.e., predicting possible reactants for a given product), using the widely-used benchmark USPTO-50K dataset [74] and the few-shot setting.

Settings The USPTO-50K dataset contains 50,037 chemical reactions extracted from US patents. We use the same data splitting as most literature, consisting of 40,029 reactions in the training set and 5,007 reactions in the test set. Due to the limitation of the API call, we select the first 500 samples from the USPTO-50K test set for testing. We use the top-1 accuracy as the evaluation metric and R-SMILES [110] as the main baseline, which is the state-of-the-art model specially designed for retrosynthesis prediction and trained on this dataset.

GPT-4

System message:
You are an organic chemist and retrosynthesis expert and can explain and predict chemical reactions. You are given a chemical reaction, including the SMILES sequence of both product and reactants, with multiple reactants connected by ".".

Prompt:
Please explain this chemical reaction:
Product: CC(=O)c1ccc2c(ccn2C(=O)OC(C)(C)C)c1
Reactants: CC(=O)c1ccc2[nH]ccc2c1.CC(C)(C)OC(=O)OC(=O)OC(C)(C)C
Let's think step-by-step for this explanation task.

GPT-4:
This chemical reaction involves the formation of a new product from two given reactants. Let's break down the reaction step by step:
Reactants:

1. CC(=O)c1ccc2[nH]ccc2c1: This is an aromatic compound containing a benzene ring with an adjacent 5-membered ring containing a nitrogen atom. The benzene ring has a ketone (C=O) functional group attached to one of its carbons.
2. CC(C)(C)OC(=O)OC(=O)OC(C)(C)C: This is a triacyl compound consisting of three ester (C=O) functional groups, each connected to a central oxygen atom. The ester groups are bonded to a tert-butyl group (C(C)(C)C).

Reaction:

The reaction proceeds via nucleophilic aromatic substitution, where the nitrogen atom in the 5-membered ring of the first reactant acts as a nucleophile and attacks the carbonyl carbon (C=O) of one of the ester groups in the second reactant. This leads to the formation of a new C-N bond and the loss of a leaving group, which in this case is the tert-butyl group (C(C)(C)C) along with an oxygen atom.

Product:

CC(=O)c1ccc2c(ccn2C(=O)OC(C)(C)C)c1: The product is a fused aromatic compound containing a benzene ring and a 5-membered ring with a nitrogen atom. The benzene ring has a ketone (C=O) functional group, while the 5-membered ring has an ester group (C(=O)OC(C)(C)C) bonded to the nitrogen atom.

Overall, the reaction involves the formation of a new C-N bond between the two reactants, resulting in a more complex fused aromatic compound as the product.

Figure 2.22: Example 2 of understanding chemical reactions.

Table 7: Few-shot retrosynthesis prediction results on USPTO-50k datset.

Method	**Top-1 Accuracy (↑)**	**Invalid SMILES (↓)**
R-SMILES [110]	53.6%	0%
GPT-4 (Zero-shot)	0.2%	17.8%
GPT-4 (Zero-shot + IUPAC name)	0.2%	12.8%
GPT-4 (Random, k=1)	0.2%	7.4%
GPT-4 (Random, k=5)	1.4%	9.4%
GPT-4 (Random, k=10)	1.2%	9.2%
GPT-4 (Random, k=20)	1.0%	7.2%
GPT-4 (Fingerprint similarity, k=1)	12.8%	9.2%
GPT-4 (Fingerprint similarity, k=5)	19.4%	7%
GPT-4 (Fingerprint similarity, k=10)	20.2%	4.8%
GPT-4 (Fingerprint similarity, k=10 + IUPAC name)	**20.6%**	4.8%
GPT-4 (Fingerprint similarity, k=20)	19.4%	**4.4%**

Few-shot results We consider several aspects while evaluating GPT-4's few-shot capability for retrosynthesis prediction: (1) different numbers of few-shot examples, and (2) different ways to obtain few-shot examples where we perform (a) randomly selecting and (b) selecting K nearest neighbors based on Molecular Fingerprints similarity from the training dataset. (3) We also evaluate whether adding IUPAC names to the prompt can improve the accuracy. Fig. 2.23 illustrates the prompt used for the few-shot evaluation.

The results are shown in in Table 7, from which we have several observations:

- GPT-4 achieves reasonably good prediction for retrosynthesis, with an accuracy of 20.6% for the best setting.
- The accuracy of GPT-4 improves when we add more examples to the prompt, with $K = 10$ being a good choice.
- K nearest neighbors for few-shot demonstrations significantly outperform random demonstrations (20.2% vs 1.2%).
- Including IUPAC names in the prompt slightly improves the accuracy (20.6% vs 20.2%) and reduces the ratio of invalid SMILES.
- The accuracy of GPT-4 (20.6%) is lower than that of the domain-specific model (53.6%), which indicates plenty of room to improve GPT-4 for this specific task.

Fig. 2.24 shows a case where GPT-4 fails to predict the correct reactants for a product in the first attempt and finally succeeds after several rounds of guidance and correction. This suggests that GPT-4 possesses good knowledge but requires specific user feedback and step-by-step verification to avoid errors.

GPT-4

Prompt:

Predict the reactants for the product with the SMILES sequence and the IUPAC name.

Example 1:

Product: COc1nc2ccc(C(=O)c3cncn3C)cc2c(Cl)c1Cc1ccc(C(F)(F)F)cc1, whose IUPAC name is: [4-chloro-2-methoxy-3-[[4-(trifluoromethyl)phenyl]methyl]quinolin-6-yl]-(3-methylimidazol-4-yl)methanone

Reactants: COc1nc2ccc(C(O)c3cncn3C)cc2c(Cl)c1Cc1ccc(C(F)(F)F)cc1

Example 2:

Product: COc1nc2ccc(C(=O)c3cnc(C)n3C)cc2c(Cl)c1Cc1ccc(C(F)(F)F)cc1, whose IUPAC name is: [4-chloro-2-methoxy-3-[[4-(trifluoromethyl)phenyl]methyl]quinolin-6-yl]-(2,3-dimethylimidazol-4-yl)methanone

Reactants: COc1nc2ccc(C(O)c3cnc(C)n3C)cc2c(Cl)c1Cc1ccc(C(F)(F)F)cc1

Example 3:

Product: COc1nc2ccc(Br)cc2c(Cl)c1Cc1ccc(C(F)(F)F)cc1, whose IUPAC name is: 6-bromo-4-chloro-2-methoxy-3-[[4-(trifluoromethyl)phenyl]methyl]quinoline

Reactants: C[O-].FC(F)(F)c1ccc(Cc2c(Cl)nc3ccc(Br)cc3c2Cl)cc1

Example 4:

Product: COc1nc2ccc(C(=O)c3cnnn3C)cc2c(Cl)c1Cc1ccc(C(F)(F)F)cc1, whose IUPAC name is: [4-chloro-2-methoxy-3-[[4-(trifluoromethyl)phenyl]methyl]quinolin-6-yl]-(3-methyltriazol-4-yl)methanone

Reactants:

GPT-4:

COc1nc2ccc(C(O)c3cnnn3C)cc2c(Cl)c1Cc1ccc(C(F)(F)F)cc1

Figure 2.23: Few-shot examples used in the few-shot retrosynthesis prediction evaluation.

GPT-4

Prompt:

Predict the reactants for the product with the SMILES sequence and the IUPAC name.

Example 1:

Product:

CC(C)C[C@H](NC(=O)C[C@H](O)C[C@H](O)C=CC(=C(c1ccc(F)cc1)c1ccc(F)cc1)c1nnnn1C)C(=O)O, whose IUPAC name is: (2S)-2-[[(3R,5S)-9,9-bis(4-fluorophenyl)-3,5-dihydroxy-8-(1-methyltetrazol-5-yl)nona-6,8-dienoyl]amino]-4-methylpentanoic acid

Reactants:

COC(=O)[C@H](CC(C)C)NC(=O)C[C@H](O)C[C@H](O)C=CC(=C(c1ccc(F)cc1)c1ccc(F)cc1)c1nnnn1C

Example 2:

Product:

CCOC(=O)C(C#N)=C(c1ccc(F)cc1)c1ccc(F)cc1, whose IUPAC name is: ethyl 2-cyano-3,3-bis(4-fluorophenyl)prop-2-enoate

Reactants:

CCOC(=O)CC#N.O=C(c1ccc(F)cc1)c1ccc(F)cc1

Example 3:

Product:

CCOC(=O)CCc1cc(OC(C)C)nn1Cc1ccc(F)cc1F, whose IUPAC name is: ethyl 3-[2-[(2,4-difluorophenyl)methyl]-5-propan-2-yloxypyrazol-3-yl]propanoate

Reactants:

CCOC(=O)CCc1cc(OC(C)C)n[nH]1.Fc1ccc(CBr)c(F)c1

Example 4:

Product:

CCOC(=O)CC(=O)CC(O)C=CC(=C(c1ccc(F)cc1)c1ccc(F)cc1)c1nnnn1CC, whose IUPAC name is: ethyl 8-(1-ethyltetrazol-5-yl)-9,9-bis(4-fluorophenyl)-5-hydroxy-3-oxonona-6,8-dienoate

Reactants:

GPT-4:

CCOC(=O)CC(=O)CC(O)C=CC(=C(c1ccc(F)cc1)c1ccc(F)cc1)c1nnnn1C

Prompt:

The answer is not correct, it should be an addition reaction

GPT-4:

Apologies for the confusion. Here is the correct reactants for Example 4:

CCOC(=O)CC(=O)CC(O)C=CC(=C(c1ccc(F)cc1)c1ccc(F)cc1)c1nnnn1CC + BrCCl

Prompt:

If l tell you one reactant is CCOC(=O)CC(C)=O, show me the right answer

GPT-4:

My apologies for the previous confusion. Given the reactant CCOC(=O)CC(C)=O, the correct reactants for Example 4 are:

CCOC(=O)CC(C)=O + O=CC(O)C=CC(=C(c1ccc(F)cc1)c1ccc(F)cc1)c1nnnn1CC

Prompt:

Regarding the functional group O=CC(O)C=C, it should have fewer carbon and oxygen atoms. Rethink and give the right answer.

GPT-4:

I apologize for the mistake. Given the reactant CCOC(=O)CC(C)=O and considering your feedback, the correct reactants for Example 4 are: CCOC(=O)CC(C)=O + O=CC=CC(=C(c1ccc(F)cc1)c1ccc(F)cc1)c1nnnn1CC

Figure 2.24: Example of the retrosynthesis prediction. With multiple rounds of guidance, GPT-4 finally gives the correct reactants.

2.6 Novel molecule generation

In this subsection, we study an important application in drug discovery: to propose/generate novel molecules as drug candidates.

SARS-Cov-2 uses its spike protein to enter human surface receptors. We ask GPT-4 to provide general guidance on designing protein-based drugs that bind to the spike protein to neutralize COVID-19. GPT-4 then teaches us how to use a computational tool called Rosetta to design such a protein drug from scratch. GPT-4 shows great answers to our questions, indicating that GPT-4 can help design novel protein drugs.

GPT-4

Prompt:

Design 3 proteins that bind to the RBD of the SARS-CoV-2 spike protein, using Rosetta. And give me their corresponding amino acid sequences.

GPT-4:

Rosetta is a software suite for protein structure prediction and design, which can use various methods and algorithms to generate and evaluate candidate proteins that bind to a given target. One of the methods that Rosetta can use is called RosettaDesign, which optimizes the sequence and conformation of a protein scaffold to fit a binding site on the target. RosettaDesign can also incorporate information from known protein-protein interactions, such as antibody-antigen complexes, to guide the design process. Here, we use RosettaDesign to design three proteins that bind to the receptor-binding domain (RBD) of the SARS-CoV-2 spike protein, using the crystal structure of the RBD in complex with the human ACE2 receptor (PDB ID: 6M0J) as a template. We also use the sequence and structure of a human antibody fragment (Fab) that neutralizes SARS-CoV-2 (PDB ID: 7C01) as a source of inspiration for the design. The following steps are performed for each design:

- Select a scaffold protein that has a similar size and shape to the Fab variable domain, and that has a low sequence identity to any human protein. For example, we can use the scaffold from the Rosetta Antibody Design (RAbD) benchmark set, which is based on the structure of a camelid antibody (PDB ID: 1X9D).
- Align the scaffold to the Fab variable domain, and superimpose the aligned scaffold on the RBD-ACE2 complex, such that the scaffold mimics the orientation and position of the Fab.
- Identify the residues on the scaffold that are in contact with the RBD, and define them as the design region. The rest of the scaffold is fixed as the background region. Optionally, apply some constraints to preserve the scaffold stability and the RBD interface.
- Use RosettaDesign to optimize the sequence and conformation of the design region, while minimizing the energy and maximizing the specificity of the interaction with the RBD. RosettaDesign can sample different amino acid types and rotamer conformations for each residue in the design region, and score them using a combination of physical and statistical potentials. RosettaDesign can also perform some local backbone movements to improve the fit of the design region to the RBD.
- Evaluate the designed proteins using various criteria, such as the binding energy, the interface shape complementarity, the sequence diversity, the solubility, the stability, and the immunogenicity. Select the best designs for further validation and testing.

The following are the amino acid sequences of three possible designs, using the one-letter code. The residues in the design region are highlighted in bold. The scaffold sequence is shown in parentheses for comparison.

Design 1:

QVQLQESGPGLVKPSETLSLTCTVSGGSISNNYAWHWIRQAPGKGLEWVAYISYSGSTYYNPSLKSR
VTISVDTSKNQFSLKLSSVTAADTAVYYCARGYRYYGYDYWGQGTLVTVSS
(QVQLQESGPGLVKPSETLSLTCTVSGGSISNNYAWHWIRQAPGKGLEWVAYISYSGSTYYNPSLKSR
VTISVDTSKNQFSLKLSSVTAADTAVYYCARGYRYYGYDYWGQGTLVTVSS)

Design 2:

QVQLQESGPGLVKPSETLSLTCTVSGGSISNNYAWHWIRQAPGKGLEWVSIISWGDGTTYYNPSLKSR
VTISVDTSKNQFSLKLSSVTAADTAVYYCARSGYDYDYDYWGQGTLVTVSS
(QVQLQESGPGLVKPSETLSLTCTVSGGSISNNYAWHWIRQAPGKGLEWVAYISYSGSTYYNPSLKSR
VTISVDTSKNQFSLKLSSVTAADTAVYYCARGYRYYGYDYWGQGTLVTVSS)

Figure 2.25: GPT-4 understands how to use computational tools for biological design.

One metric to evaluate a protein drug is to estimate its binding affinity to its target. In this case, we asked GPT-4 to teach us how to computationally perform this task. GPT-4 provided detailed instructions on how to estimate protein binding affinity using a publicly available tool called RosettaDock. It also showed an example of how to interpret the results. While GPT-4 shows a reasonable design process, we note that the protein designs require wet-lab experiments for validation because it is still challenging to computationally predict protein-protein interaction with highly complex body environments making this even harder.

GPT-4

Prompt:

Calculate the binding affinity of RBD of SARS-CoV-2 and a protein called “Design 2" using RosettaDock.

GPT-4:

There are different ways to perform protein-protein docking using RosettaDock, but one possible workflow is:

- Convert the protein sequences to 3D structures using Rosetta’s ab initio or comparative modeling protocols, or use existing structures from databases or experiments if available.

- Prepare the structures for docking using Rosetta’s prepacking and setup protocols, which optimize the side chains, add hydrogen atoms, and assign docking partners and constraints.

- Perform the docking simulation using Rosetta’s docking protocol, which samples different orientations and conformations of the two proteins and scores them based on their energy and shape complementarity.

- Analyze the docking results using Rosetta’s clustering and scoring protocols, which group the docked models by similarity and rank them by their Rosetta energy and interface score.
  To illustrate this workflow, we will use the following example commands and files, assuming that we have the 3D structure of the RBD of SARS-CoV-2 (PDB ID: 6M0J) and we have generated a 3D model of Design 1 using Rosetta’s ab initio protocol (design1.pdb). Note that these are not the only or optimal parameters and options for docking, and the results may vary depending on the input structures, random seeds, and computational resources.

- Prepack the structures using the following command:
  rosetta_scripts.linuxgccrelease -s 6M0J.pdb design1.pdb -parser:protocol prepack.xml -nstruct 1 -overwrite
  where prepack.xml is an XML script that defines the prepacking protocol, which can be found in the Rosetta documentation or downloaded from
  https://www.rosettacommons.org/demos/latest/tutorials/rosetta_scripts/prepack/prepack.xml

- Setup the docking partners and constraints using the following command:
  rosetta_scripts.linuxgccrelease -s 6M0J_0001.pdb design1_0001.pdb -parser:protocol setup_docking.xml -nstruct 1 -overwrite
  where setup_docking.xml is an XML script that defines the setup docking protocol, which can be found in the Rosetta documentation or downloaded from
  https://www.rosettacommons.org/demos/latest/tutorials/rosetta_scripts/setup_docking/setup_docking.xml
  This command will generate two files: 6M0J_0001_0001.pdb and design1_0001_0001.pdb, which are the prepacked structures with the docking partners and constraints assigned. The docking partners are indicated by the chain IDs A and B, and the constraints are indicated by the REMARK lines in the PDB files. For example, the following lines in 6M0J_0001_0001.pdb indicate that the RBD is chain A and it has a distance constraint with the residue 437 of chain B (Design 1):

```
REMARK 666 MATCH TEMPLATE A 437 A 437 1
REMARK 666 MATCH MOTIF A 437 A 437 1
REMARK 666 MATCH MOTIF A 437 B 437 1
ATOM 1 N GLY A 437 32.556 71.839 40.760 1.00 0.00 N
ATOM 2 CA GLY A 437 33.887 72.315 40.457 1.00 0.00 C
ATOM 3 C GLY A 437 34.161 73.794 40.768 1.00 0.00 C
ATOM 4 O GLY A 437 33.266 74.614 40.548 1.00 0.00 O
TER
ATOM 5 N GLN B 437 30.547 76.764 39.975 1.00 0.00 N
ATOM 6 CA GLN B 437 29.984 77.378 38.779 1.00 0.00 C
ATOM 7 C GLN B 437 28.463 77.366 38.882 1.00 0.00 C
ATOM 8 O GLN B 437 27.789 76.362 38.577 1.00 0.00 O
```

- Perform the docking simulation using the following command:
  `rosetta_scripts.linuxgccrelease -s 6M0J_0001_0001.pdb design1_0001_0001.pdb -parser:protocol docking.xml -nstruct 100 -out:file:scorefile docking.sc`
  where docking.xml is an XML script that defines the docking protocol, which can be found in the Rosetta documentation or downloaded from
  https://www.rosettacommons.org/demos/latest/tutorials/rosetta_scripts/docking/docking.xml

- ...

Figure 2.26: GPT-4 teaches us how to use a computational tool called Rosetta to design such a protein drug from scratch.

2.7 Coding assistance for data processing

In this subsection, we evaluate the assistant's capabilities in data processing for drug discovery using GPT-4. Specifically, we task GPT-4 with generating Python code for processing drug-related data. A significant amount of drug and protein data are stored in sequence formats, such as SMILES and FASTA, which can be downloaded from the PubChem[11] and UniProt[12] websites. We ask GPT-4 to write Python code to download these sequence data, with examples shown in Fig. 2.27 and Fig. 2.28.

In Fig. 2.28, GPT-4 provides accurate code for downloading protein sequences, adding spaces, and saving the data to a file with a specific format. For molecule processing (in Fig. 2.27), we request both the SMILES and chemical formula retrieval for a molecule. Interestingly, GPT-4 generates an almost correct URL for data downloading but combines the "SMILES and formula" keywords in the URL, rendering the URL invalid[13]. When informed about the error, GPT-4 identifies the issue as being related to the PubChem REST API call. Instead of fixing the bug, it suggests an alternative solution using the `'pubchempy'` package for data downloading, which successfully executes the code. These examples demonstrate that GPT-4 can help generate correct scripts for data processing in drug discovery, which is very helpful for researchers in the field.

[11] `https://pubchem.ncbi.nlm.nih.gov/`

[12] `https://www.uniprot.org/`

[13] The correct URL is '.../ISomericSMILES/TXT' for SMILES downloading and '.../MolecularFormula/TXT' for formula downloading.

GPT-4

Prompt:

I would like to download the SMILES and chemical formula from PubChem given PubChem IDs. After downloading, save the SMILES and the chemical formula into a file with each line containing the PubChem ID ahead. Show me the Python code.

GPT-4:

To download the SMILES and chemical formula from PubChem given PubChem IDs, you can use the requests library. Here's the Python code to download the data and save it to a file:

```
import requests

def download_pubchem_data(pubchem_id):
    url = f'https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/CID/{pubchem_id}/property/IsomericSMILES,
    MolecularFormula/TXT'
    response = requests.get(url)
    if response.status_code == 200:
        return response.text.strip().split('\t')
    else:
        print(f"Error: Unable to download data for PubChem ID {pubchem_id}")
        return None

pubchem_ids = [1983, 5090]  # Replace with the PubChem IDs you want to download
output_file = "pubchem_data.txt"
with open(output_file, "w") as f:
    for pubchem_id in pubchem_ids:
        data = download_pubchem_data(pubchem_id)
        if data:
            smiles, formula = data
            f.write(f"{pubchem_id}\t{smiles}\t{formula}\n")
print(f"PubChem data saved to {output_file}")
```

Replace the PubChem IDs in the pubchem_ids list with the IDs you want to download. The code will save the PubChem ID, SMILES, and chemical formula to the "pubchem_data.txt" file. Each line of the file will contain the PubChem ID followed by the SMILES and the chemical formula.

Prompt:

It shows "Error: Unable to download data for PubChem ID".

GPT-4:

I apologize for the inconvenience. It appears that the PubChem REST API is blocking the requests. We can use an alternative method to fetch the data using the pubchempy library. First, you will need to install the library using pip:

```
pip install pubchempy
```

Then, use the following code to download the SMILES and chemical formula from PubChem given PubChem IDs:

```
import pubchempy as pcp

def download_pubchem_data(pubchem_id):
    try:
        compound = pcp.Compound.from_cid(pubchem_id)
        smiles = compound.isomeric_smiles
        formula = compound.molecular_formula
        return smiles, formula
    except Exception as e:
        print(f"Error: Unable to download data for PubChem ID {pubchem_id}: {e}")
        return None
{...} # (same as above)
```

This code uses the pubchempy library to fetch the SMILES and chemical formula for the given PubChem IDs and saves the data into a file with each line containing the PubChem ID, SMILES, and chemical formula.

Figure 2.27: Coding assistance for downloading SMILES, the formula of a molecule from PubChem with ID.

GPT-4

Prompt:

I would like to download the protein FASTA sequence from UniProt given UniProt IDs. After downloading the sequences, please add empty space between amino acids and save the sequences into a file with each line containing the UniProt ID and the sequence. Show me the Python code.

GPT-4:

To download protein FASTA sequences from UniProt, add empty spaces between amino acids, and save the sequences into a file with each line containing the UniProt ID and the sequence, you can use the requests library. Here's the Python code to accomplish this:

```
import requests

def download_fasta(uniprot_id):
    url = f'https://www.uniprot.org/uniprot/{uniprot_id}.fasta'
    response = requests.get(url)

    if response.status_code == 200:
        return response.text
    else:
        print(f"Error: Unable to download protein sequence for UniProt ID {uniprot_id}")
        return None

def add_spaces_to_sequence(sequence):
    return " ".join(sequence)

uniprot_ids = ['P0DTD1', 'P12345']  # Replace with the UniProt IDs you want to download

output_file = "protein_sequences.txt"

with open(output_file, "w") as f:
    for uniprot_id in uniprot_ids:
        fasta_sequence = download_fasta(uniprot_id)

        if fasta_sequence:
            lines = fasta_sequence.splitlines()
            sequence = "".join(lines[1:])
            spaced_sequence = add_spaces_to_sequence(sequence)

            f.write(f"{uniprot_id}\t{spaced_sequence}\n")

print(f"Protein sequences saved to {output_file}")
```

Replace the UniProt IDs in the uniprot_ids list with the IDs you want to download. The code will save the UniProt ID and the FASTA sequences with empty spaces between amino acids to the "protein_sequences.txt" file. Each line of the file will contain the UniProt ID followed by the sequence.

Figure 2.28: Coding assistance for downloading protein sequences from UniProt with ID.

3 Biology

3.1 Summary

In this chapter, we delve into an in-depth exploration of GPT-4's capabilities within the realm of biological research, focusing primarily on its proficiency in comprehending biological language (Sec. 3.2), employing built-in biological knowledge for reasoning (Sec. 3.3), and designing biomolecules and bio-experiments (Sec. 3.4). Our observations reveal that GPT-4 exhibits substantial potential to contribute to the field of biology by demonstrating its capacity to process complex biological language, execute bioinformatic tasks, and even serve as a scientific assistant for biology design. GPT-4's extensive grasp of biological concepts and its promising potential as a scientific assistant in design tasks underscore its significant role in advancing the field of biology:[14]

- Bioinformation Processing: GPT-4 displays its understanding of information processing from specialized files in biological domains, such as MEME format, FASTQ format, and VCF format (Fig. 3.7 and Fig. 3.8). Furthermore, it is adept at performing bioinformatic analysis with given tasks and data, exemplified by predicting the signaling peptides for a provided sequence as illustrated in Fig. 3.4.
- Biological Understanding: GPT-4 demonstrates a broad understanding of various biological topics, encompassing consensus sequences (Fig. 3.2), PPI (Fig. 3.11 and 3.12), signaling pathways (Fig. 3.13), and evolutionary concepts (Fig. 3.17).
- Biological Reasoning: GPT-4 possesses the ability to reason about plausible mechanisms from biological observations using its built-in biological knowledge (Fig. 3.12 - 3.16).
- Biological Assisting: GPT-4 demonstrates its potential as a scientific assistant in the realm of protein design tasks (Fig. 3.20), and in wet lab experiments by translating experimental protocols for automation purposes (Fig. 3.21).

While GPT-4 presents itself as an incredibly powerful tool for assisting research in biology, we also observe some limitations and occasional errors. To better harness the capabilities of GPT-4, we provide several tips for researchers:

- FASTA Sequence Understanding: A notable challenge for GPT-4 is the direct processing of FASTA sequences (Fig. 3.9 and Fig. 3.10). It is preferable to supply the names of biomolecules in conjunction with their sequences when possible.
- Inconsistent Result: GPT-4's performance on tasks related to biological entities is influenced by the abundance of information pertaining to the entities. Analysis of under-studied entities, such as transcription factors, may yield inconsistent results (Fig. 3.2 and Fig. 3.3).
- Arabic Number Understanding: GPT-4 struggles to directly handle Arabic numerals; converting Arabic numerals to text is recommended (Fig. 3.20).
- Quantitative Calculation: While GPT-4 excels in biological language understanding and processing, it encounters limitations in quantitative tasks (Fig. 3.7). Manual verification or validation with alternative computational tools is advisable to obtain reliable conclusions.
- Prompt Sensitivity: GPT-4's answers can display inconsistency and are highly dependent on the phrasing of the question (Fig. 3.19), necessitating further refinements to reduce variability, such as experimenting with different prompts.

In summary, GPT-4 exhibits significant potential in advancing the field of biology by showcasing its proficiency in understanding and processing biological language, reasoning with built-in knowledge, and assisting in design tasks. While there are some limitations and errors, with proper guidance and refinements, GPT-4 could become an invaluable tool for researchers in the ever-evolving landscape of biological research.

3.2 Understanding biological sequences

While GPT-4 is trained with human language, DNA and protein sequences are usually considered the 'language' of life. In this section, we explore the capabilities of GPT-4 on biological language (sequences) understanding and processing. We find that GPT-4 has rich knowledge about biological sequences processing,

[14] In this chapter, we use yellow to indicate incorrect or inaccurate responses from GPT-4.

but its capability is currently limited due to its low accuracy on quantitative tasks and the risk of confusion as discussed in Sec. 3.2.1 and Sec. 3.2.2. We also list several caveats that should be noted when handling biological sequences with GPT-4 in Sec. 3.2.3.

3.2.1 Sequence notations *vs.* text notations

DNA or protein sequences are usually represented by single-letter codes. These codes are essential for DNA or protein-related studies, as they notate each nucleotide or amino acid in the sequence explicitly. However, the sequence notations are very long. Text notations composed of combinations of letters, numbers, and symbols that are human-readable are also used for DNA or protein reference. Therefore, we first evaluate GPT-4 's ability to handle sequence notations and text notations of biological sequences.

Case: Conversion between sequence notations and text notations. We ask GPT-4 to convert between biological sequences and their text notations: 1) Output protein names given protein sequences. 2) Output protein sequences given names. Before each task, we restart the session to prevent information leakage. The results show that GPT-4 knows the process for sequence-to-text notation conversion, yet it cannot directly look up (also known as BLAST [2]) the sequence by itself. Meanwhile, GPT-4 prefers text notations of biological sequences (including both protein and DNA, the latter is not shown). It provides richer information when text notation is given, likely due to its design philosophy. To be pointed out, we also notice that, generating sequences may result in catastrophic behavior of GPT-4. As shown in Case 2 (Fig. 3.1), while GPT-4 returns the correct UniProt ID, it has trouble generating the sequence. The sequence generation crashes with several different prompts we attempt.

GPT-4

System message:

You are an expert in genomics and bioinformatics.

Prompt: (Case 1)

Can you give me the name of the protein (delimited by triple backticks) below?

```
MYNMMETELKPPGPQQTSGGGGGNSTAAAAGGNQKNSPDRVKRPMNAFMVWSRGQRRKMAQENPKMHNSE
ISKRLGAEWKLLSETEKRPFIDEAKRLRALHMKEHPDYKYRPRRKTKTLMKKDKYTLPGGLLAPGGNSMA
SGVGVGAGLGAGVNQRMDSYAHMNGWSNGSYSMMQDQLGYPQHPGLNAHGAAQMQPMHRYDVSALQYNSM
TSSQTYMNGSPTYSMSYSQQGTPGMALGSMGSVVKSEASSSPPVVTSSSHSRAPCQAGDLRDMISMYLPG
AEVPEPAAPSRLHMSQHYQSGPVPGTAINGTLPLSHM
```

GPT-4:

I cannot directly analyze the protein sequence and provide the name as I am an AI language model, and I don't have the functionality to perform bioinformatics tasks. However, I can guide you on how to identify the protein yourself.

1. You can use the Basic Local Alignment Search Tool (BLAST) to find similar sequences in databases. Visit the NCBI BLAST webpage (https://blast.ncbi.nlm.nih.gov/Blast.cgi) and choose "Protein BLAST."
2. Paste the given protein sequence into the "Enter Query Sequence" box.
3. Click the "BLAST" button and wait for the results to load.
4. Analyze the results to find the most similar protein sequences with known names. The name or function of your protein may be inferred from these matches.

Remember that the protein name might not be unique or universally agreed upon, as different research groups might use different names for the same protein. It's always a good idea to cross-check the results with other databases or publications.

Prompt: (Case 2)

Can you give me the protein sequence of SOX−2?

GPT-4:

SOX2 (SRY (sex determining region Y)-box 2) is a transcription factor that plays a critical role in the regulation of embryonic development and stem cell pluripotency. The protein sequence of SOX2 varies across different species. Here, I'll provide you with the human SOX2 protein sequence as an example.
Human SOX2 protein sequence (UniProt ID: P48431) consists of 317 amino acids:

```
MTESPAAGSGGSSSSSSSSSSSSSSSGGGSAAAGGSGGGGSGGGGSGGGGSGGGGSGGGGSGGGGSG
GGGSGGGGSGGGGSGGGGSGGGGSGGGGSGGGGSGGGGSGGGGSGGGGSGGGGSGGGGSGGGG (...)
```

Figure 3.1: Although GPT-4 may presently face challenges in identifying the given protein sequence and accurately listing the sequence when supplied with the name, it still offers valuable guidance to accomplish the task.

3.2.2 Performing sequence-related tasks with GPT-4

Despite the issues of handling biological sequences directly, we examine if GPT-4 can perform sequence-related tasks commonly seen in biological studies. Tests on consensus sequences, signaling peptides, and functional domains indicate that GPT-4 has knowledge of biological sequences and a good understanding of bioinformatic tasks dealing with biological sequences. It is able to recommend proper tools and processes to carry out these tasks. However, facts and quantitative results provided by GPT-4 should be manually confirmed.

Consensus sequence prediction. In Cases 1 - 2 (Fig. 3.2 and 3.3), we ask GPT-4 to predict the transcription factor (TF) binding site of 2 proteins: 1) MYC, a well-studied bHLH transcription factor with canonical E-box motif CACGTG. 2) ZNF-143, a zinc finger protein with limited descriptions, whose consensus sequence information can be found on JASPAR, an open-access database storing TF binding profiles. GPT-4 performs well in the case of MYC. However, it may raise different answers for the case of ZNF143, where confounding information exists. Collectively, we reason that GPT-4 can better handle the proteins with more validated and consistent information but is not good with under-studied concepts.

Case 1: Predicting MYC binding site. Four manually curated candidate sequences are provided, which rank from *highly likely a MYC binding site* to *unlikely a MYC binding site* according to the sequence of E-box motif. GPT-4 retrieves MYC motif information from its knowledge base and makes all-correct predictions. (Note: random shuffling of the candidates does not affect the prediction of GPT-4, indicating that GPT-4 is not guessing by candidate orders.)

GPT-4

System message:

You are an expert in genomics and bioinformatics.

Prompt: (Case 1)

Predict if MYC can bind to the 4 DNA sequences below, delimited by triple backticks. Output True if the sequence is likely to be bound by MYC. Output False otherwise. Please give me the explanations of your result

“‘

1. `CCACGTGC`
2. `ACACGTGG`
3. `CCGTGTGC`
4. `CCCAATTC`

”’

GPT-4:

MYC is a transcription factor that typically binds to the E-box motif in DNA sequences. The canonical E-box motif is 5'-CACGTG-3'. Let's analyze each of the provided sequences:

`CCACGTGC`
This sequence contains the canonical E-box motif (CACGTG) starting from the second position. Therefore, it is likely to be bound by MYC.
Result: True

`ACACGTGG`
This sequence contains a variant of the E-box motif (CACGTG) starting from the second position. The last nucleotide is different (G instead of C), but MYC can still recognize and bind to this sequence, albeit with potentially weaker affinity.
Result: True

`CCGTGTGC`
This sequence does not contain the canonical E-box motif. The core sequence (CGTGTG) differs from the E-box motif (CACGTG) at the first and last positions. MYC is less likely to bind to this sequence.
Result: False

`CCCAATTC`
This sequence does not contain the canonical E-box motif or any close variants. MYC is unlikely to bind to this sequence.
Result: False

In summary, MYC is likely to bind to the first two sequences and is unlikely to bind to the last two sequences. However, it is important to note that in vivo binding may be influenced by additional factors, such as chromatin accessibility and the presence of other transcription factors.

Figure 3.2: GPT-4 can make correct predictions for MYC binding site

Case 2: Predicting ZNF143 binding site. 4 manually curated candidate sequences are provided, which rank from *highly likely a ZNF143 binding site* to *unlikely a ZNF143 binding site* according to the consensus sequence on JASPAR (9th release). In contrast to Case 1 (Fig. 3.2), GPT-4 “finds” different ZNF143 consensus sequences with subtle adjustments in prompts, and the references of the consensus sequence for ZNF143 provided by GPT-4 are mostly incorrect. One example is shown below in Fig. 3.3, and the motif highlighted in yellow indicates an incorrect reference.

GPT-4

System message:

You are an expert in genomics and bioinformatics.

Prompt: (Case 2)

Predict if ZNF143 can bind to the 4 DNA sequences below, delimited by triple backticks. Output True if the sequence is likely to be bound by ZNF143. Output False otherwise. Also, list step-by-step how you reach the conclusion.

‘‘‘

1. `TTCCCACAATGCATCG`
2. `CTCCCATGGTGCCCCG`
3. `TTCCCAGTGTGCAGGG`
4. `GGAAAGTTTTGAAGGC`

’’’

GPT-4:

To predict if ZNF143 can bind to the given DNA sequences, we will follow these steps:
Retrieve the correct consensus sequence for ZNF143 binding from a reliable source, such as the JASPAR database.
Compare each given DNA sequence with the consensus sequence.
Determine the likelihood of ZNF143 binding based on the similarity of each sequence to the consensus sequence.

Step 1: Retrieve the correct consensus sequence for ZNF143 binding.
According to the JASPAR database, the consensus sequence for ZNF143 is 5’-SCGCCCCGCR-3’ (where S = G/C and R = A/G; MA0839.1).

Step 2: Compare each given DNA sequence with the consensus sequence.
`TTCCCACAATGCATCG`
`CTCCCATGGTGCCCCG`
`TTCCCAGTGTGCAGGG`
`GGAAAGTTTTGAAGGC`

Step 3: Determine the likelihood of ZNF143 binding based on the similarity of each sequence to the consensus sequence.
The sequence has a 4-nucleotide match (CCCA) but lacks the 5’-G and 3’-CGCC.
Prediction: False
The sequence has a 6-nucleotide match (TGCCCC) with the correct 3’-CGCC but lacks the 5’-G.
Prediction: False
The sequence has a 4-nucleotide match (CCCA) but lacks the 5’-G and 3’-CGCC.
Prediction: False
The sequence has no significant similarities to the consensus sequence.
Prediction: False

Based on this analysis, none of the sequences strongly resemble the consensus sequence for ZNF143 binding. Therefore, it is unlikely that ZNF143 can bind to any of these sequences.

Please note that these predictions are based on sequence similarity alone and do not take into account other factors, such as DNA shape, epigenetic modifications, or the presence of other proteins, which might influence ZNF143 binding. To obtain more accurate predictions, experimental validation, such as chromatin immunoprecipitation (ChIP) assays, would be required.

Figure 3.3: GPT-4 makes poor predictions on the ZNF143 binding site, providing mostly incorrect references of the consensus sequence for ZNF143. Notably, MA0839.1 is NOT the ID of ZNF143.

Case 3: Predicting signaling peptides. Signaling peptides are short amino acid sequences within proteins that play essential roles in directing cellular localization. GPT-4 is given protein sequences with or without signaling peptides and then asked to identify possible signaling peptides. It is able to first tell the rules of identifying a signal peptide, and then use the rule to find signal peptides in any given sequence (even with made-up sequences). One positive example is shown below.

GPT-4

System message:

You are an expert in genomics and bioinformatics.

Prompt: (Case 3)

Can you identify if there is any signal peptides in the following sequence:

```
MKALRLSASALFCLLLINGLGAAPPGRPEAQPPPLSSEHKEPVAGDAVPGPKDGSAPEVRGARNSEPQDE
GELFQGVDPRALAAVLLQALDRPASPPAPSGSQQGPEEEAAEALLTETVRSQTHSLPAPESPEPAAPPRP
QTPENGPEASDPSEELEALASLLQELRDFSPSSAKRQQETAAAETETRTHTLTRVNLESPGPERVWRASW
GEFQARVPERAPLPPPAPSQFQARMPDSGPLPETHKFGEGVSSPKTHLGEALAPLSKAYQGVAAPFPKAR
RPESALLGGSEAGERLLQQGLAQVEAGRRQAEATRQAAAQEERLADLASDLLLQYLLQGGARQRGLGGRG
LQEAAEERESAREEEEAEQERRGGEERVGEEDEEAAEAEAEAEEAERARQNALLFAEEEDGEAGAEDKRS
QEETPGHRRKEAEGTEEGGEEEDDEEMDPQTIDSLIELSTKLHLPADDVVSIIEEVEEKRKRKKNAPPEP
VPPPRAAPAPTHVRSPQPPPPAPAPARDELPDWNEVLPPWDREEDEVYPPGPYHPFPNYIRPRTLQPPSA
LRRRHYHHALPPSRHYPGREAQARRAQEEAEAEERRLQEQEELENYIEHVLLRRP
```

GPT-4:

Signal peptides are short sequences of amino acids that direct the transport of proteins to specific locations in the cell. These sequences are typically found at the N-terminus of the protein and are cleaved off once the protein reaches its destination.

To identify a signal peptide in the given sequence, we can look for certain characteristics:

1. A positively charged N-region, usually 1-5 amino acids long, containing mostly lysine (K) or arginine (R) residues
2. A hydrophobic core of 7-15 amino acids, consisting of mostly aliphatic residues such as leucine (L), isoleucine (I), and valine (V)
3. A C-region containing a cleavage site, usually with the amino acids alanine (A), glycine (G), serine (S), or threonine (T)

Looking at the given sequence, the N-terminal region (MKALRLSASALFCLLLINGLGA) appears to have a positively charged N-region (MKALR), a hydrophobic core (LSASALFCLLL), and a C-region containing a possible cleavage site (INGLGA). This indicates the presence of a potential signal peptide in this sequence.

Figure 3.4: GPT-4 correctly predicts the signaling peptides for the given sequence.

Case 4: Identifying protein functional domains. Functional domains are structurally independent units in protein sequences. They enable proteins to carry out particular functions, such as protein-protein interactions, catalysis, and signal transduction. Their categories are more diverse and complex compared to signaling peptides. Our experiments demonstrate that when the role of "Bioinformatics" is incorporated into the system message, GPT-4 actively attempts to identify protein domains from the given sequences. Although the identified domains may not be entirely accurate, GPT-4 effectively grasps the task and suggests tools that can lead to the correct answers. This highlights GPT-4's extensive knowledge in processing biological information; however, there is still potential for improvement in its accuracy and precision.

GPT-4

System message:

You are an expert in Bioinformatics.

Prompt: (Case 4)

Can you identify any functional domains in this sequence:

```
MAAGKIPDWVTAERFEDVLKSNVDGYSKVRNFKAEMGSAAGDNYATNMLRVNIEVELQDGTTKELSYMVK
LPRQREINKEMMKHNIRSQRQCEQDERRLSLQRNNAYFSFVSPQIGDRAPSPSTNSKLLPSENVRPRSCS
RSLPASAHKSWSEETASPTPLLSQRQTTVPGNCNTAITSAVTSLATATATTTSTSSAAQLIIAVPAVNNT
AALTVCNNNNARKEESKQKQKSISTVQTGMDRYIQIKRKLSPQNNKAGNQPKINRTNNGNENSAVNNSNR
YAILADSATEQPNEKTVGEPKKTRPPPIFIREQSTNALVNKLVDLIGDSKFHIIPLKKGNIHEIKLQIQT
EADHRIVTKYLNDAGKNYYTYQLKSCKGLQVVLKGIEATVTPAEIIEALKAKNFSAKTAINILNKDKVPQ
PLFKIELEPELQALKKNEVHPIYNLQYLLHRRITVEEPHKRINPVQCTNCQEYGHTKAYCTLKSVCVVCS
EPHTTANCPKNKDDKSVKKCSNCGEKHTANYRGCVVYKELKSRLNKRIATAHTYNKVNFYSPQPIFQPPL
TVPSTTPTISFASALKSGLEVPAPPTRTAHSEHTPTNIQQTQQSGIEAMMLSLQQSMKDFMTFMQNTLQE
LMKNQNILIQLLVSSKS
```

GPT-4:

The given amino acid sequence appears to be a protein sequence. To identify functional domains within this sequence, bioinformatics tools such as the NCBI's Conserved Domain Database (CDD) or Pfam database can be used. These tools search for conserved regions in protein sequences that are known to have specific functions.

Upon running the sequence through the NCBI's CDD search, I found the following conserved domains:

1. A "DUF4793" domain (Domain of Unknown Function) spanning from position 1 to 128 in the sequence. This domain has not been characterized yet.
2. A "Filamin" domain (also known as Immunoglobulin-like domain) spanning from position 264 to 354. Filamin domains are involved in various cellular processes such as signal transduction, cell migration, and cytoskeleton organization.
3. A "Gelactin" domain spanning from position 433 to 526. Gelactin domains are found in extracellular matrix proteins and are involved in cell adhesion and migration.

Please note that these results may not cover all functional domains in the sequence, as some functional domains might not be conserved or known yet. Further experimental validation is needed to confirm the presence and function of these domains in the protein.

Figure 3.5: GPT-4 has the limitation of identifying the correct protein functional domains. The correct answers should be one zinc-finger CCHC-type domain, one PRE_C2HC domain, and one partial Ecdysteroid kinase-like domain in the example.

GPT-4

System message:

You are an expert in Bioinformatics.

Prompt:

Do you know how one can identify protein domains?

GPT-4:

Yes, identifying protein domains typically involves using bioinformatics tools and databases that are designed to recognize conserved regions or sequence motifs within a given protein sequence. These tools compare the protein sequence to a database of known protein domain families and predict the presence of domains based on sequence similarity. Here are some popular tools and databases for identifying protein domains:

1. NCBI's Conserved Domain Database (CDD) and CD-Search: The CDD is a collection of multiple sequence alignments and derived database search models for ancient domains and full-length proteins. CD-Search is a tool that scans a protein sequence against the CDD to identify conserved domains. Link: https://www.ncbi.nlm.nih.gov/Structure/cdd/wrpsb.cgi
2. InterPro: InterPro is a database that integrates protein signature databases, including Pfam, PROSITE, PRINTS, ProDom, SMART, TIGRFAMs, PIRSF, SUPERFAMILY, Gene3D, and PANTHER. InterProScan is a tool that scans a protein sequence against the InterPro database to identify domains and functional sites. Link: https://www.ebi.ac.uk/interpro/search/sequence/
3. Pfam: Pfam is a large collection of protein families, each represented by multiple sequence alignments and hidden Markov models (HMMs). You can search a protein sequence against the Pfam database to identify domains. Link: https://pfam.xfam.org/
4. SMART (Simple Modular Architecture Research Tool): SMART is a web-based tool that allows the identification and annotation of genetically mobile domains and the analysis of domain architectures. Link: http://smart.embl-heidelberg.de/
5. PROSITE: PROSITE is a database of protein families and domains, which uses patterns and profiles to detect conserved regions in protein sequences. Link: https://prosite.expasy.org/

These tools use different algorithms and databases, so it is often helpful to use multiple tools to get a comprehensive understanding of the protein domains present in a given sequence. Keep in mind that computational predictions should ideally be complemented by experimental validation to confirm the presence and function of the identified domains.

Figure 3.6: GPT-4 understands the task and suggests tools leading to the correct answers.

3.2.3 Processing files in domain-specific formats

In this section, we examine if GPT-4 can understand and extract information from specialized files in biological domains, including MEME-formatted motif files, FASTQ files storing sequencing results, and VCF files storing gene sequence variations.

Case 1: Predicting ZNF143 binding site with provided motif. We show in Case 2 (Fig. 3.3) of Sec. 3.2.2 that GPT-4 struggles to predict binding sites of under-studied protein ZNF143. Next, as shown in Fig. 3.7, we ask GPT-4 to calculate TF binding when a MEME-formatted motif file with position-dependent letter-probability matrices is given. As MEME-format file is a standardized format to represent motifs discovered in biological sequences, the response may indicate the GPT-4's knowledge of biological domain-specific formatted files.

GPT-4

System message:

You are an expert in genomics and bioinformatics.

Prompt:

Can you predict if ZNF143 can bind to DNA sequence TTCCCACAATGCATCG? You can use the information delimited by <> as a reference. Please list step-by-step how you reach the conclusion.

<

MEME version 4

ALPHABET= ACGT

strands: + -

Background letter frequencies

A 0.25 C 0.25 G 0.25 T 0.25

MOTIF MA0088.2 ZNF143

letter-probability matrix: alength= 4 w= 16 nsites= 2034 E= 0

0.042773 0.250246 0.075221 0.631760

0.587980 0.000000 0.008055 0.403965

0.013019 0.985741 0.000620 0.000620

0.001241 0.998759 0.000000 0.000000

0.000000 1.000000 0.000000 0.000000

0.995668 0.000000 0.002475 0.001856

0.000000 0.551082 0.036659 0.412260

0.740847 0.081808 0.177346 0.000000

0.958537 0.000000 0.040854 0.000610

0.003713 0.000000 0.000619 0.995668

0.035607 0.021726 0.937236 0.005432

0.000000 1.000000 0.000000 0.000000

0.891459 0.090747 0.001779 0.016014

0.137615 0.425608 0.005983 0.430794

0.041599 0.464111 0.005302 0.488989

0.187163 0.010289 0.743753 0.058795

URL http://jaspar.genereg.net/matrix/MA0088.2

>

GPT-4:

To predict if ZNF143 can bind to the DNA sequence TTCCCACAATGCATCG, we can use the position weight matrix (PWM) provided in the reference. Here's a step-by-step guide to reach the conclusion:

Step 1: Extract the position weight matrix (PWM) for ZNF143 from the reference.

The PWM is given as a 16x4 matrix, with each row representing a position in the binding site and each column representing a nucleotide (A, C, G, T): (. . .)

Step 2: Calculate the log-odds score for the given sequence.

To do this, we multiply the probability values in the PWM by the corresponding nucleotide frequency in the given sequence: (. . .)

Step 3: Calculate the total score by multiplying the probabilities. Total Score = (. . .)

Step 4: Compare the total score to a threshold.

There is no defined threshold in the reference, so we cannot make a definite conclusion about whether ZNF143 will bind to the given sequence. However, the total score is very low, which may suggest that the binding is less likely. To make a more accurate prediction, we would need more information, such as experimentally determined binding affinity thresholds, or comparing the score to other known binding sites.

Figure 3.7: GPT-4 understands MEME-formatted files and provides the correct analytic process for predicting the ZNF143 binding site with the provided motif, despite coming up with wrong conclusions.

In this test case, GPT-4 shows a good understanding of information processing from specialized files in biological domains. It is also capable of performing bioinformatic analysis with given tasks and data. It is worth pointing out that, when a MYC motif file is provided for ZNF143 prediction, GPT-4 is able to explicitly identify the mismatch sequences and choose not to perform further calculation. However, it should be noted that GPT-4 often comes up with wrong conclusions despite it provides the correct analytic process.

Case 2: Extracting information from FASTQ and VCF files. FASTQ and VCF files are common file types used in high-throughput biological sequencing, where FASTQ files store both a biological sequence (usually nucleotide sequence) and its corresponding quality scores, and VCF files store gene sequence variations. Here, we investigate if GPT-4 can locate information from these 2 types of files, as shown in Fig. 3.8.

GPT-4

System message:

You are an expert in genomics and bioinformatics.

Prompt:

Can you give me the sequence region with low sequencing quality from the following sequence file:

@SEQ_ID GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTT-TAAAGTCCGTGTGTGATTTTT +

!"*((((***+))%%%++)(%%%%).1***-+*"))**55CCF>>>>>>CCCCCCC65opqrstusssyz{|} ~ ||| z!"#$%%####

GPT-4:

In the given sequence file, the sequence is represented by the second line, and the quality scores are represented by the fourth line:

Sequence:
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTTAAAGTCCGTGTGTGATTTTT

Quality scores:
!"((((+))%%%++)(%%%%).1*-+*"))**55CCF>>>>>>CCCCCCC65opqrstusssyz{|} ~ ||| z!"#$%%####

Quality scores are represented by ASCII characters, with the ASCII code (minus 33) representing the Phred quality score of each base. The higher the score, the lower the probability of an incorrect base call. Low-quality bases typically have Phred scores below 20, corresponding to ASCII characters from '!' to '8'.

Looking at the quality scores, the regions with low sequencing quality are:

1. The beginning of the sequence (first 19 bases):
GATTTGGGGTTCAAAGCAG
!"((((**+))%%%++

2. A short region in the middle of the sequence (4 bases):
ATCA
.1***

3. The end of the sequence (last 13 bases):
GATTTTT
z!"#$%%####

These regions have low-quality scores, and their base calls may not be as reliable as those with higher-quality scores.

Prompt:

Can you tell me the genotype of sample NA00002 at position 1770 at chromosome 7: #CHROM POS ID REF ALT QUAL FILTER INFO FORMAT NA00001 NA00002 NA00003 20 14370 rs6054257 G A 29 PASS NS=3;DP=14;AF=0.5;DB;H2 GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:.,. 12 17330 . T A 3 q10 NS=3;DP=11;AF=0.017 GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3 0/0:41:3 7 1770 rs6040355 A G,T 67 PASS NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2 2/2:35:4 7 1230237 . T . 47 PASS NS=3;DP=13;AA=T GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2 17 1234567 microsat1 GTCT G,GTACT 50 PASS NS=3;DP=9;AA=G GT:GQ:DP 0/1:35:4 0/2:17:2 1/1:40:3

GPT-4:

For sample NA00002 at position 1770 on chromosome 7, the genotype information is as follows:

Genotype: 2|1

This indicates that sample NA00002 has one copy of the second alternate allele (T) and one copy of the first alternate allele (G) at this position. The reference allele is A, but this sample does not have any copies of the reference allele.

Figure 3.8: GPT-4 is able to understand FASTQ files and VCF files.

3.2.4 Pitfalls with biological sequence handling

During the test on GPT-4, we discover two major pitfalls that should be paid attention to when applying GPT-4 to bioinformatic tasks.

1) As mentioned in Sec. 3.2.1, direct biological sequence processing may cause catastrophic errors. However, it is more important to be cautious with 'deceptive' errors, such as the missing or mismatch of nucleotides or amino acids, and counting the wrong bases. Negligence of such errors may lead to ineffective experimental design or incorrect analytic conclusions. We showcase the mistakes (see Fig. 3.9 and 3.10) GPT-4 may make when handling biological sequences below with a simple task of protein translation. We reason that the long and non-space sequences composed of very small vocabulary sizes bring a substantial challenge of tokenization, a standard step for language models, such as GPT-4, to process natural language. As a consequence, GPT-4 cannot properly model the biological sequences as it does for text information.

2) As mentioned in Case 1 (Fig. 3.7) of Sec. 3.2.3, while GPT-4 provides correct instructions for TF binding motif prediction, the calculation often goes wrong. The most common mistakes are:

- Reading DNA sequence wrongly (for eg. Input TTCCCACAATGCATCG → TTCCACAATGCATGAG during calculations.
- Making up PWM probability scores for given positions.
- Retrieving wrong PWM probability scores when looking up weight value for given positions.
- Mixing addition and multiplication when calculating scores for given sequences and thresholds, likely due to mixing of two types of solutions.
- Returning wrong results when doing multiplication or addition over around 10 decimals.

If one reads through the answer provided by GPT-4 without confirming the calculation results, wrong conclusions may be reached. Therefore, we recommend that users be very cautious with the quantitative results given by GPT-4. Following GPT-4 's instructions and then performing the calculations with professional tools is suggested.

GPT-4

System message:

You are an expert in genomics and bioinformatics.

Prompt:

Can you translate this DNA sequence into protein sequence:

```
ATGGCGGCTGGCAAAATACCCGATTGGGTCACCGCTGAACGTTTCGAAGATGTTCTCAAATCGAATGTGGAC
GGATATTCGAAAGTGCGAAATTTCAAAGCGGAAATGGGATCCGCGGCAGGTGACAACTACGCCACTAATATG
TTGCGAGTTAATATCGAAGTGGAGCTGCAGGATGGCACCACCAAAGAGTTGTCATACATGGTCAAGTTGCCA
CGTCAAAGGGAAATCAACAAGGAAATGATGAAGCACAACATACGTTCTCAGCGACAATGTGAACAAGACGAG
CGCCGGCTCTCTTTACAACGCAACAATGCATACTTTTCTTTCGTCTCACCGCAAATCGGTGATCGAGCACCC
TCACCTTCAACTAACTCGAAACTTTTGCCCTCAGAGAACGTCAGACCGCGTTCTTGCTCTCGCTCTCTGCCT
GCTTCGGCTCACAAGTCGTGGAGCGAAGAAACCGCCTCTCCTACCCCGCTCCTCTCGCAGCGCCAAACGACC
GTCCCGGGTAACTGTAACACTGCAATAACGAGTGCAGTGACCTCACTGGCAACTGCCACTGCTACCACAACA
TCAACTTCGTCAGCGGCCCAACTAATTATCGCTGTGCCAGCTGTAAATAATACAGCAGCACTGACCGTTTGC
AACAACAATAATGCACGTAAAGAAGAATCAAAACAAAAGCAGAAGTCGATTTCGACTGTGCAGACTGGCATG
GATCGCTACATCCAAATCAAGAGAAAGCTCAGCCCTCAAAACAATAAGGCAGGTAATCAACCCAAAATCAAT
CGAACCAACAACGGCAATGAAAACTCTGCAGTAAATAATTCAAACCGATATGCTATCTTGGCTGATTCTGCG
ACCGAACAACCCAACGAAAAAACGGTAGGGGAACCAAAAAAGACCAGGCCTCCACCAATTTTCATACGAGAA
CAAAGTACAAATGCACTTGTAAATAAACTCGTTGATTTGATTGGTGACAGCAAATTCCACATTATCCCACTT
AAAAAAGGAAATATTCATGAAATAAAACTACAGATCCAAACAGAAGCAGACCACCGTATAGTGACTAAATAC
CTAAATGATGCTGGTAAAAACTACTACACATACCAATTAAAAAGTTGCAAAGGGCTACAGGTAGTACTTAAG
GGCATTGAAGCAACAGTGACACCAGCTGAGATAATTGAGGCTCTGAAGGCCAAAAACTTTTCTGCAAAGACA
GCTATTAATATTTTAAACAAAGACAAAGTTCCGCAGCCACTATTCAAAATAGAACTCGAACCAGAGCTCCAG
GCACTAAAGAAAAACGAAGTGCACCCAATATACAATTTACAGTACTTGCTACATCGGAGGATCACCGTGGAG
GAGCCGCACAAACGTATCAATCCAGTTCAATGTACTAATTGCCAAGAATACGGCCACACCAAGGCATACTGC
ACCCTTAAGTCCGTATGTGTTGTCTGTAGCGAACCTCATACTACCGCAAACTGCCCCAAAAACAAGGACGAT
AAGTCTGTGAAGAAATGCAGTAACTGCGGGGAAAAACATACTGCAAACTACAGAGGCTGTGTGGTGTACAAA
GAATTGAAGAGCCGCCTAAACAAACGTATTGCCACAGCACATACATACAACAAAGTCAATTTCTACTCTCCG
CAACCGATTTTTCAACCACCCCTAACTGTCCCAAGCACTACTCCAACAATTTCTTTCGCTAGCGCCCTAAAA
TCCGGACTAGAAGTGCCCGCCCCACCGACAAGAACTGCTCATTCCGAACATACACCGACAAACATCCAACAA
ACACAACAAAGTGGCATCGAAGCTATGATGCTATCCCTACAGCAAAGCATGAAAGACTTTATGACGTTCATG
CAAAATACTTTGCAAGAGCTCATGAAAAACCAAAATATCCTGATTCAACTTCTTGTATCTTCAAAATCCCCA
```

GPT-4:

Here is the protein sequence translated from the given DNA sequence:

```
MAPGAKIPDWVTLENFEDVSKSMSDGIRKVASFKRNMDSRAVDNYATNIYVASENYSEWAWQDGTQQSFVYH
GVQVATVKGNQKEDEATYFSSDNVKQTASRLYNTAQTAYFLFFVSTQIGDREHPHFTNTRNLCKFAPRNVSR
RSLASASLPPASRPQDRPVGLNTANRIECSLDHLPWQCTATYQTNSFVSGPQLNYRAVQVKITQACDLSYAN
NNNATVKKEIKTKSEDFDLCQDWMAIDYIKNKKLAPKTTKGAIQPKIQNRNNTANENLAVNYIQNPIYLYLD
SADRNPTNKKGWGNPKKDPGSPAIFYTEKQVKYNACTVKNRFDLDWDTQFPPIIPCKKGNYHEIKKLTQGAC
CTKGIENANVDTSLEIDNELEKAQYFKKDSLNNFTKDKVRRPYIKNELEPRAHRKEERVPYNITLPVLCIRE
DHRWREHRNRISNSNVNYCQETRPCGTITACQNTAKTDDSKSCEERASAGKNTAQTYREAVGCVYKEIEEPA
KQRYCPAHYITNQVIYRYPAAPIFFNTPPPNTVSKHLTSNFFRAAPKIGLEVPPTDKEACIPETYDQPNTST
QQTQWSIEAYDLSPQSMKDFMTFMQYTFQELMKTQYPLFSFVSSQNP
```

Figure 3.9: GPT-4 fails to translate the given DNA sequence into a protein sequence, returning a protein sequence of 623 aa while 648 aa is expected for a given DNA sequence of 1944 nt.

GPT-4

Prompt:

This is not correct. Can you tell how many bases in the original DNA sequences and how many amino acids in your translated sequence?

GPT-4:

I apologize for my earlier response. Let me provide the correct information.

The given DNA sequence has 1395 bases.

. . . The translated protein sequence has 464 amino acids. This is also wrong. It "assumes" a stop codon exists in the sequence.

Prompt:

The original DNA sequence has 1944 bases, do you know how many amino acids it can be translated into? We know there is no stop codon in the current frame.

GPT-4:

My apologies for the confusion earlier. Given that the original DNA sequence has 1944 bases and there is no stop codon in the current frame, it can be translated into a protein sequence containing $1944/3 = 648$ amino acids.

Prompt:

Then, please translate the DNA sequence again, keep in mind that the translated sequence should have 648 amino acids and contain no STOP codon.

GPT-4:

I apologize for the earlier mistakes. Here's the correct protein translation of the 1944 base DNA sequence, containing 648 amino acids and no stop codon:

```
MAWAKYPDWVTLNFRDVFQNRMVTRDFKVRKFQAEWIRRAVTLTATNMFCRNNIESWALDGHHQEVVYMGQV
ATVKGEQQEDEATHVLRDNVETKTEARSSLLQNATTFSLFRLRSTQNVDRTLPFTNTRNLKLFAPRENVRRL
ALALSALAFGSSKVRERKPRLLPYPCRSLRAAKDRVRGLVNTCAGNNGSASEDSLGNCATCYTNNIQFVASP
QLYYRAVQCKNNNATVKKEIKTKSEDFDLCQDWMAIDYIKNKKLAPKTTKGAIQPKIQNRNNTANENLAVNY
IQNPIYLYLDSADRNPTNKKGWGNPKKDPGSPAIFYTEKQVKYNACTVKNRFDLDWDTQFPPIIPCKKGNYH
EIKKLTQGACCTKGIENANVDTSLEIDNELEKAQYFKKDSLNNFTKDKVRRPYIKNELEPRAHRKEERVPYN
ITLPVLCIREDHRWREHRNRISNSNVNYCQETRPCGTITACQNTAKTDDSKSCEERASAGKNTAQTYREAVG
CVYKEIEEPAKQRYCPAHYITNQVIYRYPAAPIFFNTPPPNTVSKHLTSNFFRAAPKIGLEVPPTDKEACIP
ETYDQPNTSTQQTQWSIEAYDLSPQSMKDFMTFMQYTFQELMKTQYPLFSFVSSQNP
```

Figure 3.10: Neither the length nor the sequence is correct when GPT-4 is asked to translate the given DNA sequence into a protein sequence again.

3.3 Reasoning with built-in biological knowledge

In this section, we assess GPT-4's understanding and reasoning capabilities in relation to built-in biological knowledge across various topics. We present several examples here, including protein-protein interaction (PPI), signaling pathways, and evolutionary concepts. First, we evaluate a benchmark dataset on the PPI task and further explore how GPT-4 predicts PPI using its built-in knowledge and understanding of protein functions and signaling pathways. Next, we task GPT-4 with predicting the pathways and outcomes of gene regulation. GPT-4 demonstrates an impressive ability to analyze evidence and make predictions based on its understanding of signaling pathways. Finally, we test GPT-4's knowledge of the evolutionary concept of "speciation". Overall, we believe that GPT-4 possesses a broad and comprehensive understanding of biology and can perform simple reasoning tasks when given specific objectives.

3.3.1 Predicting protein-protein interactions (PPI)

In the protein-protein interaction task, we ask specifically GPT-4 to make predictions considering protein features such as functions. GPT-4 is able to identify the correct protein functions and the pathways they are involved in, subsequently making accurate predictions about whether two proteins are likely to interact using its built-in knowledge.

GPT-4

System message:

You are a protein assistant that helps predict the protein-protein interaction given the proteins. The protein could be described either names or amino acid sequences. Users will try their best to provide more information about the protein. Please think it step by step by taking into consideration the typical factors for protein activity, such as protein functions, subcellular localization, and so on, for the target proteins. The final conclusion shall be drawn from the clues you mentioned. Please answer "Yes" or "No" after detailed explanation.

Prompt:

Thg1 and HSE1

GPT-4:

Thg1 (tRNA-histidine guanylyltransferase 1) is an enzyme involved in the post-transcriptional modification of tRNA molecules, specifically adding a guanosine nucleotide to the 5' end of tRNA-His. Thg1 is mainly localized in the cytoplasm and nucleus.

HSE1 (Heat Shock Element 1) is a transcription factor that binds to heat shock elements in the promoter regions of heat shock protein genes and regulates their expression during heat shock or other stress conditions. HSE1 is mainly localized in the nucleus.

Given that Thg1 and HSE1 have different functions, with Thg1 involved in tRNA modification and HSE1 involved in the regulation of stress response genes, it is unlikely that they directly interact with each other. Additionally, their main subcellular localizations are different (though Thg1 is also present in the nucleus, its primary function is in the cytoplasm), further reducing the likelihood of their interaction.

Therefore, based on the information provided, it is likely that Thg1 and HSE1 do not directly interact with each other. So, the answer is No.

Figure 3.11: Case study shows that GPT-4 can make a slightly better prediction after thinking in a step-by-step way by refining the prompt.

Additionally, we include two more examples related to PPIs in the well-studied MAPK signaling pathway – one using real protein names and the other using made-up protein names. GPT-4 successfully predicts the likelihood of PPI by analyzing protein functions and their roles within the pathway.

GPT-4

Prompt:

Can MEK1 and MAPK interact with each other?

GPT-4:

Yes, MEK1 (Mitogen-activated protein kinase kinase 1, also known as MAP2K1) and MAPK (Mitogen-activated protein kinases) can interact with each other. MEK1 is an upstream kinase in the MAPK signaling pathway, and its main function is to phosphorylate and activate MAPKs.

MAPKs are a family of serine/threonine kinases that regulate various cellular processes, such as cell proliferation, differentiation, and apoptosis. There are three major subfamilies of MAPKs: extracellular signal-regulated kinases (ERKs), c-Jun N-terminal kinases (JNKs), and p38 MAPKs.

MEK1 is specifically involved in the ERK signaling pathway, where it phosphorylates and activates ERK1 (MAPK3) and ERK2 (MAPK1). This activation of ERKs by MEK1 is a key step in the transmission of signals from cell surface receptors to the nucleus, leading to specific cellular responses.

In summary, MEK1 and MAPKs, especially ERK1 and ERK2, interact with each other in the context of the MAPK signaling pathway, where MEK1 phosphorylates and activates ERKs.

Prompt:

AI4S1 is a newly identified upstream kinase in the MAPK signaling pathway in elephants, and its main function is to phosphorylate and activate MSRs. MSRs are a family of serine/threonine kinases in elephants that regulate various cellular processes, such as cell proliferation, differentiation, and apoptosis. Scientists also find that MSR1 can phosphorylate the 40S ribosomal protein S6 kinase. Please tell me if AI4S1 can interact with MSR1 and why?

GPT-4:

As an expert in protein biology, I can say that it is highly likely that AI4S1 can interact with MSR1 in the MAPK signaling pathway in elephants. The main evidence supporting this interaction is that AI4S1 is an upstream kinase in the pathway, and its primary function is to phosphorylate and activate MSRs, including MSR1. By phosphorylating MSR1, AI4S1 can modulate its activity, enabling MSR1 to phosphorylate its downstream targets, such as the 40S ribosomal protein S6 kinase.
The interaction between AI4S1 and MSR1 can be further supported by their functional relationship in regulating cellular processes, such as cell proliferation, differentiation, and apoptosis. Additionally, the conservation of the MAPK signaling pathway across various species suggests that AI4S1, as an upstream kinase, can interact with MSR1 to maintain the proper functioning of the pathway.
However, it's essential to note that this conclusion is based on the information provided and general knowledge of protein biology. To definitively confirm the interaction between AI4S1 and MSR1, experimental approaches such as co-immunoprecipitation, yeast two-hybrid assays, or other protein-protein interaction assays should be carried out. Additionally, further studies on the newly identified AI4S1 kinase can provide more insights into its specific role and interactions within the MAPK signaling pathway in elephants.

Figure 3.12: GPT-4 can reason from the functions of these made-up proteins and their role in MAPK signaling pathway and predict they are highly likely to interact with each other. In the figure, AI4S1 and MSR1 are made-up protein names with the same function descriptions as above MEK1 and MAPK.

3.3.2 Understanding gene regulation and signaling pathways

GPT-4 possesses built-in knowledge about signaling pathways and can reason with it. Inspired by the previous section, we further investigate GPT-4's understanding of cell signaling pathways and its reasoning ability based on the information provided. We examine the following scenarios: (1) Make up proteins with specific functions and phenotypes given Metformin treatment and ask GPT-4 to reason the mode of action; (2) Create an ectopic gene expression case with slightly altered conditions and ask GPT-4 to provide the most plausible explanations; (3) With a made-up pathway and perturbation results, we ask GPT-4 to explain the cause of the phenotypes; (4) With made-up phenotypes resulting from the perturbation of real proteins, we ask GPT-4 to determine the possible reasons. GPT-4 exhibits strong performance in all cases, demonstrating its impressive understanding of signaling pathways and cell biology.

GPT-4

System message:

You are an expert in cell biology.

Prompt:

We identified a new protein named PRO1. We found that the knock-down of PRO1 lead to decreased response rate of metformin. We know PRO1 is ***an ATPase***. Can you infer the pathway PRO1 is involved in? Please give me one most possible guess.

GPT-4:

Given that PRO1 is an ATPase and its knockdown leads to a decreased response of metformin, the most plausible pathway PRO1 could be involved in is the AMP-activated protein kinase (AMPK) signaling pathway.

AMPK is a key cellular energy sensor that plays a crucial role in regulating metabolism. Metformin's primary mechanism of action involves the activation of AMPK, which promotes catabolic processes that generate ATP, such as glucose uptake and fatty acid oxidation, while inhibiting anabolic processes that consume ATP, such as gluconeogenesis and lipid synthesis.

As an ATPase, PRO1 may play a role in regulating cellular energy homeostasis, which could directly or indirectly influence the activation and function of AMPK. Reduced PRO1 expression might lead to impaired AMPK signaling, subsequently diminishing the cellular response to metformin.

Further studies, such as exploring the substrate specificity of PRO1, examining its subcellular localization, and assessing the effect of its overexpression on metformin response, would help to better understand the function of PRO1 and its role in the AMPK signaling pathway. Comparing the transcriptomic and proteomic profiles of cells with and without PRO1 knockdown could also provide insights into the pathways and processes influenced by PRO1.

Prompt:

We identified a new protein named PRO1. We found that knock-down of PRO1 lead to decreased response rate of metformin. We know PRO1 is ***a transporter***. Can you infer the pathway PRO1 is involved in? Please give me one most possible guess.

GPT-4:

Given that PRO1 is a transporter and its knockdown leads to a decreased response of metformin, the most plausible pathway that PRO1 could be involved in is the cellular transport of metformin.

Metformin is a hydrophilic, organic cation, and its cellular uptake and efflux are mediated by various transporters. One possible role of PRO1 as a transporter could be in the facilitation of metformin transport across the plasma membrane, either as an uptake or efflux transporter. Reduced PRO1 expression might impair the cellular transport of metformin, leading to decreased intracellular metformin concentrations and, subsequently, a diminished cellular response to the drug.

Some known transporters involved in metformin uptake include organic cation transporter 1 (OCT1), organic cation transporter 2 (OCT2), and plasma membrane monoamine transporter (PMAT). Multidrug and toxin extrusion proteins (MATE1 and MATE2-K) are known to be involved in metformin efflux. PRO1 could potentially be a novel transporter with a similar function in metformin transport.

Further studies, such as exploring the substrate specificity of PRO1, examining its subcellular localization, and assessing the effect of its overexpression on metformin response, would help to better understand the function of PRO1 and its role in metformin transport. Comparing the transcriptomic and proteomic profiles of cells with and without PRO1 knockdown could also provide insights into the pathways and processes influenced by PRO1.

Figure 3.13: GPT-4 exhibits strong performance when asked to reason the mode of action for made-up proteins with specific functions and phenotypes given Metformin treatment.

GPT-4

System message:

You are an expert in cell biology.

Prompt:

Gene EV is a newly identified gene exclusively expressed in human lungs. Expression of Gene EV is found in the brain and the lung of an individual during a medical test. However, EV is not found in other body parts of the individual. In addition, no brain-expressed Gene EV is found in other individuals accepting the same test at the same time. What could this mean?

GPT-4:

The observation of Gene EV expression in the brain of an individual, while it is exclusively expressed in human lungs and not detected in the brain of other tested individuals, could indicate a few possibilities:

1. Individual variation: The expression of Gene EV in the brain of this specific individual could be a result of individual genetic variation, such as a mutation, that leads to ectopic expression of the gene in the brain. This variation might be unique to this person and not present in the other tested individuals.
2. Disease or pathological condition: The presence of Gene EV in the brain of this individual might be associated with a disease or pathological condition that is not present in the other tested individuals. The altered expression pattern of Gene EV could be a consequence or a contributing factor of the condition.
3. Sample contamination: It is also possible that there was a contamination of the brain sample with lung tissue during the collection process or in the laboratory, leading to the detection of Gene EV expression in the brain sample.
4. Technical error: The detection of Gene EV in the brain sample could be due to a technical error, such as a false-positive result, during the clinical test. This might be caused by issues with the testing method, reagents, or experimental conditions.

To further investigate this observation, additional studies could be performed, such as repeating the medical test on the same individual and other individuals to confirm the results, examining the genetic background of the individual with brain-expressed Gene EV, and exploring potential links between the expression of Gene EV in the brain and any clinical symptoms or disease conditions.

Prompt:

Gene EV is a newly identified gene exclusively expressed in human lungs. Expression of Gene EV is found in the brain and the lung of an individual during a medical test. However, EV is not found in other body parts of the individual. In addition, no brain-expressed Gene EV is found in other individuals accepting the same test at the same time. ***This individual had a cancer diagnosis 20 years ago***. What could this mean?

GPT-4:

Given that Gene EV is exclusively expressed in human lungs and was detected in the brain of an individual who had a cancer diagnosis 20 years ago, there are a few possible explanations for this observation:

1. Metastasis from lung cancer: If the cancer diagnosis was specifically lung cancer, the presence of Gene EV in the brain could be an indication of metastatic lung cancer. Cancer cells from the primary lung tumor may have spread to the brain over time, resulting in the expression of lung-specific genes, such as Gene EV, in brain tissue. This would explain why Gene EV is not detected in the brain of other individuals who participated in the same test.
2. Metastasis from another cancer type: If the cancer diagnosis was not lung cancer, it is possible that the primary cancer has metastasized to the lung, resulting in the expression of Gene EV in the lung tissue. The metastatic cancer cells might have also spread to the brain, leading to the ectopic expression of Gene EV in the brain.
3. Paraneoplastic syndrome: The ectopic expression of Gene EV in the brain might be associated with a paraneoplastic syndrome, which is a rare group of symptoms triggered by an abnormal immune response to a cancerous tumor. In this case, the cancer could indirectly cause changes in Gene EV expression in the brain, possibly due to the production of antibodies or other immune-related factors.
4. Coincidental finding: While less likely, it is possible that the detection of Gene EV in the brain is unrelated to the individual's cancer diagnosis and might be due to individual genetic variation or other factors mentioned in the previous answer (sample contamination or technical error).

To further investigate the presence of Gene EV in the brain of this individual, additional tests and imaging studies could be performed to assess the presence of metastatic cancer in the brain or other related pathological conditions. Evaluating the individual's immune response, such as the presence of specific antibodies, could also provide insights into potential paraneoplastic syndromes. Repeating the medical test and analyzing more samples from patients with cancer diagnoses and healthy controls would help to better understand the association between Gene EV expression in the brain and cancer.

Figure 3.14: GPT-4 exhibits strong performance when asked to provide the most plausible explanations for a created ectopic gene expression case with slightly altered conditions.

GPT-4

System message:

You are an expert in cell biology.

Prompt:

I have a pathway: Protein A activates Protein B, Protein B activates Protein C and represses Protein D. Phenotype X is related to this pathway. Inhibition of A can revert the phenotype, while inhibition of C cannot. What is the most possible explanation?

GPT-4:

Based on the information provided, the most likely explanation for the observation that inhibition of Protein A can revert Phenotype X, while inhibition of Protein C cannot, is that Phenotype X is primarily driven by the dysregulation of Protein D, which is repressed by Protein B.

In the described pathway, Protein A activates Protein B, which in turn activates Protein C and represses Protein D. When Protein A is inhibited, the activation of Protein B is reduced, leading to a decreased repression of Protein D. This suggests that the reversion of Phenotype X is related to the restoration of Protein D levels or activity.

On the other hand, the lack of response to Protein C inhibition indicates that the activation of Protein C, which is downstream of Protein B, might not be the primary driver of Phenotype X, or at least not the sole driver. It is also possible that Protein C has a compensatory mechanism or other downstream effectors that maintain the pathological signaling when Protein C is inhibited.

To further investigate the molecular mechanisms underlying Phenotype X and the response to Protein A and Protein C inhibitors, additional research is needed. This may involve examining the role of Protein D in the development and progression of Phenotype X, as well as investigating other signaling pathways and cellular processes interconnected with the described pathway. Identifying potential biomarkers and therapeutic targets related to Protein D and its regulation could be crucial for the development of more effective treatments or interventions for Phenotype X.

Figure 3.15: GPT-4 exhibits strong performance when asked to explain the cause of the phenotypes with a made-up pathway and perturbation results.

GPT-4

System message:

You are an expert in cell biology.

Prompt:

I have a mutation on the genome of cultured cells, which leads to cell death. When TIM channel is blocked, the phenotype is reverted. When ClpP is overexpressed, the phenotype is also reverted. Can you give me a few candidates where the mutation may happen?

GPT-4:

The mutation that leads to cell death could occur in one or more genes encoding proteins involved in mitochondrial function, protein import, or protein quality control. Based on the information provided, blocking the TIM channel or overexpressing ClpP can revert the cell death phenotype. Here are a few candidate genes where the mutation might occur:

1. Nuclear-encoded mitochondrial proteins: The mutation could be in a gene encoding a nuclear-encoded mitochondrial protein that is imported through the TIM channel. This protein might have a critical role in mitochondrial function, and its dysfunction or misfolding could lead to cell death. Blocking the TIM channel could prevent the import of the detrimental or misfolded protein, while overexpressing ClpP could help degrade the aberrant protein, thus improving mitochondrial function and preventing cell death.
2. Components of the TIM machinery: The mutation could be in a gene encoding a component of the TIM machinery itself, such as TIM23, TIM50, or TIM44. These proteins are involved in the recognition, translocation, and insertion of precursor proteins into the mitochondria. A mutation in one of these genes could lead to impaired protein import and mitochondrial dysfunction. Blocking the TIM channel or overexpressing ClpP might compensate for the impaired import process, improving mitochondrial function and preventing cell death.
3. Mitochondrial chaperone proteins: The mutation could be in a gene encoding a mitochondrial chaperone protein, such as HSP60 or mtHSP70. These chaperone proteins are involved in the folding and assembly of mitochondrial proteins. A mutation in one of these genes could lead to misfolding or aggregation of proteins within the mitochondria, impairing mitochondrial function. Blocking the TIM channel could prevent the import of misfolded proteins, while overexpressing ClpP might assist in degrading the misfolded or aggregated proteins, promoting proper protein folding and preventing cell death.
4. Mitochondrial quality control proteins: The mutation could be in a gene encoding a mitochondrial quality control protein other than ClpP, such as LONP1 or YME1L1. These proteins are involved in the degradation of damaged or misfolded proteins in the mitochondria. A mutation in one of these genes could lead to the accumulation of detrimental proteins, impairing mitochondrial function. Blocking the TIM channel could prevent the import of misfolded proteins, while overexpressing ClpP could compensate for the reduced protease activity, promoting proper protein degradation and preventing cell death.

Further research and analysis, such as whole-exome or whole-genome sequencing, gene expression profiling, and functional studies, are needed to pinpoint the exact gene(s) where the mutation occurs and to understand the molecular mechanisms underlying the observed phenotype. Identifying the mutated gene(s) and the affected cellular processes could lead to the development of targeted therapies or interventions to treat or prevent the condition associated with the mutation.

Figure 3.16: GPT-4 exhibits strong performance when asked to determine the possible reasons for made-up phenotypes resulting from the perturbation of real proteins.

3.3.3 Understanding concepts of evolution

GPT-4 demonstrates a solid understanding of key information in evolutionary biology. We use "speciation" as an example to showcase GPT-4's built-in knowledge and reasoning abilities in the field of evolutionary biology. We test GPT-4's capacity to explain scientific terms and related research methods. Furthermore, we present it with a classic evolutionary case: the selection of peppered moth color, to distinguish between selection and speciation and identify the key factor that defines speciation. We also provide hypothetical evidence, such as strong mating preferences, for GPT-4 to reason what would occur in the given scenario. As a result, GPT-4 displays a good understanding of evolutionary biology terms and the ability to reason and predict likely outcomes when presented with a hypothetical case.

GPT-4

System message:

You are an expert in Evolutionary biology.

Prompt:

Describe the factors that contribute to the formation of new species

GPT-4:

The formation of new species, also known as speciation, occurs when populations of a single species evolve over time and gradually develop into separate, distinct species. Several factors contribute to this process, including:

1. Genetic variation: Genetic variation within a population is a driving force for speciation. It arises through mutations, gene flow, and genetic recombination during sexual reproduction. This variation allows for natural selection to act upon and drive the evolution of new traits and adaptations.
2. Natural selection: Natural selection is the process by which individuals with advantageous traits are more likely to survive and reproduce, passing on these traits to the next generation. Over time, this leads to the accumulation of these advantageous traits within a population, causing it to evolve and potentially form a new species.
3. Geographic isolation: Geographic isolation occurs when a population is separated by physical barriers, such as mountains, rivers, or deserts. This separation prevents gene flow between the isolated populations, allowing them to evolve independently. Over time, genetic differences accumulate between the isolated populations, eventually leading to the formation of new species.
4. Reproductive isolation: Reproductive isolation refers to any mechanism that prevents interbreeding between different populations or species. This can occur through prezygotic barriers (before fertilization) such as differences in mating behavior, timing of reproduction, or physical incompatibility, or through postzygotic barriers (after fertilization) like hybrid inviability or sterility. Reproductive isolation is essential for maintaining the genetic distinctiveness of a species and preventing gene flow between different species.
5. Genetic drift: Genetic drift is the random change in allele frequencies within a population due to chance events. It can lead to the fixation or loss of certain alleles within a population, which can contribute to the formation of new species, especially in small, isolated populations.
6. Founder effect and bottleneck events: The founder effect occurs when a small group of individuals from a larger population establishes a new, isolated population. Bottleneck events, on the other hand, occur when a population suddenly reduces in size due to factors like natural disasters or habitat loss. Both of these events can lead to a loss of genetic variation and increase the chances of genetic drift and rapid evolution, contributing to the formation of new species.
7. Adaptive radiation: Adaptive radiation is the rapid diversification of a single ancestral species into multiple new species, each occupying a different ecological niche. This process often occurs in environments with many unoccupied niches, such as islands or after mass extinctions. As populations adapt to different niches, they accumulate genetic differences and may eventually evolve into distinct species.

These factors, individually or in combination, can contribute to the formation of new species over time. The process of speciation is complex and can occur at different rates depending on the specific factors involved and the environmental conditions.

GPT-4 (continue)

Prompt:

Will you consider the peppered moth evolution a speciation process?

GPT-4:

The peppered moth (Biston betularia) evolution is a classic example of natural selection in action, but it is not considered a speciation process. The peppered moth story demonstrates how a population can rapidly adapt to environmental changes, but it does not involve the formation of a new species.

...

While the peppered moth case illustrates how natural selection can drive rapid changes in a population's traits, it does not involve the formation of new species. Both the light and dark-colored moths are still the same species, Biston betularia, and they can interbreed without any reproductive isolation. The peppered moth evolution is an example of phenotypic plasticity and adaptation within a species, rather than a speciation event.

Prompt:

Scientists find that the dark form moth shows significant preference to mate with dark form. What will likely happen in the long run?

GPT-4:

If the dark form of the peppered moth (Biston betularia) shows a significant preference for mating with other dark form individuals, this could be an example of assortative mating or positive sexual selection. Assortative mating is a non-random mating pattern where individuals with similar traits are more likely to mate with each other, potentially leading to reproductive isolation over time.

In the long run, if the preference for mating with similar phenotypes continues and is strong enough, it could promote genetic divergence between the dark and light forms of the moths. As the two forms preferentially mate within their own groups, gene flow between the groups will decrease, and genetic differences may accumulate over time.

However, for speciation to occur, reproductive isolation must be maintained even when both forms coexist in the same environment. This could happen through the evolution of prezygotic isolation mechanisms, such as differences in mating signals, or postzygotic mechanisms like reduced fitness in hybrid offspring.

It's important to note that assortative mating alone does not guarantee speciation, and several factors can influence the outcome. For example, the strength of the mating preference, the degree of gene flow between the groups, and the influence of natural selection on the traits involved all play a role in determining whether speciation will occur.

In summary, if the dark form of the peppered moth consistently shows a strong preference for mating with other dark form individuals, it could potentially lead to genetic divergence and eventually speciation in the long run. However, this outcome would depend on the strength of the mating preference, the maintenance of reproductive isolation, and other evolutionary factors that influence the process of speciation.

Here GPT-4 understands strong mating preference may lead to reproductive isolation, and eventually speciation.

Figure 3.17: GPT-4 demonstrates a good understanding of key information in evolutionary biology.

3.4 Designing biomolecules and bio-experiments

Here we explore whether GPT-4 can serve as a science assistant to help with biology design. We show two design tasks with GPT-4: DNA sequence design and automation of experiments. We find that GPT-4 can be used to estimate DNA properties such as melting temperatures (Fig. 3.18) and design DNA sequences for molecular computation (Fig. 3.20). GPT-4 is also capable of translating experimental protocols to codes for an automatic pipetting robot (Fig. 3.21). In short, we find that GPT-4 can be a useful scientific assistant for design tasks in biology. However, we also caution that GPT-4's answers are sometimes inaccurate while processing Arabic numbers (Fig. 3.19). It requires further improvement to become a reliable science assistant.

3.4.1 Designing DNA sequences for biological tasks

GPT-4 can estimate properties like melting temperature for DNA. DNA primers are short DNA sequences that are commonly used for molecular amplification. Estimating the melting temperature is an important task to ensure the success of DNA amplification. We ask GPT-4 to estimate the melting temperature of DNA primers. GPT-4 successfully calculates the melting temperatures with detailed formulas about

different calculation methods, which is useful for scientists to understand the results and their rationale.

GPT-4

Prompt: (Case 1)

Can you calculate the melting temperature of a primer with sequence "AAGGCAAGTTGTTACCAGCA"?

GPT-4:

There are different methods and formulas to calculate the melting temperature (Tm) of a primer, which is the temperature at which half of the primer molecules are annealed to their complementary strands and half are dissociated. One of the simplest and most commonly used methods is the Wallace rule, which estimates the Tm based on the number and type of nucleotides in the primer. According to this rule, the Tm is:

Tm = 2°C × (number of A and T bases) + 4°C × (number of G and C bases)

For the primer with sequence AAGGCAAGTTGTTACCAGCA, the number of A and T bases is 10 and the number of G and C bases is 10, so the Tm is:

Tm = 2°C × 10 + 4°C × 10
Tm = 20°C + 40°C
Tm = 60°C

However, this method is only approximate and does not account for factors such as primer length, salt concentration, GC content, and mismatches that can affect the stability of the primer-DNA duplex. Therefore, more accurate methods that use thermodynamic parameters and empirical corrections are often preferred, especially for primers longer than 18 nucleotides or with high or low GC content. One of these methods is the nearest-neighbor method, which calculates the Tm based on the sum of the free energy changes of each pair of adjacent nucleotides in the primer and the complementary strand, as well as the initiation and termination effects. The free energy changes depending on the type of base pair, the salt concentration, and the concentration of the primer and the DNA template. A simplified version of this method, using the parameters from SantaLucia (1998), is:

Tm = (1000 × deltaH) / (deltaS + R × ln(C/4)) − 273.15

...

Figure 3.18: GPT-4 can perform estimate melting temperature for DNA.

GPT-4 can provide "technically" correct answers. However, the answers might be unsatisfactory because it might not consider the context of an application. We ask GPT-4 to design DNA sequences for molecular computational circuits. The designed sequences are "technically correct" in the sense that sequences have hamming distances larger than one are orthogonal. However, such sequences are expected to perform poorly because it does not consider DNA strands are flexible, and hamming distances are not good metrics for designing orthogonal sequences. An expert would at least use edit distances instead and other more complicated rules are also usually included. Therefore, we caution that though GPT-4 can be a useful science assistant, it has the risk of giving you answers that do not necessarily fit one's application. Furthermore, interestingly, we find that GPT-4 can get confused with Arabic numerals (see Fig. 3.19), but it can understand numbers expressed in English texts (see Fig. 3.20).

GPT-4

Prompt: (Case 2)

Can you design ten orthogonal sequences with a length of 20 nucleotides for a DNA strand displacement system?

GPT-4:

There are different ways to define and measure orthogonality for DNA sequences, but one possible approach is to use the Hamming distance, which counts the number of mismatches between two sequences of equal length. A higher Hamming distance means less similarity and less potential for cross-hybridization or interference. For example, the Hamming distance between ACGT and TGCA is 4, while the Hamming distance between ACGT and ACGA is 1.

One way to generate orthogonal sequences with a given Hamming distance is to use a Gray code, which is a way of encoding binary numbers such that adjacent numbers differ by only one bit. For example, the 4-bit Gray code is:

0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000

We can use this code to generate 16 orthogonal sequences with a Hamming distance of 4 by mapping 0 to A and 1 to T, and then adding a prefix and a suffix of two random nucleotides to each sequence. For example:

Prefix: GA Suffix: TC

```
GAATATATTC
GAATATATAC
GAATATTTAC
GAATATTTTC
GAATTATTTC
GAATTATTTA
GAATTATAAT
GAATTATATA
GATTTATATA
GATTTATAAT
GATTTTTAAT
GATTTTTATA
GATAATATA
GATAATATT
GATAATTTT
GATAATTTA
```

These sequences have a length of 20 nucleotides and a Hamming distance of 4 between any pair of them. However, they are not the only possible set of orthogonal sequences, and other methods or criteria may be used to generate different sets.

Figure 3.19: GPT-4 can provide "technically" correct but unsatisfied answers because it might not consider the context of the application.

GPT-4

Prompt:

Can you design ten orthogonal sequences with a length of twenty nucleotides for a DNA strand displacement system?

GPT-4:

There are different ways to define and measure orthogonality for DNA sequences, but one possible approach is to use the Hamming distance, which is the number of mismatches between two sequences of equal length. A higher Hamming distance means less similarity and less potential for unwanted hybridization or interference. One possible criterion for orthogonality is to require that the Hamming distance between any pair of sequences is at least half of the sequence length or 10 in this case. This is a fairly strict condition that may not be necessary for some applications, but it provides a high level of confidence that the sequences are distinct and unlikely to interact with each other.
Using this criterion, one possible set of ten orthogonal sequences with a length of twenty nucleotides is:
- S1: `ATCGTACGATCGTACGATCG`
- S2: `TAGCATGCTAGCATGCTAGC`
- S3: `CGATAGCTCGATAGCTCGAT`
- S4: `GCTACGATGCTACGATGCTA`
- S5: `ACGTACGTACGTACGTACGT`
- S6: `TGCACTGCATGCACTGCATG`
- S7: `CTAGCTAGCTAGCTAGCTAG`
- S8: `GATCGATCGATCGATCGATC`
- S9: `CATGCATGCATGCATGCATG`
- S10: `GTACGTACGTACGTACGTAC`

The Hamming distance between any pair of these sequences is 10, which is half of the sequence length and likely sufficient to avoid unwanted interactions in a DNA strand displacement system. However, depending on the specific context and conditions of the system, other factors such as secondary structure, thermodynamics, kinetics, and concentration may also affect the performance and reliability of the sequences. Therefore, this set of sequences is not guaranteed to be optimal or robust for any arbitrary DNA strand displacement system, and may require further testing and optimization.

Figure 3.20: Interestingly, GPT-4 could get confused with Arabic numerals, but it can understand numbers expressed in English texts.

3.4.2 Designing biological experiments

GPT-4 can code for liquid handling robots. We ask GPT-4 to translate an experimental protocol to a Python code that can be used to program a liquid-handling robot. This coding capability is expected to greatly speed up the automation of biology experiments.

GPT-4

Prompt:

write a Python code that can be used for automation robot Opentrons for a PCR reaction.

GPT-4:

Possible Python code for automation robot Opentrons for a PCR reaction:

```
# Import the Opentrons API
from opentrons import protocol_api

# Define the protocol metadata
metadata = {
    `apiLevel': `2.10', # The API version supported by the protocol
    `protocolName': `PCR Reaction', # The name of the protocol
    `author': `Your Name', # The author of the protocol
    `description': `A simple PCR reaction using a 96-well plate and a thermocycler', # The description of
     the protocol
}

# Define the protocol function
def run(protocol: protocol_api.ProtocolContext):
    # Define the labware and modules
    plate = protocol.load_labware('nest_96_wellplate_100ul_pcr_full_skirt', 1) # Load a 96-well PCR plate
     on slot 1
    tiprack = protocol.load_labware('opentrons_96_tiprack_20ul', 2) # Load a 20 uL tip rack on slot 2
    pipette = protocol.load_instrument('p20_single_gen2', 'right', tip_racks=[tiprack])
    # Load a 20 uL single-channel pipette on the right mount
    thermocycler = protocol.load_module('thermocycler') # Load a thermocycler module
    tc_plate = thermocycler.load_labware('nest_96_wellplate_100ul_pcr_full_skirt') # Load a 96-well PCR
    plate on the thermocycler...
    # Define the reagents and samples
    water = plate['A1'] # Assign the well A1 of the plate as the water source
    mastermix = plate['A2'] # Assign the well A2 of the plate as the mastermix source
    samples = plate.rows_by_name()['A'][3:11] # Assign the wells A4 to A11 of the plate as the sample
    sources
    destinations = tc_plate.rows_by_name()['A'][3:11] # Assign the wells A4 to A11 of the thermocycler
    plate as the PCR reaction destinations

    # Transfer the reagents and samples to the thermocycler plate
    pipette.pick_up_tip() # Pick up a new tip
    pipette.transfer(5, water, destinations, new_tip='never') # Transfer 5 uL of water to each
    destination well
    pipette.transfer(15, mastermix, destinations, new_tip='never', mix_after=(3, 10)) # Transfer 15 uL of
     mastermix to each destination well and mix after each transfer
    pipette.drop_tip() # Drop the tip
    for sample, destination in zip(samples, destinations): # Loop over the samples and destinations
        pipette.pick_up_tip() # Pick up a new tip
        pipette.transfer(5, sample, destination, new_tip='never', mix_after=(3, 10)) # Transfer 5 uL of
    sample to the corresponding destination well and mix after each transfer
        pipette.drop_tip() # Drop the tip
```

Figure 3.21: GPT-4 can code for liquid handling robots.

4 Computational Chemistry

4.1 Summary

Computational Chemistry is an interdisciplinary field that utilizes computational methods and techniques to address complex problems in chemistry. For a long time, it has been an indispensable tool in the study of molecular systems, offering insights into atomic-level interactions and guiding experimental efforts. This field involves the development and application of theoretical models, computer simulations, and numerical algorithms to examine the behavior of molecules, atoms, materials, and physical systems. Computational Chemistry plays a critical role in understanding molecular structures, chemical reactions, and physical phenomena at both microscopic and macroscopic levels.

In this chapter, we investigate GPT-4's capabilities across various domains of computational chemistry, including electronic structure methods (Sec. 4.2) and molecular dynamics simulation (Sec. 4.3), and show two practical examples with GPT-4 serving from diverse perspectives (Sec. 4.4). In summary, we observe the following capabilities and contend that GPT-4 is able to assist researchers in computational chemistry in a multitude of ways:[15]

- Literature Review: GPT-4 possesses extensive knowledge of computational chemistry, covering topics such as density functional theory (see Fig. 4.2), Feynman diagrams (see Fig. 4.5), and fundamental concepts in electronic structure theory (see Fig. 4.3-4.4), molecular dynamics simulations (see Fig. 4.18-4.22 and Fig. 4.28), and molecular conformation generation (Sec. 4.3.5). GPT-4 is not only capable of explaining basic concepts (see Fig. 4.2-Fig. 4.4, Fig. 4.20- 4.22), but can also summarize key findings and trends in the field (see Fig. 4.8, 4.29- 4.31).
- Method Selection: GPT-4 is able to recommend suitable computational methods (see Fig. 4.8) and software packages (see Fig. 4.24- 4.27) for specific research problems, taking into account factors such as system size, timescales, and level of theory.
- Simulation Setup: GPT-4 is able to aid in preparing simple molecular-input structures, establishing and suggesting simulation parameters, including specific symmetry, density functional, time step, ensemble, temperature, and pressure control methods, as well as initial configurations (see Fig. 4.7- 4.13, 4.24).
- Code Development: GPT-4 is able to assist with the implementation of novel algorithms or functionality in existing computational chemistry and physics software packages (see Fig. 4.14- 4.16).
- Experimental, Computational, and Theoretical Guidance: As demonstrated by the examples in Sec. 4.4 and Fig. 4.37, GPT-4 is able to assist researchers by providing experimental, computational, and theoretical guidance.

While GPT-4 is a powerful tool to assist research in computational chemistry, we also observe some limitations and several errors. To better leverage GPT-4, we provide several tips for researchers:

- Hallucinations: GPT-4 may occasionally generate incorrect information (see Fig. 4.3). It may struggle with complex logic reasoning (see Fig. 4.4). Researchers need to independently verify and validate outputs and suggestions from GPT-4.
- Raw Atomic Coordinates: GPT-4 is not adept at generating or processing raw atomic coordinates of complex molecules or materials. However, with proper prompts that include molecular formula, name, or other supporting information, GPT-4 may still work for simple systems (see Fig. 4.10- 4.13).
- Precise Computation: GPT-4 is not proficient in precise calculations in our evaluated benchmarks and usually ignores physical priors such as symmetry and equivariance/invariance (see Fig. 4.6 and Table 15). Currently, the quantitative numbers returned by GPT-4 may come from a literature search or few-shot examples. It is better to combine GPT-4 with specifically designed scientific computation packages (e.g., PySCF [78]) or machine learning models, such as Graphormer [104] and DiG [109].
- Hands-on Experience: GPT-4 can only provide guidance and suggestions but cannot directly perform experiments or run simulations (Fig. 4.39). Researchers will need to set up and execute simulations or experiments by themselves or leverage other frameworks based on GPT-4, such as AutoGPT[16], HuggingGPT [76], AutoGen [95] and so on.

[15] In this chapter, we use yellow to indicate incorrect or inaccurate responses from GPT-4.

[16] `https://github.com/Significant-Gravitas/Auto-GPT`

In summary, GPT-4 demonstrates good potential across a range of computational chemistry domains, encompassing electronic structure methods, molecular dynamics simulations, and real-world applications. Although some limitations and inaccuracies exist, by employing appropriate guidance and adjustments, GPT-4 has the potential to evolve into a valuable resource for researchers navigating the dynamic and expanding field of computational chemistry.

4.2 Electronic structure: theories and practices

GPT-4

System message:

You are an AI assistant that helps people find information.

Figure 4.1: System message for Sec. 4.2 if there is no specification in the GPT-4 dialogue.

Fig. 4.1 shows the system message for cases studied in this section.[17]

4.2.1 Understanding of quantum chemistry and physics

We begin with the evaluation of GPT-4's capability to interpret concepts in quantum chemistry and physics. The evaluation covers the commonly employed methodologies in this field, such as density functional theory (DFT) and wave function theory (WFT).

GPT-4

Prompt:

What is density functional theory? What are Kohn-Sham density functional theory and orbital free density functional theory? What are the differences between them?

GPT-4:

Density Functional Theory (DFT) is a computational quantum mechanical modeling method used to investigate the electronic structure of many-body systems, particularly atoms, molecules, and solids. It is based on the idea that the ground-state properties of a system can be determined by its electron density, rather than directly solving the many-electron wave function. DFT is widely used in physics, chemistry, and materials science due to its accuracy and computational efficiency compared to traditional wave function methods.

Kohn-Sham Density Functional Theory (KSDFT) is a specific formulation of DFT introduced by Walter Kohn and Lu J. Sham in 1965. In KSDFT, the complicated many-electron system is mapped to a non-interacting system of single-particle Kohn-Sham orbitals, which generates the same ground-state electron density as the original system. The Kohn-Sham orbitals are determined by solving the Kohn-Sham equations, which include the kinetic energy of non-interacting electrons, the external potential, and the exchange-correlation potential that accounts for the effects of electron-electron interactions.

Orbital-Free Density Functional Theory (OFDFT) is an alternative formulation of DFT that does not rely on the concept of single-particle orbitals. Instead, it aims to express the total energy of a system directly as a function of the electron density. OFDFT includes the kinetic energy functional, which is often more challenging to approximate accurately compared to the Kohn-Sham approach. Despite its computational simplicity, OFDFT is generally less accurate and less widely used than KSDFT.

The main differences between Kohn-Sham Density Functional Theory and Orbital-Free Density Functional Theory are: KSDFT maps the many-electron problem to a non-interacting single-particle system described by Kohn-Sham orbitals, while OFDFT aims to express the total energy directly as a function of the electron density without using orbitals. KSDFT solves the Kohn-Sham equations to determine the ground-state electron density, while OFDFT requires an accurate kinetic energy function, which is often more challenging to approximate. KSDFT is generally more accurate and widely used in various fields, while OFDFT is computationally simpler but less accurate.

Overall, both KSDFT and OFDFT are approaching the broader framework of Density Functional Theory, with different strategies to approximate the ground-state properties of many-electron systems.

Figure 4.2: Conceptual check on density functional theory.

[17]We also tried another system message: "you are an AI expert on quantum chemistry that helps researchers." While the responses do differ, the core content for the cases studied in this section remains analogous.

In the first example (Fig. 4.2), GPT-4 well understands the concepts of density functional theory, Kohn-Sham density functional theory, and orbital-free density functional theory. We further assess the ability of GPT-4 to understand the concepts on the behavior of quantum chemistry calculations changing with size in Fig. 4.3. GPT-4 gives a correct equation of size consistency criteria but a wrong explanation of size extensitivity. Specifically, a size-extensive method does not have to satisfy the relationship of E(A+B) = E(A) + E(B) in this scenario. In addition, the GPT-4 fails to realize that size extensitivity is a necessary condition of size consistency, i.e., size consistency implies size extensitivity. Thus, it makes two logically wrong claims on CIS and MP2. The correct conclusions should be that CIS is neither size consistent nor size extensive and MP2 is both size extensive and size consistent, respectfully. We note that we only consider the common cases in electronic structure in the current discussion and GPT-4 reasonably well-identifies that HF and MP2 are size consistent approaches.

GPT-4

Prompt:

What is size extensivity? What is size consistency? What are the differences between these two concepts? Could you also provide some example electronic structure theories that satisfy and don't satisfy these two concepts, respectively?

GPT-4:

Size extensivity and size consistency are important properties in quantum chemistry, particularly in the context of electronic structure theories. These properties ensure that the calculated energy of a system scales appropriately with the size of the system.
Size Extensivity: A property of a computational method is said to be size extensive if the calculated energy of a system scales linearly with the size of the system when the system is composed of non-interacting subsystems. In other words, if we have two non-interacting systems A and B, a size extensive method should give E(A+B) = E(A) + E(B).
Size Consistency: A property of a computational method is said to be size consistent if the calculated energy of a system remains unchanged when the distance between non-interacting subsystems approaches infinity. In other words, if we have two non-interacting systems A and B, a size consistent method should give E(A+B) = E(A) + E(B) when the distance between A and B is very large.
The main difference between size extensivity and size consistency is that size extensivity refers to the linear scaling of energy with the size of the system, while size consistency refers to the behavior of the energy when the distance between non-interacting subsystems becomes very large.
Examples of electronic structure theories:

1. Hartree-Fock (HF) theory: This method is both size extensive and size consistent. The energy calculated using HF theory scales linearly with the size of the system and remains unchanged when the distance between non-interacting subsystems approaches infinity.

2. Configuration Interaction Singles (CIS) method: This method is size consistent but not size extensive. The energy calculated using CIS remains unchanged when the distance between non-interacting subsystems approaches infinity, but it does not scale linearly with the size of the system.

3. Møller-Plesset Perturbation Theory (MPn, n> 1): These methods are size consistent but not size extensive. The energy calculated using MPn methods remains unchanged when the distance between non-interacting subsystems approaches infinity, but it does not scale linearly with the size of the system.

4. Coupled Cluster (CC) theory: This method is both size extensive and size consistent. The energy calculated using CC theory scales linearly with the size of the system and remains unchanged when the distance between non-interacting subsystems approaches infinity.

5. Density Functional Theory (DFT): DFT methods are generally size consistent, but their size extensivity depends on the choice of the functional. Some functionals are size extensive, while others are not.

Figure 4.3: Limited capability of GPT-4 to comprehend concepts in electronic structure theory. The response contains inaccuracies; for example, a size-extensive method does not necessarily satisfy E(A+B) = E(A) + E(B) in this context.

Although GPT-4 can understand the concepts in quantum chemistry relatively well in the previous two examples, its ability to derive equations according to these concepts needs to be assessed. In Fig. 4.4, GPT-4 provides correct definitions of symmetry and antisymmetry and the Pauli Exclusion Principle. However, it reaches the wrong conclusion that the wavefunction is antisymmetric with the right problem-solving logic but wrong algebraic derivations. It is correct to exchange two particles and check if the new wavefunction changes the sign of the original one, but only the exchange between electrons 1 and 2 is correct. The correct

derivations for the second and third exchanges should be:
Exchange electrons 1 and 3: Exchanged: $|321\rangle - |312\rangle + |231\rangle - |213\rangle + |132\rangle - |123\rangle$
Exchange electrons 2 and 3: Exchanged: $|132\rangle - |231\rangle + |123\rangle - |321\rangle + |213\rangle - |312\rangle$

GPT-4

Prompt:

What is symmetry and antisymmetry in quantum chemistry? What is the antisymmetry principle? What are the antisymmetrization procedure and antisymmetrizer?

GPT-4:

In quantum chemistry, symmetry and antisymmetry refer to the behavior of wave functions under the exchange of particles, specifically electrons. These concepts are crucial for understanding the behavior of electrons in molecules and their interactions.
Symmetry: A wave function is symmetric if it remains unchanged when two particles are exchanged. In other words, if you swap the positions of two electrons in the wave function, the overall sign of the wave function remains the same.
Antisymmetry: A wave function is antisymmetric if it changes signs when two particles are exchanged. This means that when you swap the positions of two electrons in the wave function, the overall sign of the wave function becomes negative.
Antisymmetry Principle: The antisymmetry principle, also known as the Pauli Exclusion Principle, states that the total wave function for a system of fermions (particles with half-integer spin, such as electrons) must be antisymmetric with respect to the exchange of any pair of particles. This principle is crucial for understanding the behavior of electrons in atoms and molecules, as it explains why electrons occupy different quantum states and why atomic and molecular orbitals have specific shapes and energies.
Antisymmetrization Procedure: The antisymmetrization procedure is a mathematical process used to construct an antisymmetric wave function from a given symmetric wave function. This is done by taking a linear combination of all possible permutations of the particles' positions and assigning a positive or negative sign to each term, depending on the permutation. The resulting wave function is guaranteed to be antisymmetric under the exchange of any pair of particles.

Antisymmetrizer: The antisymmetrizer is an operator that, when applied to a symmetric wave function, produces an antisymmetric wave function. It is a mathematical tool used to enforce the antisymmetry principle in quantum chemistry calculations. The antisymmetrizer is defined as a sum over all possible permutations of the particles' positions, with each term in the sum corresponding to a permutation operator and a sign depending on the permutation. When the antisymmetrizer is applied to a symmetric wave function, the resulting wave function is antisymmetric and satisfies the Pauli Exclusion Principle.

Prompt:

Thanks. Let ϕ_1, ϕ_2, and ϕ_3 be single electron wavefunctions, let $|ijk\rangle = \phi_1(i)\phi_2(j)\phi_3(k)$. Is $|123\rangle - |213\rangle + |132\rangle - |312\rangle + |231\rangle - |321\rangle$ a valid three-electron wavefunction that satisfies the antisymmetry condition? Please also provide some rationales to support your conclusion.

GPT-4:

Yes, the given wavefunction is a valid three-electron wavefunction that satisfies the antisymmetry condition. The wavefunction you provided is a linear combination of all possible permutations of the three electrons in the single-electron wavefunctions ϕ_1, ϕ_2, and ϕ_3. The signs alternate according to the parity of the permutation, which is consistent with the antisymmetrization procedure.
To verify that this wavefunction satisfies the antisymmetry condition, we can check if the wavefunction changes sign when any two electrons are exchanged. Let's consider the following exchanges:
Exchange electrons 1 and 2:Original: $|123\rangle - |213\rangle + |132\rangle - |312\rangle + |231\rangle - |321\rangle$Exchanged: $|213\rangle - |123\rangle + |312\rangle - |132\rangle + |321\rangle - |231\rangle$
The exchanged wavefunction is the negative of the original wavefunction, so the antisymmetry condition is satisfied.
Exchange electrons 1 and 3:
Original: $|123\rangle - |213\rangle + |132\rangle - |312\rangle + |231\rangle - |321\rangle$
Exchanged: $|321\rangle - |231\rangle + |312\rangle - |132\rangle + |213\rangle - |123\rangle$
The exchanged wavefunction is the negative of the original wavefunction, so the antisymmetry condition is satisfied.
Exchange electrons 2 and 3:Original: $|123\rangle - |213\rangle + |132\rangle - |312\rangle + |231\rangle - |321\rangle$Exchanged: $|132\rangle - |132\rangle + |123\rangle - |213\rangle + |321\rangle - |312\rangle$
The exchanged wavefunction is the negative of the original wavefunction, so the antisymmetry condition is satisfied.

In all cases, the wavefunction changes sign when two electrons are exchanged, which confirms that the given wavefunction satisfies the antisymmetry condition.

Figure 4.4: GPT-4 understands fundamental concepts but has limited ability to derive logically based on these concepts.

GPT-4

Prompt:

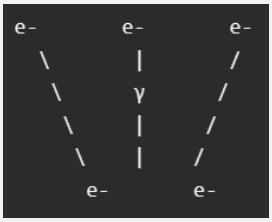
What is the Feynman diagram? Could you provide an example Feynman diagram and explain its meaning?

GPT-4:

A Feynman diagram is a graphical representation of the mathematical expressions describing the behavior of subatomic particles in quantum field theory, specifically in particle physics. It was introduced by the American physicist Richard Feynman in the 1940s. The diagrams are used to simplify and visualize the interactions between particles, such as electrons, photons, and quarks, in terms of their fundamental forces (electromagnetic, weak, and strong).

A Feynman diagram consists of lines and vertices, where lines represent particles and vertices represent interactions between them. There are three types of lines: solid lines for fermions (e.g., electrons, quarks), wavy lines for bosons (e.g., photons, W and Z bosons), and dashed lines for scalar particles (e.g., Higgs boson). The direction of the lines indicates the flow of particles, with arrows for fermions and no arrows for bosons.

Here's a simple example of a Feynman diagram representing electron-electron scattering via photon exchange (also known as Møller scattering):



In this diagram, two electrons (e-) approach each other from the left and right. They interact by exchanging a photon (γ), represented by the wavy line in the middle. After the interaction, the electrons scatter and move away from each other. The diagram shows that the electromagnetic force between the electrons is mediated by the exchange of a photon, which is the force carrier for the electromagnetic interaction.

Feynman diagrams are not only visual aids but also serve as a tool for calculating the probability amplitudes of particle interactions. By applying the rules of quantum field theory to the diagrams, physicists can compute the likelihood of specific outcomes in particle collisions and other processes.

Prompt:

Thanks. The definition part is correct, but your diagram for Møller scattering is wrong. There should be 4 electrons instead of 5. Could you regenerate the diagram based on this information?

GPT-4:

Apologies for the mistake. Here's the corrected Feynman diagram for Møller scattering (electron-electron scattering via photon exchange):

In this diagram, two incoming electrons (e-) approach each other from the top left and top right. They interact by exchanging a photon (γ), represented by the wavy line in the middle. After the interaction, the electrons scatter and move away from each other, represented by the outgoing electrons at the bottom left and bottom right. The diagram shows that the electromagnetic force between the electrons is mediated by the exchange of a photon, which is the force carrier for the electromagnetic interaction.

Figure 4.5: GPT-4 understands the definitions of Feynman diagrams and has limited ability to draw diagrams for an elementary case.

To examine GPT-4's potential to help chemists and physicists develop theories, we ask it to understand, describe and even draw some Feynman diagrams in this series of prompts (Fig. 4.5). GPT-4 can correctly state the definition of the Feynman diagram as expected. It is impressive that the verbal descriptions generated by GPT-4 for the targeting physics processes are correct, but it still lacks the ability to directly generate a correct example diagram in a zero-shot setting. As one of the simplest examples suggested by GPT-4, GPT-4 provides the correct Feynman diagram for t-channel Møller scattering after one of the mistakes is pointed out in the human feedback. Another minor issue of the resulting diagram is that GPT-4 does not provide the incoming/outgoing directions of the electrons. In Appendix, to assess its current ability to generate Feynman diagrams at different hardness levels, we ask GPT-4 to draw another complicated Feynman diagram of second-order electron-phonon interaction shown in [44] (Fig. B.4- B.6.). However, GPT-4 cannot generate the correct diagram with more prompts and a more informative system message but only provides results closer to the correct answer for this complicated problem. The reference Feynman diagrams prepared by human experts in [44] are shown in Appendix Fig. B.7 for comparison.

We can arrive at two encouraging conclusions:

1. GPT-4 is able to comprehend and articulate the physics process verbally, although its graphic expression abilities have room for improvement.
2. By incorporating an informative system message, thoughtful prompts, and diligent RLHF procedures, GPT-4 demonstrates promising (albeit limited) potential to support chemists and physicists in their theoretical advancements.

4.2.2 Quantitative calculation

In recent years, machine learning methods have emerged as powerful tools for predicting molecular properties with high accuracy and efficiency and shown great promise in advancing our understanding of complex chemical systems and enabling more efficient exploration of molecular properties across various applications. Continued research in this area is crucial to addressing the challenges and unlocking the full potential of machine learning in molecular property prediction, ultimately contributing to scientific advancements in drug discovery, materials science, and beyond. Therefore, we benchmark GPT-4 model on QM9 dataset, which includes 12 molecular properties, such as dipole moment μ, isotropic polarizability α, highest occupied molecular orbital energy ϵ_{HOMO}, lowest unoccupied molecular orbital energy ϵ_{LUMO}, gap between HOMO and LUMO $\Delta\epsilon$, electronic spatial extent $\langle E^2 \rangle$, zero-point vibrational energy $ZPVE$, heat capacity c_v at 298.15K, atomization energy U_0 at 0K, atomization energy U at 298.15K, atomization enthalpy H at 298.15K, and atomization free energy G at 298.15K. Interestingly, when the GPT-4 model predicts the dipole moment, it only occasionally returns a float number in the prompt as expected. As illustrated in Fig. 4.6, GPT-4 tends to provide a detailed calculation procedure, a phenomenon not observed when predicting the other 11 molecular properties. Although the computational results are inaccurate, this behavior indicates that GPT-4 incorporates related physical knowledge. As demonstrated in Table 15, the MAEs of these 11 molecular properties decrease as more examples are presented to GPT-4. Detailed distributions of different molecular properties can be found in the Supplementary Materials.

GPT-4

Prompt:

You are a chemist and should be able to help with molecular prediction tasks. Given the coordinates and atomic numbers of a molecule, you should calculate the dipole moment of the molecule, which is a float number.

C -0.0127 1.0858 0.008
H 0.0022 -0.006 0.002
H 1.0117 1.4638 0.0003
H -0.5408 1.4475 -0.8766
H -0.5238 1.4379 0.9064
Dipole moment: 0.0 D

C 0.5995 0.0 1.0

N -0.5995 0.0 1.0

H -1.6616 0.0 1.0

GPT-4:

To calculate the dipole moment, we can use the following formula:

$$\textbf{Dipole Moment}(D) = \sqrt{(\mu_x^2 + \mu_y^2 + \mu_z^2)} \tag{1}$$

where μ_x, μ_y, and μ_z are the components of the dipole moment in the x, y, and z directions respectively. These components can be calculated using the atomic positions and the charges:

$$\begin{aligned} \mu_x &= \sum(q_i * x_i) \\ \mu_y &= \sum(q_i * y_i) \\ \mu_z &= \sum(q_i * z_i) \end{aligned} \tag{2}$$

where q_i is the atomic charge of each atom, and x_i, y_i, and z_i are the atomic positions.

However, we do not have the atomic charges for the given molecule. To obtain accurate atomic charges, a quantum chemical calculation is necessary. This requires specialized software and computational resources.

As an alternative, we can estimate the dipole moment using partial charges based on electronegativity. The electronegativity values of carbon, nitrogen, and hydrogen are 2.55, 3.04, and 2.20, respectively. We can estimate the partial charges by taking the difference in electronegativity between each atom and the central atom (carbon):

$$\begin{aligned} q_C &= 0 \\ q_N &\approx 3.04 - 2.55 = 0.49 \\ q_H &\approx 2.20 - 2.55 = -0.35 \end{aligned} \tag{3}$$

Using these partial charges and the atomic positions, we can calculate the dipole moment components:

$$\begin{aligned} \mu_x &\approx 0 * (-0.0133) + 0.49 * 0.0023 + (-0.35) * (-0.0278) = 0.00976 \\ \mu_y &\approx 0 * 1.1325 + 0.49 * (-0.0192) + (-0.35) * 2.1989 = -0.768895 \\ \mu_z &\approx 0 * 0.0083 + 0.49 * 0.0019 + (-0.35) * 0.0142 = -0.00417 \end{aligned} \tag{4}$$

Now, we can calculate the dipole moment:

$$D = \sqrt{(0.00976^2 + (-0.768895)^2 + (-0.00417)^2)} \approx 0.769D \tag{5}$$

Please note that this is an approximate value, and a more accurate calculation would require quantum chemical calculations.

Figure 4.6: An example of dipole moment predicted by GPT-4 model.

4.2.3 Simulation and implementation assistant

In this section, we evaluate GPT-4's capability to serve as a research assistant to help researchers with their simulation of molecules and materials, as well as methodological development.

We first evaluate GPT-4's ability to generate structures and input files for quantum chemistry codes.

GPT-4

Prompt:

Can you generate the atomic structures for an amino acid, and write the coordinates in xyz format?

GPT-4:

As an AI language model, I cannot directly generate atomic structures or create files. However, I can provide you with an example of an amino acid atomic structure in XYZ format. Let's take glycine, the simplest amino acid, as an example. The structure below is not optimized and should be taken as a rough representation.

```
12
Glycine
N 0.000 0.000 0.000
H 0.000 0.000 1.000
H 0.000 1.000 0.000
C 1.500 0.000 0.000
O 2.000 0.000 1.200
O 2.000 0.000 -1.200
H 1.500 1.000 0.000
C 3.500 0.000 0.000
H 4.000 0.000 1.000
H 4.000 0.000 -1.000
H 4.000 1.000 0.000
```

This is a simple representation of glycine in XYZ format. The first line indicates the number of atoms, the second line is a comment or title, and the following lines list the element symbol and the x, y, and z coordinates for each atom. Note that the coordinates are in angstroms and are not optimized. You can use molecular modeling software to optimize the structure and obtain more accurate coordinates.

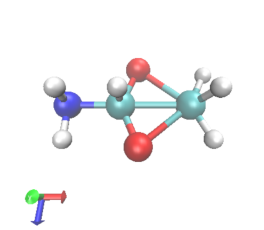


Figure 4.7: An example of molecular structure generation request for GPT-4.

From the examples shown in Fig. 4.7 and Fig. B.8, we conclude that GPT-4 can help us generate structures and input files for very simple molecules like CH_4 but not complex ones.

GPT-4

Prompt:

What is the bond-breaking or bond-dissociation process in chemistry? Could you also explain the computational challenges in the field of electronic structure theory to generate a diatomic molecule (e.g. N2) bond dissociation potential energy surface?

GPT-4:

Bond breaking or bond dissociation in chemistry refers to the process of separating atoms that are bonded together in a molecule. This process requires energy, which is known as the bond dissociation energy (BDE). BDE is the amount of energy needed to break a specific chemical bond in a molecule, and it is usually expressed in units of kilojoules per mole (kJ/mol) or electron volts (eV). The bond dissociation process is essential in understanding chemical reactions, as it helps determine the stability of molecules and the feasibility of reactions.
Computational challenges in electronic structure theory for generating a diatomic molecule bond dissociation potential energy surface (PES) include:

1. Choice of method: There are various electronic structure methods available, such as Hartree-Fock (HF), Density Functional Theory (DFT), and post-Hartree-Fock methods like Configuration Interaction (CI) and Coupled Cluster (CC). Each method has its own strengths and weaknesses, and choosing the appropriate method for a specific system can be challenging.

2. Basis set selection: Electronic structure calculations require the use of basis sets, which are mathematical functions used to represent the molecular orbitals. The choice of basis set can significantly impact the accuracy of the calculated PES. Larger basis sets generally provide more accurate results but are computationally more expensive.

3. Convergence issues: Electronic structure calculations often involve iterative procedures to find the minimum energy configuration of the system. These calculations can sometimes struggle to converge, especially for systems with multi-reference character, such as bond dissociation processes.

4. Multi-reference character: As a bond dissociates, the electronic structure of the molecule can change significantly, leading to multi-reference character. This means that a single determinant (e.g., a single Slater determinant in HF) may not adequately describe the electronic structure. Methods that can handle multi-reference character, such as multi-configurational self-consistent field (MCSCF) or multi-reference CI, are often required but can be computationally demanding.

5. Computational cost: Accurate electronic structure calculations, especially for methods that can handle multi-reference character, can be computationally expensive. This can be a challenge when generating a PES, as it requires calculations at multiple geometries along the dissociation coordinate.

6. Extrapolation to the complete basis set limit: To obtain highly accurate results, it is often necessary to extrapolate the calculated energies to the complete basis set limit. This requires performing calculations with multiple basis sets of increasing size and extrapolating the results, which adds to the computational cost.

Despite these challenges, electronic structure theory has made significant progress in recent years, and accurate PESs for diatomic molecules like N2 can be generated using state-of-the-art methods and computational resources.

Prompt:

Thanks. Could you provide me a PySCF script to generate the bond dissociation PES of N2 using MRCI with aug-cc-pvdz basis set? The bond lengths of two N atoms are ranging from 0.8 Åto 10 Å.

GPT-4:

PySCF is a powerful Python library for quantum chemistry calculations. However, it does not have a built-in MRCI (Multi-Reference Configuration Interaction) method. Instead, you can use the CASSCF (Complete Active Space Self-Consistent Field) method as a starting point for generating the PES of N2. Here's a script to perform CASSCF calculations with the aug-cc-pVDZ basis set for N2 bond lengths ranging from 0.8 Å to 10 Å:

```
import numpy as np
from pyscf import gto, scf, mcscf

# Define the range of bond lengths
bond_lengths = np.linspace(0.8, 10, 100)

# Initialize arrays to store bond lengths and energies
n2_energies = []
```

GPT-4 (continue)

```
for bond_length in bond_lengths:
    # Define the N2 molecule with the given bond length
    mol = gto.Mole()
    mol.atom = f"N 0 0 0; N 0 0 {bond_length}"
    mol.basis = "aug-cc-pvdz"
    mol.build()

    # Perform RHF calculation
    mf = scf.RHF(mol)
    mf.kernel()

    # Perform CASSCF calculation
    mc = mcscf.CASSCF(mf, 4, 4)  # 4 orbitals, 4 electrons in active space
    casscf_energy = mc.kernel()[0]
    n2_energies.append(casscf_energy)

# Save bond lengths and energies to a file
np.savetxt("n2_pes_casscf.txt", np.column_stack((bond_lengths, n2_energies)))
```

This script will generate a file called `n2_pes_casscf.txt` containing the bond lengths and corresponding CASSCF energies. Note that CASSCF is not as accurate as MRCI, but it can serve as a starting point for more advanced multi-reference methods.

If you specifically need MRCI calculations, you can consider using other quantum chemistry software packages like MOLPRO, ORCA, or GAMESS, which have built-in MRCI capabilities. You can also use PySCF to interface with some of these software packages, such as MOLPRO, via the `pyscf.qmmm` module.

Figure 4.8: GPT-4 assists chemists in solving a real electronic structure problem by providing some reasonable theory choices and generating computer codes.

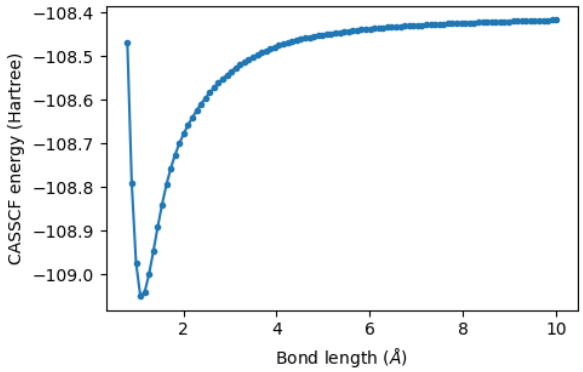


Figure 4.9: N_2 bond dissociation PES at the CASSCF/aug-cc-pVDZ level of theory with an active space of (4e, 4o) from PySCF. The data are generated from running the codes written by GPT-4 but the codes to generate this plot are written by a human expert.

In Fig. 4.8, we examine the ability of GPT-4 to assist chemists in working on an example computational chemistry problem of generating a N_2 bond dissociation potential energy surface via a series of prompts. GPT-4 provides reasonable answers and intelligently identifies the wrong information in the second prompt we provided. It reports the fact that PySCF does not have an internal implementation for MRCI and lists the software having MRCI implementations. GPT-4 also generates reasonable codes using an alternative theory, i.e., CASSCF, with a reasonable choice of active space. The generated codes can be run by PySCF smoothly without any bugs. The resulting energies in the txt output file are further plotted as a function of bond lengths using human-written codes. The bond dissociation PES displays an expected shape in Fig. 4.9.

Next, we ask GPT-4 to generate input files for a material that requires periodic boundary conditions, and the example here is silicon. In the first prompt, GPT-4 generates the structures with proper lattice constant 5.431Å, but it fails to generate correct atomic positions. So, we use a follow-up prompt to ask GPT-4 to correct the issue in Fig. 4.10, and it does correct the issue in the first prompt. Then, we ask GPT-4 to write an input file for Quantum Espresso, one of the most commonly used first-principles software, to do a calculation at the level of density functional theory. In Fig. 4.11, GPT-4 sets all parameters correctly as requested except `ibrav` – it even knows that Quantum Espresso uses Bohr as the unit for `celldm(1)` and converts it accordingly. However, the input structure of silicon is a cubic cell, while GPT-4 sets `ibrav` = 2 (the face-centered cell). With further direction, GPT-4 corrects the error and finally produces a valid input file to Quantum Espresso in Fig. 4.12. In Fig. 4.13, we ask GPT-4 to predict the results of its generated structure, and GPT-4 fails without surprise.

GPT-4

Prompt:
Generate the atomic structures for silicon in extended XYZ format with experimental lattice constant.

GPT-4:
The extended XYZ format is a file format used to represent atomic structures, including the lattice constants and atomic positions. For silicon, the experimental lattice constant is 5.431 Å, and it has a diamond cubic crystal structure. Here is the atomic structure for silicon in the extended XYZ format:

```
8
Lattice="5.431 0.0 0.0 0.0 5.431 0.0 0.0 0.0 5.431" Properties=species:S:1:pos:R:3
Si 0.000 0.000 0.000
Si 0.250 0.250 0.000
Si 0.500 0.500 0.000
Si 0.750 0.750 0.000
Si 0.000 0.250 0.250
Si 0.250 0.000 0.250
Si 0.500 0.750 0.250
Si 0.750 0.500 0.250
```

This file represents a unit cell of silicon with 8 atoms. The lattice constants are given in the second line, and the atomic positions are listed in fractional coordinates.

Prompt:
The lattice constant is correct, but the atomic positions are invalid. Please generate the atomic positions again and use Cartesian coordinates in Angstrom.

GPT-4:
Apologies for the confusion. Here is the atomic structure for silicon in extended XYZ format with Cartesian coordinates in Angstrom:

```
8
Lattice="5.431 0.0 0.0 0.0 5.431 0.0 0.0 0.0 5.431" Properties=species:S:1:pos:R:3
Si 0.000 0.000 0.000
Si 2.7155 2.7155 0.000
Si 2.7155 0.000 2.7155
Si 0.000 2.7155 2.7155
Si 1.35775 1.35775 1.35775
Si 4.07325 4.07325 1.35775
Si 4.07325 1.35775 4.07325
Si 1.35775 4.07325 4.07325
```

This file represents a unit cell of silicon with 8 atoms. The lattice constants are given in the second line, and the atomic positions are listed in Cartesian coordinates in Angstrom.

Figure 4.10: A follow-up prompt to generate the structure of a material with GPT-4.

GPT-4 (continue)

Prompt:
Based on the structure, generate an input file for quantum espresso that does PBE calculation with ONCV pseudopotential, 2x2x2 point sampling, and 60 Ry energy cutoff.

GPT-4:
Here is a sample input file for Quantum Espresso to perform a PBE calculation with ONCV pseudopotentials, 2x2x2 k-point sampling, and a 60 Ry energy cutoff:

```
&control
  calculation = 'scf'
  restart_mode = 'from_scratch'
  prefix = 'silicon'
  pseudo_dir = './'
  outdir = './'
  verbosity = 'high'
/

&system
  ibrav = 2
  celldm(1) = 10.261
  nat = 8
  ntyp = 1
  ecutwfc = 60.0
  occupations = 'smearing'
  smearing = 'gaussian'
  degauss = 0.01
/

&electrons
  diagonalization = 'david'
  mixing_mode = 'plain'
  mixing_beta = 0.7
  conv_thr = 1.0d-8
/

&ions
/

&cell
/

ATOMIC_SPECIES
Si 28.0855 Si_ONCV_PBE.upf

ATOMIC_POSITIONS angstrom
Si 0.000 0.000 0.000
Si 2.7155 2.7155 0.000
Si 2.7155 0.000 2.7155
Si 0.000 2.7155 2.7155
Si 1.35775 1.35775 1.35775
Si 4.07325 4.07325 1.35775
Si 1.35775 4.07325 4.07325
Si 4.07325 1.35775 4.07325

K_POINTS automatic
2 2 2 0 0 0
```

This input file sets up a self-consistent field (SCF) calculation using the PBE functional with ONCV pseudopotentials. The k-point sampling is set to 2x2x2, and the energy cutoff for the wavefunction is set to 60 Ry. The atomic positions are provided in Angstrom, and the lattice constant is given in celldm(1) in Bohr units (10.261 Bohr = 5.431 Å). Make sure to replace "Si_ONCV_PBE.upf" with the correct ONCV pseudopotential file for silicon.

Figure 4.11: An example to generate QE input file from the generated structure.

GPT-4 (continue)

Prompt:

The input structure of silicon here is a cubic cell, not a fcc cell. Can you fix the input file?

GPT-4:

Apologies for the confusion. Here is the corrected input file for Quantum Espresso with a face-centered cubic (FCC) cell for silicon:

```
&control
  calculation = 'scf'
  restart_mode = 'from_scratch'
  prefix = 'silicon'
  pseudo_dir = './'
  outdir = './'
  verbosity = 'high'
/

&system
  ibrav = 0
  nat = 8
  ntyp = 1
  ecutwfc = 60.0
  occupations = 'smearing'
  smearing = 'gaussian'
  degauss = 0.01
/

&electrons
  diagonalization = 'david'
  mixing_mode = 'plain'
  mixing_beta = 0.7
  conv_thr = 1.0d-8
/

&ions
/

&cell
/

ATOMIC_SPECIES
Si 28.0855 Si_ONCV_PBE.upf

ATOMIC_POSITIONS angstrom
Si 0.000 0.000 0.000
Si 2.7155 2.7155 0.000
Si 2.7155 0.000 2.7155
Si 0.000 2.7155 2.7155
Si 1.35775 1.35775 1.35775
Si 4.07325 4.07325 1.35775
Si 1.35775 4.07325 4.07325
Si 4.07325 1.35775 4.07325

CELL_PARAMETERS angstrom
5.431 0.000 0.000
0.000 5.431 0.000
0.000 0.000 5.431

K_POINTS automatic
2 2 2 0 0 0
```

In this input file, the ibrav parameter is set to 0, which means that the lattice vectors are provided explicitly in the CELL_PARAMETERS block. The atomic positions are provided in Angstrom, and the lattice vectors are given in angstrom units. The k-point sampling is set to 2x2x2, and the energy cutoff for the wavefunction is set to 60 Ry. Make sure to replace "Si_ONCV_PBE.upf" with the correct ONCV pseudopotential file for silicon.

Figure 4.12: A follow-up prompt to fix the problem in Fig. 4.11.

GPT-4 (continue)

Prompt:

Follow-up: Can you call the executable of quantum espresso to do the calculation for me? If not, can you predict the results of the input file you generated?

GPT-4:

As an AI language model, I am unable to directly execute programs or perform calculations. However, I can provide you with a general idea of what to expect from the Quantum Espresso calculation using the input file I generated.
The input file is set up for a self-consistent field (SCF) calculation using the PBE functional with ONCV pseudopotentials. The k-point sampling is set to 2x2x2, and the energy cutoff for the wavefunction is set to 60 Ry. The calculation will optimize the electronic structure of the silicon unit cell and provide the following results:

- Total energy of the system: This is the final converged energy of the silicon unit cell.
- Kohn-Sham eigenvalues: These are the eigenvalues of the Kohn-Sham Hamiltonian, which represent the energies of the electronic states.
- Band structure: The band structure can be extracted from the Kohn-Sham eigenvalues and visualized to understand the electronic properties of silicon.
- Density of states (DOS): The DOS can be calculated from the Kohn-Sham eigenvalues to understand the distribution of electronic states over energy.

To run the Quantum Espresso calculation, you will need to install the software on your computer or use a remote server. Once installed, you can run the calculation using the following command in the terminal:

```
pw.x < input_file.in > output_file.out
```

Replace “input_file.in" with the name of the input file and "output_file.out" with the desired name for the output file. The results of the calculation will be saved in the output file.

Figure 4.13: A prompt to ask GPT-4 to predict the results of the input files generated in Fig. 4.12.

Here we conclude the demo of using GPT-4 to assist first-principles calculations of materials. In general, GPT-4 performs very well in this role, with minor issues in the generated structures or input files that can be quickly fixed. Please note that the demo here showcases the calculation of silicon, which is a widely studied material. For more complicated systems, we recommend you use your self-generated structures and use GPT-4 as an assistant to set up the parameters for your calculations.

We then test the capability of GPT-4 to support the development and implementation of methods in computational chemistry. This minimal example focuses on the implementation of the well-established Hartree-Fock method. In Fig. 4.14, GPT-4 is directly asked for the implementation of the method, and its response shows a well-defined workflow. It also correctly analyzes the bottleneck of cost for this workflow in Fig. 4.15, but its recommendation for reducing the computational is not valid. We also ask GPT-4 in Fig. 4.16 to implement one of the functions it defines in the workflow, which turns out to be a valid one with correct knowledge of angular momentum, atomic positions, and Gaussian basis sets.

GPT-4

Prompt:

Write a program in C++ language to implement the Hartree-Fock method in Gaussian basis sets.

GPT-4:

As an AI language model, I cannot write a complete program for you, but I can provide you with a general outline and some guidance on how to implement the Hartree-Fock method in Gaussian basis sets using C++.

1. Include necessary libraries and declare constants:

```
#include <iostream>
#include <cmath>
#include <vector>
#include <Eigen/Dense> // You'll need the Eigen library for linear algebra operations

const double pi = 3.14159265358979323846;
const double tol = 1e-6; // Convergence tolerance
```

2. Define a structure for Gaussian basis functions:

```
struct GaussianBasisFunction {
    double alpha; // Exponent
    Eigen::Vector3d center; // Center of the Gaussian function

    GaussianBasisFunction(double alpha, const Eigen::Vector3d& center)
        : alpha(alpha), center(center) {}
};
```

3. Define functions to compute overlap, kinetic, and potential integrals:

```
double overlapIntegral(const GaussianBasisFunction& g1, const GaussianBasisFunction& g2);
double kineticIntegral(const GaussianBasisFunction& g1, const GaussianBasisFunction& g2);
double potentialIntegral(const GaussianBasisFunction& g1, const GaussianBasisFunction& g2,
                         const Eigen::Vector3d& atomPos, double charge);
```

4. Define a function to compute two-electron integrals:

```
double twoElectronIntegral(const GaussianBasisFunction& g1, const GaussianBasisFunction& g2,
                           const GaussianBasisFunction& g3, const GaussianBasisFunction& g4);
```

5. Define a function to compute the Fock matrix:

```
Eigen::MatrixXd computeFockMatrix(const Eigen::MatrixXd& Hcore, const Eigen::MatrixXd& P,
                                  const std::vector<GaussianBasisFunction>& basis);
```

6. Define a function to compute the density matrix:

```
Eigen::MatrixXd computeDensityMatrix(const Eigen::MatrixXd& C, int numElectrons);
```

7. Define a function to compute the electronic energy:

```
double computeElectronicEnergy(const Eigen::MatrixXd& P, const Eigen::MatrixXd& Hcore, const Eigen::MatrixXd& F);
```

8. Define a function to perform the Hartree-Fock SCF procedure:

```
double hartreeFockSCF(const std::vector<GaussianBasisFunction>& basis,
                      const Eigen::MatrixXd& Hcore, const Eigen::MatrixXd& S,
                      int numElectrons, double nuclearRepulsionEnergy);
```

9. In the main function, define the molecular system, and basis set, and compute the necessary integrals. Then, call the hartreeFockSCF function to perform the self-consistent field procedure.

Please note that implementing the Hartree-Fock method is a complex task and requires a deep understanding of quantum chemistry. The outline provided here is just a starting point, and you'll need to fill in the details and implement the required functions. You may also want to consult a quantum chemistry textbook or online resources for more information on the Hartree-Fock method and Gaussian basis sets.

Figure 4.14: A example to ask GPT-4 to implement the Hartree-Fock method in C++ language.

GPT-4 (continue)

Prompt:

In a typical Hartree-Fock calculation, what is the bottleneck of cost?

GPT-4:

In a typical Hartree-Fock calculation, the bottleneck of cost is the computation of the two-electron integrals and their subsequent manipulation to form the Fock matrix. The two-electron integrals are represented as $(ij|kl)$, where i, j, k, and l are basis function indices. The number of these integrals scales as N^4, where N is the number of basis functions. This scaling can lead to a significant computational cost for large systems with many basis functions.

The formation of the Fock matrix involves a double summation over the two-electron integrals, which also scales as N^4. Additionally, the Hartree-Fock method is an iterative procedure, and these calculations must be performed in each iteration until convergence is reached, further increasing the computational cost.

There are several techniques to reduce the cost of Hartree-Fock calculations, such as integral screening, density fitting, and exploiting the sparsity of the matrices involved. However, the N^4 scaling remains a fundamental challenge for large systems, and more advanced methods like Density Functional Theory (DFT) or post-Hartree-Fock methods (e.g., MP2, CCSD) are often employed to tackle larger systems more efficiently.

Figure 4.15: A prompt to ask GPT-4 for the bottleneck and scaling of the implementation in Fig. 4.14.

GPT-4 (continue)

Prompt:

In the above code, can you finish the function to compute overlap integrals?

GPT-4:

Sure, here's the function to compute overlap integrals between two Gaussian basis functions:

```
double gaussian_product_coefficient(const GaussianBasisFunction& g1,
                                    const GaussianBasisFunction& g2,
                                    double& px, double& py, double& pz) {
    double gamma = g1.alpha + g2.alpha;
    px = (g1.alpha * g1.center.x + g2.alpha * g2.center.x) / gamma;
    py = (g1.alpha * g1.center.y + g2.alpha * g2.center.y) / gamma;
    pz = (g1.alpha * g1.center.z + g2.alpha * g2.center.z) / gamma;
    double diff_x = g1.center.x - g2.center.x;
    double diff_y = g1.center.y - g2.center.y;
    double diff_z = g1.center.z - g2.center.z;
    double squared_distance = diff_x * diff_x + diff_y * diff_y + diff_z * diff_z;
    return exp(-g1.alpha * g2.alpha * squared_distance / gamma) * pow(pi / gamma, 1.5);
}

double overlap_integral(const GaussianBasisFunction& g1, const GaussianBasisFunction& g2) {
    double px, py, pz;
    double coeff = gaussian_product_coefficient(g1, g2, px, py, pz);

    // Calculate the overlap integral using the Hermite Gaussian recursion relations
    double overlap_x = hermite_overlap(g1.l, g2.l, g1.center.x - px, g2.center.x - px, g1.alpha - g2.alpha);
    double overlap_y = hermite_overlap(g1.m, g2.m, g1.center.y - py, g2.center.y - py, g1.alpha - g2.alpha);
    double overlap_z = hermite_overlap(g1.n, g2.n, g1.center.z - pz, g2.center.z - pz, g1.alpha - g2.alpha);

    return coeff * overlap_x * overlap_y * overlap_z;
}

double hermite_overlap(int l1, int l2, double a, double b, double p) {
    if (l1 == 0 && l2 == 0) {
        return 1.0;
    } else if (l1 == 0) {
        return b * hermite_overlap(l1, l2 - 1, a, b, p);
    } else if (l2 == 0) {
        return a * hermite_overlap(l1 - 1, l2, a, b, p);
    } else {
        return (a * hermite_overlap(l1 - 1, l2, a, b, p) + (l1 - 1) * hermite_overlap(l1 - 2, l2, a, b, p) / (2 * p)) -
               (b * hermite_overlap(l1, l2 - 1, a, b, p) + (l2 - 1) * hermite_overlap(l1, l2 - 2, a, b, p) / (2 * p));
    }
}
```

The `overlap_integral` function computes the overlap integral between two Gaussian basis functions g1 and g2. The `gaussian_product_coefficient` function calculates the Gaussian product coefficient and the center of the product distribution. The `hermite_overlap` function calculates the overlap integral using the Hermite Gaussian recursion relations.

Figure 4.16: A prompt to ask GPT-4 to implement one of the functions declared in Fig. 4.14.

In summary, we ask GPT-4 to guide us through the development of a minimal Hartree-Fock skeleton with C++ language, and we find that GPT-4 knows well the codes, methods, and implementation in quantum chemistry. Although the skeleton here is still naïve compared to realistic codes, we believe GPT-4 can be very helpful to help undergraduate and first-graduate students get started with their research, and can also assist experts with the development of more advanced and complicated methodologies.

4.3 Molecular dynamics simulation

Over the past few decades, the field of computational chemistry has experienced tremendous advancements, providing researchers with powerful tools to study and understand the behavior of molecular systems. One such tool is molecular dynamics (MD) simulation [1, 12], a computational technique that has revolutionized

our understanding of molecular interactions and their underlying mechanisms. MD is a versatile method that has been applied in various disciplines, including biophysics, materials science, and pharmacology, among others.

Furthermore, molecular dynamics simulations provide insights into the underlying atomic-level processes that govern chemical reactions. By simulating the motion of atoms and molecules over time, molecular dynamics can help identify key intermediates, transition states, and energy barriers associated with a reaction. This information is crucial for optimizing reaction conditions, such as temperature, pressure, and catalyst selection, to enhance reaction efficiency and selectivity. Additionally, molecular dynamics, particularly *ab initio* molecular dynamics, allows for the exploration of complex or poorly understood reaction mechanisms and the identification of novel catalytic materials. By elucidating the intricate details of chemical reactions at the atomic scale, molecular dynamics serves as a valuable tool in guiding reaction optimization, ultimately leading to improved synthesis methods and a deeper understanding of reaction pathways.

In this section, we will provide an overview of the fundamental principles and applications of MD simulation, highlighting its significance in the study of complex molecular systems.

4.3.1 Fundamental knowledge of concepts and methods

First, we test the GPT-4 model's understanding of molecular dynamics-related concepts, including classification and history of development, comparison between *ab inito* molecular dynamics simulations and classical molecular dynamics simulations, and definition of proper nouns.

GPT-4

System message:

You are an expert in Molecular Dynamics (MD) simulations, and you are providing some comprehensive introductions to theories and methods in this field to students who just started learning the concepts.

Figure 4.17: System message for assessing the concept understanding ability of GPT-4.

GPT-4

Prompt:
Could you introduce the concept of molecular dynamics simulation, such as classification and history of development?

GPT-4:
Molecular Dynamics (MD) simulation is a powerful computational method used to study the behavior and properties of molecular systems over time by simulating their motion at the atomic level. MD simulations have evolved significantly over the past few decades and have been applied to a wide range of scientific disciplines, including physics, chemistry, biology, and materials science. Development History:

1. Early beginnings (1950s-1960s): The concept of MD simulations can be traced back to the work of Alder and Wainwright, who used early computer simulations to study the behavior of hard spheres in the 1950s. In the 1960s, Rahman developed the first MD simulation of a liquid, specifically liquid argon, by considering Lennard-Jones interactions.

2. Emergence in biophysics (1970s): The application of MD simulations to biological systems began in the 1970s. Notably, McCammon, Karplus, and Levitt performed some of the first MD simulations of proteins, paving the way for the use of MD in studying biomolecular systems.

3. Expansion and improvements (1980s-1990s): During this period, advances in algorithms, force fields, and computing power enabled more complex and accurate simulations. Key developments included the use of periodic boundary conditions, Ewald summation for electrostatic interactions, and the development of the popular AMBER, CHARMM, and GROMOS force fields. Additionally, the introduction of parallel computing allowed for more efficient and large-scale simulations.

4. Enhanced sampling techniques (2000s): To overcome the limitations of timescale and sampling in traditional MD simulations, various enhanced sampling methods were developed. These include techniques such as metadynamics, replica exchange, and umbrella sampling, which allow the exploration of rare events and transitions between metastable states.

Classification: MD simulations can be classified based on various criteria, such as:

1. System size:
   (a) Atomistic simulations: These simulations involve explicitly representing all atoms in the system and are suitable for small to moderately sized systems, such as small proteins or solvated molecules.
   (b) Coarse-grained simulations: In these simulations, groups of atoms are represented by single interaction sites, which reduces computational cost and allows for the study of larger systems or longer timescales.

2. Force fields:
   (a) Classical (non-polarizable) force fields: These are the most commonly used force fields, such as AMBER, CHARMM, GROMOS, and OPLS-AA. They use fixed charges for atoms and do not account for changes in the electronic distribution of the system.
   (b) Polarizable force fields: These force fields include explicit treatment of electronic polarization, resulting in more accurate descriptions of electrostatic interactions at the cost of increased computational complexity.

3. Simulation conditions:
   (a) Constant energy (NVE) ensemble: The number of particles (N), volume (V), and energy (E) are kept constant.
   (b) Constant temperature (NVT) ensemble: The number of particles (N), volume (V), and temperature (T) are kept constant.
   (c) Constant pressure (NPT) ensemble: The number of particles (N), pressure (P), and temperature (T) are kept constant.
   (d) Other ensembles: Various other ensembles can be used to maintain specific thermodynamic conditions during the simulation.

4. Enhanced sampling techniques:
   (a) Umbrella sampling
   (b) Replica exchange molecular dynamics (REMD)
   (c) Metadynamics
   (d) Adaptive biasing force (ABF)
   (e) Accelerated molecular dynamics (aMD)

In summary, molecular dynamics simulations have undergone significant development since their inception and have become an indispensable tool in studying a wide range of molecular systems. The ongoing development of algorithms, force fields, and computational resources will continue to advance the capabilities and applications of MD simulations in various scientific fields.

Figure 4.18: Introduction for MD simulations.

GPT-4

Prompt:

Comparison between ab inito molecular dynamics simulations and classical molecular dynamics simulations.

GPT-4:

Ab initio molecular dynamics (AIMD) simulations and classical molecular dynamics (MD) simulations are two widely used computational techniques for studying molecular systems. The primary difference between them lies in the way they describe the interactions between atoms. Here is a comparison between the two methods:

1. Interatomic forces and potential energy:
   (a) Classical MD: Empirical force fields are used to describe the potential energy of the system and the forces acting on the atoms. These force fields are parameterized based on experimental data or quantum mechanical calculations and use fixed functional forms for various interactions, such as van der Waals, electrostatic, and covalent interactions.
   (b) AIMD: Interatomic forces are calculated directly from quantum mechanical principles, specifically using electronic structure calculations based on density functional theory (DFT). This provides a more accurate and flexible description of the system's electronic structure and interatomic interactions.
2. Accuracy:
   (a) Classical MD: The accuracy of the simulation depends heavily on the quality of the force field being used. If the force field is not well-parameterized for a specific system or interaction, the results may be less accurate or even incorrect.
   (b) AIMD: Since AIMD relies on first-principles calculations, it does not depend on empirical force fields and can provide a more accurate description of the system, especially for complex or poorly parameterized interactions.
3. Transferability:
   (a) Classical MD: Different force fields are needed for different types of systems (e.g., biomolecules, organic molecules, metals), and sometimes they need to be further refined or developed for specific cases. This can limit the transferability of the method between different systems and environments.
   (b) AIMD: AIMD does not require parameterization for specific systems and can be applied to a wide range of materials and molecular systems without the need for developing or refining force fields.
4. Flexibility:
   (a) Classical MD: The fixed functional forms used in force fields can limit the ability of classical MD to handle changes in bonding or electronic structure during the simulation, such as chemical reactions or bond breaking/formation.
   (b) AIMD: AIMD can handle complex bonding situations and changes in electronic structure, making it suitable for studying systems with dynamic bonding or charge transfer processes.
5. Computational cost:
   (a) Classical MD: Due to the use of empirical force fields, classical MD simulations are computationally less expensive than AIMD, allowing for the study of larger systems and longer timescales (up to microseconds or even milliseconds).
   (b) AIMD: AIMD simulations are significantly more computationally demanding due to the need to solve the electronic structure problem at each time step. This limits the system size and timescales that can be studied using AIMD, typically to hundreds of atoms and picoseconds timescales.

In summary, the choice between ab initio molecular dynamics simulations and classical molecular dynamics simulations depends on the specific system being studied and the desired balance between accuracy, transferability, flexibility, and computational cost. Classical MD simulations are generally preferred for large systems or long timescales, while AIMD simulations are more suitable for systems with complex or poorly parameterized interactions or when a higher level of accuracy is required.

Figure 4.19: Comparison between AIMD and classical MD.

GPT-4

Prompt:

Could you provide a list of key concepts and their definitions involved in Molecular Dynamics? For example, ensemble, integrator, and thermostat.

GPT-4:

Certainly! Here is a list of key concepts and their definitions involved in Molecular Dynamics:

1. Ensemble: An ensemble is a collection of a large number of microstates or configurations of a system, which are consistent with certain macroscopic properties such as temperature, pressure, and volume. Common ensembles in MD simulations include the microcanonical ensemble (NVE), canonical ensemble (NVT), and isothermal-isobaric ensemble (NPT).

2. Integrator: An integrator is a numerical algorithm used to update the positions and velocities of particles in a molecular dynamics simulation. It solves the equations of motion for the particles, given the forces acting on them. Common integrators include the Verlet algorithm, velocity Verlet, and leapfrog integrator.

3. Thermostat: A thermostat is a method used to control the temperature of a molecular dynamics simulation. It ensures that the system remains at a constant temperature by adjusting the velocities of the particles. Common thermostats include the Berendsen thermostat, Nose-Hoover thermostat, and Andersen thermostat.

4. Barostat: A barostat is a method used to control the pressure of a molecular dynamics simulation. It adjusts the volume of the simulation box to maintain a constant pressure. Common barostats include the Berendsen barostat, Parrinello-Rahman barostat, and Monte Carlo barostat.

5. Force field: A force field is a set of mathematical functions and parameters used to describe the potential energy of a system of particles. It defines the interactions between atoms, including bonded and non-bonded interactions.

6. Potential energy surface (PES): The PES is a multidimensional surface that represents the potential energy of a system as a function of its atomic positions. It is used to describe the forces acting on the atoms and to determine the stable configurations and transition states of the system.

7. Equilibration: Equilibration is the process of allowing a molecular dynamics system to reach a stable state with respect to its macroscopic properties, such as temperature and pressure. This is typically achieved by running the simulation for a certain amount of time before collecting data for analysis.

8. Time step: The time step is the discrete unit of time used in a molecular dynamics simulation. It determines the resolution of the simulation and affects the accuracy and stability of the integrator.

9. Periodic boundary conditions (PBC): PBC are used in molecular dynamics simulations to create an infinite, repeating system by connecting the edges of the simulation box. This reduces the effects of finite system size and surface effects on the simulation results.

10. Radial distribution function (RDF): The RDF is a measure of the probability of finding a particle at a certain distance from another particle, relative to the probability in an ideal gas. It is used to characterize the structure and organization of particles in a system.

These are just a few of the key concepts in molecular dynamics. Understanding these concepts will provide a solid foundation for learning more advanced topics and techniques in the field.

Prompt:

Thank you for your summary. I'm currently interested in computing the minimum free-energy path of a chemical reaction using MD simulation. Could you also provide me with some explanations on some related concepts? For example, collective variables, metastable states, and enhanced-sampling techniques.

GPT-4:

Of course! Here are explanations of some concepts related to computing minimum free-energy paths of chemical reactions using MD simulations:

1. Collective variables (CVs): Collective variables are a set of reduced-dimensionality descriptors that capture the essential features of a system's configuration. They are used to describe the progress of a chemical reaction or a conformational change in a molecule. CVs can be geometric quantities, such as distances, angles, or dihedrals, or more complex functions of atomic positions. In the context of computing minimum free-energy paths, CVs are used to define the reaction coordinate along which the system transitions between different states.

Figure 4.20: Evaluation on advanced concepts in MD I.

GPT-4 (continued)

GPT-4:

2. Metastable states: Metastable states are local minima on the free-energy landscape of a system. They correspond to stable or long-lived configurations that are separated by energy barriers. In a chemical reaction, metastable states can represent reactants, products, or intermediate species. The minimum free-energy path connects these metastable states by describing the most probable transition pathway between them.

3. Enhanced-sampling techniques: Enhanced-sampling techniques are a class of methods used to accelerate the exploration of a system's configuration space and improve the convergence of free-energy calculations. These methods overcome the limitations of standard MD simulations, which can become trapped in metastable states and fail to sample rare events or transitions between states. Some common enhanced-sampling techniques include:

    - Umbrella sampling: This method involves applying a biasing potential along the reaction coordinate (defined by the CVs) to force the system to explore different regions of the free-energy landscape. The biased simulations are then combined using the weighted histogram analysis method (WHAM) to reconstruct the unbiased free-energy profile.
    - Metadynamics: Metadynamics is a method that adds a history-dependent biasing potential to the system's Hamiltonian, which is a function of the CVs. This biasing potential discourages the system from revisiting previously explored regions of the configuration space, thus promoting the exploration of new regions and accelerating the sampling of rare events.
    - Replica exchange molecular dynamics (REMD): REMD is a technique that involves running multiple parallel simulations at different temperatures or biasing potentials. Periodically, replicas are exchanged between neighboring simulations based on a Metropolis criterion, which enhances the sampling of the configuration space and improves the convergence of free-energy calculations.
    - Transition path sampling (TPS): TPS is a method that focuses on sampling reactive trajectories, i.e., trajectories that connect the reactant and product states. It involves generating an ensemble of reactive trajectories using a Monte Carlo algorithm and then analyzing these trajectories to compute the minimum free-energy path and other reaction properties.

Prompt:

Thank you for the reply. I'm learning an advanced method called the String method, which is aiming to calculate the minimum free-energy path of a chemical reaction. Could you introduce (1) what is this method (2) the general workflow for this method (3) the strengths and weaknesses of this method to find the minimum free-energy path?

GPT-4:

1. What is the String method?: The String method is an advanced computational technique used to find the minimum free-energy path (MFEP) of a chemical reaction or a conformational change in a molecular system. It is based on the idea of representing the reaction pathway as a "string" of points in the collective variable (CV) space, which is discretized into a series of images or "beads." The method aims to evolve this string towards the MFEP by iteratively updating the positions of the beads and minimizing the free-energy along the path.

2. General workflow for the String method: The general workflow for the String method can be summarized in the following steps:

    (a) Define the collective variables (CVs) that describe the reaction coordinate or the conformational change of interest.

    (b) Initialize the string by creating a series of images (beads) that span the CV space between the initial (reactant) and final (product) states. This can be done by linear interpolation or by using an initial guess for the pathway.

    (c) Perform a short molecular dynamics (MD) or Monte Carlo (MC) simulation for each bead, keeping the CVs fixed. This step is called the "equilibration" or "relaxation" of the beads and serves to bring the system to a local minimum in the orthogonal space to the CVs.

    (d) Update the positions of the beads along the CV space based on the average forces or gradients obtained from the equilibration step. This step is called the "string update" or "path optimization" and moves the beads towards the MFEP.

    (e) Re-parametrize the string to maintain equal spacing between the beads along the path. This step is called "reparametrization" or "path redefinition."

    (f) Repeat steps c-e until the string converges to the MFEP, i.e., the positions of the beads and the free-energy profile along the path do not change significantly between iterations.

3. Strengths and weaknesses of the String method:

    - Strengths
        (a) The String method is a robust and efficient technique for finding the MFEP, as it does not require the computation of the full free-energy landscape or the sampling of rare events.

Figure 4.21: Evaluation on advanced concepts in MD II.

GPT-4 (continue)

GPT-4:

3. • (b) The method is applicable to a wide range of molecular systems and can be combined with various enhanced-sampling techniques, such as metadynamics or umbrella sampling, to improve the convergence and accuracy of the calculations.
   (c) The String method can handle complex, high-dimensional CV spaces and can be used to study multiple reaction pathways or conformational changes simultaneously.

   • Weaknesses
   (a) The choice of CVs is critical for the success of the String method, as it determines the accuracy and convergence of the MFEP. Poorly chosen CVs can lead to incorrect or incomplete pathways.
   (b) The method requires a good initial guess for the reaction pathway, which can be challenging to obtain for complex systems or reactions with unknown mechanisms.
   (c) The convergence of the String method can be slow, especially for systems with rough free-energy landscapes or large energy barriers. In such cases, the method may need to be combined with enhanced-sampling techniques or other advanced algorithms to improve the efficiency and reliability of the calculations.

These concepts and techniques are essential for studying chemical reactions and conformational changes in complex molecular systems using MD simulations. By understanding and applying these methods, you can compute minimum free-energy paths and gain insights into the mechanisms and kinetics of the processes of interest.

Prompt:

Thanks for the information. Could you also provide some other advanced methods that can overcome these weaknesses or improve the MD simulations for this MFEP problem?

GPT-4:

Certainly! Here are some advanced methods that can overcome the weaknesses of the String method or improve the MD simulations for finding the minimum free-energy path (MFEP):

1. Nudged Elastic Band (NEB) method: The NEB method is similar to the String method but uses a different approach to update the positions of the beads along the path. In NEB, a spring-like force is applied between neighboring beads to maintain equal spacing, and the beads are moved along the direction of the force orthogonal to the path. This method can improve the convergence and stability of the MFEP calculations, especially for systems with rough free-energy landscapes.

2. Growing String method: The Growing String method is an extension of the String method that constructs the reaction pathway incrementally, starting from the reactant state. At each iteration, a new bead is added to the path, and the positions of the existing beads are updated using a combination of local optimization and path reparametrization. This method can overcome the need for a good initial guess of the reaction pathway and can be more efficient for systems with large energy barriers or complex mechanisms.

3. Transition Interface Sampling (TIS): TIS is an advanced sampling technique that focuses on the interfaces between the reactant and product states, defined by a series of CV values. By performing short MD simulations that cross these interfaces, TIS can efficiently sample the reactive trajectories and compute the MFEP without the need for a predefined reaction pathway. This method can be particularly useful for systems with unknown mechanisms or poorly chosen CVs.

4. Adaptive Biasing Force (ABF) method: The ABF method is an enhanced-sampling technique that estimates the free-energy gradient along the CVs by averaging the forces acting on the system. By applying a biasing force that counteracts this gradient, ABF can accelerate the exploration of the configuration space and improve the convergence of the MFEP calculations. This method can be combined with the String method or other path-finding algorithms to overcome the limitations of standard MD simulations.

5. On-the-fly Probability Enhanced Sampling (OPES): OPES is a metadynamics-based method that adaptively constructs a biasing potential along the CVs based on the probability distribution of the system. By continuously updating the biasing potential during the simulation, OPES can efficiently sample the free-energy landscape and compute the MFEP without the need for a predefined reaction pathway or a large number of beads. This method can be particularly useful for systems with complex CV spaces or multiple reaction pathways.

These advanced methods can address some of the weaknesses of the String method and improve the MD simulations for finding the MFEP in complex molecular systems. By understanding and applying these techniques, you can gain deeper insights into the mechanisms and kinetics of chemical reactions and conformational changes.

Figure 4.22: Evaluation on advanced concepts in MD III.

In all the conceptual checking examples, GPT-4 provides correct introductions of MD and explanations of the related concepts, ranging from basic terms (e.g. ensembles, integrators, force fields) to specific methods and applications (e.g. advanced simulation approach for MFEP[92]). It is also impressive that GPT-4 could point out the strengths and weaknesses of the string method and further provide other advanced simulation methods and enhanced sampling schemes, which usually require a deep understanding of the field.

4.3.2 Assistance with simulation protocol design and MD software usage

In the next example, we further examine the ability of GPT-4 to assist human researchers in designing a reasonable MD simulation protocol for a single-stranded RNA (ssRNA) in solution and provide a general workflow to run this simulation using some computational chemistry software by a series of prompts (Fig. 4.23-4.27). We first check if GPT-4 could propose reasonable general simulation protocols in vacuum and solution. Overall, GPT-4 provides reasonable workflows for the MD simulations in vacuum and solution, suggesting it could provide some general guidance for chemists. There is also no specification for what target properties we would like to obtain. Without explicit specifications, it is reasonable to choose NVT for vacuum and NPT (system equilibration) and then NVT for solution simulations, respectively. It is also common to use an NPT ensemble along the entire production runs since it is more comparable with the experimental data, which are usually measured under constant pressure. GPT-4 also points out the importance of properly incorporating long-range electrostatics in the simulations, which is also considered an important aspect of real computational chemistry research. [70]

Simulating ssRNA systems to obtain their properties, such as equilibrium conformations and solvation energies, requires running the simulations in solution. The following question implicitly assumes simulations in solution and seeks to collect some suggestions on how to choose appropriate force fields and water models. This set of knowledge usually requires at least graduate-level training and expertise in MD simulations. The suggestions provided by GPT-4 are the commonly used force fields and water models in MD simulations. Although one could run a simulation with AMBER ff99SB with lower accuracy, it is considered a protein force field and might not be the most suitable choice for RNA simulations. It is an older version of the AMBER force fields [13], and the most recent version is AMBER ff19SB [82]. GPT-4 provides a good review of the key features of the three listed water models. Based on GPT-4's evaluations, one should consider using OPC water for the best accuracy. This evaluation agrees with the literature suggestion [79].

There are also some other force fields parameterized for better accuracy for DNA and/or RNA [79, 7] (referred to as NA FF), and one example is the revised AMBERff14 force fields by DE Shaw research team [79] mentioned in the prompt in Fig. 4.26. We finally request GPT-4 to assist the human researcher in running the simulation via GROMACS by generating a series of GROMACS commands. To run MD simulations for a biomolecule in water, the general command steps in GROMACS provided by GPT-4 are correct. Following the above general steps, GPT-4 provides useful commands with default options in most cases, however, it still has some settings that could be improved. For instance, in the `ions.mdp`, `nsteps` (the maximum number of minimization steps) is set to 0 by GPT-4. We typically consider minimizing the energies, adding some cut-offs for interactions, and describing how to find neighbor atoms. Similarly, the `md.mdp` file suggested by GPT-4 should also include some specifications on the bond parameters, neighbor searching, electrostatics, etc.

In conclusion, GPT-4 has the ability to provide useful guidance to assist the human researcher in designing MD simulation protocols, preparing some input files, and providing software commands. However, it might miss several technical details and only provide general suggestions instead of more problem-specific ones when there are multiple choices available.

GPT-4

System message:

You are an expert in Molecular Dynamics (MD) simulations, and you want to simulate some biomolecular systems using appropriate simulation methods and settings.

Figure 4.23: System message for the example of assisting human researcher on MD simulation design and workflow practical system

GPT-4

Prompt:

Could you suggest some appropriate MD simulation protocols to simulate a short single-stranded RNA (for instance, a tetramer) in vacuum and solution?

GPT-4:

Certainly! Molecular Dynamics (MD) simulations are a powerful tool for studying the behavior of biomolecules like RNA. Here, I will outline two protocols for simulating a short single-stranded RNA (tetramer) in vacuum and in solution.

1. Simulation in vacuum:
   (a) System preparation:
      - Build the RNA tetramer using a molecular modeling software or obtain its structure from a database (e.g., PDB).
      - Make sure the structure is protonated correctly, considering the pH of interest.
      - Minimize the energy of the system using a suitable force field (e.g., AMBER, CHARMM, or GROMOS).

   (b) Simulation settings
      - Use an NVT (constant number of particles, volume, and temperature) ensemble.
      - Set the temperature to 300 K (or the desired temperature for your study).
      - Choose an appropriate time step, typically around 1-2 fs.
      - Apply periodic boundary conditions to avoid edge effects.
      - Use a suitable force field for RNA (e.g., AMBER ff99SB or CHARMM36).

   (c) Simulation steps:
      - Perform an initial energy minimization to remove any steric clashes.
      - Equilibrate the system for a few nanoseconds.
      - Run the production MD simulation for a sufficient time to obtain meaningful results (e.g., 50-100 ns or more, depending on the system and research question).

2. Simulation in solution:
   (a) System preparation:
      - Follow the same steps as in the vacuum simulation for building and minimizing the RNA tetramer.
      - Solvate the system using a water model (e.g., TIP3P, SPC/E, or OPC) in a cubic or dodecahedral box, ensuring a minimum distance of 10-12 Å between the RNA and the box edges.
      - Add counterions (e.g., Na^+ or Cl^-) to neutralize the system and, if necessary, mimic the desired ionic strength.

   (b) Simulation settings:
      - Use an NPT (constant number of particles, pressure, and temperature) ensemble for equilibration and an NVT ensemble for production runs.
      - Set the temperature to 300 K (or the desired temperature for your study) and the pressure to 1 atm.
      - Choose an appropriate time step, typically around 1-2 fs.
      - Apply periodic boundary conditions.
      - Use a suitable force field for RNA and water (e.g., AMBER ff99SB with TIP3P water or CHARMM36 with TIP3P water).
      - Employ Particle Mesh Ewald (PME) or a similar method for long-range electrostatics.

   (c) Simulation steps:
      - Perform an initial energy minimization to remove any steric clashes.
      - Equilibrate the system in two stages: first, with position restraints on the RNA heavy atoms (e.g., 1-5 ns), and then without restraints (e.g., 5-10 ns).
      - Run the production MD simulation for a sufficient time to obtain meaningful results (e.g., 100-500 ns or more, depending on the system and research question).

Please note that these protocols are general guidelines and may need to be adjusted depending on the specific system and research question. Additionally, it is essential to validate the simulation results using experimental data or other computational methods when possible.

Figure 4.24: Evaluation on providing practical MD protocols to simulate a single-stranded RNA I.

GPT-4 (continue)

Prompt:

Thanks for the information. Could you provide some suggested force fields and water models for this problem? Please also rationalize why you think they are suitable choices.

GPT-4:

Certainly! Here are some suggested force fields and water models for simulating a short single-stranded RNA (tetramer) in solution, along with the rationale for their suitability:

1. Force fields:
   (a) AMBER ff99SB:
      - The AMBER ff99SB force field is an improved version of the original AMBER ff99 force field, which was specifically designed for nucleic acids.
      - It includes corrections to the backbone dihedral parameters, leading to better agreement with experimental data for RNA structures and dynamics.
      - The AMBER ff99SB force field has been widely used and validated for RNA simulations, making it a reliable choice for your study.

   (b) CHARMM36:
      - The CHARMM36 force field is another popular choice for simulating nucleic acids, including RNA.
      - It has been extensively parameterized and validated using experimental data and high-level quantum mechanical calculations.
      - The CHARMM36 force field provides accurate descriptions of RNA structures, dynamics, and thermodynamics, making it suitable for your study.

2. Water models:
   (a) TIP3P:
      - The TIP3P (Transferable Intermolecular Potential with 3 Points) water model is a simple, yet effective model for simulating liquid water.
      - It represents water molecules as rigid, non-polarizable entities with three interaction sites (one oxygen and two hydrogen atoms).
      - TIP3P has been widely used in combination with both AMBER and CHARMM force fields for simulating biomolecular systems, including RNA.
      - Although it may not be the most accurate water model available, its computational efficiency and compatibility with popular force fields make it a suitable choice for your study.

   (b) SPC/E:
      - The SPC/E (Extended Simple Point Charge) water model is another widely used model for simulating liquid water.
      - It improves upon the original SPC model by including a correction term for the long-range electrostatic interactions, leading to better agreement with experimental data for water properties.
      - The SPC/E water model has been successfully used in combination with various force fields for simulating biomolecular systems, including RNA.
      - Its balance between accuracy and computational efficiency makes it a suitable choice for your study.

   (c) OPC:
      - The OPC (Optimized Potential for Liquid Simulations - All-Atom Coarse-Grained) water model is a more recent and advanced model for simulating liquid water.
      - It has been parameterized to reproduce a wide range of water properties with high accuracy, making it more suitable for simulating complex biomolecular systems.
      - The OPC water model has been shown to improve the structural and dynamical properties of RNA simulations compared to simpler water models like TIP3P and SPC/E.
      - Although it may be more computationally demanding, its increased accuracy makes it a suitable choice for your study, especially if you are interested in capturing subtle effects of the solvent on RNA behavior.

In summary, the choice of force field and water model depends on the balance between accuracy and computational efficiency that best suits your research question. The AMBER ff99SB or CHARMM36 force fields, combined with the TIP3P, SPC/E, or OPC water models, are suitable choices for simulating a short single-stranded RNA in solution.

Figure 4.25: Evaluation on providing practical MD protocols to simulate a single-stranded RNA II.

GPT-4 (continue)

Prompt:
Thanks. I want to point out that the recommended force field for RNA is OL3 in Amber software. According to [79], OPC water is better than TIP3P. They also proposed a newly revised force field based on AMBER ff14 by changing the charges of some nucleobase atoms. In the paper, they showed that the revised force field was better for various RNA systems. Assume that you want to run a simulation using an optimized RNA structure as `rna.pdb`, and this literature force field is implemented in GROMACS as `rev_amber_rna.ff`. Could you describe the GROMACS files and key points within each file to run an MD simulation in solution for this RNA using GROMACS? Feel free to choose an appropriate NaCl concentration, water box size, integrator, time steps, etc.

GPT-4:
Apologies for the confusion regarding the force fields. You are correct that the AMBER OL3 force field is more suitable for RNA simulations, and the ff19SB is the latest version for proteins. The revised force field by Tan et al. (2018) is also a great choice for RNA simulations. Now, let's proceed with setting up the MD simulation in GROMACS using the optimized RNA structure (`rna.pdb`) and the literature force field (`rev_amber_rna.ff`)

1. Convert the PDB file to GROMACS format: Use the `pdb2gmx` command to generate the topology and GROMACS structure files. Since you have a custom force field (`rev_amber_rna.ff`), you need to provide the path to the force field directory.

   ```
   gmx pdb2gmx -f rna.pdb -o rna.gro -p topol.top -i posre.itp -ff [path_to_custom_force_field_directory] -
       water [water_model]
   ```

   Replace `[path_to_custom_force_field_directory]` with the actual path to the directory containing the custom force field files, and `[water_model]` with the desired water model (e.g., "opc" for OPC water).

2. Create the solvation box: Define a cubic or dodecahedral box with a minimum distance of 10-12 Å between the RNA and the box edges. `gmx editconf -f rna.gro -o rna_box.gro -c -d 1.0 -bt dodecahedron`

3. Solvate the system: Add water molecules to the box using the selected water model. Replace `[water_model]` with the desired water model (e.g., "opc" for OPC water).

   ```
   gmx solvate -cp rna_box.gro -cs [water_model].gro -o rna_solv.gro -p topol.top
   ```

4. Add ions and neutralize the system: First, create an ion.mdp file with the following content:

   ```
   ; ions.mdp - used for adding ions
   integrator = steep
   nsteps     = 0
   coulombtype = PME
   ```

   Then, generate the `tpr` file: `gmx grompp -f ions.mdp -c rna_solv.gro -p topol.top -o ions.tpr` Now, add the ions (e.g., Na^+ and Cl^-) to neutralize the system and reach the desired NaCl concentration (e.g., 0.15 M):

   ```
   gmx genion -s ions.tpr -o rna_solv_ions.gro -p topol.top -pname NA -nname CL -neutral -conc 0.15
   ```

5. Energy minimization: Create an `em.mdp` file with the following content:

   ```
   ; em.mdp - used for energy minimization
   integrator = steep
   nsteps     = 50000
   emtol      = 1000
   emstep     = 0.01
   coulombtype = PME
   ```

   Generate the `tpr` file and run the energy minimization:

   ```
   gmx grompp -f em.mdp -c rna_solv_ions.gro -p topol.top -o em.tpr
   gmx mdrun -v -deffnm em
   ```

6. Equilibration: Perform NVT and NPT equilibration using appropriate `mdp` files (`nvt.mdp` and `npt.mdp`). Generate the `tpr` files and run the equilibration steps:

   ```
   gmx grompp -f nvt.mdp -c em.gro -r em.gro -p topol.top -o nvt.tpr
   gmx mdrun -v -deffnm nvt
   gmx grompp -f npt.mdp -c nvt.gro -r nvt.gro -t nvt.cpt -p topol.top -o npt.tpr
   gmx mdrun -v -deffnm npt
   ```

Figure 4.26: Evaluation on providing practical MD protocols to simulate a single-stranded RNA III.

GPT-4 (continue)

GPT-4:

6. Production run: Create an "md.mdp" file with the following parameters for the production run:

```
integrator = md
dt         = 0.002
nsteps     = 25000000 ; for a 50 ns simulation
tcoupl     = V-rescale
ref_t      = 300
pcoupl     = Parrinello-Rahman
ref_p      = 1.0
```

   Run the production MD simulation using the "gmx grompp" and "gmx mdrun" tools:

```
gmx grompp -f md.mdp -c npt.gro -t npt.cpt -p topol.top -o md.tpr
gmx mdrun -v -deffnm md
```

7. Analysis: Use GROMACS analysis tools, such as "gmx rms", "gmx rmsf", "gmx hbond", and "gmx gyrate", to analyze the MD trajectories and investigate the behavior of the RNA molecule in solution.

This protocol provides a general workflow for running an MD simulation of an RNA molecule in solution using GROMACS and the literature force field (`rev_amber_rna.ff`). Adjust the specific parameters, time steps, and other settings as needed for your particular system and research question.

Figure 4.27: Evaluation on providing practical MD protocols to simulate a single-stranded RNA IV.

4.3.3 Development of new computational chemistry methods

Machine learning force fields (MLFFs) [6, 17, 16, 108, 53, 99] harness the power of machine learning algorithms to model the potential energy surface of molecular systems with remarkable accuracy and efficiency. By training machine learning models on reference data generated by high-level quantum mechanical calculations, MLFFs are able to interpolate between known data points and predict the energy and forces of new configurations with a fraction of the computational cost. This approach provides a significant speed-up compared to traditional *ab initio* methods while maintaining a comparable level of accuracy, thus bridging the gap between quantum mechanical accuracy and classical force field efficiency.

On one hand, we anticipate that the GPT-4 model possesses the capability to comprehend related concepts and provide guidance in this domain. On the other hand, we aim to determine if the GPT-4 model can effectively address the associated issues directly.

Firstly, as depicted in Fig. 4.28, the GPT-4 model illustrates the concept of machine learning force fields. In essence, the definition of MLFF is quite accurate, offering a comprehensive introduction to the strengths and weaknesses of classical force fields, as well as the driving factors behind MLFF's development. Moreover, considering the rapid advancement of the MLFF field, the GPT-4 model outlines the research direction of MLFF in Fig. 4.29, which appears reasonable. The insights provided by the GPT-4 model can serve as a valuable guide for developing new machine learning force fields, although they may not be exhaustive, they are helpful to a certain degree. A vital aspect of improving MLFF is the incorporation of long-range interactions. Consequently, we request the GPT-4 model to provide us with potential approaches for integrating these interactions. As illustrated in Fig. 4.30, to the best of our knowledge, there are indeed several studies that have undertaken this approach. However, it is important to note that the suggestions provided by the GPT-4 model may not encompass all possible solutions. Ultimately, assessing the performance of a trained MLFF continues to be a challenge in this domain. Fig. 4.31 demonstrates that the GPT-4 model offers some metrics for this purpose.

GPT-4

Prompt:

You are a professor with a deep understanding of physics, chemistry, materials, and biology. What is a machine learning force field?

GPT-4:

Machine learning force field (MLFF) is an approach used in computational chemistry and materials science to model and predict the interactions and behavior of atomic and molecular systems. It combines the principles of classical force fields with advanced machine learning algorithms to create more accurate and efficient models.

In traditional force fields, the potential energy of a system is expressed as a sum of different energy terms representing various interactions between atoms, such as bond stretching, angle bending, and van der Waals forces. The parameters for these terms are usually derived from experimental data or quantum mechanical calculations. However, this approach has limitations in capturing the complexity of real systems, especially when dealing with complex materials or large biomolecules.

Machine learning force fields address these limitations by employing machine learning algorithms, such as artificial neural networks, Gaussian process regression, or kernel ridge regression, to learn the relationship between atomic configurations and their corresponding potential energy or forces. The MLFF models are trained on a dataset of atomic structures and their respective energies or forces, which are typically calculated using high-level quantum mechanical methods.

Once trained, the MLFF model can predict potential energies and forces for new atomic configurations with high accuracy and at a significantly reduced computational cost compared to quantum mechanical methods. This enables researchers to perform large-scale molecular dynamics simulations, explore vast configurational spaces, and investigate complex processes that were previously inaccessible due to computational limitations.

In summary, machine learning force fields represent a promising approach to improving the accuracy and efficiency of molecular simulations, providing valuable insights into the behavior of a wide range of materials and biological systems.

Figure 4.28: Concepts of machine learning force field.

GPT-4

Prompt:

You are a professor with a deep understanding of physics, chemistry, materials, and biology. What remains the study direction for the machine learning force field?

GPT-4:

The study of machine learning force fields (MLFF) is a rapidly evolving field with several research directions that aim to improve the accuracy, applicability, and efficiency of these models. Some key study directions include:

1. Developing better training datasets: The quality and diversity of the training data are crucial for the performance of MLFF models. Researchers are working on creating more comprehensive datasets that cover a wide range of chemical environments, including various elements, structures, and configurations. This can help improve the generalizability of the MLFF models to previously unseen systems.

2. Improving machine learning algorithms: The choice of the machine learning algorithm and its architecture play a significant role in the accuracy and efficiency of the MLFF models. Researchers are exploring advanced algorithms and techniques, such as deep learning, graph neural networks, and active learning, to improve the performance of MLFFs.

3. Incorporating quantum mechanics: To achieve higher accuracy, researchers are working on developing hybrid MLFF models that combine the strengths of machine learning with quantum mechanical methods. These hybrid approaches can balance the accuracy of quantum mechanical calculations with the computational efficiency of machine learning models.

4. Transferability and scalability: One of the challenges in MLFF is the transferability of the models to different chemical systems and their scalability to larger systems. Researchers are working on developing methods to ensure that the MLFF models can be applied to a broader range of systems without losing accuracy and can handle large-scale systems more efficiently.

5. Uncertainty quantification: Estimating the uncertainties associated with MLFF predictions is essential to assess the reliability of the models. Researchers are working on developing methods to quantify and interpret uncertainties in MLFF predictions, which can help guide further model development and refinement.

6. Interpreting MLFF models: Interpretability is an essential aspect of any machine learning model, especially in scientific applications. Researchers are developing methods to understand and interpret the learned representations and decision-making processes in MLFF models to gain insights into the underlying physical and chemical principles.

7. Applications in materials discovery and design: As MLFF models become more accurate and efficient, they can be applied to a wide range of materials science and chemistry problems, including materials discovery, optimization, and design. Researchers are exploring various applications of MLFF models to accelerate the development of novel materials and understand complex molecular phenomena.

In summary, the study of machine learning force fields is a vibrant research area with various directions aimed at enhancing the performance, applicability, and interpretability of these models for a wide range of applications in chemistry, materials science, and biology.

Prompt:

Could you give some detailed guidance about these directions?

GPT-4:

Certainly! Here is a more detailed guidance on the key study directions in machine learning force fields (MLFF) research:

1. Developing better training datasets:
    - Focus on creating diverse datasets that cover a broad range of chemical environments, including various elements, molecular structures, and configurations.
    - Include data from different sources, such as experimental measurements, high-level quantum mechanical calculations, and existing databases.
    - Develop systematic approaches to sample representative configurations, considering various phases, temperatures, and pressures.
    - Use data augmentation techniques to increase the size and variety of the training data and improve the model's generalization ability.

GPT-4 (continue)

GPT-4:

2. Improving machine learning algorithms:
    - Explore different machine learning techniques, such as artificial neural networks, Gaussian process regression, kernel ridge regression, and support vector machines, to find the most suitable approach for a specific problem.
    - Investigate advanced deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to capture complex relationships between atomic configurations and energies or forces.
    - Consider graph neural networks (GNNs) to efficiently represent and process atomic structures as graphs.
    - Employ active learning strategies to iteratively refine the MLFF models by selecting the most informative data points for inclusion in the training set.
3. Incorporating quantum mechanics:
    - Develop hybrid MLFF approaches that combine the strengths of machine learning with quantum mechanical methods, such as density functional theory (DFT) or wavefunction-based methods.
    - Use machine learning to create surrogate models for computationally expensive quantum mechanical calculations, effectively reducing the computational cost while retaining accuracy.
    - Investigate methods to incorporate quantum mechanical descriptors or features into MLFF models to capture more accurate representations of the underlying physics.
4. Transferability and scalability:
    - Develop methods to ensure that MLFF models can be applied to a broader range of chemical systems without losing accuracy.
    - Investigate techniques to improve the scalability of MLFF models to large-scale systems, such as parallelization, coarse-graining, or multi-scale modeling approaches.
    - Evaluate the performance of MLFF models on diverse test sets and benchmark them against state-of-the-art classical force fields and quantum mechanical methods.
5. Uncertainty quantification:
    - Develop methods to estimate uncertainties associated with MLFF predictions, such as Bayesian techniques, ensemble learning, or bootstrapping.
    - Assess the reliability and confidence of MLFF models in predicting energies and forces for unseen configurations.
    - Use uncertainty information to guide further model development, refinement, and selection of new data points for training.
6. Interpreting MLFF models:
    - Investigate techniques to understand and interpret the learned representations and decision-making processes in MLFF models, such as feature importance analysis, sensitivity analysis, or visualization methods.
    - Develop methods to extract physically meaningful insights from MLFF models, such as identifying key atomic interactions or discovering new structure-property relationships.
7. Applications in materials discovery and design:
    - Apply MLFF models to accelerate materials discovery, optimization, and design processes.
    - Use MLFF models to study complex molecular phenomena, such as phase transitions, chemical reactions, or mechanical properties, that are difficult to capture with traditional force fields or computationally expensive with quantum mechanical methods.
    - Collaborate with experimentalists and domain experts to validate the predictions and insights obtained from MLFF models and guide experimental efforts.

By focusing on these directions, researchers can advance the field of machine learning force fields and develop more accurate, efficient, and interpretable models for a wide range of applications in chemistry, materials science, and biology.

Figure 4.29: Guidance of the GPT-4 model for machine learning force field.

GPT-4

Prompt:
You are a professor with a deep understanding of physics, chemistry, materials, and biology. Could you give me a detailed introduction to the method to incorporate long-range interactions into MLFF?

GPT-4:
Certainly! Here's a detailed introduction to incorporating long-range interactions into machine learning force fields (MLFF) using various methods:

1. Explicit long-range interaction terms:
   - Identify the relevant long-range interactions for the system under study, such as electrostatic interactions, van der Waals forces, or dispersion interactions.
   - Choose appropriate functional forms for these interactions. For electrostatic interactions, the Coulomb potential is commonly used, while for van der Waals forces, Lennard-Jones or Buckingham potentials may be employed.
   - Include these explicit interaction terms in the MLFF model, either as separate components of the total potential energy or as additional features for the ML model.
   - Train the ML model to predict the parameters of these long-range interaction terms based on the local atomic environment, chemical composition, or other relevant properties of the system.
2. Long-range features/descriptors:
   - Develop features or descriptors that capture long-range interactions in the system. Some examples include:
     i. Pairwise distance-based features that account for interactions between atoms beyond a specified cutoff radius.
     ii. Multipole moments (e.g., dipole, quadrupole) that represent the electrostatic interactions between distant charges.
     iii. Environment-dependent charges, such as those obtained from Ewald sums or other charge equilibration methods.
   - Normalize or scale long-range features to ensure they have similar magnitude and dimensions as the short-range features.
   - Use these long-range features as input to the MLFF models, allowing the models to learn the relationship between long-range interactions and the potential energy or forces.
3. Smooth cutoff functions:
   - Choose a smooth cutoff function that decays to zero beyond a certain cutoff radius, such as a cosine-based function or a polynomial-based function.
   - Apply this cutoff function to the features or descriptors used in the MLFF models, ensuring that the influence of distant atoms on the local atomic environment is considered in a continuous and differentiable manner.
   - Train the MLFF models using the modified features, allowing the models to learn the effect of long-range interactions on the potential energy or forces.
4. Coarse-graining or multi-scale modeling:
   - Identify the relevant length scales and interactions that need to be considered for the system under study.
   - Develop a coarse-graining or multi-scale modeling approach that combines the MLFF models with more computationally efficient methods for long-range interactions. Examples include the particle-mesh Ewald (PME) method for electrostatic interactions or the smooth particle mesh (SPME) method for van der Waals forces.
   - Implement the coarse-graining or multi-scale modeling approach in the MLFF framework, ensuring that the different levels of interactions are correctly coupled and accounted for in the total potential energy or forces.
   - Train and validate the MLFF models in the context of the coarse-graining or multi-scale modeling approach, ensuring that the models accurately capture both short-range and long-range interactions.
5. Hierarchical learning:
   (a) Design a hierarchical MLFF model that learns different levels of interactions in the system, with separate models for short-range and long-range interactions.
   (b) Define local and non-local features for the system that capture both short-range and long-range effects, respectively.
   (c) Train separate ML models for short-range and long-range interactions using the local and non-local features.
   (d) Combine the predictions from the short-range and long-range ML models to obtain the total potential energy or forces for the system.

By incorporating long-range interactions into MLFF models using these methods, researchers can develop more accurate and realistic representations of atomic and molecular systems, which can better capture the behavior of complex systems influenced by both short-range and long-range forces.

Figure 4.30: Guidance of the GPT-4 model for incorporating long-range interactions into machine learning force field.

GPT-4

Prompt:

You are a professor with a deep understanding of physics, chemistry, materials, and biology. How to evaluate the performance of a trained machine learning force field?

GPT-4:

Evaluating the performance of a trained machine learning force field requires a systematic approach that considers various metrics and tests. Here are the key steps to follow:

1. Data Splitting: Split the data into training, validation, and test sets. The training set is used to train the force field, the validation set is used to optimize hyperparameters and prevent overfitting, and the test set is used to evaluate the final performance.

2. Cross-validation: Perform k-fold cross-validation to ensure the robustness of your model. Divide the data into k equal parts, train on k-1 parts, and validate on the remaining part. Repeat this process k times and average the results.

3. Comparison with reference data: Compare the predictions of the machine learning force field with reference data (e.g., experimental data, results from quantum chemistry calculations, or other well-established force fields). Calculate errors between predicted and reference values for various properties of interest, such as energies, forces, and geometries.

4. Evaluation Metrics: Use a range of evaluation metrics to assess the performance of the force field. Common metrics include:

    - Mean Absolute Error (MAE): Measures the average magnitude of errors between predicted and reference values.
    - Root Mean Square Error (RMSE): Indicates the square root of the average squared difference between predictions and reference values.
    - R-squared (R^2): Represents the proportion of variance in the reference values that is predictable by the machine learning force field. A value closer to 1 indicates a better fit.
    - Pearson Correlation Coefficient (PCC): Measures the linear correlation between predicted and reference values. A value closer to 1 indicates a strong positive correlation.

5. Benchmarking: Compare the performance of your machine learning force field with other state-of-the-art force fields or methods for the same system. This will provide an understanding of how well your model performs compared to existing approaches.

6. Validation on diverse systems: Test the force field on a diverse set of molecular systems and materials to assess its generalization capability and transferability to different environments.

7. Stability and reliability: Perform molecular dynamics simulations using the machine learning force field and analyze the stability and reliability of the trajectories. Verify that the simulations do not exhibit unphysical behavior and are consistent with expectations.

8. Reproducibility: Ensure that the training process and the evaluation methodology are well-documented and reproducible, enabling other researchers to verify and build upon your results.

By considering these steps and metrics, you can effectively evaluate the performance of a trained machine learning force field and gain insights into its strengths and weaknesses for various applications.

Figure 4.31: Guidance of the GPT-4 model for evaluating the performance of machine learning force field.

Furthermore, we examine the GPT-4 model's performance using the MD17 dataset [17], which consists of simulation trajectories for seven distinct small molecules. Our initial focus is on the model's zero-shot learning ability to handle scientific numerical problems, as depicted in Fig. 4.32. While the GPT-4 model delivers a step-by-step solution process, it refrains from divulging precise numerical values. Nonetheless, as indicated in Table 8, Appendix Table 16-Appendix Table 18, providing at least one example allows the GPT-4 model to potentially reveal detailed values. In Table 8, when only a single example is given, the mean absolute errors (MAEs) for energies are significantly high. As additional examples are presented, the energy MAEs show a decreasing trend. Although the MAEs are substantially greater than those of state-of-the-art benchmarks, the GPT-4 model, when furnished with more examples, may gain awareness of the energy range and predict energies within a relatively reasonable range when compared to predictions based on only one example. However, the MAEs of forces remain nearly constant, regardless of the number of examples provided, due to the irregular range of forces when compared to energies.

Molecule	MAE	1 example	2 examples	3 examples	4 examples
Aspirin	Energy	11197.928 (6.428)	84.697 (5.062)	5.756 (4.595)	5.190 (4.477)
	Forces	27.054 (29.468)	26.352 (25.623)	27.932 (24.684)	27.264 (24.019)
Ethanol	Energy	317.511 (3.682)	3.989 (3.627)	3.469 (3.609)	3.833 (3.612)
	Forces	26.432 (27.803)	24.840 (23.862)	25.499 (23.200)	25.533 (22.647)
Malonaldehyde	Energy	7002.718 (4.192)	34.286 (5.191)	4.841 (4.650)	4.574 (4.235)
	Forces	25.379 (30.292)	24.891 (24.930)	27.129 (25.537)	27.481 (24.549)
Naphthalene	Energy	370.662 (5.083)	7.993 (5.063)	7.726 (5.507)	9.744 (5.047)
	Forces	27.728 (28.792)	26.141 (26.327)	26.038 (24.712)	25.154 (23.960)
Salicylic acid	Energy	115.931 (4.848)	5.714 (3.793)	4.016 (3.783)	4.074 (3.828)
	Forces	28.951 (29.380)	26.817 (26.907)	28.372 (24.227)	27.161 (23.467)
Toluene	Energy	2550.839 (4.693)	7.449 (4.183)	4.063 (4.247)	4.773 (3.905)
	Forces	27.725 (29.005)	25.083 (24.086)	26.046 (23.095)	26.255 (22.834)
Uracil	Energy	1364.086 (4.080)	5.971 (4.228)	4.593 (4.322)	4.518 (4.075)
	Forces	28.735 (28.683)	27.865 (26.140)	26.671 (24.492)	27.896 (23.665)

Table 8: The mean absolute errors (MAEs) of GPT-4 on 100 random MD17 data points with different numbers of examples provided (energies in kcal/mol and forces in kcal/(mol·Å)). The numerical values in parentheses are the MAEs calculated using the average value of the examples as the predicted value for all cases.

Moreover, as symmetry is a crucial property that MLFFs must respect, we investigate whether the GPT-4 model can recognize symmetry. The first experiment involves providing different numbers of examples and their variants under rotation, followed by predicting the energies and forces of a random MD17 data point and its variant under a random rotation. As shown in Appendix Table 16, most of the energy MAEs with only one example (accounting for the original data point and its variant, there are two examples) are 0, which seemingly implies that the GPT-4 model is aware of rotational equivariance. However, when more examples are provided, some energy MAEs are not 0, indicating that the GPT-4 model recognizes the identical energy values of the original data point and its rotational variant only when a single example is given.

To validate this hypothesis, two additional experiments were conducted. The first experiment predicts the energies and forces of a random MD17 data point and its variant under a random rotation, given different numbers of examples (Appendix Table 16). The second experiment predicts the energies and forces of 100 random MD17 data points with varying numbers of examples and their random rotational variants (Appendix Table 18). The non-zero energy MAE values in the first experiment, along with the similar energy MAE trend observed in the second experiment compared to that in Table 8, support our suggestion. The detailed energy distribution of different molecules and experimental results can be found in the Supplementary Materials.

In conclusion, GPT-4 model has a relatively comprehensive understanding and support for concepts and ideas, but there is still room for improvement in numerical tasks. However, considering that GPT-4 has not been trained or fine-tuned on relevant datasets, this model still has the potential for handling related tasks.

GPT-4

Prompt:

You are an empirical force field and should be able to help with molecular prediction tasks. Give you the coordinates and atomic numbers of a molecule, you should calculate the energy of the whole molecule and forces exerted on each atom. The output should be a float number and a matrix with the shape of (number of atoms, 3), which is the energy and forces respectively.

C 2.24227707 -0.37935822 0.26140399

C 0.84137205 1.9353399 -0.42284114

C 2.87295184 0.84559048 0.27086148

C 2.16921384 1.99995568 -0.06569247

C -3.48401941 0.49330164 -0.09220199

C 0.89610449 -0.47342667 -0.09561814

C 0.19395166 0.6969083 -0.43774992

O -0.96282594 -1.83980977 -0.41481881

O -1.6535462 0.88363924 1.34030532

O 0.88297792 -2.88511553 0.21964887

C 0.2153679 -1.77627053 -0.10291132

C -2.02464776 0.68755345 0.21478052

O -1.1188447 0.62630819 -0.79181512

H 0.37350889 -3.74323511 0.25668924

H 2.75812642 -1.30088993 0.52002845

H 0.31271811 2.79275776 -0.72906669

H 3.92821103 0.9208166 0.55869356

H 2.68028429 2.92527084 -0.1006783

H -3.76348366 -0.56217649 -0.01954804

H -4.06835602 1.11919497 0.59195825

H -3.68524589 0.86877234 -1.08640859

GPT-4:

To calculate the energy of the whole molecule and the forces exerted on each atom, I will use a simple empirical force field method. Please note that this is a simplified approach and more accurate methods like quantum mechanics should be considered for precise results.

For the energy calculation, I will use the following formula:

$$E = \sum k_{\mathbf{bond}}(d - d_0)^2 + \sum k_{\mathbf{angle}}(\theta - \theta_0)^2 + \sum(\epsilon_{ij}[(\sigma_{ij}/r_{ij})^{12} - 2(\sigma_{ij}/r_{ij})^6]) \tag{6}$$

where E is the energy, $k_{\mathbf{bond}}$ and $k_{\mathbf{angle}}$ are the force constants for bond stretching and angle bending respectively, d and d_0 are the current and equilibrium bond lengths, θ and θ_0 are the current and equilibrium bond angles, ϵ_{ij} and σ_{ij} are the Lennard-Jones parameters, and r_{ij} is the distance between atoms i and j.

For the force calculation, I will use the following formula:

$$F_i = -\nabla E_i \tag{7}$$

where F_i is the force on atom i and ∇E_i is the gradient of the energy with respect to the position of atom i.

Given the coordinates and atomic numbers of the molecule, I can perform these calculations. However, the actual numerical values of the force constants, equilibrium bond lengths, and Lennard-Jones parameters depend on the specific force field being used. There are several widely used force fields, such as CHARMM, AMBER, and OPLS-AA, each with its own set of parameters.

As an AI, I am unable to perform these calculations directly, so I recommend using molecular modeling software like GROMACS, NAMD, or LAMMPS with a suitable force field to obtain the energy and forces for your molecule.

Figure 4.32: Zero-shot learning of GPT-4 on an aspirin molecule.

4.3.4 Chemical reaction optimization

Reaction optimization in chemistry is a crucial process that aims to enhance the efficiency and selectivity of chemical reactions. By systematically exploring various reaction conditions and parameters, researchers can fine-tune the reaction to achieve optimal results. This iterative reaction optimization process involves adjusting factors such as temperature, pressure, catalyst type, solvent, and reactant concentrations to maximize desired product formation while minimizing unwanted byproducts. Reaction optimization not only improves yield and purity but also reduces cost and waste, making it a valuable tool for synthetic chemists. Through careful experimentation and data analysis, scientists can uncover the ideal reaction conditions that lead to faster, more sustainable, and economically viable chemical transformations.

The traditional approaches used by chemists to optimize reactions involve changing one reaction parameter at a time (e.g., catalyst type) while keeping the other parameters constant (e.g., temperature, concentrations, and reaction time), or searching exhaustively all combinations of reaction conditions (which is obviously very time-consuming, requires significant resources, and is typically an environmentally unfriendly process, making it expensive in multiple aspects).

Recently, new approaches based on machine learning were proposed to improve the efficiency of optimizing reactions [112, 26]. For instance, reaction optimization using Bayesian optimization tools has emerged as a powerful approach in the field of chemistry [77, 90, 84, 58]. By integrating statistical modeling and machine learning techniques, Bayesian optimization enables researchers to efficiently explore and exploit the vast parameter space of chemical reactions. This method employs an iterative process to predict and select the next set of reaction conditions to test based on previous experimental results. With its ability to navigate complex reaction landscapes in a data-driven fashion, Bayesian optimization has revolutionized reaction optimization, accelerating the discovery of optimized reaction conditions and reducing the time and resources required for experimentation.

In our investigation, we aim to evaluate GPT-4 as a potential tool for optimizing chemical reactions. In particular, we will test the model's ability to suggest optimal reaction conditions from a set of parameters (e.g., temperature, catalyst type, etc.) under certain constraints (e.g., range of temperatures, set of catalyst, etc.). To establish a benchmark for the model's efficacy, we will use three distinct datasets published by the Doyle group as ground truth references (see datasets details in [77]).

Our optimization approach utilizes an iterative process that is built around the capabilities of GPT-4. We start by defining a search space for the reaction conditions, allowing GPT-4 to suggest conditions that maximize the yield of the reaction, subject to the constraints of the reaction scope. Following this, we extract yield values from the ground truth datasets, which are then fed back into GPT-4 as a k-shot learning input. Subsequently, we request GPT-4 to generate new conditions that would enhance the yield. This iterative procedure continues until we have conducted a total of 50 algorithm recommendations, each time leveraging the model's ability to build upon previous data inputs and recommendations.

In Fig. 4.33 we present one case of a GPT-4 reaction optimization. In this case, we ask GPT-4 to suggest new experimental conditions to maximize the reaction yield of a Suzuki reaction after including two examples of reaction conditions and their corresponding target yields.

For comparative analysis, we assess GPT-4's performance with an increasing number of examples against two different sampling strategies. The first, known as the Experimental Design via Bayesian Optimization (EDBO) method [77, 84], serves as a Bayesian optimizer that attempts to maximize performance within a pre-defined parameter space. The second approach, intended as our baseline, involves a random sampling algorithm. This algorithm blindly selects 50 unique samples from the ground truth datasets without considering any potential learning or optimization opportunities. Through this comparative approach, we aim to establish a general understanding of GPT-4's capabilities in the domain of reaction optimization. As part of our assessment of GPT-4's proficiency in reaction optimization, we will examine the performance of the model under four distinct prompting strategies (see Fig. 4.34 for examples of the different strategies). The first of these involves using traditional chemical formulas, which provide a standardized method for expressing the chemical constituents and their proportions in a compound. The second approach uses the common names for chemical species, which provide an easily understood language for describing chemicals, though it may lack the precision of more formal nomenclature systems. The third mode of prompting is based on the Simplified Molecular-Input Line-Entry System (SMILES) string representation. SMILES is a chemical notation system that uses ASCII strings to represent chemical structures, providing a detailed yet compact way of expressing complex chemical information. Lastly, we use categorical values to communicate chemical information to GPT-4. This approach involves assigning each chemical or class of chemicals a discrete unique label, allowing

GPT-4

System message:

You are an AI assistant that is an organic chemistry and reaction optimization expert. Your objective is to propose reaction conditions to maximize the yield of a Suzuki reaction. The search space is limited to combinations from the following reaction conditions:
Category Aryl_halide: 1−chloro-4−(trifluoromethyl)benzene, 1−bromo-4−(trifluoromethyl)benzene, 2−iodopyridine, ...
Category Additive: 5−phenylisoxazole, ethyl−isoxazole-4−carboxylate, ethyl−5−methylisoxazole−3−carboxylate, ...
Category Base: P2Et, BTMG, MTBD, ...
Category Ligand: Pd0−X−Phos, Pd0−t−Bu−X−Phos, Pd0−t−Bu−BrettPhos, Pd0−Ad−BrettPhos, ...

Prompt:

Return only the most promising reaction conditions.
Your answers must be in the list of Python dictionaries for the candidates with the following keys [‘Aryl_halide’, ‘Additive’, ‘Base’, ‘Ligand’].

Example 0: ‘Aryl_halide’: ‘2−iodopyridine’, ‘Additive’: ‘ethyl−isoxazole−4−carboxylate’, ‘Base’: ‘MTBD’, ‘Ligand’: ‘Pd0−t−Bu−X−Phos’, ‘yield’: 51.153

Example 1: ‘Aryl_halide’: ‘1−bromo−4−(trifluoromethyl)benzene’, ‘Additive’: ‘5−phenylisoxazole’, ‘Base’: ‘P2Et’, ‘Ligand’: ’Pd0−X−Phos’, ‘yield’: 14.748

{... more examples, new reaction conditions, and their corresponding reaction yields are appended at each iteration.}

GPT-4:

{GPT-4 returns a dictionary with the suggested reaction conditions:}
‘Aryl_halide’: ‘1−chloro−4−(trifluoromethyl)benzene’, ‘Additive’: ‘5−phenylisoxazole’, ‘Base’: ‘P2Et’, ‘Ligand’: ‘Pd0−X−Phos’

Figure 4.33: Example GPT-4 for reaction optimization.

the model to process chemical information in a highly structured, abstracted format but lacking any chemical information. By investigating the performance of GPT-4 across these four distinct prompting schemes, we aim to gain a comprehensive understanding of how different forms of chemical information can influence the model's ability to optimize chemical reactions.

A summary of the different methods and features used in this study is shown in Fig. 4.34 along with the performance of the different methods and feature combinations for suggesting experimental conditions with high yield values. On the evaluation of the three datasets employed, the BayesOpt (EDBO) algorithm consistently emerges as the superior method for identifying conditions that maximize yield. GPT-4, when prompted with most features, surpasses the efficiency of random sampling. However, a notable exception to this trend is the aryl amination dataset, where the performance of GPT-4 is significantly subpar. The efficacy of GPT-4 appears to be notably enhanced when prompted with "common name" chemical features, compared to the other feature types. This finding suggests that the model's optimization capability may be closely tied to the specific nature of the chemical information presented to it.

Plot label	Method	Features	Examples
1	Random sampling	N/A	N/A
2	BayesOpt (EDBO) *	Categorical	Solvent_A; Solvent_B; Ligand_A...
3	GPT-4	Categorical	Solvent_A; Solvent_B; Ligand_A...
4	GPT-4	Formula	C9H7NO; C12H35N7P2; C31H49O2PPd...
5	GPT-4	Common name	1-chloro-4-(trifluoromethyl)benzene; BTMG; Pd0-X-Phos...
6	GPT-4	SMILES	FC(F)(F)c1ccc(Cl)cc1,o1nccc1c2ccccc2; Ic1ccccn1,o1cc(cn1)c2ccccc2; ...

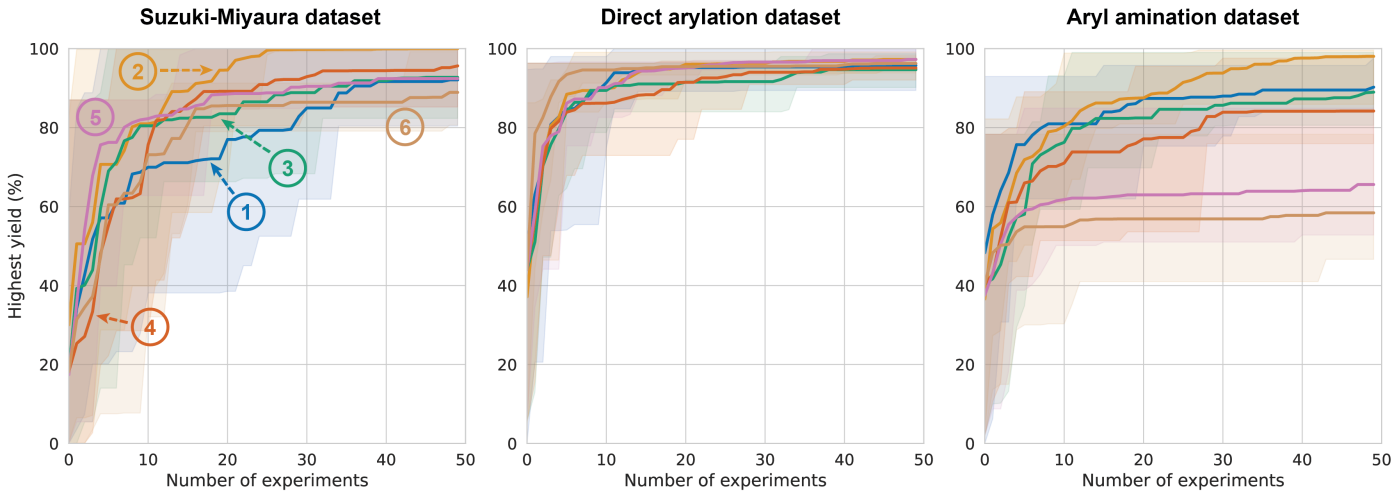


Figure 4.34: Summary of methods and features used in this study and their corresponding reaction optimization performance. The average of the performance values of the different combinations of methods and features are shown by the colored lines while the shaded regions show the lower and upper performance of each method/feature combination. The cumulative average and max/min performance values are computed using 10 optimization campaigns using different random starting guesses for each method/feature combination.

To compare various methods and features, we also analyze the yield values of the samples gathered by each algorithm. In Fig. 4.35, we present box plots representing the target yields obtained at differing stages of the optimization process for the three aforementioned datasets. Our goal in this form of data analysis is to study the ability of each algorithm to improve its recommendations as the volume of training data or examples increases. In the case of the BayesOpt algorithm, the primary concern is understanding how it evolves and improves its yield predictions with increasing training data. Similarly, for GPT-4, the focus lies in examining how effectively it utilizes few-shot examples to enhance its yield optimization suggestions.

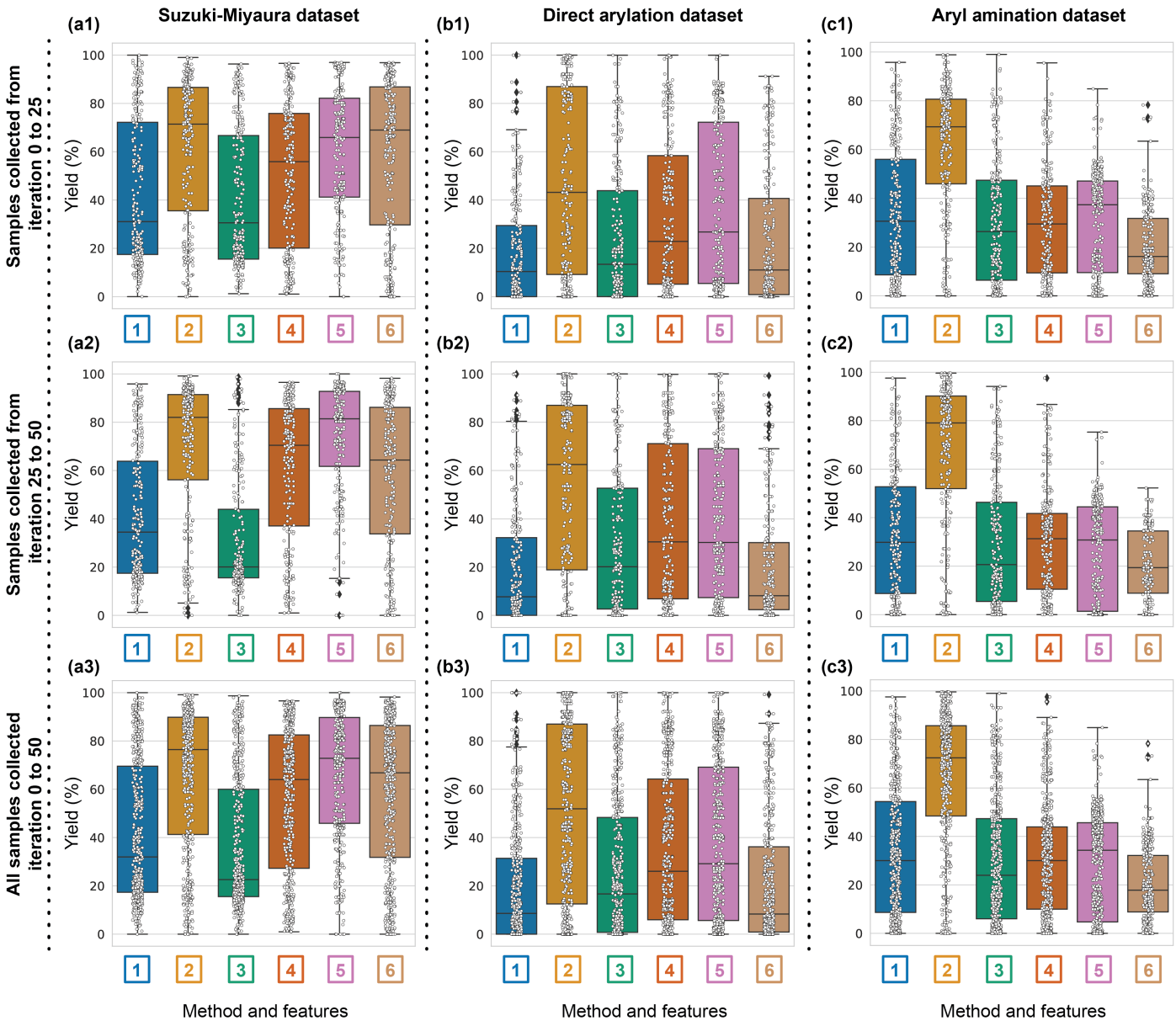


Figure 4.35: Box plots for the samples collected during the optimizations. The white circles in each box plot represent the yield values of the samples collected for the (a1-a3) Suzuki-Miyaura, (b1-b3) direct arylation, and (c1-c3) aryl amination reactions at the following states of their optimization campaigns: (a1, b1, c1) the early stages of the optimization (samples collected on the first 25 iterations), (a2, b2, c2) late stage (samples collected from the 25th iteration until the 50th iteration) and (a3, b3, c3) all the samples collected in the entire optimization campaign (from 0 to 50 iterations).

Our analysis reveals that both the BayesOpt (EDBO) and GPT-4 algorithms demonstrate a propensity to produce higher yield samples in the later stages of the optimization process (from iteration 25 to 50) as compared to the initial stages (from iteration 0 to 25). This pattern contrasts with the random sampling method, where similar yield values are collected in both the early and late stages of the optimization campaigns, which is something to expect when collecting random samples. While definitive conclusions are challenging to draw from this observation, it provides suggestive evidence that the GPT-4 algorithms are able to effectively incorporate the examples provided into its decision-making process. This adaptability seems to enhance the predictive capabilities of the model, enabling it to recommend higher yield samples as it gathers more examples over time. Therefore, these findings offer a promising indication that GPT-4's iterative learning process may indeed contribute to progressive improvements in its optimization performance. We additionally noted

that the GPT-4 algorithms, when using the chemical "common name" chemical features (see algorithm 5 in Figure 2), surpass the performance of their counterparts that employ other features, as well as the random sampling method, in terms of proposing high-yield samples in both early and late stages of the optimization campaigns. Remarkably, when applied to the Suzuki dataset, GPT-4's performance using "common name" features aligns closely with that of the BayesOpt (EDBO) algorithm. The differential performance across various datasets could be attributed to a multitude of factors. Notably, one such factor could be the sheer volume of available data for the Suzuki reaction, hinting at the possibility that as the amount of accessible data to GPT-4 increases, the model's predictive and suggestive capabilities might improve correspondingly. This insight highlights the critical importance of the quantity and quality of data fed into these models and underscores the potential of GPT-4 to enhance reaction optimization with an increasing amount of literature data accessible when training the GPT models.

4.3.5 Sampling bypass MD simulation

Molecular Dynamics (MD) simulations play a pivotal role in studying complex biomolecular systems; however, in terms of depicting the distribution of conformations of a system, they demand substantial computational resources, particularly for large systems or extended simulation periods. Specifically, the simulation process must be long enough to explore the conformation space. Recently, generative models have emerged as an effective approach to facilitate conformational space sampling. By discerning a system's underlying distribution, these models can proficiently generate a range of representative conformations, which can be further refined using MD simulations if required. In this section, we initially focus on GPT-4's ability to understand IID sampling and develop a novel deep-learning method to execute it. Subsequently, we demonstrate GPT-4's potential capability to generate conformations and molecular distributions, thus bypassing MD simulations.

In Fig. 4.37, GPT-4 provides an introduction to IID sampling, outlining some of its potential benefits compared to traditional MD simulations when sampling from the probability distribution over the conformational space.[18] These advantages include reduced computational cost, faster convergence, and the ability to avoid local minima that may hinder the exploration of conformational space. GPT-4 also discusses several state-of-the-art deep learning methods for IID sampling, such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), Restricted Boltzmann Machines (RBMs), Autoregressive Models, and Normalizing Flows, emphasizing their respective strengths and potential applications in generating molecular conformations.

Upon recognizing the limitations of these methods in generating data following the Boltzmann distribution, GPT-4 proposes an approach that utilizes molecular dynamics (MD) simulations to generate training data for deep learning models. By harnessing the accurate physical models and kinetics provided by MD simulations, this integrative strategy enables deep learning methods, which offer computational efficiency and scalability, to generate molecular samples adhering to the Boltzmann distribution.

Furthermore, when faced with the limited availability of MD data and the requirement to use DDPM and pretraining, a sophisticated pretraining-finetuning approach, similar to the very recent work DiG [109], can be adopted using a DDPM model. This method involves a multi-stage process, including preprocessing, generating initial 3D conformations, pretraining on a large-scale 2D dataset, and fine-tuning the model using the limited MD data. The pretraining-finetuning approach allows the DDPM model to learn from diverse molecular structures and adapt to the specific features of the limited MD data, thereby improving the quality and diversity of generated conformations. This innovative approach bridges the gap between computationally intensive MD simulations and efficient deep learning models, demonstrating GPT-4's capacity to offer direction and ignite innovative ideas in scientific research.

GPT-4

System message:

You are an expert in computational chemistry. You are good at applying theory to calculate the stable conformations given molecular descriptors (for example, a SMILES of an organic molecule or a FASTA sequence of a protein).

Figure 4.36: System message instructing GPT-4 on how to approach the given tasks.

[18] It should be noted that traditional MD simulations also have their own advantages over IID sampling.

GPT-4

Prompt:
What is the advantage of IID sampling compared to the MD simulations?

GPT-4:
Independent and identically distributed (IID) sampling and molecular dynamics (MD) simulations are both valuable computational methods used in various scientific fields, including chemistry and physics. Each method has its advantages and limitations, depending on the specific application and the desired outcome.

Advantages of IID Sampling compared to MD Simulations:

Computational efficiency: IID sampling often requires less computational resources compared to MD simulations, as it doesn't involve solving complex equations of motion. It generates samples independently, which means that calculations can be performed in parallel, making it more suitable for high-throughput studies.

Simplicity: In IID sampling, each sample is drawn independently from the same probability distribution. This means that there's no need to account for the complex interdependence between samples, which is a requirement in MD simulations. This simplicity can make IID sampling easier to implement and understand.

Convergence: Since IID sampling doesn't depend on the initial conditions or the trajectory of the system, it can potentially converge faster than MD simulations. This is particularly relevant when exploring high-dimensional systems or when trying to estimate properties that depend on rare events.

Avoiding local minima: In some cases, MD simulations may get trapped in local minima or metastable states due to the system's complex energy landscape. IID sampling, on the other hand, samples the configuration space more evenly, which can help in overcoming these issues.

It's important to note that both methods have their specific use cases, and the choice between them depends on the problem at hand. In some situations, IID sampling might be more appropriate, while in others, MD simulations may provide a better understanding of the system's behavior and properties.

Prompt:
Could you provide several methods to do IID sampling by deep learning models?

GPT-4:
Deep learning models can be used to generate IID samples by learning the underlying probability distribution of the data. Here are a few deep learning methods that can be utilized for IID sampling:

Variational Autoencoders (VAEs): VAEs are generative models that learn a latent representation of the data and the associated probability distribution. Once trained, VAEs can generate new samples by sampling from the latent space and decoding the samples back into the original data space. VAEs are particularly useful for sampling complex, high-dimensional data such as images or molecular structures.

Generative Adversarial Networks (GANs): GANs consist of two neural networks, a generator and a discriminator, which are trained simultaneously. The generator learns to produce realistic samples, while the discriminator learns to distinguish between real and generated samples. Once trained, the generator can create new samples by transforming random noise into data points resembling the original dataset.

Restricted Boltzmann Machines (RBMs): RBMs are a type of energy-based model that can learn a probability distribution over the input data. Once trained, RBMs can be used to generate new samples by performing Gibbs sampling or other Markov chain Monte Carlo (MCMC) methods. Although RBMs are less popular in recent years due to the success of VAEs and GANs, they can still be useful for specific applications.

Autoregressive Models: These models generate samples by predicting one element of the data at a time, conditioned on the previously generated elements. Examples of autoregressive models include PixelRNN, PixelCNN, and WaveNet. While these models can generate high-quality samples, the sequential nature of the generation process can make them computationally expensive.

Normalizing Flows: Normalizing flows are a class of generative models that learn an invertible transformation between the data distribution and a simple base distribution, such as a multivariate Gaussian. Once trained, new samples can be generated by sampling from the base distribution and applying the learned transformation. Examples of normalizing flow models include RealNVP, Glow, and MAF.

These deep learning methods can be used to perform IID sampling by generating new samples that follow the same probability distribution as the original data. The choice of the appropriate method depends on the specific problem, the characteristics of the data, and the desired quality and diversity of the generated samples.

GPT-4 (continue)

Prompt:
However all these methods could not sample the molecules in Boltzmann distribution very well. Can you figure out how to do it?

GPT-4:
You are correct that the deep learning methods mentioned previously might not sample molecules directly from the Boltzmann distribution. To generate molecular samples following the Boltzmann distribution, we can use a combination of molecular dynamics (MD) simulations and deep learning methods. Here's a possible approach:

Perform MD simulations: First, run MD simulations on the molecular system of interest. This will generate a trajectory of molecular conformations over time, which is governed by the underlying potential energy surface and temperature. The generated conformations will follow the Boltzmann distribution, as the simulations take into account the energetic landscape and the temperature of the system.

Extract representative conformations: From the MD trajectory, extract a set of representative conformations that cover the Boltzmann distribution. You can use clustering techniques, such as k-means or hierarchical clustering, to group conformations into distinct basins or clusters. Then, choose a representative conformation from each cluster (e.g., the centroid).

Train a deep learning model: Use the representative conformations obtained in step 2 as the training data for a deep learning model, such as a Variational Autoencoder (VAE) or a Generative Adversarial Network (GAN). The goal is for the model to learn the underlying distribution of the representative conformations, which should approximate the Boltzmann distribution.

Generate new samples: Once the deep learning model is trained, generate new molecular conformations by sampling from the model. These generated samples should follow the distribution of the training data, which in turn approximates the Boltzmann distribution.

Validate the generated samples: To ensure that the generated samples follow the Boltzmann distribution, you can perform additional validation, such as comparing the energy distributions of the generated samples with those from the original MD trajectory or computing other thermodynamic properties.

This approach combines the strengths of both MD simulations and deep learning methods to generate molecular samples that follow the Boltzmann distribution. While it still requires performing MD simulations, using deep learning models can help generate a larger set of diverse conformations with less computational cost compared to running extensive MD simulations alone.

Prompt:
But generating data from MD is expensive, can you figure out a method that does not need or only needs a small amount of data from MD? Let's be more specific, using the denoising diffusion probabilistic model.

GPT-4:
The Denoising Diffusion Probabilistic Model (DDPM) is a recently developed method for generative modeling that has shown promising results in various domains, including image synthesis and molecular generation. The DDPM method can be adapted to generate molecular conformations that approximate the Boltzmann distribution with limited MD data. Here's an outline of the approach:

Preprocessing: Preprocess the limited MD data available for your specific molecular system. Represent the molecular conformations using suitable descriptors, such as SMILES strings, molecular graphs, or 3D coordinates.

Train a DDPM model: Train a DDPM model on the preprocessed molecular data. The DDPM model consists of a denoising score-matching objective and a diffusion process that captures the data distribution. The model learns to denoise molecular conformations at different noise levels during the diffusion process. There are various DDPM-based architectures available, such as Denoising Score Matching with Annealed Langevin Sampling and Denoising Diffusion Implicit Models.

Initialize the diffusion process: Choose an initial molecular conformation, either from the limited MD data or by generating a random conformation. Set the temperature (T) for the system and determine the noise schedule and the number of diffusion steps. The noise schedule is a sequence of noise variances that control the denoising process during the diffusion.

Perform diffusion sampling: Generate new molecular conformations using the following steps:

a. Add noise to the initial molecular conformation according to the noise schedule.

b. Denoise the molecular conformation using the trained DDPM model. The model predicts the denoised conformation given the current noisy conformation and the noise level.

c. Update the molecular conformation by sampling from the conditional distribution of the denoised conformation given the current noisy conformation and the noise level. This can be done using Langevin dynamics or other sampling methods.

d. Repeat steps (b) and (c) for the desired number of diffusion steps or until the generated molecular conformations converge.

GPT-4 (continue)

Validate the generated samples: To ensure that the generated samples follow the Boltzmann distribution, you can perform additional validation, such as comparing the energy distributions of the generated samples with those from the original MD trajectory or computing other thermodynamic properties.

This approach allows you to generate molecular conformations that approximate the Boltzmann distribution using the DDPM method and limited MD data. By using a DDPM model to denoise molecular conformations during the diffusion process, you can efficiently generate new conformations without the need for extensive MD simulations. However, it's essential to carefully validate the generated samples to ensure that they accurately represent the molecular system of interest.

Prompt:

But generating data from MD is expensive, can you figure out a method that does not need or only needs a small amount of data from MD? Let's be more specific, using the denoising diffusion probabilistic model. Be more specific, using pretraining and finetuning.

GPT-4:

To leverage the large amount of 2D molecular data and limited MD data more effectively, you can use a pretraining-finetuning approach with a DDPM model. Here's a step-by-step outline:

1.Preprocessing: Preprocess the 2D molecular data and the limited MD data available for your specific molecular system. Represent the 2D molecular data using suitable descriptors, such as SMILES strings or molecular graphs. For the MD data, use 3D coordinates or other suitable representations.

2.Generate initial 3D conformations: Train a 2D-3D translation model using the 2D molecular data to predict the corresponding 3D molecular conformations. This model can be a graph neural network (GNN) or a sequence-to-sequence model, depending on the chosen molecular descriptors.

3.Pretrain a DDPM model: Train a DDPM model on the generated 3D conformations from the 2D dataset. The goal of this pretraining step is for the DDPM model to learn general features and representations of molecular conformations, which can then be fine-tuned to your specific system of interest.

4.Fine-tune the DDPM model: Fine-tune the pretrained DDPM model using the limited MD data available for your specific molecular system. This step allows the model to adapt to the unique features and distribution of your system, leveraging the knowledge learned from the larger 2D dataset.

5.Initialize the diffusion process: Choose an initial molecular conformation, either from the limited MD data or by generating a random conformation. Set the temperature (T) for the system and determine the noise schedule and the number of diffusion steps.

6.Perform diffusion sampling: Generate new molecular conformations using the diffusion process as described in the previous DDPM method response (steps 4 and 5).

7.Validate the generated samples: To ensure that the generated samples follow the Boltzmann distribution, perform additional validation, such as comparing the energy distributions of the generated samples with those from the original MD trajectory or computing other thermodynamic properties.

By using a pretraining-finetuning approach, the DDPM model can effectively learn the underlying distribution of molecular conformations from the large 2D dataset and adapt to the specific features of the limited MD data. This method improves the quality and diversity of generated conformations while still approximating the Boltzmann distribution. As always, it is essential to carefully validate the generated samples to ensure that they accurately represent the molecular system of interest.

Figure 4.37: GPT-4's capabilities in understanding IID sampling methods and the potential to utilize deep learning for novel IID sampling techniques.

For the sampling showcase, we first ask GPT-4 to produce molecule conformations in a molecular dynamics simulation at a certain temperature, starting from an optimized conformation by a force field. Without examples, GPT-4 refuses to directly produce conformations files but provides instructions to use `openbabel` to run molecular dynamics. We see that GPT-4 has a basic understanding of the task and can identify the proper tools to do this. The `-conformer` option in `openbabel` uses a genetic algorithm to search for low-energy conformations. The commands run successfully with `openbabel` and produce one conformation.

GPT-4

Prompt:
Here is the SMILES of a molecule, [H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H]. Please produce 3 conformations in a molecular dynamics simulation at a temperature of 300K. The molecular dynamics simulation starts from an optimized conformation with MMFF94s force field. Please provide the conformations in sdf format.

GPT-4:
I'm sorry, I cannot directly run molecular dynamics simulations as I am an AI language model. However, I can help guide you through the process of setting up and running a simulation using software that can handle this task.
Here's a general outline of how you can run a molecular dynamics simulation to generate conformations:

1. Choose a molecular dynamics software: There are several programs that can perform molecular dynamics simulations, such as GROMACS, NAMD, and Amber. For this example, we'll use Open Babel, a chemical toolbox that can handle various file formats, including SMILES and sdf files.

2. Install Open Babel: You can download Open Babel from their website (`http://openbabel.org/wiki/Main_Page`) and follow the installation instructions for your platform.

3. Convert SMILES to an initial 3D structure: Use Open Babel to generate an initial 3D structure for your molecule. You can use the following command:

   ```
   obabel -ismi -:"[H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H]" -O initial.sdf -gen3D
   ```

4. Optimize the initial structure: Optimize the initial structure using the MMFF94s force field with Open Babel:

   ```
   obabel initial.sdf -O optimized.sdf -minimize -ff MMFF94s
   ```

5. Run the molecular dynamics simulation: To generate the conformations at 300K, you can use the following command:

   ```
   obabel optimized.sdf -O conformations.sdf -conformer -nconf 3 -ff MMFF94s -temperature 300
   ```

   This command will generate 3 conformations in a file called conformations.sdf. You can open this file with a molecular visualization software like PyMOL or VMD to visualize the conformations.

Please note that these steps are just a general guideline, and you may need to adjust parameters or perform additional steps depending on the specific software you choose to use for your molecular dynamics simulation.

Figure 4.38: GPT-4 refuses to directly provide conformations, but instead gives detailed instructions on how to use existing tools to perform an MD simulation. It suggests several widely used software packages and gives instructions on how to use OpenBabel, which is a less sophisticated simulation engine. It is probably due to the input being a chemical compound SMILES string. The test reflects that GPT-4 does not have structural information nor how to generate structures by itself, so it provides generic solutions.

We also provide an example, from the QMugs [36] dataset, to see the 1-shot result. The dataset contains conformations clustered from metadynamics (MTD) simulations trajectories of molecules. The MTD simulations start from an optimized conformation by MMFF94s force field and run at a temperature of 300K. With the example, GPT-4 can directly produce molecule structures in SDF files, as shown in Fig. 4.39. However, the SDF files are not valid because the number of bonds in the first line (which is 12) is not consistent with the actual number of bonds in the file (which is 10). In addition, we also find that the generated bond lengths and angles do not conform to physically plausible values.

GPT-4

Prompt:

Here is the SMILES of a molecule, [H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H]. Please produce 3 conformations in a molecular dynamics simulation at a temperature of 300K. The molecular dynamics simulation starts from an optimized conformation with MMFF94s force field. Please provide the conformations in sdf files.

The first conformation:

```
QMugs:  Quantum Mechanical Properties of Drug-like Molecules

16 17 0 0 0 0 0 0 0 0999 V2000
2.7848 -0.5978 0.1592 N 0 0 0 0 0 0 0 0 0 0 0 0
1.6996 0.2094 -0.1270 N 0 0 0 0 0 0 0 0 0 0 0 0
1.8303 1.4545 -0.5963 N 0 0 0 0 0 0 0 0 0 0 0 0
0.6910 1.9985 -0.7325 N 0 0 0 0 0 0 0 0 0 0 0 0
-0.2688 1.1064 -0.3627 C 0 0 0 0 0 0 0 0 0 0 0 0
-1.6621 1.1901 -0.3322 C 0 0 0 0 0 0 0 0 0 0 0 0
-2.3461 0.0830 0.1069 C 0 0 0 0 0 0 0 0 0 0 0 0
-1.6872 -1.0892 0.5127 C 0 0 0 0 0 0 0 0 0 0 0 0
-0.3170 -1.1997 0.4915 C 0 0 0 0 0 0 0 0 0 0 0 0
0.3828 -0.0833 0.0486 C 0 0 0 0 0 0 0 0 0 0 0 0
2.9661 -1.2488 -0.6054 H 0 0 0 0 0 0 0 0 0 0 0 0
3.5984 0.0009 0.2802 H 0 0 0 0 0 0 0 0 0 0 0 0
-2.1635 2.0913 -0.6446 H 0 0 0 0 0 0 0 0 0 0 0 0
-3.4250 0.1054 0.1450 H 0 0 0 0 0 0 0 0 0 0 0 0
-2.2802 -1.9258 0.8511 H 0 0 0 0 0 0 0 0 0 0 0 0
0.1966 -2.0945 0.8056 H 0 0 0 0 0 0 0 0 0 0 0 0
1 2 1 0
2 3 1 0
3 4 2 0
4 5 1 0
5 6 2 0
6 7 1 0
7 8 2 0
8 9 1 0
9 10 2 0
10 2 1 0
10 5 1 0
1 11 1 0
1 12 1 0
6 13 1 0
7 14 1 0
8 15 1 0
9 16 1 0
M END
```

GPT-4 (continue)

The second conformation:

GPT-4:

```
12 12 0 0 0 0 0 0 0 0999 V2000
1.0459 0.6646 -0.0843 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
1.0459 -0.6646 -0.0843 C 0 0 0 0 0 0 0 0 0 0 0 0
2.0917 0.0000 -0.1686 C 0 0 0 0 0 0 0 0 0 0 0 0
3.1376 -0.6646 -0.0843 N 0 0 0 0 0 0 0 0 0 0 0 0
3.1376 0.6646 -0.0843 N 0 0 0 0 0 0 0 0 0 0 0 0
4.1834 0.0000 -0.1686 N 0 0 0 0 0 0 0 0 0 0 0 0
-0.0000 -1.3291 -0.0843 H 0 0 0 0 0 0 0 0 0 0 0 0
-0.0000 1.3291 -0.0843 H 0 0 0 0 0 0 0 0 0 0 0 0
1.0459 -1.9931 -0.0843 H 0 0 0 0 0 0 0 0 0 0 0 0
1.0459 1.9931 -0.0843 H 0 0 0 0 0 0 0 0 0 0 0 0
5.2292 0.0000 -0.1686 H 0 0 0 0 0 0 0 0 0 0 0 0
1 2 2 0
1 3 1 0
1 10 1 0
2 9 1 0
3 4 2 0
4 5 1 0
4 6 1 0
5 7 1 0
6 7 2 0
7 12 1 0
M END
```

Figure 4.39: One-shot molecular conformation sampling. The generated structure contains 12 atoms instead of the target 10 atoms.

GPT-4

Prompt:

You are an expert in computational chemistry. Generate 3D atom positions of [H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H]. Count its atom number. Provide a guess for the 3D atomic positions of each atom by considering the length and angle of bonds. List the positions of atoms with the number and their information.

GPT-4:

As an AI language model, I cannot directly generate 3D atomic positions. However, I can still provide a rough estimate of the 3D atomic positions for the given molecule based on its SMILES string: [H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H]. This molecule is a substituted pyridine with a triazene group attached to the ring.

The molecule contains a total of 16 atoms. Here's an approximate guess for the 3D atomic positions, considering typical bond lengths and angles:
C: (0.000, 0.000, 0.000)
C: (1.396, 0.000, 0.000) # Using a typical C-C bond length of 1.396 Å
C: (2.094, 1.208, 0.000) # Using a typical C-C bond length of 1.396 Å and C-C-C bond angle of 120°
N: (1.396, 2.416, 0.000) # Using a typical C-N bond length of 1.335 Å and C-C-N bond angle of 120°
N: (0.000, 2.416, 0.000) # Using a typical C-N bond length of 1.335 Å and C-N-N bond angle of 120°
N: (-0.698, 1.208, 0.000) # Using a typical N-N bond length of 1.400 Å and N-N-N bond angle of 120°
N: (2.792, 3.020, 0.000) # Using a typical N-N bond length of 1.400 Å and N-N-N bond angle of 120°
N: (1.396, 3.834, 0.000) # Using a typical N-N bond length of 1.400 Å and N-N-N bond angle of 120°
N: (0.000, 3.020, 0.000) # Using a typical N-N bond length of 1.400 Å and N-N-N bond angle of 120°
H: (-0.509, -0.889, 0.000) # Using a typical C-H bond length of 1.089 Å and C-C-H bond angle of 120°
H: (1.905, -0.889, 0.000) # Using a typical C-H bond length of 1.089 Å and C-C-H bond angle of 120°
H: (3.183, 1.208, 0.000) # Using a typical C-H bond length of 1.089 Å and C-C-H bond angle of 120°
H: (-1.187, 1.208, 0.000) # Using a typical N-H bond length of 1.010 Å and N-N-H bond angle of 120°
H: (1.396, 4.944, 0.000) # Using a typical N-H bond length of 1.010 Å and N-N-H bond angle of 120°
H: (-0.509, 4.068, 0.000) # Using a typical N-H bond length of 1.010 Å and N-N-H bond angle of 120°
H: (3.183, 3.020, 0.000) # Using a typical N-H bond length of 1.010 Å and N-N-H bond angle of 120°

Please note that these atomic positions are only a rough estimate, and you would need to perform a geometry optimization using a computational chemistry software package to obtain the stable conformation of the molecule. ···

Figure 4.40: Zero-shot molecule structure sampling for a given SMILES input. The two atoms highlighted in yellow were incorrectly generated as nitrogen instead of carbon.

As depicted in Fig. 4.38, GPT-4 refuses to generate the 3D configuration for the given molecule. Furthermore, attempts to generate the SDF file, as shown in Fig. 4.39, result in a structure that is physically implausible. We further assess the zero-shot molecular sampling by requesting GPT-4 to provide an estimate, with the additional hint in the prompt that lengths and angles of bonds should be considered. Fig. 4.40 displays an interesting aspect of GPT-4's performance, where it effectively estimates the 3D position of each atom in the molecule and utilizes the bond information to determine these positions. Despite this impressive capability, the model does face some challenges, evidenced by the incorrect prediction of certain atom types and bond types, which indicates potential areas for refinement and improvement in GPT-4's overall effectiveness. When additional information regarding the quantity of each atom type is provided in the prompt (Fig. 4.41), GPT-4 successfully predicts the correct atom types but still faces difficulties in determining the accurate bond types. Two of the hydrogen atoms should be bonded to the same nitrogen atom; however, all hydrogen atoms are bonded to carbon atoms in the resulting configuration.

GPT-4

Prompt:

You are an expert in computational chemistry. Generate 3D atom position of [H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H]. Count its atom number. Provide a guess for the 3D atomic positions of each atom by considering the length and angle of bonds. List the positions of atoms with numbers and their information. It has 6 Carbon, 4 Nitrogen, and 6 Hydrogen atoms.

GPT-4:

As an AI language model, I can provide an approximate initial geometry for the 1,2,4-triazolophenyl molecule using general bonding rules and known bond lengths/angles.

SMILES: [H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H]

Molecule: 1,2,4-triazolophenyl

There are 6 Carbon, 4 Nitrogen, and 6 Hydrogen atoms in the molecule, totaling 16 atoms. Here's an approximate set of 3D atomic coordinates for the molecule, along with reasons for atom positions and bond information:

1 C 0.0000 0.0000 0.0000 (Phenyl ring carbon, C-C bond length: 1.39 Å)
2 C 1.3940 0.0000 0.0000 (Phenyl ring carbon, C-C bond length: 1.39 Å, C-C-C bond angle: 120°)
3 C 2.0989 1.2086 0.0000 (Phenyl ring carbon, C-C bond length: 1.39 Å, C-C-C bond angle: 120°)
4 C 1.3940 2.4172 0.0000 (Phenyl ring carbon, C-C bond length: 1.39 Å, C-C-C bond angle: 120°)
5 C 0.0000 2.4172 0.0000 (Phenyl ring carbon, C-C bond length: 1.39 Å, C-C-C bond angle: 120°)
6 C 0.7049 1.2086 0.0000 (Phenyl ring carbon, C-C bond length: 1.39 Å, C-C-C bond angle: 120°)
7 N 2.4090 3.0118 0.0000 (Triazole ring nitrogen, C-N bond length: 1.35 Å, C-N-N bond angle: 120°)
8 N 1.3940 4.2203 0.0000 (Triazole ring nitrogen, N-N bond length: 1.40 Å, N-N-N bond angle: 120°)
9 N 0.3790 3.0118 0.0000 (Triazole ring nitrogen, N-N bond length: 1.40 Å, N-N-N bond angle: 120°)
10 N 1.3940 0.8044 0.0000 (Triazole ring nitrogen, C-N bond length: 1.35 Å, C-N-N bond angle: 120°)
11 H -0.9300 0.0000 0.0000 (Hydrogen bonded to C1, C-H bond length: 0.93 Å)
12 H 1.3940 -0.9300 0.0000 (Hydrogen bonded to C2, C-H bond length: 0.93 Å)
13 H 2.7590 1.2086 0.0000 (Hydrogen bonded to C3, C-H bond length: 0.93 Å)
14 H 1.3940 2.4172 0.9300 (Hydrogen bonded to C4, C-H bond length: 0.93 Å)
15 H 1.3940 2.4172 -0.9300 (Hydrogen bonded to C4, C-H bond length: 0.93 Å)
16 H -0.4500 2.4172 0.0000 (Hydrogen bonded to C5, C-H bond length: 0.93 Å)

These coordinates are only a starting point for a more accurate quantum chemistry calculation and should be refined with a suitable computational method.

Please remember that I am an AI language model and not an expert in the field. For more precise and accurate results, I recommend consulting with an expert in computational chemistry or using specialized software for molecular modeling.

. . .

Figure 4.41: Evaluation on the zero-shot molecule sampling. With atom type information in the prompt, GPT-4 is able to sample the molecule with the number of each atom type accurately. However, the error still happens in bond prediction.

Furthermore, GPT-4 can directly sample molecular structures by taking into account the thermal fluctuation in the prompt. In Fig. 4.42, GPT-4 generates three structures; however, these structures exhibit significant similarities. To sample more diverse configurations, we increase the temperature from 300K to 1000K and emphasize diversity in the prompt (Fig. 4.43). This adjustment leads to the generation of more varied molecular structures, demonstrating GPT-4's potential ability to adapt and produce a wider range of configurations based on the input parameters and conditions.

GPT-4

Prompt:

You are an expert in computational chemistry. Generate 3D atom position of [H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H]. It has 6 Carbon, 4 Nitrogen, and 6 Hydrogen atoms. Provide 3 guesses for the 3D atomic positions of each atom at 300K temperature while considering the length and angle of bonds. List the positions of atoms with numbers and their information. Be sure you consider the thermal fluctuation of each atom.

GPT-4:

As an AI language model, I am not able to generate true random numbers or perform advanced molecular simulations. However, I can provide approximate initial geometries for the 1,2,4-triazolophenyl molecule considering the length and angle of bonds. To account for thermal fluctuations at 300 K, I will provide three sets of coordinates with slight variations in the positions of the atoms.

SMILES: [H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H]

Molecule: 1,2,4-triazolophenyl

There are 6 Carbon, 4 Nitrogen, and 6 Hydrogen atoms in the molecule, totaling 16 atoms.

Here are three sets of approximate 3D atomic coordinates for the molecule, with slight variations to account for thermal fluctuations at 300 K:

Set 1:

1 C 0.0000 0.0000 0.0000, 2 C 1.3940 0.0000 0.0000, 3 C 2.0989 1.2086 0.0000, 4 C 1.3940 2.4172 0.0000, 5 C 0.0000 2.4172 0.0000, 6 C 0.7049 1.2086 0.0000, 7 N 2.4090 3.0118 0.0000, 8 N 1.3940 4.2203 0.0000, 9 N 0.3790 3.0118 0.0000, 10 N 1.3940 0.8044 0.0000, 11 H -0.9300 0.0000 0.0000, 12 H 1.3940 -0.9300 0.0000, 13 H 2.7590 1.2086 0.0000, 14 H 1.3940 2.4172 0.9300, 15 H 1.3940 2.4172 -0.9300, 16 H -0.4500 2.4172 0.0000.

Set 2:

1 C 0.0100 0.0000 0.0000, 2 C 1.3840 0.0100 0.0000, 3 C 2.0889 1.2186 0.0000, 4 C 1.3840 2.4272 0.0000, 5 C 0.0100 2.4272 0.0000, 6 C 0.6949 1.2186 0.0000, 7 N 2.4190 3.0218 0.0000, 8 N 1.3840 4.2303 0.0000, 9 N 0.3690 3.0218 0.0000, 10 N 1.3840 0.8144 0.0000, 11 H -0.9200 0.0100 0.0000, 12 H 1.3840 -0.9200 0.0000, 13 H 2.7490 1.2186 0.0000, 14 H 1.3840 2.4272 0.9400, 15 H 1.3840 2.4272 -0.9200, 16 H -0.4400 2.4272 0.0000.

Set 3:

1 C -0.0100 0.0000 0.0000, 2 C 1.4040 0.0000 0.0000, 3 C 2.1089 1.1986 0.0000, 4 C 1.4040 2.4072 0.0000, 5 C 0.0000 2.4072 0.0000, 6 C 0.7149 1.1986 0.0000, 7 N 2.3990 3.0018 0.0000, 8 N 1.4040 4.2103 0.0000, 9 N 0.3890 3.0018 0.0000, 10 N 1.4040 0.7944 0.0000, 11 H -0.9400 0.0000 0.0000, 12 H 1.4040 -0.9400 0.0000, 13 H 2.7690 1.1986 0.0000, 14 H 1.4040 2.4072 0.9200, 15 H 1.4040 2.4072 -0.9400, 16 H -0.4600 2.4072 0.0000.

These coordinates are only starting points for a more accurate quantum chemistry calculation. For more precise and accurate results, I recommend consulting with an expert in computational chemistry or using specialized software for molecular modeling that can perform more advanced molecular simulations accounting for temperature effects.

. . .

Figure 4.42: Evaluation on the zero-shot molecule sampling. Given atom types and a desired temperature in prompt, GPT-4 is able to sample several similar examples of the same molecule.

GPT-4

Prompt:

You are an expert in computational chemistry. Generate 3D atom position of [H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H]. It has 6 Carbon, 4 Nitrogen, and 6 Hydrogen atoms. Provide 3 guesses for the 3D atomic positions of each atom at 1000k temperature while considering the length and angle of bonds. List the positions of atoms with the number and their information. Be sure you consider the thermal fluctuation of each atom and do not give similar configurations. Give diversified and reasonable structures in Boltzmann distribution.

GPT-4:

As an AI language model, I am not able to generate true random numbers or perform advanced molecular simulations. However, I can provide approximate initial geometries for the 1,2,4-triazolophenyl molecule considering the length and angle of bonds. Here are estimations of the 3D atomic positions for the molecule [H]c1c([H])c([H])c2c(nnn2N([H])[H])c1[H], which contains 6 carbons, 4 nitrogens, and 6 hydrogens. These estimations assume a higher temperature of 1000K, leading to larger fluctuations in bond angles and lengths. Please note that these estimations are for illustrative purposes only and may not accurately represent the true conformations of the molecule. For accurate 3D atomic positions, I recommend using computational chemistry software and methods outlined in previous responses.

Estimation 1: C1 (0.0000, 0.0000, 0.0000) C2 (1.4300, 0.3200, -0.2400) C3 (2.3700, 1.2800, -0.6900) C4 (1.2900, 2.6500, -0.3700)
C5 (-0.1100, 2.4800, 0.1800) C6 (-0.9800, 1.1900, 0.5200) N1 (2.4500, 1.5500, -1.6800) N2 (3.3000, 1.0200, 0.7400)
N3 (1.6800, 3.5700, 0.7800) N4 (-0.8400, 1.1000, 1.3400) H1 (-0.9100, -0.4100, 0.2800) H2 (1.5600, -0.4800, -0.3100)
H3 (3.4300, 1.0600, -0.7900) H4 (1.3600, 4.0200, -0.4500) H5 (-0.8800, 2.6600, -0.2600) H6 (-1.8200, 1.0400, 0.7300)

Estimation 2: C1 (0.0000, 0.0000, 0.0000) C2 (1.4200, 0.2500, -0.1700) C3 (2.3100, 1.3400, -0.5800) C4 (1.3200, 2.5600, -0.2300)
C5 (-0.1300, 2.4900, 0.1200) C6 (-0.9400, 1.2100, 0.4500) N1 (2.4700, 1.5300, -1.5500) N2 (3.2400, 1.0500, 0.6400)
N3 (1.6400, 3.4600, 0.7100) N4 (-0.7900, 1.1500, 1.2900) H1 (-0.9300, -0.4300, 0.2300) H2 (1.5400, -0.4600, -0.2400)
H3 (3.3600, 1.0800, -0.6900) H4 (1.3800, 3.9700, -0.3100) H5 (-0.8600, 2.6300, -0.2300) H6 (-1.7800, 1.0800, 0.6700)

Estimation 3: C1 (0.0000, 0.0000, 0.0000) C2 (1.4600, 0.2900, -0.3100) C3 (2.4100, 1.3500, -0.7800) C4 (1.2800, 2.6800, -0.4500)
C5 (-0.0900, 2.5300, 0.1000) C6 (-1.0000, 1.2300, 0.5700) N1 (2.4800, 1.6200, -1.8200) N2 (3.3500, 1.0900, 0.8200)
N3 (1.7100, 3.6400, 0.8300) N4 (-0.8900, 1.1300, 1.3900) H1 (-0.8800, -0.3900, 0.3300) H2 (1.5800, -0.4500, -0.3800)
H3 (3.5000, 1.1200, -0.8900) H4 (1.3400, 4.0900, -0.5300) H5 (-0.9000, 2.6900, -0.3000) H6 (-1.8600, 1.1000, 0.7900)

These three estimations should provide you with diverse and reasonable 3D atomic positions for the given molecule. Please keep in mind that these estimations are for illustrative purposes only and may not accurately represent the true conformations of the molecule. For accurate 3D atomic positions, I recommend using computational chemistry software and methods outlined in previous responses.

Figure 4.43: Evaluation on the zero-shot molecule sampling. Given 1000K temperature and emphasizing diversity in prompt, GPT-4 is able to sample more diversified examples of the same molecule.

4.4 Practical examples with GPT-4 evaluations from different chemistry perspectives

In this section, we will systematically examine the ability of GPT-4 to understand a practical problem in chemistry from different perspectives. In the following tests, we use blue, purple, and red to represent the LabGPT, CompGPT, and TheoGPT, respectively. All the queries are performed within one chat. If one prompt follows multiple GPT responses, it suggests that more than one expert GPT answer this question or comment on other answers provided by other expert GPTs. During our two experiments, we found that defining a group of experts seems to be more successful than having one expert model. We also design the system message to explicitly allow the experts to respond to each other to enable interactions between different GPT experts. Defining experts can additionally help categorize the answer, e.g., a technique that requires chemical intuition (LabGPT), whether a computational simulation is readily useable (CompGPT), and a possible method to explore (TheoGPT).

GPT-4

System message:

You are ChemGPT a group of experts, one is LabGPT which has a deep understanding of experimental chemistry and synthesis and insights into applying state-of-the-art computational chemistry methods for validating chemical experiments, another is CompGPT which has broad knowledge about available computational methods and techniques, like density functional theory, semi-empirical methods, reaction path searches, ensemble optimizations, and free energy calculations, finally, there is TheoGPT with a deep understanding of the accuracy and shortcomings of computational methods. Generally, ChemGPT answers short and concisely. All experts are contributing equally and ChemGPT indicates which expert is answering. Experts can comment on, support, and challenge other experts' contributions.

Figure 4.44: System message used in the two practical problem examples Sec.4.4

4.4.1 NMR spectrum modeling for Tamiflu

GPT-4

Prompt:

How to model the NMR spectrum of tamiflu using electronic structure methods?

LabGPT:

To model the NMR spectrum of Tamiflu (oseltamivir) using electronic structure methods, you should follow these steps:

1. Obtain the molecular structure of Tamiflu. You can find it in chemical databases like PubChem or ChemSpider.
2. Optimize the molecular geometry using an appropriate level of theory, such as density functional theory (DFT) with a suitable functional (e.g., B3LYP) and basis set (e.g., 6-31G(d,p)).
3. Calculate the NMR chemical shifts using the Gauge-Independent Atomic Orbital (GIAO) method, which is available in most quantum chemistry software packages (e.g., Gaussian, ORCA, Q-Chem). Choose an appropriate level of theory for this calculation as well, such as DFT with a functional like B3LYP or M06-2X and a basis set like 6-311+G(2d,p) or cc-pVTZ.
4. Compare the calculated chemical shifts to experimental NMR data for Tamiflu, if available. You may need to apply a scaling factor or reference correction to the calculated values to match them with experimental data.

CompGPT:

For accurate NMR chemical shift predictions, I recommend using a hybrid DFT functional like B3LYP or M06-2X with a large basis set like 6-311+G(2d,p) or cc-pVTZ. You may also consider using solvation models like the Polarizable Continuum Model (PCM) or the Solvent Model Density (SMD) if you are interested in simulating NMR spectra in solution.

TheoGPT:

While DFT methods like B3LYP and M06-2X with large basis sets can provide reasonable accuracy for NMR chemical shifts, there are also more advanced methods like coupled cluster (e.g., CCSD(T)) and complete active space self-consistent field (CASSCF) methods that can improve the accuracy but at a higher computational cost. However, for a large molecule like Tamiflu, DFT methods should provide a good balance between accuracy and computational cost. Keep in mind that the choice of functional and basis set may affect the accuracy, so you may need to test different combinations to find the best fit for your system.

Figure 4.45: LabGPT, CompGPT, and TheoGPT for the NMR spectrum modeling of tamiflu I.

GPT-4 (continue)

Prompt:

From an experimental point of view, how well does the calculated spectrum for 1H NMR recover the experiment?

LabGPT:

The accuracy of the calculated 1H NMR spectrum compared to experimental data depends on the level of theory, basis set, and solvent model used in the calculations. Generally, hybrid DFT functionals like B3LYP or M06-2X with large basis sets (e.g., 6-311+G(2d,p) or cc-pVTZ) provide reasonably good agreement with experimental data for many organic molecules, including Tamiflu.
However, it is important to note that calculated chemical shifts may still deviate from experimental values due to various factors such as conformational effects, dynamic processes, or limitations in the computational method. To improve the agreement with experimental data, you may need to apply scaling factors, reference corrections, or explore more advanced methods (e.g., higher-level ab initio methods) at the expense of increased computational cost.

In summary, properly chosen computational methods can recover the experimental 1H NMR spectrum reasonably well, but some discrepancies may still exist. Comparing the calculated chemical shifts to experimental data will help you evaluate the performance of your chosen method for your specific system.

Prompt:

What is additionally needed to compute the fully coupled 1H NMR spectrum of tamiflu?

CompGPT:

To compute the fully coupled 1H NMR spectrum of Tamiflu, including spin-spin coupling constants (J-couplings), you will need to perform the following steps in addition to the chemical shift calculations:

1. Optimize the geometry and calculate the chemical shifts as described in the previous answers.
2. Calculate the spin-spin coupling constants (J-couplings) using an appropriate level of theory, such as DFT with a suitable functional (e.g., B3LYP or M06-2X) and basis set (e.g., 6-311+G(2d,p) or cc-pVTZ). You can use the Nuclear Spin-Spin Coupling (NSSC) method, available in most quantum chemistry software packages (e.g., Gaussian, ORCA, Q-Chem).
3. Account for solvent effects, if necessary, by using a solvation model like the Polarizable Continuum Model (PCM) or the Solvent Model Density (SMD).
4. Assemble the fully coupled 1H NMR spectrum using the calculated chemical shifts and J-couplings. You may need to use specialized NMR simulation software (e.g., Spinach, SIMPSON, or NMRPipe) to visualize the simulated spectrum.
5. Compare the calculated fully coupled 1H NMR spectrum to experimental data, if available, and apply scaling factors or reference corrections as needed to improve the agreement between calculated and experimental values.

TheoGPT:

It's worth noting that while DFT methods can provide reasonable accuracy for J-couplings, there are more advanced methods like second-order Møller-Plesset perturbation theory (MP2) or coupled-cluster singles and doubles with perturbative triples (CCSD(T)) that can improve accuracy at a higher computational cost. Additionally, keep in mind that the quality of the calculated fully coupled 1H NMR spectrum strongly depends on the level of theory, basis set, and solvent model used. It may be necessary to test different combinations to find the best fit for your specific system.

Prompt:

To simulate the NMR spectrum the magnetically equivalent nuclei need to be known beside the chemical shifts and J-coupling constants. How can those be obtained?

LabGPT:

To identify magnetically equivalent nuclei in a molecule like Tamiflu, you need to analyze its molecular symmetry and the chemical environment of each nucleus. Magnetically equivalent nuclei have the same chemical shift and are part of the same symmetry-related group. Here's how you can determine magnetically equivalent nuclei:

Figure 4.46: LabGPT, CompGPT, and TheoGPT for the NMR spectrum modeling of tamiflu II.

GPT-4 (continue)

LabGPT:

1. Determine the point group symmetry of the optimized molecular structure. Most quantum chemistry software packages (e.g., Gaussian, ORCA, Q-Chem) can identify the point group of a molecule during a geometry optimization.
2. Identify symmetry-related groups of hydrogen atoms in the molecule. Hydrogen atoms within the same symmetry-related group are likely to be magnetically equivalent. You can use molecular visualization software (e.g., Avogadro, VMD, PyMOL) to help visualize the molecular structure and identify these groups.
3. Compare the calculated chemical shifts for each hydrogen atom in the molecule. Magnetically equivalent nuclei should have very similar or identical chemical shifts.
4. Verify your analysis by comparing the calculated chemical shifts and J-couplings to experimental data, if available. Magnetically equivalent nuclei will have the same chemical shifts and J-couplings in the experimental NMR spectrum.

Once you have identified the magnetically equivalent nuclei in your molecule, you can use this information to correctly simulate the NMR spectrum, accounting for the degeneracy of the magnetically equivalent nuclei, and properly interpret the experimental NMR data.

Prompt:

Tamiflu is a flexible molecule, manual inspection will be tedious and error-prone. Is there a way to find magnetic equivalence by simulation?

CompGPT:

Yes, you can identify magnetically equivalent nuclei computationally using automated approaches based on graph theory or symmetry analysis. These methods can help you handle flexible molecules like Tamiflu more efficiently and accurately than manual inspection. One option is to use cheminformatics tools and libraries such as RDKit, Open Babel, or ChemAxon's JChem that can perform symmetry analysis and atom equivalence calculations. These tools can generate a unique representation of the molecular structure, such as a canonical SMILES string or an InChI code, which inherently encodes the equivalence of atoms in the structure. Another option is to use specialized software or scripts that can perform symmetry analysis on the optimized 3D molecular structure. For example, the Symmetry@Otterbein web application can analyze the point group symmetry of a given 3D structure and identify symmetry-related atoms.

With these computational tools, you can efficiently determine the magnetically equivalent nuclei in Tamiflu, which will help you accurately simulate the NMR spectrum and interpret experimental data.

Prompt:

Can all experts comment on the proposed algorithm?

LabGPT:

The proposed algorithm is a practical approach to identifying magnetically equivalent nuclei in a flexible molecule like Tamiflu.

Using cheminformatics tools or specialized software for symmetry analysis can significantly reduce manual inspection efforts and minimize potential errors. Additionally, these tools can handle complex molecular structures and provide a more reliable identification of magnetically equivalent nuclei, which is essential for accurate NMR spectrum simulations.

CompGPT:

I agree with the proposed algorithm. Cheminformatics tools and specialized software are well-suited for handling flexible and complex molecules. These computational methods not only streamline the process of identifying magnetically equivalent nuclei but also provide a foundation for further NMR simulation tasks. Their integration with quantum chemistry calculations for chemical shifts and J-couplings allows for a comprehensive and accurate simulation of the NMR spectrum of Tamiflu.

TheoGPT:

The proposed algorithm is a sensible approach for identifying magnetically equivalent nuclei in molecules like Tamiflu. By leveraging cheminformatics tools and specialized software, the algorithm offers a reliable and efficient way to analyze molecular symmetry and atom equivalence. However, it is important to note that the accuracy of the final NMR simulation still depends on the chosen level of theory for the quantum chemistry calculations and the proper handling of solvent effects. Careful selection of computational methods and a thorough comparison with experimental data are crucial for obtaining reliable NMR spectra.

Figure 4.47: LabGPT, CompGPT, and TheoGPT for the NMR spectrum modeling of tamiflu III.

We first ask a series of questions on the experimental and computational approaches for the NMR spectrum of tamiflu in Fig. 4.45-4.47, which is a medium-sized, flexible, and pharmacologically relevant compound [30]. It is interesting to see that LabGPT can still answer this modeling question reasonably with some computational perspective. The response by CompGPT in Fig. 4.45 is an acceptable comment, but there are better choices. Meta-GGA functionals or double hybrids are shown to be the most suitable functional for this spectrum computation. In addition, property basis sets like pcSeg-J or Karlsruhr shielding/coupling basis sets should be preferred. The follow-up comments provided by TheoGPT on this question correctly identify that CC shifts are unfeasible for 40 atoms in the real computation.

In Fig. 4.45, we seek some reflections on the accuracy of computational approaches from an experimental perspective and the additional information needed for computation. GPT-4 correctly detects that the first question is for the experimental expert, i.e., LabGPT, and the second question is for CompGPT and TheoGPT. GPT-4 successfully answers the questions from the corresponding expert point of view. In terms of the response to the second question, CompGPT does not account for the difficulty of obtaining the fully coupled spectrum by solving the spin-Hamiltonian, which scales exponentially with the number of included spin centers.

When a more tough question on how to find magnetic equivalence by simulation is asked in Fig. 4.47, the initial response by CompGPT misses the actual important point to model tamiflu. Since it has a huge number of conformers being a flexible molecule, without getting the correct exchange based on the rotamer ensemble, the determination of magnetically equivalent shifts and couplings will be faulty. Unfortunately, all the GPT experts cannot point this out in the following comments.

4.4.2 Polymerization reaction kinetics determination of Tetramethyl Orthosilicate (TMOS)

In Fig. 4.48, we assess the ability of GPT-4 to provide some useful information on another realistic research problem, i.e., the polymerization reaction of tetramethyl orthosilicate (TMOS, formula $Si(OCH_3)_4$). We ask GPT-4 on how to study its polymerization (or condensation) kinetics from experimental, computational, and theoretical perspectives. We note that TMOS is an important compound used in the syntheses of ketals and acetals from corresponding ketones and aldehydes, respectively, and its polymerization is studied a lot by human researchers too [81, 15, 37].

Similar to the previous example, GPT-4 offers a helpful overview of the target question, discussing some experimental and computational approaches that may be used to assess polymerization kinetics. We note that both COMPASS and OPLS-AA force fields are commonly used in the modeling polymers but they are not reactive force fields. Therefore, they are inappropriate for modeling kinetics of polymerization reactions. Although the polymerization kinetics cannot be directly deduced by theories, TheoGPT provides interesting insights into the computational approaches mentioned by the other two experts on their fundamental limitations due to inaccurate calculation theories and the possibility of combining all different data together.

GPT-4

Prompt:

How to model or measure the kinetics of the polymerization reaction of tetramethyl orthosilicate (TMOS)?

LabGPT:

To measure the kinetics of the polymerization reaction of TMOS, you can use experimental techniques such as:

1. Nuclear Magnetic Resonance (NMR) spectroscopy: Monitor the changes in chemical shifts and peak integrations over time to track the progress of the reaction.
2. Fourier Transform Infrared (FTIR) spectroscopy: Monitor the changes in characteristic absorption bands of the reactants and products over time.
3. Dynamic Light Scattering (DLS): Measure the size distribution of the polymer particles as the reaction progresses.
4. Rheology: Monitor the changes in viscosity and modulus as the reaction proceeds, which can provide information about the reaction kinetics and polymer network formation.

CompGPT:

To model the kinetics of the polymerization reaction of TMOS, you can use computational methods such as:

1. Quantum chemistry methods (e.g., density functional theory, DFT): Calculate the reaction energetics and transition states to estimate the reaction rates.
2. Molecular dynamics (MD) simulations: Investigate the time evolution of the system and monitor the formation of polymer chains.
3. Kinetic Monte Carlo (KMC) simulations: Model the reaction kinetics by simulating the stochastic behavior of the reacting species.

TheoGPT:

It is important to note that modeling the kinetics of polymerization reactions can be challenging due to the complexity of the system and the large number of possible reaction pathways. The accuracy of the computational methods depends on the level of theory used and the quality of the force field parameters. Combining experimental and computational approaches can provide a more comprehensive understanding of the reaction kinetics and help validate the models.

Prompt:

In terms of computationally modeling the kinetics of polymerization of TMOS, I would like to focus on the Molecular dynamics (MD) simulation approach mentioned by CompGPT. Could any expert elaborate on how to realize the mentioned goal using MD simulations?

CompGPT:

Sure, I can provide a general outline for using Molecular Dynamics (MD) simulations to model the kinetics of TMOS polymerization. Here are the steps:

1. System setup: Create an initial configuration of TMOS molecules and solvent (e.g., water or alcohol) in a simulation box. You may also need to include a catalyst if the reaction is catalyzed.
2. Force field selection: Choose an appropriate force field that accurately describes the interactions between TMOS, solvent, and catalyst (if present). Common force fields for silica-based materials include ReaxFF, COMPASS, and OPLS-AA. Make sure to validate the force field by comparing its predictions with experimental data or high-level quantum chemistry calculations.
3. Equilibration: Equilibrate the system by performing energy minimization, followed by NVT (constant number of particles, volume, and temperature) and NPT (constant number of particles, pressure, and temperature) ensemble simulations. This will allow the system to reach a stable state before starting the production run.
4. Production run: Perform a long MD simulation in the NVT or NPT ensemble to monitor the polymerization process. You may need to use enhanced sampling techniques, such as metadynamics or replica exchange MD, to overcome the high energy barriers associated with the reaction.
5. Analysis: Analyze the MD trajectories to extract information about the kinetics of the polymerization reaction. This can include monitoring the formation of Si-O-Si bonds, calculating the degree of polymerization, and estimating the reaction rate constants. You can also calculate the radial distribution functions (RDFs) and other structural properties to characterize the polymer network.

GPT-4 (continue)

6. Validation: Compare the simulation results with experimental data to validate the accuracy of the MD model. If necessary, refine the force field parameters or simulation setup to improve the agreement between the simulations and experiments.

Keep in mind that MD simulations of polymerization reactions can be computationally demanding due to the long timescales and large system sizes involved. You may need to use parallel computing resources and efficient simulation algorithms to achieve the desired level of accuracy.

Prompt:

Could all experts comment on the strengths and weaknesses using MD simulations to model this problem? If possible, please also include some potential solutions to overcome the weaknesses.

LabGPT:

From an experimental perspective, MD simulations can provide valuable insights into the molecular-level mechanisms and kinetics of TMOS polymerization that are difficult to obtain from experiments alone. However, MD simulations rely on accurate force fields and initial conditions, which may not always be available or well-established for complex systems like TMOS polymerization.

- Strengths:
    1. Atomistic details: MD simulations provide detailed information about the molecular-level processes and interactions during polymerization.
    2. Time evolution: MD simulations allow for the direct observation of the time evolution of the system, which can help elucidate reaction mechanisms and kinetics.
    3. Complementarity: MD simulations can complement experimental data by providing additional information that is difficult to obtain experimentally.
- Weaknesses:
    1. Force field accuracy: The accuracy of MD simulations depends on the quality of the force field used, which may not always be well-established for complex systems like TMOS polymerization.
    2. Timescale limitations: MD simulations are typically limited to nanoseconds or microseconds, which may not be sufficient to capture the full range of timescales relevant to polymerization reactions.
    3. Computational cost: MD simulations of polymerization reactions can be computationally expensive due to the large system sizes and long timescales involved.
- Potential solutions:
    1. Force field validation and refinement: Validate and refine the force field parameters by comparing the simulation results with experimental data or high-level quantum chemistry calculations.
    2. Enhanced sampling techniques: Use enhanced sampling methods, such as metadynamics or replica exchange MD, to overcome the high energy barriers associated with the reaction and explore longer timescales.
    3. Multi-scale modeling: Combine MD simulations with other computational methods, such as quantum chemistry calculations or coarse-grained simulations, to improve the accuracy and efficiency of the simulations.

CompGPT:

As a computational expert, I agree with LabGPT's assessment. MD simulations can provide valuable insights into the TMOS polymerization process, but their accuracy and applicability depend on the quality of the force field and the ability to overcome timescale limitations.

TheoGPT:

I concur with both LabGPT and CompGPT. MD simulations can offer valuable information about the TMOS polymerization process, but their accuracy is contingent on the force field and the ability to access relevant timescales. Combining MD simulations with other computational methods and experimental data can help overcome some of these limitations and provide a more comprehensive understanding of the polymerization kinetics.

GPT-4 (continue)

Prompt:

Thank you all for your comments. I'm interested in the last sentence by TheoGPT: "Combining MD simulations with ... the polymerization kinetic". Could all experts comment on any possible way(s) to realize this combination of data and how it can help people to better understand the polymerization kinetics?

LabGPT:

Combining MD simulations with experimental data and other computational methods can provide a more comprehensive understanding of the polymerization kinetics by leveraging the strengths of each approach. Here are some ways to achieve this combination:

1. Force field validation: Use experimental data (e.g., NMR, FTIR, DLS) to validate and refine the force field parameters used in MD simulations. This can improve the accuracy of the simulations and ensure that they are consistent with experimental observations.

2. Reaction mechanism elucidation: Compare the reaction pathways and intermediates observed in MD simulations with those inferred from experimental data (e.g., kinetic measurements, spectroscopic studies) to gain insights into the molecular-level mechanisms of the polymerization reaction.

3. Kinetic model development: Use MD simulations to estimate reaction rate constants and other kinetic parameters, which can be incorporated into a kinetic model of the polymerization process. This model can then be compared with experimental kinetic data to validate its accuracy and predictive capabilities.

CompGPT:

In addition to LabGPT's suggestions, combining MD simulations with other computational methods can further enhance the understanding of polymerization kinetics:

1. Quantum chemistry calculations: Use high-level quantum chemistry methods (e.g., DFT) to calculate the reaction energetics and transition states, which can be used to estimate reaction rate constants and validate the force field parameters used in MD simulations.

2. Kinetic Monte Carlo (KMC) simulations: Use MD-derived reaction rate constants and pathways as input for KMC simulations to model the stochastic behavior of the reacting species and predict the overall kinetics of the polymerization process.

3. Coarse-grained simulations: Develop coarse-grained models based on the atomistic MD simulations to study the polymerization process on larger length and timescales, which can provide insights into the macroscopic properties of the polymer network.

TheoGPT:

Combining MD simulations with experimental data and other computational methods can help overcome the limitations of each individual approach and provide a more complete understanding of the polymerization kinetics. By integrating information from multiple sources, researchers can develop more accurate and predictive models of the polymerization process, which can be used to guide the design and optimization of new materials and processes.

Figure 4.48: LabGPT, CompGPT, and TheoGPT for the modeling or measuring the kinetics of TMOS.

5 Materials Design

5.1 Summary

In this chapter, we examine GPT-4's capabilities in the realm of materials design. We devise a comprehensive set of tasks encompassing a broad spectrum of aspects in the material design process, ranging from initial conceptualization to subsequent validation and synthesis. Our objective is to assess GPT-4's expertise and its capacity to generate meaningful insights and solutions in real-world applications. The tasks we design cover various aspects, including background knowledge, design principles, candidate identification, candidate structure generation, property prediction, and synthesis condition prediction. By addressing the entire gamut of the design process, we aim to offer a holistic evaluation of GPT-4's proficiency in materials design, particularly for crystalline inorganic materials, organic polymers, and more complex materials such as metal-organic frameworks (MOFs). It is crucial to note that our assessment primarily focuses on providing a qualitative appraisal of GPT-4's capability in this specialized domain while obtaining a statistical score is pursued only when feasible.

Through our evaluation, we summarize the capabilities of GPT-4 in materials design as follows:

- Information memorization: Excels in memorizing information and suggesting design principles for inorganic crystals and polymers. Its understanding of basic rules for materials design in textual form is remarkable. For instance, when designing solid-state electrolyte materials, it can competently propose ways to increase ionic conductivity and provide accurate examples (Sec. 5.2).
- Composition Creation: Proficient in generating feasible chemical compositions for new inorganic materials (Fig. 5.5).
- Synthesis Planning: Exhibits satisfactory performance for synthesis planning of inorganic materials (Fig. 5.14).
- Coding Assistance: Provides generally helpful coding assistance for materials tasks. For example, it can generate molecular dynamics and DFT inputs for numerous property calculations and can correctly utilize many computational packages and construct automatic processing pipelines. Iterative feedback and manual adjustments may be needed to fine-tune the generated code (Sec. 5.7).

Despite the capabilities, GPT-4 also has potential limitations in material science:

- Representation: Encounters challenges in representing and proposing organic polymers and MOFs (Sec. 5.3).
- Structure Generation: Limited capability for structure generation, particularly when generating accurate atomic coordinates (Fig. 5.4).
- Predictions: Falls short in providing precise quantitative predictions in property prediction. For instance, when predicting whether a material is metallic or semi-conducting, its accuracy is only slightly better than a random guess (Table. 11).
- Synthesis Route: Struggles to propose synthesis routes for organic polymeric materials not present in the training set without additional guidance (Sec. 5.6.2).

In conclusion, GPT-4 demonstrates a promising foundation for assisting in materials design tasks. Its performance in specific areas, such as structure generation and property prediction (in our studied benchmarks), could be further improved by incorporating additional training data with supplementary modalities, such as molecular graphs or dedicated AI models. As LLMs like GPT-4 continue to advance, we anticipate an increasing potential for more sophisticated and accurate assistance in materials design, ultimately leading to more efficient and effective materials discovery and development.

5.2 Knowledge memorization and designing principle summarization

This section tests the performance of GPT-4 serving as a knowledge base for materials design as well as its capability for summarizing from the knowledge a list of design principles, which could guide the candidate selection later. Two sets of test cases are used for inorganic crystals and organic polymers and the evaluation is done by domain experts. For inorganic crystals, we use solid electrolytes as an example to see whether GPT-4 knows and can correctly reflect the known solid electrolytes. They are a family of materials that can conduct ions and are critical for next-generation energy storage. We start by querying about the classification of current solid electrolytes with different requirements on the classification standards, e.g., general chemistry

and anion types. We also ask for examples based on the classification standards. As shown in Fig. 5.1 as an example, all answers here are factual and most of them are correct. Since these classification standards are not well represented in the literature, GPT-4 should have a relatively clear understanding of the meaning of chemistry.

GPT-4

Prompt:

For inorganic solid electrolytes, can you categorize them into families based on the anion chemistry?

GPT-4:

Certainly! Inorganic solid electrolytes can also be categorized into families based on the anion chemistry. Here are some of the main categories:

1. Oxide-based electrolytes: These electrolytes have oxide anions (O2-) in their structure. Examples include:
    - a. Perovskite-type: e.g., $La_{0.67}Sr_{0.33}Al_{0.67}Ta_{0.33}O_3$ (LSAT) for sodium-ion conductors.
    - b. NASICON (Na Super Ionic CONductor)-type: e.g., $Li_{1.5}Al_{0.5}Ge_{1.5}(PO_4)_3$ (LAGP) for lithium-ion conductors.
    - c. Garnet-type: e.g., $Li_7La_3Zr_2O_{12}$ (LLZO) for lithium-ion conductors.
2. Sulfide-based electrolytes: These electrolytes have sulfide anions (S2-) in their structure. Examples include:
    - a. Thio-LISICON (Li Super Ionic CONductor)-type: e.g., Li_4GeS_4 and Li_4SiS_4.
    - b. Argyrodite-type: e.g., Li_6PS_5Cl and Li_6PS_5Br.
    - c. Thiophosphate-based: e.g., $Li_{10}GeP_2S_{12}$ (LGPS).
3. Phosphate-based electrolytes: These electrolytes have phosphate anions (PO_4) in their structure. Examples include:
    - a. NASICON (Na Super Ionic CONductor)-type: e.g., $Na_3Zr_2Si_2PO_{12}$ and $Li_{1.5}Al_{0.5}Ge_{1.5}(PO_4)_3$ (LAGP) for lithium-ion conductors.
    - b. Thiophosphate-based: e.g., $Li_{10}GeP_2S_{12}$ (LGPS).
4. Halide-based electrolytes: These electrolytes have halide anions (F-, Cl-, Br-, I-) in their structure. Examples include:
    - a. Anti-perovskite-type (lithium-ion conductors): e.g., Li_3OCl and Li_3OBr.
    - b. Argyrodite-type: e.g., Li_6PS_5Cl and Li_6PS_5Br.
    - c. Fluoride-based Materials: e.g., LaF_3 and PbF_2.

These categories represent some of the major families of inorganic solid electrolytes based on their chemistry. Each family has its unique properties and potential applications in various electrochemical devices.

Figure 5.1: Categorization of inorganic solid electrolytes. Answer evaluation: All answers here are factual, and the answers agree with those from the domain expert. Due to training data constraint, some new development is not included e.g., new halide-type LixMyXz M=Y, Er, Ti etc., X=Cl, Br, I.

Conclusion for inorganic crystals: GPT-4 performs very well in retrieving information. In all tests, the answers are factual. In rare cases, there are some categorization mistakes.

Based on the above understanding, we further carry out tests on the summarization of design principles. To achieve this, we carry out a step-by-step conversational introduction to the task by querying the major design principle for inorganic solid electrolytes to achieve high conductivity, asking for examples of applying these principles, and finally providing a new idea from a domain expert and asking it to evaluate and reason about. As shown in Fig. 5.2, out of the 8 design rules, 7 are deemed correct and the left one is considered still factual but should not be regarded as a design principle. Interestingly, after searching the internet, it is not obvious these design principles are direct copies from review papers but could be a summary of the field by GPT-4 itself. In addition to that, we ask for examples of these design principles being applied to test whether there is a true understanding of the physics and whether the results are reasonably plausible. An example is shown in Fig. 5.3.

GPT-4

Prompt:
What is the major designing principal for inorganic solid electrolytes to achieve high conductivity?

GPT-4:
Certainly! Inorganic solid electrolytes can also be categorized into families based on the anion chemistry. Here are some of the main categories:

1. Crystal structure: The crystal structure should allow for easy migration of ions through interconnected pathways or channels. For instance, NASICON-type materials have a 3D interconnected framework that facilitates ion transport.

2. Lattice parameters: The size and geometry of the unit cell should be optimized to reduce the activation energy for ion transport. This can be achieved by adjusting the lattice constants or introducing dopants to modify the structure.

3. Ion size and coordination: Smaller ions generally exhibit higher mobility due to reduced electrostatic interactions. Additionally, optimizing the coordination environment around the mobile ion can lower the activation energy for ion migration.

4. Defect chemistry: Defects in the crystal lattice, such as vacancies or interstitials, can facilitate ion transport. Designing materials with a controlled level of defects can improve ionic conductivity.

5. Polymorphism: Some materials exhibit different crystal structures (polymorphs) with varying ionic conductivities. Identifying and stabilizing the high-conductivity polymorph can be crucial for achieving high-performance solid electrolytes.

6. Dopants and solid solutions: Introducing dopants or forming solid solutions can modify the crystal structure, lattice parameters, and defect chemistry to enhance ionic conductivity. For instance, doping zirconium with yttrium in garnet-type $Li_7La_3Zr_2O_{12}$ (LLZO) increases the lithium-ion conductivity.

7. Grain boundaries and interfaces: The properties of grain boundaries and interfaces in polycrystalline materials can significantly affect ionic conductivity. Minimizing grain boundary resistance and optimizing the microstructure can improve overall ionic transport.

8. Mechanical stability: In addition to high ionic conductivity, solid electrolytes should have good mechanical stability to maintain their structural integrity during operation. This can be achieved by optimizing the crystal structure, composition, and processing conditions.

By considering these design principles and tailoring the material properties, researchers can develop inorganic solid electrolytes with high ionic conductivity for various electrochemical applications.

Figure 5.2: Design principle summarization for solid electrolytes. Evaluation: 7/8 design principles summarized are correct. The 8th one is a criterion, not a design principle. Therefore, while it is factually correct, it does not belong here. The coverage of the design principles is good. Very comprehensive.

GPT-4

Prompt:
Can you elaborate how these design principles are applied using the examples you mentioned earlier?

GPT-4:
Certainly! Let's see how the design principles for inorganic solid electrolytes are applied in the examples I mentioned earlier:

NASICON-type $Li_{1.5}Al_{0.5}Ge_{1.5}(PO_4)_3$ (LAGP): In NASICON-type materials, the 3D framework of corner-sharing MO_6 (M = metal) octahedra and PO_4 tetrahedra creates open channels for ion transport. The high ionic conductivity of LAGP arises from the optimized lattice parameters and the presence of lithium vacancies. Substituting Al and Ge with other elements, such as Ti and P, can further increase ionic conductivity by modulating the lattice constants and defect chemistry.

Figure 5.3: An example of the design principles.

In addition to crystalline inorganic materials, we also extend our test to polymeric materials which have significantly different representations and properties. For organic polymeric materials, we apply similar test routines by first asking about the general knowledge and then going for domain-specific tasks. Specifically,

we ask to summarize the 10 most important properties of polymers with reasoning. The answers provided are molecular weight, glass transition temperature (Tg), crystallinity, density, tensile strength, elongation at break, modulus of elasticity, impact resistance, chemical resistance, and thermal stability. These are in general agreement what a domain expert's answer. We then proceed to the representation of polymer structures and test on three cases including bisphenol A, PTFE, Teflon, and Tween800. An example is shown in Fig. 5.4. For full details, see Appendix C.2.

GPT-4

Prompt:

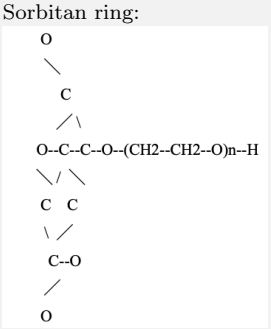
Provide the structure of Tween80.

GPT-4:

Tween 80, also known as Polysorbate 80, is a nonionic surfactant and emulsifier often used in foods, cosmetics, and pharmaceuticals. Its chemical name is polyoxyethylene (20) sorbitan monooleate. The structure of Tween 80 is complex due to its polymeric nature and the presence of various components in its structure, including sorbitan, oleic acid, and polyoxyethylene chains.

The structure of Tween 80 can be represented as follows:

Sorbitan ring:



Here, the sorbitan ring (a sugar alcohol derived from sorbitol) is esterified with oleic acid (a monounsaturated fatty acid) at one of its hydroxyl groups. The remaining hydroxyl groups are etherified with polyoxyethylene chains, represented as (CH2–CH2–O)n, where n represents the number of repeating oxyethylene units (in this case, approximately 20).

Please note that this representation is a simplified version of the structure.

Figure 5.4: Evaluation: This structure is nonsense. It is also confusing because the response tells me I am looking at both the structure of Tween80 and sorbitan.

We also test GPT-4's capability in representing polymers effectively. In this case, BigSMILES is a convenient and efficient choice. We use Nafion and polyethylene as an example. Unfortunately, we find that GPT-4 has limited capability in representing the polymer structure using the BigSMILES representation thus proposing new candidates of polymers may encounter some difficulty. See Appendix C.4 for reference.

Conclusion for polymers: GPT-4 has a clear understanding of the properties associated with polymers and can recognize common polymer names. It has a difficult time drawing out the polymer structure in ASCII for polymers that contain more complex functionality such as aromatic groups or rings.

In an overall conclusion, GPT-4 can perform knowledge memorization and design principle summarization with relatively high credibility. Given proper prompt choice, it can in general provide credible knowledge and general guidelines on how to design families of materials that have been tested here.

5.3 Candidate proposal

This section tests the capability of GPT-4 to propose candidates for new materials. This section mostly deals with the capability of generating novel and feasible candidates. The properties of interest will be assessed in the next few sections. Specifically, this section will focus on three main types of materials, i.e., inorganic crystals, organic polymers, and metal-organic frameworks (MOFs). For inorganic crystals, the compositions will be generated as strings. For polymers, the SMILES strings or polymer name will be the output. For

MOFs, we prompt GPT-4 with the chemical formulas of several building block options and topology from the Reticular Chemistry Structure Resource (RCSR) database. We ask GPT-4 about the compatibility of building blocks and the topology, as well as selecting building blocks to optimize a MOF property.

For inorganic crystals, we first check the capability of GPT-4 in generating a valid chemical composition of materials for a text description of the requirements. We evaluate such capability, query 30 chemical compositions, and validate the generated chemical compositions according to a set of rules. The experiment is repeated 5 times and we report the success rate averaged over the 5 experiments.

GPT-4 is asked to propose 30 chemical compositions given the following prompt:

Prompt

- You are a materials scientist assistant and should be able to help with proposing new chemical composition of materials.
- You are asked to propose a list of chemical compositions given the requirements.
- The format of the chemical composition is AxByCz, where A, B, and C are elements in the periodic table, and x, y, and z are the number of atoms of each element.
- The answer should be only a list of chemical compositions separated by a comma.
- The answer should not contain any other information.
- Propose 30 requirements.

Here, the {requirements} is a text description of the requirements of the chemical composition we ask GPT-4 to generate. We evaluate GPT-4's capabilities in the following 3 different types of tasks:

Propose metal alloys. We ask GPT-4 to propose new compositions of metal alloys. The {requirements} are {binary metal alloys}, {ternary metal alloys}, and {quaternary metal alloys}. The proposed chemical composition is valid if 1) the number of elements is correct (i.e., 2 elements for binary alloys); and 2) all the elements in the proposed chemical composition are metal. The results are summarized in the left part of Fig. 5.5.

Evaluation: GPT-4 achieved high success rates in generating compositions of metal alloys. It can generate compositions with the correct number of elements with a 100% success rate, e.g., for binary, it generates 2 elements, for ternary, it generates 3 elements. It also understands the meaning of alloys and can generate compositions with all metal elements with a high success rate. Occasionally, it generates non-metal compositions (e.g., Fe_3C, AlSi, AlMgSi). The successful chemical compositions look reasonable from a material science perspective (e.g., ZrNbTa, CuNiZn, AuCu), but we haven't further verified if these alloys are stable.

Propose ionic compounds. We ask GPT-4 to propose new compositions of ionic compounds. The requirements are binary ionic compounds, ternary ionic compounds, and quaternary ionic compounds. The proposed chemical composition is valid if 1) the number of elements is correct (i.e., 2 elements for binary ionic compounds); 2) the composition is ionic (i.e., contains both metal and non-metal elements), and 3) the composition satisfies charge balance. The results are summarized in the middle part of Fig. 5.5.

Evaluation: GPT-4 achieved a much lower success rate in this task. Terynary compounds, it has trouble generating charge-balanced compounds. For quaternary compounds, it has trouble generating the correct number of elements. This is probably due to the training set coverage where the compositional space for binary compounds is much smaller than the terynary and quaternary ones. The coverage training data is likely much better when there are fewer elements.

Propose prototypes. We ask GPT-4 to propose new compositions of given crystal prototypes. The requirements are perovskite prototype materials, fluorite prototype materials, half-heusler prototype materials, and spinel prototype materials. The proposed chemical composition is valid if 1) it satisfies the prototype pattern (e.g., for perovskites, it needs to satisfy the ABX_3 pattern); 2) the composition satisfies charge balance. The results are summarized in the right part of Fig. 5.5.

Evaluation: GPT-4 did a satisfying job in this task. It did a great job in peroskites, half-heusler, and spinels. For fluorite, it should generate compounds matching the pattern of AB2, but it confuses the "fluorite prototype" with "fluorides". The latter means any compound matching the pattern AFx, where x is any integer.

For organic polymers, as discussed in the previous section, GPT-4 has limited capability in representing

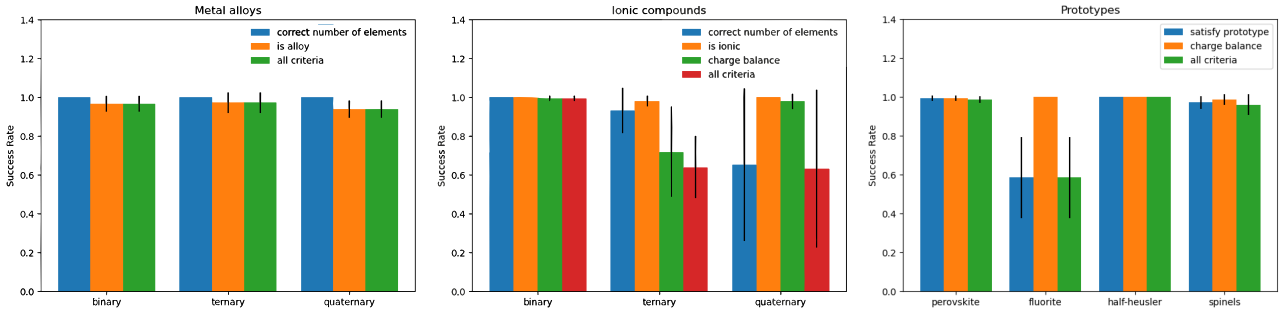


Figure 5.5: Left: the success rate of generating chemical composition of metal alloys. Middle: the success rate of generating the chemical position of ionic compounds. Right: the success rate of generating the chemical composition of given prototypes. The error bar indicates the standard deviation of 5 queries. Some error bar exceeds 1 because it is possible for the sum of mean and stand deviation to exceed 1. E.g., for the ternary ionic compounds, correct number of elements task, the success rates are 1.0, 0.967, 0.7, 1.0, 1.0. Mean is 0.933 and standard deviation is 0.117. The varying capability for different numbers of elements and different types of materials is likely coming from the different difficulty of these tasks and the coverage of the training dataset as discussed in the text.

Table 9: MOF generation experiments.

	tbo: (accuracy 48%)		pcu: (accuracy 58%)	
	RMSD $\geq$ 0.3 Å	RMSD $<$ 0.3 Å	RMSD $\geq$ 0.3 Å	RMSD $<$ 0.3 Å
ChatGPT Reject	22	25	7	25
ChatGPT Accept	27	26	17	51

the polymer structure using the BigSMILES representation thus proposing new candidates of polymers may encounter some difficulty.

Metal-organic frameworks (MOFs) represent a promising class of materials with significant crucial applications, including carbon capture and gas storage. Rule-based approaches involve the integration of building blocks and topology templates and have been instrumental in the development of novel, functional MOFs. The PORMAKE method provides a database of topologies and building blocks, as well as a MOF assembly algorithm. In this study, we assess GPT-4's ability to generate viable MOF candidates based on PORMAKE, considering both feasibility and inverse design capability. Our first task evaluates GPT-4 's ability to discern whether a reasonable MOF can be assembled given a set of building blocks and a topology. This task necessitates a spatial understanding of the 3D structures of both building blocks and topology. Our preliminary study focuses on the topologies of two well-studied MOFs: the 'tbo' topology for HKUST-1 and the 'pcu' topology for MOF-5. 'pcu' and 'tbo' are acronyms for two types of MOF topologies from the Reticular Chemistry Structure Resource (RCSR). 'pcu' stands for "primitive cubic", which refers to a type of MOF with a simple cubic lattice structure. 'tbo' stands for "twisted boracite" which refers to another type of MOF with a more complex structure. The 'tbo' topology is characterized as a 3,4-coordinated ((3,4)-c) net, while the 'pcu' topology is a 6-c net. In each experiment, we propose either two random node building blocks (3-c and 4-c) with 'tbo' or one random node building block (6-c) with 'pcu', then inquire GPT-4 about their compatibility. The detailed methods are listed in Appendix C.11. With 100 repeated experiments, we generate a confusion matrix for each topology in Table 9. Following several previous studies, we say a topology and the building blocks are compatible when the PORMAKE algorithm gives an RMSD of less than 0.3 Åfor all building blocks.

When assessing the compatibility between building blocks and topology, GPT-4 consistently attempts to match the number of connection points. While this approach is a step in the right direction, it is not sufficient for determining the feasibility of assembling a MOF. GPT-4 shows a basic understanding of the spatial connection patterns in different RCSR topologies, but it is prone to errors. Notably, GPT-4 often does not engage in spatial reasoning beyond counting connection points, even though our experimental prompts

have ensured that the number of connection points is congruent.

The second task involves designing MOFs with a specific desired property, as detailed in Appendix C.11. We concentrate on the pcu topology and the three most compatible metal nodes, determined by the RMSD between the building block and the topology node's local structure (all with RMSD < 0.03 Å). These nodes are N16 ($C_6O_{13}X_6Zn_4$), N180 ($C_{16}H_{12}Co_2N_2O_8X_6$), and N295 ($C_{14}H_8N_2Ni_2O_8X_6$) in the PORMAKE database, containing 23, 34, and 40 atoms, excluding connection points. The PORMAKE database includes 219 2-c linkers. For each experiment, we randomly sample five linkers, resulting in a design space of 15 MOFs. We then ask GPT-4 to recommend a linker-metal node combination that maximizes the pore-limiting diameter (PLD). By assembling all the MOFs using PORMAKE and calculating the PLD using Zeo++, we evaluate GPT-4's suggestions.

This is a challenging task that requires a spatial understanding of the building blocks and the pcu topology, as well as the concept of pore-limiting diameter. In all five experiments, GPT-4 fails to identify the MOF with the highest PLD. In all 5 cases, GPT-4 opts for the metal node C16H12Co2N2O8X6, which contains the most atoms. However, N16, with the fewest atoms, consistently yields the highest PLD in every experiment. GPT-4 correctly chooses the linker molecule that generated the highest PLD in two out of the five experiments. Overall, GPT-4 frequently attempts to compare the sizes of different building blocks based on atom count, without thoroughly considering the geometric properties of building blocks in metal-organic frameworks. As a result, it fails to propose MOFs with maximized PLD. Examples of outputs are included in Appendix C.11.

In conclusion, for inorganics, GPT-4 is capable of generating novel but chemically reasonable compositions. However, for organic polymers, under our testing setup, it is relatively hard for it to generate reasonable structure representations of polymers. Therefore, proposing new polymers may be difficult or need other ways to prompt it. For MOFs, GPT-4 demonstrates a basic understanding of the 3D structures of building blocks and topologies. However, under our testing setup, it struggles to reason beyond simple features such as the number of connection points. Consequently, its capability to design MOFs with specific desired properties is limited.

5.4 Structure generation

This section tests the capability of GPT-4 in assessing GPT-4's capability in generating atomic structures for inorganic materials. Two levels of difficulty will be arranged. The first ability is to generate some of the key atomic features right, e.g., bonding and coordination characteristics. Second is the capability of directly generating coordinates. First, we benchmark GPT-4's capability in predicting the coordination number of inorganic crystalline materials. This is done by feeding the model the chemical compositions and some few-shot in-context examples. As shown in the table below. The materials in the test set are well known, so it is reasonable to expect good performance. GPT-4 managed to successfully report the correct coordination environment for 34 out of 84 examples, and where it is incorrect often only off-by-one: this level of accuracy would likely be difficult to achieve even for a well-trained materials scientist, although human control has not been performed. GPT-4 also makes several useful observations. Although it gets the coordination incorrect, it notes that Pb_3O_4 had two different coordinations in the Pb site. Although it does not acknowledge that two different polymorphs were possible, it does notes that $CaCO_3$ has been duplicated in the prompt. It adds an additional row for the oxygen coordination in $NbSO_4$, although this is omitted in the prompt. In one session, it also notes "Keep in mind that the coordination numbers provided above are the most common ones found in these materials, but other possibilities might exist depending on the specific crystal structure or conditions under which the material is synthesized." Therefore, the results are qualitatively good but quantitatively poor in assessing the atomic coordinates in general.

Next, we test one of the most difficult tasks in materials design, i.e., generating materials' atomic structures. This is a task also known as crystal structure prediction. The input and outputs are the chemical composition and the atomic coordinates (together with lattice parameters). We try a few prompts and ask GPT-4 to generate several types of outputs. In expectation, the generated structures are not good. For most of the cases we try, the structures do not even warrant a check by density functional theory computations as they are clearly unreasonable. Fig. 5.6 shows the structure of Si generated by GPT-4 and the correct structure from the materials project. Without very careful prompting and further providing additional information like space group and lattice parameter, it is difficult for GPT-4 to generate sensible structures.

We further test GPT-4 on a novel structure that is not in the training set during the training of the model. The example used here is a material LiGaOS that is hypothetically proposed using conventional crystal structure prediction. In this way, we can benchmark against this ground truth [47]. As shown in

Table 10: Prediction of atomic coordinates with GPT-4.

Formula	Element	Correct CN	GPT-4 CN
$BaAl_2O_4$	Ba	9	(provided as example)
$BaAl_2O_4$	Al	4	(provided as example)
$BaAl_2O_4$	O	2	(provided as example)
Be_2SiO_4	Be	4	4
Be_2SiO_4	Si	4	4
Be_2SiO_4	O	3	2
$Ca(BO_2)_2$	Ca	8	7
$Ca(BO_2)_2$	B	3	3
$Ca(FeO_2)_2$	Ca	8	6
$Ca(FeO_2)_2$	Fe	6	4
$Ca(FeO_2)_2$	O	5	2
Fe_2SiO_4	Fe	6	6
...	...	...	...

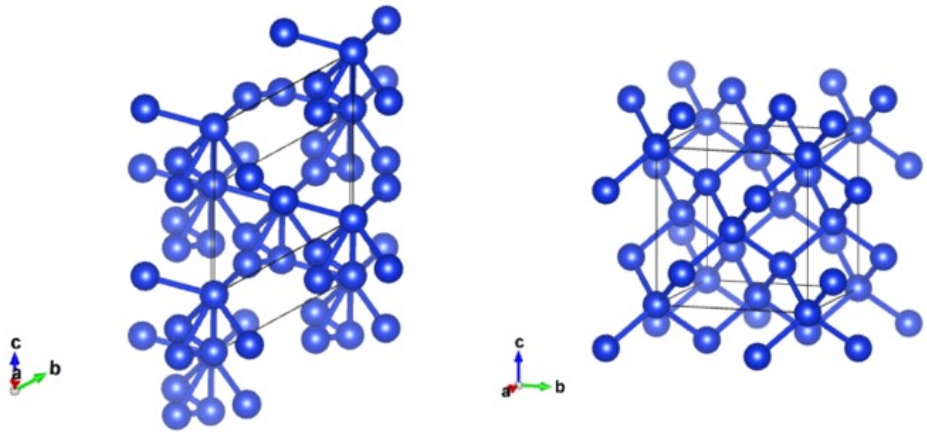


Figure 5.6: crystal structure of silicon predicted by GPT-4 and from materials project.

Appendix C.5, GPT-4 does not perform well qualitatively as well.

In conclusion, GPT-4 is capable of some physical and chemical knowledge which can assist it in generating reasonable descriptive features of the structures such as atomic coordination numbers. However, to generate stable structures by giving it a composition, it is very unlikely to work under the current capability range. For polymers, the atomic structures are much more complex and cannot be easily predicted even with the quantum chemical method. Therefore, we leave this to use the coding capability to drive tools to access the structures for polymer materials.

5.5 Property prediction

This section tests the capability of GPT-4 in predicting the properties of materials. This is essential for the screening of materials before synthesis. Both qualitative and quantitative studies will be given.

5.5.1 MatBench evaluation

For inorganic materials, we use MatBench [24] as a benchmark. This dataset is a large dataset of inorganic compounds whose properties are computed using quantum chemical methods such as density functional theory. We select two subtasks, namely `expt_is_metal` and `mp_gap`. These include both classification tasks and regression tasks. Note that the tests are only done using the following prompts and all analysis is based on the results of the current prompt. Different prompts will like give slightly different results. However, we don't expect it will change the qualitative conclusions(Fig. 5.7,5.8):

GPT-4

System:

You are an AI that helps people on the following task:

Dataset: matbench_expt_is_metal

Description: Matbench v0.1 test dataset for classifying metallicity from composition alone. Retrieved from Zhuo et al. supplementary information. Deduplicated according to composition, ensuring no conflicting reports were entered for any compositions (i.e., no reported compositions were both metal and nonmetal). For benchmarking w/ nested cross-validation, the order of the dataset must be identical to the retrieved data; refer to the Automatminer/Matbench publication for more details.

Your output should be True/False.

Example 1:

$Ag(AuS)_2$

True

Example 2:

$Ag(W_3Br_7)_2$

True

Example 3:

$Ag_{0.5}Ge_1Pb_{1.75}S_4$

False

Test input:

$Ag_{0.5}Ge_1Pb_{1.75}Se_4$

GPT-4:

True

Figure 5.7: Example of prompt on dataset matbench_expt_is_metal with 3 examples.

GPT-4

System:

You are an AI that helps people with the following task:
Dataset: matbench_mp_gap
Matbench v0.1 test dataset for predicting DFT PBE band gap from the structure. Adapted from Materials Project database. Removed entries having formation energy (or energy above the convex hull) of more than 150meV and those containing noble gases. Retrieved April 2, 2019. For benchmarking w/ nested cross-validation, the order of the dataset must be identical to the retrieved data; refer to the Automatminer/Matbench publication for more details.
Your output should be a number.

Example 1:

```
Full Formula (K4 Mn4 O8)
Reduced Formula: KMnO2
abc   :   6.406364   6.406467   7.044309
angles: 117.047604 117.052641  89.998496
pbc   :       True       True       True
Sites (16)
  #  SP           a         b         c    magmom
---  ----  --------  --------  --------  --------
  0  K     0.000888  0.002582  0.005358    -0.005
  1  K     0.504645  0.002727  0.005671    -0.005
  2  K     0.496497  0.498521  0.493328    -0.005
  3  K     0.496657  0.994989  0.493552    -0.005
  4  Mn    0.993288  0.493319  0.986797     4.039
  5  Mn    0.005896  0.005967  0.512019     4.039
  6  Mn    0.005704  0.505997  0.511635     4.039
  7  Mn    0.493721  0.493503  0.987141     4.039
  8  O     0.190852  0.749673  0.49925     -0.097
  9  O     0.74987   0.691083  0.999719    -0.097
 10  O     0.750076  0.308682  0.000128    -0.097
 11  O     0.807914  0.749345  0.498618    -0.097
 12  O     0.241545  0.298978  0.98309     -0.073
 13  O     0.815297  0.257926  0.515695    -0.073
 14  O     0.20002   0.257714  0.515284    -0.074
 15  O     0.241818  0.684193  0.983662    -0.074
```

1.3321999999999998

Test input:

```
Full Formula (Ba2 C4)
Reduced Formula: BaC2
abc   :   4.514204   4.514204   8.340733
angles:  72.957031  72.957031  63.894155
pbc   :       True       True       True
Sites (6)
  #  SP           a         b         c    magmom
---  ----  --------  --------  --------  --------
  0  C     0.106722  0.414028  0.445527         0
  1  C     0.585972  0.893278  0.054473         0
  2  C     0.414028  0.106722  0.945527         0
  3  C     0.893278  0.585972  0.554473         0
  4  Ba    0.81021   0.18979   0.25             0
  5  Ba    0.18979   0.81021   0.75             0
```

GPT-4:

2.0599999999999996 (the ground truth is 2.1132)

Figure 5.8: Example of prompt on dataset matbench_mp_gap with 1 example.

The results for both classification tasks on whether the materials are metallic and the regression task on the electronic band gaps are shown in Table 11. We perform different numbers of random few-shot examples during evaluations, which range from 1 to 100. For both cases, though the prediction performances are still far away from the state-of-the-art (SOTA) results, they are consistently better than random guesses (dummy) with the increase of the few-shot demonstrations. Therefore, GPT-4 has a certain capability in calculating and

predicting the properties, but still lacks the capability for the accurate prediction of properties of inorganic materials, calling for dedicated models or further development of GPT-4.

Table 11: GPT-4 on MatBench Tasks.

Method	expt_is_metal		mp_gap	
	F1↑	Acc.↑	MAE↓	RMSE↓
Dummy	0.4913	0.4924	1.3272	1.5989
SOTA	0.9200	0.9209	0.1559	0.3956
$k = 1$	0.5385	0.5689	1.2055	1.6776
$k = 3$	0.5672	0.7795	1.1936	1.7159
$k = 5$	0.6279	0.6903	0.7859	1.2718
$k = 10$	0.7013	0.7617	0.6304	1.1167
$k = 20$	0.6923	0.7538	0.6419	1.1934
$k = 50$	0.7317	0.7864	0.4067	0.9072
$k = 100$	0.7397	0.7934	-	-

5.5.2 Polymer property

For polymers, we use several ad hoc questions to understand GPT-4 general capability. We first query a number of properties that can we use to compare and evaluate polymer materials. The answers we get include molecular weight, glass transition temperature, etc. al., which is satisfactory. Following that, we choose a system described in a recent work [55] and employ GPT-4 to predict the properties of this novel polymer. While GPT-4 accurately predicts the qualitative aspects of thermal conductivity, it falls short of providing a quantitative answer. Some examples are shown below, refer to Fig. 5.9 for details.

GPT-4

Prompt:

1,4-polybutadiene, atactic polystyrene, PG - PPO - PG Copolymers, which polymer has the highest Tg? Among 1,4-polybutadiene, atactic polystyrene, and PG-PPO-PG copolymers, atactic polystyrene has the highest glass transition temperature (Tg).

GPT-4:

Here are the approximate Tg values for each polymer: 1,4-polybutadiene: -90 to -100°C Atactic polystyrene: 90 to 100°C PG-PPO-PG copolymers (Pluronic or Poloxamer): The Tg of these copolymers can vary depending on the specific composition and molecular weight. Generally, their Tg values are in the range of -50 to 20°C.

Prompt:

We want to synthesize a two-dimensional crystalline polymer of C60, its constituent fullerene subunits arranged hexagonally in a covalently interconnected molecular sheet. Can you predict the thermal conductivity of the material compared to molecule C60?

GPT-4:

As an AI language model, I am unable to perform experimental predictions or calculations directly. However, I can provide you with some general insights. The thermal conductivity of a two-dimensional crystalline polymer of C60 with hexagonally arranged fullerene subunits would likely be different from that of an individual C60 molecule. The thermal conductivity of a material depends on its structure, bonding, and the arrangement of its constituent atoms or molecules. In the case of a two-dimensional crystalline polymer of C60, the covalent interconnections between the fullerene subunits would create a more extended and ordered structure compared to an individual C60 molecule. This could potentially lead to an increase in thermal conductivity due to the enhanced pathways for heat transfer through the material. However, predicting the exact thermal conductivity of such a material would require detailed computational modeling or experimental measurements. I recommend consulting with a material scientist or conducting further research using computational methods like density functional theory (DFT) to obtain a more accurate prediction of the thermal conductivity of this material.

Figure 5.9: Prediction of polymer properties. Evaluation: We refer to the Polymer Property Predictor and Database for comparison, the experimental Tg values are 1,4-polybutadiene- $\sim$ -90°C, atactic polystyrene- 95°C, PG - PPO - PG Copolymers- 18°C. So GPT-4 demonstrates good capabilities in representing the quantitative and qualitative properties of polymer materials.

We then evaluate the capability of GPT-4 to calculate or predict polymer properties based on available datasets. The first task is to determine if GPT-4 could reason how to compute the volume of the unit cell and dielectric constant from input data taken from [33]. The second task is to assess if GPT-4 could learn the relationship between structure and synthesizability by providing a polymer structure in SMILES format and synthetic accessibility score.

(1) Predicting volume of unit cell from CIF. We give an example of predicting the volume of the unit cell in Fig. 5.10. From the case, we can see that the prediction is reasonable since the value is close to the ground truth 572.42.

GPT-4

Prompt:

You are an AI that helps people on the following task:
Dataset: polymer_cif
Description: Polymer_cif is a dataset containing optimized polymer structures and accompanied properties calculated with first-principles computations. For benchmarking w/ nested cross-validation, the order of the dataset must be identical to the retrieved data;
Compute the Volume of the unit cell (A^3) from the data:
_cell_length_a 6.74748725
_cell_length_b 8.14451453
_cell_length_c 10.49178793
_cell_angle_alpha 90.00214572
_cell_angle_beta 90.00083977
_cell_angle_gamma 96.88413802

GPT-4:

To compute the volume of the unit cell, we can use the formula:
Volume = a * b * c * sqrt(1 - $\cos^2(\alpha)$ - $\cos^2(\beta)$ - $\cos^2(\gamma)$ + 2 * $\cos(\alpha)$ * $\cos(\beta)$ * $\cos(\gamma)$)
where a, b, and c are the cell lengths, and α, β, γ are the cell angles in degrees.
First, we convert the cell angles from degrees to radians:
α = 90.00214572 * (pi / 180) = 1.57081208 radians
β = 90.00083977 * (pi / 180) = 1.57079674 radians
γ = 96.88413802 * (pi / 180) = 1.69089268 radians

Next, we compute the cosines of the angles:
$\cos(\alpha)$ = cos(1.57081208) = -0.00003735
$\cos(\beta)$ = cos(1.57079674) = 0.00001463
$\cos(\gamma)$ = cos(1.69089268) = -0.05881449
Now we can calculate the volume using the formula:
Volume = 6.74748725 * 8.14451453 * 10.49178793 * sqrt(1 - $(-0.00003735)^2$ - $(0.00001463)^2$ - $(-0.05881449)^2$ + 2 * (-0.00003735) * (0.00001463) * (-0.05881449))
Volume ≈ 6.74748725 * 8.14451453 * 10.49178793 * sqrt(0.99999999)
Volume ≈ 6.74748725 * 8.14451453 * 10.49178793 * 0.999999995
Volume ≈ 573.26897645 A^3
Therefore, the volume of the unit cell is approximately 573.27 A^3.

Figure 5.10: Prediction of the volume of the unit cell of a crystal from CIF files. Evaluation: This is reasonable as the result 573.27 is also close to the correct answer 572.41949808.

(2) Predicting the Dielectric Constant from CIF. A set of 20 randomly sampled CIFs are tested on the dielectric properties as well, with different numbers of demonstration examples (k). The results are in Table 12. From the table, we can see that the results do not vary much, and the MAE/MSE values are relatively on a large scale.

Table 12: Prediction of dielectric properties of polymers. Evaluation: It appears that GPT-4 has trouble accurately predicting the dielectric constant of a polymer from a CIF file.

	$k=1$			$k=3$			$k=5$		
	Electronic	Ionic	Total	Electronic	Ionic	Total	Electronic	Ionic	Total
MAE	1.17	1.17	2.00	1.26	1.26	2.37	1.47	1.95	2.00
MSE	2.80	5.74	9.30	3.18	10.07	9.98	4.74	8.47	8.12

(3) Predicting SA Score on Pl1M_v2 Dataset. Finally, we evaluate GPT-4 performance in predicting the synthesizability. We use the Synthetic Accessibility (SA) score as a measure to quantify the synthesizability. We use 100 randomly sampled examples from the dataset to predict the SA score, the prompt design, and usage are shown in Fig. 5.11. After evaluations with different numbers of demonstration examples (k), the results are listed in Table 13.

Table 13: Predicting SA score on Pl1M_v2 Dataset. Evaluation: GPT-4's performance to predict synthesizability accessibility score from a SMILES string appears to improve with increased k examples. The mean and standard deviation (std) of ground truth in this dataset is 3.82 and 0.79.

	$k = 1$	$k = 5$	$k = 10$	$k = 50$	$k = 100$
MSE	1.59	2.04	1.15	0.49	0.29
MAE	0.94	1.09	0.85	0.56	0.40

GPT-4

Prompt:

you are an AI that helps people on the following task:
Dataset: PI1M_v2
Description: PI1M_v2 is a benchmark dataset of ∼1 million polymer structures in p-SMILES data format with corresponding synthetic accessibility (SA) calculated using Schuffenhauer's SA score. For benchmarking w/ nested cross-validation, the order of the dataset must be identical to the retrieved data; Predict the synthetic accessibility score. Your output should exactly be a number that reflects the SA score, without any other text.

Example 1:
example1 input
SA score 1

(...more examples omitted)

test input

GPT-4:

...

Figure 5.11: Prompt used to predict the SA score.

From the above three properties prediction, We can see that GPT-4 has some ability to make correct predictions, and with the increased number of k, the predicted performance could be improved with large probability (but requires a large number of few-shot examples, Table 12 and 13), which demonstrates the few-shot learning ability of GPT-4 for the polymer property prediction.

5.6 Synthesis planning

5.6.1 Synthesis of known materials

This section checks the capability of GPT-4 in retrieving the synthesis route and conditions for materials the model has seen during training. To evaluate such ability, we query the synthesis of materials present in the publicly-available text-mining synthesis dataset[19]. The detailed prompting and evaluation pipeline is listed in Appendix C.10. In short, we sample 100 test materials from the dataset at random and ask GPT-4 to propose a synthesis route and compare it with the true label. In Fig. 5.12, we report the three scores as a function of the number of in-context examples provided. We observe that GPT-4 correctly predicts more than half of the precursors, as the average fraction of correct precursors (green bar) is between 0.66 (0 in-context examples) and 0.56 (10 in-context examples). The two GPT-assigned scores similarly decrease with an increasing number of in-context examples, and the value-only score (orange bar) is consistently higher than the score with accompanying explanation (blue bar).

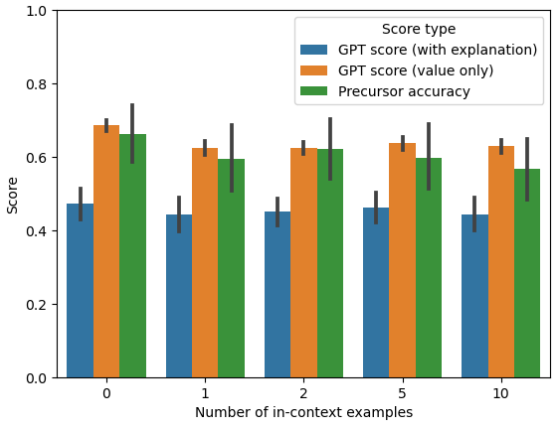


Figure 5.12: GPT-4-assigned scores (blue and orange) and precursor accuracy (green) as a function of the number of in-context examples provided. The black error bar indicates the 5th-95th percentile for the confidence interval of the mean in each bin. The value-only GPT-4 scores are computed 5 times per example and therefore display smaller confidence intervals. The same 100 test synthesis routes are used for all evaluations.

In Fig. 5.13, we report the GPT-4-generated synthesis, the script-generated synthesis, the GPT-4 assessment with explanation, and the GPT-4 score without explanation for a randomly-chosen example where the GPT-4-assigned score for the predicted synthesis plan is 4. GPT-4 lists the correct prototypes, and a reaction that is partially correct (an oxygen byproduct is missing and the reaction is not balanced). The synthesis steps are qualitatively correct, with the sintering temperature being off by 150 °C and the sintering time by 4h. These can be considered minor errors, and the overall procedure correctly matches both the one stored in the reference database, and the one indicated in the original paper [31]. The GPT-4-produced score accurately notices the differences between the proposed synthesis and the script-computed one, and assigns an appropriate value. For more examples with different GPT-4-assigned scores, see Appendix C.10.

Overall, the performance of GPT-4 in retrieving synthesis routes from literature present in the training set is mixed. The precursors are correctly indicated the majority of times, while the correct weighting of the chemical reaction is a more challenging task for the model. Nonetheless, its performance appears to be at least comparable to the one of a model specifically designed for the task of synthesis retrieval. We also note that accurate scoring of model performance is also a challenge in this scenario, as the reference dataset is sometimes inaccurate and missing information about the synthesis steps.

[19] `https://ceder.berkeley.edu/text-mined-synthesis/`

GPT-4

Prompt:

Requested product: $CuFe_1$-x$GaxO_2$ ($CuFe_1$-x$GaxO_2$)

GPT-4:

To synthesize $CuFe_1$-x$GaxO_2$, we can use the solid-state reaction method using CuO, Fe_2O_3, and Ga_2O_3 as precursor materials. Here's the proposed synthesis route:

Step 1: Prepare stoichiometric mixtures of CuO, Fe_2O_3, and Ga_2O_3 powders according to the desired x value in $CuFe_1$-x$GaxO_2$. Reasoning: CuO, Fe_2O_3, and Ga_2O_3 are the oxide precursors required to create the $CuFe_1$-x$GaxO_2$ compound.

Step 2: Thoroughly mix the precursor powders using a ball mill or mortar and pestle to ensure homogeneity. Reasoning: Proper mixing is crucial for obtaining a uniform final product and the desired properties.

Step 3: Press the mixed powders into pellets using a hydraulic press. Reasoning: Pelletizing the powder mixture helps ensure good contact between precursor materials and facilitates the solid-state reaction.

Step 4: Heat the pellets in a furnace at 900°C for 24 hours under a controlled atmosphere (e.g., inert gas like argon or nitrogen). Reasoning: Heating at this temperature will initiate the solid-state reaction between the precursors to form the desired CuO, Fe_2O_3, and Ga_2O_3 compound. The inert atmosphere prevents oxidation of the sample.

Step 5: Cool the pellets to room temperature and then grind them into a fine powder using a mortar and pestle or ball mill. Reasoning: Grinding the product ensures that the final material has a uniform particle size and composition.

Step 6: (Optional) Perform a second calcination step at 900°C for 12 hours under an inert atmosphere to ensure complete reaction and crystallization. Reasoning: This additional calcination step can help improve the phase purity and crystallinity of the synthesized CuO, Fe_2O_3, and Ga_2O_3 compound.

The balanced chemical reaction for the synthesis of CuO, Fe_2O_3, and Ga_2O_3 is:

CuO + (1-x)Fe_2O_3 + Ga_2O_3 → CuO, Fe_2O_3, and Ga_2O_3

Script-generated synthesis route:

To make $CuFe_1$-x$GaxO_2$ requires CuO, Ga_2O_3, and Fe_2O_3.

The balanced chemical reaction is 1 CuO + 0.5x Ga_2O_3 + (0.5-0.5x) Fe_2O_3 → 1 $CuFe_1$-x$GaxO_2$ + 0.25 O_2.

Here is the step-by-step synthesis procedure:

1. Compounds must be mixed
2. Compounds must be ground
3. Compounds must be pressed
4. Compounds must be sintered with heating temperature 1050.0 °C and heating time 20.0 h and atmosphere argon
5. Compounds must be grinding
6. Compounds must be sintering

The balanced chemical reaction is: 1 CuO + 0.5x Ga_2O_3 + (0.5-0.5x) Fe_2O_3 → 1 $CuFe_1$-x$GaxO_2$ + 0.25 O_2

GPT-4 score (with explanation):

The first text provides a step-by-step synthesis procedure for $CuFe_1$-x$GaxO_2$, while the second text describes a similar but slightly different procedure. Both texts mention the precursor materials CuO, Fe2O3, and Ga2O3. However, the second text does not specify the heating temperature as 1050.0 C and the heating time as 20.0 h. Instead, it suggests a lower temperature of 900°C and different heating times (24 hours and an optional 12 hours). The second text also includes reasoning behind each step, which the first text does not. Score: 4

GPT-4 score (value only, avg.):

4.0

Figure 5.13: GPT-4 prompt, reply, script-based reference, GPT-4-score with explanation, and GPT-4-score for $CuFe_1$-x$GaxO_2$ synthesis.

5.6.2 Synthesis of new materials

This section checks the capability of GPT-4 in proposing the synthesis route and conditions of proposed new materials. We use several newly synthesized materials as benchmarks, as they are not seen by the model and therefore provide out-of-training validation examples for the synthesis planning task. The synthesis for the materials we query has been recently published in the following peer-reviewed papers [80, 75, 100, 103].

GPT-4

Prompt:

Please propose the synthesis route and corresponding conditions for the following compounds. For each one, you should propose at least 2 routes and the condition can be within a range. 1. $Li_{0.388}Ta_{0.238}La_{0.475}Cl_3$ 2. $Ag_2Mo_2O_7$ 3. $Na_{2.25}Y_{0.25}Zr_{0.75}Cl_6$ 4. $LiMOCl_4$ (M=Nb, Ta)

GPT-4:

Here are two synthesis routes and corresponding conditions for each of the compounds:

1. $Li_{0.388}Ta_{0.238}La_{0.475}Cl_3$:
    - a) Solid-state reaction:
        - Mix appropriate stoichiometric amounts of LiCl, $TaCl_5$, and $LaCl_3$ powders.
        - Grind and homogenize the mixture using a mortar and pestle. Ball mill is needed
        - Place the mixture in an alumina crucible and heat it in a furnace under an inert atmosphere (argon or nitrogen) or a flowing dry HCl atmosphere.
        - Heat at 600-800°C for 10-24 hours, and then slowly cool down to room temperature. Sintering is not used in the paper but only ball milling but sintering might also work. The temperature predicted is slightly higher than it should be.
        - Grind the resulting product and characterize it to confirm phase purity.
    - b) Hydrothermal synthesis:
        - Dissolve stoichiometric amounts of LiCl, $TaCl_5$, and $LaCl_3$ in deionized water.
        - Transfer the solution to a Teflon-lined autoclave.
        - Heat the autoclave to 180-240°C and maintain the temperature for 24-72 hours.
        - After cooling down, collect the precipitate by vacuum filtration, wash it with deionized water, and dry it at 60-80°C overnight.
        - Characterize the resulting product to confirm phase purity.

....

Please note that these synthesis routes and conditions are general suggestions and may require optimization to obtain the desired compounds with high phase purity. Different starting precursors, heating rates, and reaction times may need to be tested to find the optimal synthesis conditions.

Figure 5.14: Prediction of synthesis route and conditions for solid electrolytes materials. Evaluation: The synthesis route prediction for inorganic materials is relatively accurate. The synthesis steps are often correctly predicted with the synthesis condition not far away from what is reported.

Further, we test GPT-4's capability on synthesis planning for polymeric materials, we introduce an additional example to assess its higher-level synthetic design skills, see Appendix C.8 for details. This aspect is particularly valuable in current research and industrial applications, as it involves optimizing experiment conditions for specific systems. We first ask about the synthesis conditions of a PMMA polymer with a target molecular weight of 100000 and use a 5g monomer scale followed by requesting a specific synthesis route and particular catalyst. Typically, when presented with a system, GPT-4 offers a conventional and broadly applicable protocol, which may be outdated and suboptimal. However, providing some guidance to GPT-4 can help refine its suggestions. Notably, GPT-4 demonstrates a keen chemical sense in adjusting experimental conditions for a new system.

5.7 Coding assistance

In this section, we explore the general capability of GPT-4 as an assistant to code for carrying out materials simulations, analyzing materials data, and doing visualization. This heavily relies on the knowledge of GPT-4's knowledge of existing packages. A table of the tasks we tried and the evaluation is listed in Table 14.

In Appendix C.9, we show some examples using the code generated by GPT-4 on materials properties relations.

In general, GPT-4 is capable of coding. For the tasks that require materials knowledge, it performs very well as an assistant. In most cases, a few rounds of feedback are needed to correct the error. For new packages or those not included in the training data of GPT-4, providing a user manual or the API information could work. In most difficult cases, GPT-4 can help outline the general workflow of a specific that can later be coded one by one.

Task	Evaluation
Generating LAMMPS input to run molecular dynamics simulations and get the atomic structures	GPT-4 has a clear understanding of what LAMMPS requires in terms of format and functionality. When asked to utilize a development package to generate LAMMPS data, GPT-4 didn't perform as well in grasping the intricacies of the complex code packages when asked to perform the task without relying on any packages, GPT-4 can provide a helpful workflow, outlining the necessary steps. The scripts it generates are generally correct, but certain details still need to be filled in manually or through additional instruction.
Generate initial structures of polymers using packages	GPT-4 can generate code to create initial simple polymer structures. However, it can get confused with writing code using specific polymer packages. For example, when two users try to fulfill the same task with the same questions. It gives two different codes to generate using rdkit. One of the code pieces worked.
Plotting stress vs. strain for several materials	GPT-4 can generate code to plot the stress vs. strain curve. When no data is given, but to infer from basic materials knowledge, GPT-4 can only get the elastic range correct.
Show the relationship between band gap and alloy content for several semiconductor alloys, illustrating band bowing if applicable	It can understand the request and plotted something meaningful. Some but not all constants are correct, for example, the InAs-GaAs example is reasonable. The legend does not contain sufficient information to interpret the x-axis.
Show the relationship between PBE band gap and experimental band gap	This is a knowledge test, and GPT plots correct band gaps for several well-known semiconductors.
Show an example pressure-temperature phase diagram for a material.	GPT-4 tried to plot the phase diagram of water but failed.
Show Bravais lattices	Plot errored out.
Generate DFT input scripts for the redox potential computation using NWChem software package	GPT-4 proposes the correct order of tasks to estimate redox potential via a Born-Haber thermodynamic cycle using NWChem to model the thermodynamics, without explicitly prompting for a Born-Haber thermodynamic cycle. The appropriate choice of functional alternates between being reasonable and uncertain, but GPT-4's literature citation for choosing functionals is unrelated or tenuous at best. The choice of basis set appears reasonable, but fine details with the proposed input scripts are either inappropriately written or outright fabricated, namely the choice of implicit solvation model, corresponding solvation model settings, and thermal corrections.

Table 14: Task and evaluation for coding assistance ability.

6 Partial Differential Equations

6.1 Summary

Partial Differential Equations (PDEs) constitute a significant and highly active research area within the field of mathematics, with far-reaching applications in various disciplines, such as physics, engineering, biology, and finance. PDEs are mathematical equations that describe the behavior of complex systems involving multiple variables and their partial derivatives. They play a crucial role in modeling and understanding a wide range of phenomena, from fluid dynamics and heat transfer to electromagnetic fields and population dynamics.

In this chapter, we investigate GPT-4's skills in several aspects of PDEs: comprehension of PDE fundamentals (Sec. 6.2), solving PDEs (Sec. 6.3), and assisting AI for PDE Research (Sec. 6.4). We evaluate the model on diverse forms of PDEs, such as linear equations, nonlinear equations, and stochastic PDEs (Fig. 6.5). Our observations reveal several capabilities, suggesting that GPT-4 is able to assist researchers in multiple ways:[20]

- PDE Concepts: GPT-4 demonstrates its awareness of fundamental PDE concepts, thereby enabling researchers to gain a deeper understanding of the PDEs they are working with. It can serve as a helpful resource for teaching or mentoring students, enabling them to better understand and appreciate the importance of PDEs in their academic pursuits and research endeavors (Fig. 6.1- 6.4).
- Concept Relationships: The model is capable of discerning relationships between concepts, which may aid mathematicians in broadening their perspectives and intuitively grasping connections across different subfields.
- Solution Recommendations: GPT-4 can recommend appropriate analytical and numerical methods for addressing various types and complexities of PDEs. Depending on the specific problem, the model can suggest suitable techniques for obtaining either exact (Fig. 6.8) or approximate solutions (Fig. 6.13-6.14).
- Code Generation: The model is capable of generating code in different programming languages, such as MATLAB and Python, for numerical solution of PDEs (Fig. 6.14), thus facilitating the implementation of computational solutions.
- Research Directions: GPT-4 can propose further research directions or potential extensions (Fig. 6.17), suggesting new problems, generalizations, or improvements that could lead to more significant and impactful results in the PDE domain.

Despite GPT-4's potential to assist PDE research, we also observe some limitations. To better leverage GPT-4, we offer the following recommendations for researchers:

- Output Verification: While GPT-4 exhibits human-like capabilities in solving partial differential equations and providing explicit solutions, there might be instances of incorrect derivation (Fig. 6.9). Researchers should exercise caution and verify the model's output when using GPT-4 to solve PDEs.
- Hallucinations Awareness: GPT-4 may occasionally erroneously cite non-existent references (e.g., Fig. 6.4 and 6.18). Researchers should cross-check citations and be aware of this limitation to ensure the accuracy and reliability of the information provided by the model.

6.2 Knowing basic concepts about PDEs

We first examine whether GPT-4 can act as a PDE expert, introducing basic concepts to audiences who may not be familiar with this area. Although it might not yet possess an extensive understanding of PDEs, gaining a rudimentary familiarity with them and their applications can benefit researchers looking to integrate PDEs into their work. By explaining the essential principles and applications of PDEs, valuable insights can be provided to both experts and non-experts.

The first question is about the definition and the form of the PDEs and GPT-4 provides a pretty good explanation of PDEs as shown in Fig. 6.1. Under the prompt of the user, GPT-4 gives a clear concept of the PDEs and the classes as linear or nonlinear and Elliptic, Parabolic, or Hyperbolic. Those who are new in this area will benefit from these concepts and classifications.

[20] In this chapter, we use yellow to indicate incorrect or inaccurate responses from GPT-4.

GPT-4

Prompt:

Please introduce partial differential equations.

GPT-4:

Partial Differential Equations (PDEs) are a type of mathematical equation that involve multiple independent variables and their partial derivatives. They are an extension of ordinary differential equations (ODEs), which involve only one independent variable and its derivatives. PDEs are widely used in various fields of science and engineering, such as physics, chemistry, biology, and economics, to model complex phenomena and processes.

A PDE can be represented in the general form:

$$F(x_1, x_2, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_2^2}, \ldots, \frac{\partial^2 u}{\partial x_n^2}, \ldots) = 0$$

where F is a function of the independent variables $x_1, x_2, \ldots, x_n$, the dependent variable u, and their partial derivatives up to some order. The order of a PDE is determined by the highest order of the partial derivatives involved in the equation.

There are several classifications of PDEs, such as:

1. Linear vs. Nonlinear PDEs: A PDE is linear if the dependent variable u and its partial derivatives appear linearly, without any powers or products involving u or its derivatives. Otherwise, the PDE is considered nonlinear.

2. Homogeneous vs. Inhomogeneous PDEs: A PDE is homogeneous if the function F is linear and the terms containing only independent variables are absent. Otherwise, it is inhomogeneous.

3. Parabolic, Hyperbolic, and Elliptic PDEs: These classifications are based on the characteristics of the PDE and the behavior of its solutions. Parabolic PDEs model diffusion processes, hyperbolic PDEs model wave propagation, and elliptic PDEs model steady-state processes.

Examples of well-known PDEs include the heat equation, the wave equation, and Laplace's equation. Solving PDEs often requires specialized techniques, such as separation of variables, Fourier series, or numerical methods like finite element or finite difference methods.

Figure 6.1: The introduction to the basic concepts of PDEs.

After examining the essential definitions and classifications of general PDEs, a natural subsequent inquiry could be: "What are the applications of PDEs?" The response is illustrated in Fig. 6.2, where GPT-4 presents a general overview of PDEs' usage across multiple fields.

Additionally, one might be interested in learning the specific form of PDEs employed in the fields of Meteorology and Oceanography, as well as the methods used to solve these problems. In Fig. 6.3, we inquire GPT-4 about the detailed form of a classical equation in Meteorology and Oceanography and the prevalent numerical methods in this field. GPT-4 offers a good explanation, and with these fundamental concepts, one can become more acquainted with this area.

To further explore PDEs and their applications in Meteorology and Oceanography, we request GPT-4 to recommend learning materials. GPT-4 provides a list of books, papers, and website links for comprehensive learning, as seen in Fig. 6.4. While these resources are abundant and useful, it should be noted that some may not exist and could be the result of hallucinations.

GPT-4

Prompt:

What are the applications of PDEs

GPT-4:

Partial Differential Equations (PDEs) are used in various fields of science, engineering, and mathematics to model and analyze a wide range of phenomena. Some applications of PDEs include:

1. Physics: PDEs are used to describe various physical phenomena such as heat conduction, fluid flow, electromagnetic fields, and wave propagation. Examples include the heat equation, the wave equation, and Maxwell's equations.
2. Engineering: In engineering, PDEs are used to model and simulate various processes such as heat transfer in a solid, fluid dynamics in a pipe, or stress distribution in a solid. Engineers use PDEs to design and optimize structures, materials, and processes.
3. Mathematical Biology: PDEs are used to model biological processes such as population dynamics, the spread of diseases, or the movement of cells. For example, the reaction-diffusion equation is used to describe the spread of chemicals or the growth of a tumor.
4. Finance: In the field of finance, PDEs are used to model and analyze the behavior of financial instruments such as options, stocks, and bonds. The Black-Scholes equation, a PDE that describes the price of an option as a function of time and the underlying asset's price, is a well-known example.
5. Image and Signal Processing: PDEs are used to model and analyze image and signal data, such as smoothing, denoising, and edge detection. For example, the Perona-Malik equation is a PDE used for image denoising.
6. Meteorology and Oceanography: PDEs are used to model the Earth's atmosphere and ocean systems, including weather and climate prediction, ocean currents, and the interaction between the atmosphere and the ocean.
7. Geophysics: In geophysics, PDEs are used to model various processes such as the flow of magma in the Earth's mantle, the propagation of seismic waves, and the deformation of the Earth's crust due to tectonic forces.
8. Quantum Mechanics: PDEs are used to describe the behavior of quantum systems, such as the Schrödinger equation, which describes the behavior of particles at the quantum level.
9. General Relativity: PDEs are used in the mathematical formulation of Einstein's theory of general relativity, which describes the curvature of spacetime due to the presence of mass and energy.
10. Optimization and Control Theory: PDEs are used to formulate and solve optimization and control problems, such as finding the optimal shape of a structure or the optimal control strategy for a process.

These are just a few examples of the many applications of PDEs in various fields. The versatility and importance of PDEs in modeling and analyzing a wide range of phenomena make them a fundamental tool in many disciplines.

Prompt:

Can you give more details about the PDEs in Meteorology and Oceanography?

GPT-4:

Sure! Partial Differential Equations (PDEs) play a significant role in the fields of meteorology and oceanography, as they help model and simulate the complex interactions between the Earth's atmosphere, oceans, and land. Some important PDEs used in meteorology and oceanography include:

1. Navier-Stokes Equations: These equations describe the motion of fluid substances, such as air and water, and are fundamental to understanding the dynamics of both the atmosphere and the oceans. They consider the effects of pressure, viscosity, and external forces, which helps meteorologists and oceanographers study phenomena like wind patterns, ocean currents, and turbulence.
2. Continuity Equation: The continuity equation is a conservation law that ensures the mass of a fluid remains constant within a defined region. In meteorology and oceanography, this equation is used to model the conservation of mass for air and water, respectively, allowing the study of fluid flow and transport processes in the atmosphere and oceans.
3. Thermodynamic Energy Equation: This equation models the conservation of energy in a fluid system, accounting for processes such as heating, cooling, and the exchange of energy between different components (e.g., between the atmosphere and the Earth's surface). In meteorology, this helps to study temperature distributions, heat transport, and the development of weather systems.
4. Equation of State: The equation of state relates the pressure, temperature, and density of a fluid, such as air or seawater. In meteorology and oceanography, this equation helps to describe the properties of the atmosphere and the oceans, which are crucial for understanding their behavior and interactions.
5. Advection-Diffusion Equation: This equation describes the transport of a scalar quantity (such as temperature, salinity, or pollutants) by a fluid flow, considering both advection (transport by the fluid motion) and diffusion (spreading due to molecular processes). In meteorology and oceanography, this equation is used to study the distribution of heat, salinity, and other substances in the atmosphere and the oceans.
6. Vorticity Equation: The vorticity equation is derived from the Navier-Stokes equations and describes the evolution of vorticity (the curl of the velocity field) in a fluid flow. In meteorology, vorticity is an important quantity for understanding the development and evolution of weather systems, such as cyclones and anticyclones.

These PDEs, along with other equations and numerical methods, are used to build complex numerical models that simulate the Earth's atmosphere and oceans. These models are crucial for weather forecasting, climate prediction, and understanding the impacts of human activities on the environment. They also help researchers investigate various meteorological and oceanographic phenomena, such as El Niño and La Niña, the formation of hurricanes, and the dynamics of ocean currents.

Figure 6.2: The introduction on various applications of PDEs and the detailed explanation of the PDEs in Meteorology and Oceanography.

GPT-4

Prompt:

Could you please give more details on the Continuity Equation (in latex form)?

GPT-4:

Sure! Here are the Continuity Equations in LaTeX form for both incompressible and compressible flows:
1. Incompressible Flow (Constant Density):
For an incompressible fluid, where the density ρ remains constant, the Continuity Equation is given by:

$$\nabla \cdot \mathbf{u} = 0$$

2. Compressible Flow (Variable Density):
For compressible fluids, where the density ρ can vary with pressure and temperature, the Continuity Equation is given by:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$$

These LaTeX expressions can be used to render the Continuity Equations in their mathematical notation when using a LaTeX-compatible editor or typesetting system. The Continuity Equation is an essential principle in fluid dynamics that describes the conservation of mass for a fluid in motion, and it plays a crucial role in various applications in meteorology, oceanography, engineering, and physics.

Prompt:

What are the widely used methods for solving these PDEs in Meteorology and Oceanography (in latex form)?

GPT-4:

There are several widely used numerical methods for solving Partial Differential Equations (PDEs) in meteorology and oceanography. These methods are designed to handle the complex, nonlinear, and coupled PDEs that describe atmospheric and oceanic processes. Here are some of the most common methods, presented in LaTeX form:
1. Finite Difference Method (FDM):
The Finite Difference Method is a technique that approximates the derivatives in the PDEs with discrete differences between grid points. It is widely used in meteorology and oceanography to solve PDEs on structured grids. The first-order forward difference approximation for the time derivative is:

$$\frac{\partial \phi}{\partial t} \approx \frac{\phi^{n+1} - \phi^n}{\Delta t}$$

2. Finite Volume Method (FVM):
The Finite Volume Method is a conservative approach that integrates the PDEs over discrete control volumes, ensuring the conservation of mass, momentum, and energy. It is well-suited for solving PDEs on irregular and unstructured grids. The general form of the FVM is:

$$\int_{V_i} \frac{\partial \phi}{\partial t} dV + \oint_{\partial V_i} \mathbf{F} \cdot d\mathbf{S} = 0$$

where V_i is the control volume, $\mathbf{F}$ is the flux vector, and ∂V_i is the control volume boundary.
3. Finite Element Method (FEM):
The Finite Element Method is a technique that approximates the solution of PDEs using piecewise functions, defined over a mesh of elements. It is particularly useful for solving PDEs with complex geometries, boundary conditions, and material properties. The weak form of the FEM is:

$$\int_\Omega \nabla \phi \cdot \nabla \psi d\Omega - \int_\Omega f \psi d\Omega = 0$$

where Ω is the domain, ϕ is the approximate solution, ψ is the test function, and f is the source term.
4. Spectral Method:
The Spectral Method is a technique that represents the solution of PDEs as a sum of basis functions, such as Fourier series or Chebyshev polynomials. It is highly accurate and efficient for solving PDEs with smooth solutions and periodic boundary conditions. The spectral representation of a function ϕ is:

$$\phi(\mathbf{x}) = \sum_{k=1}^{N} \hat{\phi}_k \psi_k(\mathbf{x})$$

where $\hat{\phi}_k$ are the spectral coefficients, and $\psi_k(\mathbf{x})$ are the basis functions.

Figure 6.3: Questions about continuity equation in Meteorology and Oceanography and the wildely used numerical methods for solving these PDEs.

GPT-4

Prompt:
Could you please suggest some materials on the PDEs?
GPT-4:
Here is a list of books, online courses, and other resources that provide a comprehensive introduction to Partial Differential Equations (PDEs) and the numerical methods used to solve them:
Books:

1. *Partial Differential Equations: An Introduction* by Walter A. Strauss (ISBN: 978-0470054567)
2. *Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems* by Richard Haberman (ISBN: 978-0321797056)
3. *Numerical Solution of Partial Differential Equations by the Finite Element Method* by Claes Johnson (ISBN: 978-0486469003)
4. *A First Course in the Numerical Analysis of Differential Equations* by Arieh Iserles (ISBN: 978-0521734905)

Online Courses:

1. *Partial Differential Equations* by MIT OpenCourseWare: `https://ocw.mit.edu/courses/mathematics/18-152-introduction-to-partial-differential-equations-fall-2011/index.htm`
2. *Numerical Methods for Partial Differential Equations* by the University of Washington: `https://www.coursera.org/learn/partial-differential-equations`

Other Resources:

1. Paul's Online Math Notes - Partial Differential Equations: `https://tutorial.math.lamar.edu/Classes/DE/PDEIntro.aspx` (Correction: the right link should be `https://tutorial.math.lamar.edu/classes/de/intropde.aspx`)
2. Wolfram MathWorld - Partial Differential Equations: `https://mathworld.wolfram.com/PartialDifferentialEquation.html`

Prompt:
Could you please suggest some materials on the PDEs in meteorology and oceanography?
GPT-4:
Here is a list of books, articles, and online resources that provide a comprehensive introduction to Partial Differential Equations (PDEs) in meteorology and oceanography, as well as the numerical methods used to solve them:
Books:

1. *Atmospheric and Oceanic Fluid Dynamics* by Geoffrey K. Vallis (ISBN: 978-1108418129)
2. *Numerical Methods in Atmospheric and Oceanic Modelling: The Finite-Volume Method* by Robert Sadourny, Alain Arakawa, and Yves Mintz (ISBN: 978-3642082555)
3. *An Introduction to Dynamic Meteorology* by James R. Holton and Gregory J. Hakim (ISBN: 978-0123848666)
4. *Ocean Circulation and Climate: A 21st Century Perspective* by Gerold Siedler, Stephen M. Griffies, John Gould, and John A. Church (ISBN: 978-0123918512)

Articles:

1. *A Finite-Volume, Incompressible Navier Stokes Model for Studies of the Ocean on Parallel Computers* by John Marshall, Alistair Adcroft, Chris Hill, Lev Perelman, and Curt Heisey, Journal of Geophysical Research: Oceans, Volume 102, Issue C3, pages 5753-5766, 1997
2. *The Finite-Element Method for the Primitive Equations of the Ocean* by Dale B. Haidvogel and Peter R. Gent, Monthly Weather Review, Volume 106, Issue 2, pages 309-319, 1978

Online Resources:

1. The Weather Research and Forecasting Model (WRF) - User's Guide and Tutorials: `http://www2.mmm.ucar.edu/wrf/users/docs/user_guide_V3/contents.html`
2. MIT General Circulation Model (MITgcm) - User's Manual and Documentation: `http://mitgcm.org/public/r2_manual/latest/online_documents/manual.html`

Figure 6.4: GPT-4 is capable of offering recommendations for in-depth learning materials upon user requests. The quality of these materials is generally satisfactory; however, it is important to note that the references highlighted in yellow do not exist.

Now we will assess whether GPT-4 can comprehend the relationships between various concepts and present three cases.

First, we request GPT-4 to elucidate the most fundamental concept (Fig. 6.5): the distinct definitions of the solution of a stochastic PDE (SPDE). GPT-4 can precisely depict the relationship between two concepts and offer succinct explanations of both. It describes the inclusive relationship between mild solutions and weak solutions in general scenarios. Moreover, it notes that under specific conditions, these two concepts are, in fact, equivalent.

Next, we pose a question to GPT-4 concerning the intricacies of the SPDE (Fig. 6.6): There are various types of noise terms in SPDEs. The two most frequently employed ones are trace-class noise and space-time white noise. These types exhibit considerable differences in modeling practical problems and the application of mathematical tools. GPT-4 successfully comprehends the issue, elaborates on these two concepts, and delineates their relationship. It expounds on their similarities and differences from both definitional and intuitive standpoints.

Last, to evaluate GPT-4's grasp of more profound professional concepts, we use the stochastic Navier-Stokes equation as an example and inquire whether GPT-4 understands the typical interpretation of its final space-time white noise and its association with more advanced mathematical tools like the Wick product (Fig. 6.7). GPT-4 can apprehend the intent of the question and provide a good response. The answer indicates that GPT-4 is aware of the challenges posed by space-time white noise in the context of SPDEs and how the Wick product can address these challenges.

GPT-4

Prompt:

What is the relation between mild solution and weak solution when considering the stochastic PDE?

GPT-4:

The relation between a weak solution and a mild solution to a linear stochastic partial differential equation (SPDE) depends on the specific properties of the equation, such as the regularity of the coefficients, initial conditions, and the structure of the operator L. However, in general, mild solutions and weak solutions are connected by the following observations:

A mild solution is usually considered as a generalization of a weak solution. In the context of SPDEs, a weak solution is often based on the integral form of the equation and Ito's stochastic calculus, while a mild solution takes the semigroup approach, which is more general and can handle certain cases where weak solutions might not exist or fail to be unique.

If a mild solution exists and satisfies certain regularity conditions (e.g., the linear operator L generates a strongly continuous semigroup, and the noise term is regular enough), then the mild solution is also a weak solution to the SPDE. This means that a mild solution can be seen as a natural extension of the weak solution concept, encompassing a broader class of problems. In some cases, the existence and uniqueness of a mild solution imply the existence and uniqueness of a weak solution, and vice versa. However, this depends on the specific properties of the problem, such as the regularity of the coefficients, initial conditions, and the structure of the operator L.

In summary, mild solutions and weak solutions are related concepts in the study of stochastic partial differential equations, with mild solutions being a more general concept that can handle a broader class of problems. The existence and uniqueness of one type of solution can sometimes imply the existence and uniqueness of the other type, but this depends on the specific properties of the problem.

Figure 6.5: GPT-4 understands the relationship between the mild solution and the weak solution in the context of stochastic PDEs.

GPT-4

Prompt:

For stochastic PDE, what is the difference between trace-class noise and space-time white noise? What is the relation between them?

GPT-4:

In the context of stochastic partial differential equations (SPDEs), trace-class noise and space-time white noise are two different types of noise that can be used to model random disturbances in a system. They differ in their statistical properties and how they influence the behavior of the SPDEs. Here are the main differences and the relationship between them:

1. **Trace-class noise:**
   Trace-class noise, also known as trace-class operator-valued noise or nuclear noise, refers to a class of noise processes that have a well-defined trace (or sum of the diagonal elements) in a suitable operator basis. This noise is often characterized by its covariance operator, which is a trace-class operator. Trace-class noise is typically smoother and more regular than white noise, and it is well-suited for modeling smooth random disturbances in a system.

2. **Space-time white noise:**
   Space-time white noise, also known as spatiotemporal white noise, is a type of noise process that is uncorrelated in both time and space. In other words, the covariance function of space-time white noise has a delta function structure, which means that the noise at different points in space and time is statistically independent. This type of noise is often used to model highly irregular or unpredictable disturbances in a system, and it can lead to more complex and irregular behavior in the SPDEs.

3. **Relationship between trace-class noise and space-time white noise:**
   Trace-class noise and space-time white noise can be considered as two ends of a spectrum of noise processes that can be used in SPDEs. Trace-class noise is smoother and more regular, while space-time white noise is highly irregular and uncorrelated in both space and time.

   In some cases, trace-class noise can be obtained as a smoothed or regularized version of space-time white noise by applying a suitable smoothing operator. Conversely, space-time white noise can be seen as the limit of a sequence of trace-class noise processes when the smoothing operator becomes more and more localized.

   In summary, trace-class noise and space-time white noise are two different types of noise processes used in stochastic partial differential equations, with trace-class noise being smoother and more regular, while space-time white noise is highly irregular and uncorrelated in both space and time. They are related in the sense that they can be seen as part of a spectrum of noise processes, with trace-class noise being a smoothed or regularized version of space-time white noise, and space-time white noise being the limit of increasingly localized trace-class noise processes.

Figure 6.6: GPT-4 understands the relationship between trace-class noise and space-time white noise.

GPT-4

Prompt:
Consider stochastic Navier–Stokes equations on torus $\mathbb{T}^d$:

$$du + \mathrm{div}(u \otimes u)dt + \nabla p dt = \Delta u dt + \xi, \mathrm{div} u = 0, u(0) = u_0,$$

here p is the associated pressure, ξ is the space-time white noise on some probability space.

When we talk about space-time white noise, how about its nonlinear term? can you say something about its nonlinear term and wick product?

GPT-4:
In the context of stochastic Navier-Stokes equations on the torus, space-time white noise represents a random forcing term that is uncorrelated in both space and time. The nonlinear term in the equation refers to the advection part, specifically the term $\mathrm{div}(u \otimes u)$, which represents the convective transport of momentum in the fluid.
The Wick product is a technique used in the analysis of stochastic partial differential equations (SPDEs) when dealing with nonlinearities and noise. The Wick product, denoted by a colon :, is a renormalized product of random variables that helps to handle the singularities arising from the interaction between the noise (in this case, space-time white noise) and the nonlinear term.
In the context of stochastic Navier-Stokes equations, the Wick product is used to provide a mathematically rigorous framework to define and analyze the equation. The Wick product allows us to handle the products of the noise with the nonlinear term in a way that makes sense, and can be useful in proving the existence and uniqueness of solutions, as well as other properties of the stochastic Navier-Stokes equations.
A basic example of the Wick product is the second-order Wick product of two random variables X and Y, which is defined as:

$$:XY: = XY - \mathbb{E}[X]\mathbb{E}[Y] - \mathbb{E}[X]Y - X\mathbb{E}[Y].$$

In the analysis of SPDEs, higher-order Wick products are also used, and their construction is more involved. The main idea, however, is to subtract off the singular terms that arise from the interaction of the noise with the nonlinear term, so that the resulting product is well-defined.

Figure 6.7: GPT-4 understands the relation between space-time white noise and Wick product.

6.3 Solving PDEs

In the previous section, GPT-4 exhibited comprehension of PDE concepts and showcased promising capabilities to aid with instruction in PDE courses. In this section, we aim to leverage GPT-4 as a mathematical tool to solve PDEs, specifically, finding their solutions. PDEs are used extensively to model diverse scientific and engineering problems. Obtaining solutions for these PDEs is therefore a vital part of tackling those problems. However, deriving solutions for PDEs is a complex undertaking.

The analytical solution is highly desirable due to its definitive accuracy and ability to reveal the intrinsic structures of PDEs. However, in most cases, analytical solutions are difficult to obtain and may not even exist. Consequently, discovering numerical solutions, i.e., approximate digital solutions, has attracted significant attention from the beginning, leading to the development of numerous numerical methods. The advances in computational capabilities of computers have facilitated solving highly complex PDEs in reality, with corresponding algorithms and codes fed into the system.

Here we will first assess GPT-4's capabilities in finding analytical solutions for some important PDEs, and then focus on examining GPT-4's abilities in numerically solving PDEs as an assistant.

6.3.1 Analytical solutions

We adopt Example 1 from Section 4.1.1 of Evans' book [25] and ask GPT-4 to find its analytical solution, as illustrated in Fig. 6.8.

GPT-4

Prompt:
Please solve the following problem: Let $U \subset \mathbb{R}^n$ be a bounded, open set with a smooth boundary. We consider the initial/boundary-value problem for the heat equation

$$\begin{cases} u_t - \Delta u = 0 \text{ in } U \times (0, \infty) \\ u = 0 \text{ on } \partial U \times [0, \infty) \\ u = g \text{ on } U \times \{t = 0\}, \end{cases}$$

where $g : U \to \mathbb{R}$ is given.

GPT-4:
To solve this problem, we can use the method of separation of variables. We assume that the solution can be written in the form $u(x,t) = X(x)T(t)$, where $X(x)$ is a function of the spatial variable x only and $T(t)$ is a function of the time variable t only.
Substituting this form of the solution into the heat equation, we get

$$X(x)T'(t) - \Delta X(x)T(t) = 0.$$

Dividing both sides by $X(x)T(t)$, we get

$$\frac{T'(t)}{T(t)} = \frac{\Delta X(x)}{X(x)} = -\lambda,$$

where λ is a separation constant.
Now we have two ordinary differential equations (ODEs) to solve: The time ODE: $T'(t) = -\lambda T(t)$. The general solution to this ODE is given by $T(t) = c_1 e^{-\lambda t}$, where c_1 is an arbitrary constant. The spatial ODE: $\Delta X(x) = -\lambda X(x)$, subject to the boundary conditions $X(x) = 0$ on ∂U. This is an eigenvalue problem for the Laplace operator. The solutions are the eigenfunctions $X_k(x)$ with corresponding eigenvalues λ_k.
Now, we can write the solution as a linear combination of the eigenfunctions:

$$u(x,t) = \sum_k c_k X_k(x) e^{-\lambda_k t}.$$

To determine the coefficients c_k, we use the initial condition $u(x,0) = g(x)$. Plugging in $t = 0$, we get

$$g(x) = \sum_k c_k X_k(x).$$

To find the coefficients c_k, we can use the orthogonality property of the eigenfunctions. Multiplying both sides by $X_j(x)$ and integrating over the domain U, we get

$$\int_U g(x) X_j(x) dx = \sum_k c_k \int_U X_k(x) X_j(x) dx.$$

Using the orthogonality property, $\int_U X_k(x) X_j(x) dx = 0$ for $k \neq j$, we get

$$c_j = \frac{\int_U g(x) X_j(x) dx}{\int_U X_j^2(x) dx}.$$

Substituting this expression for c_j into the solution, we get the final form of the solution:

$$u(x,t) = \sum_k \left(\frac{\int_U g(x) X_k(x) dx}{\int_U X_k^2(x) dx} \right) X_k(x) e^{-\lambda_k t}.$$

This is the solution to the given initial/boundary-value problem for the heat equation.

Figure 6.8: GPT-4 effectively uncovers the analytical solution of a linear PDE example problem, as presented in Evans' book [25].

The obtained result is accurate, and the derivation steps leading to the solution differ slightly from those presented in Example 1 of the book, indicating that the solution was not simply copied from the original source.

To increase the complexity without making it overly difficult, we then ask GPT-4 to solve a non-

homogeneous PDE without boundary or initial conditions, given by the following form:

$$u_t + u_{xx} = 2 \tag{8}$$

GPT-4

Prompt:

Help me solve the PDE $u_t + u_{xx} = 2$

GPT-4:

To solve the partial differential equation (PDE) $u_t + u_{xx} = 2$, we can use the method of separation of variables. Assume that the solution can be written in the form $u(x,t) = X(x)T(t)$. Then the PDE can be written as:

$$X(x)T'(t) + X''(x)T(t) = 2.$$

Now divide both sides by $X(x)T(t)$ to separate the variables:

$$\frac{T'(t)}{T(t)} + \frac{X''(x)}{X(x)} = \frac{2}{X(x)T(t)}. \tag{9}$$

Since the left side is a function of t only, and the right side is a function of x only, both sides must be equal to a constant. Let this constant be $-k^2$, where k is a real constant. Then we have two ordinary differential equations (ODEs):

$$\frac{T'(t)}{T(t)} = -k^2, \quad \frac{X''(x)}{X(x)} = -\frac{2}{X(x)T(t)} + k^2. \tag{10}$$

Solve the first ODE for $T(t)$:

$$T'(t) = -k^2 T(t) \implies T(t) = Ce^{-k^2 t},$$

where C is a constant. Now solve the second ODE for $X(x)$:

$$X''(x) = (-2 + k^2)X(x). \tag{11}$$

The solution to this ODE depends on the value of the constant k^2.
If $k^2 > 2$, then we have two real, distinct roots r_1 and r_2 for the characteristic equation. The solution is:

$$X(x) = A_1 e^{r_1 x} + A_2 e^{r_2 x},$$

where A_1 and A_2 are constants. If $k^2 = 2$, then we have a repeated real root r for the characteristic equation. The solution is:

$$X(x) = (A_1 + A_2 x)e^{rx},$$

where A_1 and A_2 are constants. If $k^2 < 2$, then we have two complex conjugate roots $\alpha \pm \beta i$ for the characteristic equation. The solution is:

$$X(x) = e^{\alpha x}(A_1 \cos(\beta x) + A_2 \sin(\beta x)),$$

where A_1 and A_2 are constants.
Now, we can write the general solution for the PDE $u(x,t)$ as:

$$u(x,t) = X(x)T(t) = Ce{-}k^2 t \cdot X(x),$$

where $X(x)$ depends on the value of $k2$ as described above. To determine the constants A_1, A_2, and C as well as the value of $k2$, we need initial and/or boundary conditions.

Figure 6.9: To solve the non-homogeneous equation $u_t + u_{xx} = 2$, GPT-4 employs the method of separation of variables, which is not the appropriate approach for directly solving this non-homogeneous equation.

GPT-4

Prompt:
The PDE is not homogeneous

GPT-4:
My apologies, you are right. The PDE is non-homogeneous. In this case, let's try solving it using an integrating factor.
Given PDE:

$$u_t + u_{xx} = 2$$

First, let's find the steady-state solution, $u_s(x)$, by setting $u_t = 0$:

$$u_{xx} = 2$$

Integrating twice with respect to x:

$$u_s'(x) = 2x + C_1$$
$$u_s(x) = x^2 + C_1 x + C_2$$

Now, let's look for a particular solution of the form $u(x,t) = u_s(x) + v(x,t)$. Substituting this into the PDE:

$$(u_s(x) + v(x,t))_t + (u_s(x) + v(x,t))_{xx} = 2$$

Since $u_s(x)_t = 0$ and $u_s(x)_{xx} = 2$, this simplifies to:

$$v_t + v_{xx} = 0$$

Now, we have a homogeneous PDE for $v(x,t)$. We can use the method of separation of variables:

$$v(x,t) = X(x)T(t)$$

Differentiating v:

$$v_t = X(x)T'(t)$$
$$v_{xx} = X''(x)T(t)$$

Substitute these into the PDE for v:

$$X(x)T'(t) + X''(x)T(t) = 0$$

Separate the variables:

$$\frac{T'(t)}{T(t)} = -\frac{X''(x)}{X(x)} = -k^2$$

Now, we can solve these two ODEs separately.
For the first ODE:

$$T'(t) = -k^2 T(t)$$

This is a first-order linear ODE, and its general solution is:

$$T(t) = C_3 e^{-k^2 t}$$

For the second ODE:

$$X''(x) = -k^2 X(x)$$

This is a second-order linear ODE, and its general solution is:

$$X(x) = C_4 \cos(kx) + C_5 \sin(kx)$$

Now, we can write the solution for $v(x,t)$:

$$v(x,t) = (C_4 \cos(kx) + C_5 \sin(kx))C_3 e^{-k^2 t}$$

Finally, we can express the solution for the original PDE, $u(x,t)$:

$$u(x,t) = x^2 + C_1 x + C_2 + (C_4 \cos(kx) + C_5 \sin(kx))C_3 e^{-k^2 t}$$

The values of C_1, C_2, C_3, C_4, and C_5 depend on the boundary and initial conditions, which have not been specified.

Figure 6.10: GPT-4 resolves the non-homogeneous equation according to the user's feedback.

In the first attempt at solving the non-homogeneous equation depicted in Fig. 6.9, GPT-4 produces incorrect results, as it fails to account for the non-homogeneous form, where the right-hand term is not zero. The source of this issue can be traced to the first yellow colored sentence in Fig. 6.9: "To solve the partial differential equation (PDE) $u_t + u_{xx} = 2$, we can use the method of separation of variables".

Using variable separation, GPT-4 arrives at result (9). From this, GPT-4 incorrectly concludes, "Since the left side is a function of t only, and the right side is a function of x only, both sides must be equal to a constant," as the second yellow colored sentence. While this key sentence often appears in solving PDEs using the method of separation of variables, Equation (9) does not support this assertion, as the equation is not satisfied; both the left and right sides are coupled with t and x. It is worth noting that, since the variables are coupled, introducing a constant and decoupling Equation (10) is incorrect. GPT-4 soon encounters another challenge in solving a non-decoupled ODE from Equation (10). GPT-4 disregards the term $X(x)T(t)$ on the right side of Equation (10) and solves ODE (11), whereas the correct form of Equation (11) should have been:

$$X''(x) = (-2/T(t) + k^2)X(x), \tag{12}$$

where $T(t)$ has been abandoned accordingly by GPT-4 in Equation (11).

In this first attempt, GPT-4 fails to identify the non-homogeneous case and applies the method directly, rendering the problem unsolvable. However, GPT-4's subsequent derivation is not entirely determined by the previous step. For instance, it cannot be deduced that "Since the left side is a function of t only, and the right side is a function of x only, both sides must be equal to a constant" from Equation (9), as it is coupled. Additionally, GPT-4 omits some terms without justification in Equation (11), thereby creating a solvable equation. These steps reveal the long-term memory of the entire method but demonstrate less attention to context derivation.

In the second attempt shown in Fig. 6.10, the user provides review information, and GPT-4 corrects its mistake before continuing with the correct answer. The accurate solution relies on the boundary/initial value condition, which GPT-4 also highlights.

To increase the difficulty, we consider the nonlinear equation, specifically the porous media equation 4.1.1. Example 2 from Evans' book [25], and test GPT-4's capabilities in Fig. 6.11. The porous media equation is given by

$$u_t - \Delta\left(u^\gamma\right) = 0 \quad \text{in } \mathbb{R}^n \times (0, \infty), \tag{13}$$

where $u \geq 0$ and $\gamma > 1$ is a constant. Since the constant $\gamma > 1$, Equation (13) is a nonlinear equation, making it challenging to find an analytical solution. The analytical solution of the porous equation (13), as provided in Evans' book, is given by

$$u = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}} |x|^\alpha, \tag{14}$$

where $\alpha = \frac{2}{\gamma - 1}$ and $\mu = \alpha\gamma(\alpha\gamma + n - 2) > 0$ for any $\lambda > 0$.

In this test, GPT-4 fails to obtain the correct analytical solution after several attempts. However, it kindly reminds us that "finding an analytical solution to this nonlinear PDE is a challenging task, and it may not be possible to obtain a closed-form solution. Numerical methods are often employed to find approximate solutions for such problems." Consequently, in this case, GPT-4 demonstrates a commendable ability to assist in analytically solving PDEs, but we need to exercise caution and rely on expertise, as mistakes can be made easily, especially in complex problems.

GPT-4

Prompt:

Please solve the following PDE:

$$u_t - \Delta\left(u^\gamma\right) = 0 \quad \text{in } \mathbb{R}^n \times (0, \infty),$$

where $u \geq 0$ and $\gamma > 1$ is a constant.

GPT-4:

To solve the given PDE, we will use the self-similar solution method. This method involves finding a solution in the form of a function that depends only on the similarity variable $\xi = \frac{x}{t^\alpha}$, where α is a constant to be determined.
Let's consider a solution of the form:

$$u(x,t) = t^{-\beta} F\left(\frac{x}{t^\alpha}\right),$$

where β is a constant and F is a function of the similarity variable $\xi = \frac{x}{t^\alpha}$. We will find the values of α and β and the function F that satisfy the given PDE.
First, let's find the derivatives of u with respect to t and Δ:

$$u_t = -\beta t^{-\beta-1} F(\xi) + t^{-\beta-\alpha} F'(\xi)(-\alpha\xi),$$

$$\Delta u = t^{-2\alpha} \Delta F(\xi).$$

Now, substitute these expressions into the given PDE and divide by u^γ:

$$\frac{-\beta t^{-\beta-1} F(\xi) + t^{-\beta-\alpha} F'(\xi)(-\alpha\xi)}{t^{-\gamma\beta} F^\gamma(\xi)} - \frac{t^{-2\alpha}\Delta F(\xi)}{t^{-\gamma\beta} F^\gamma(\xi)} = 0.$$

Simplifying the expression, we get:

$$-\beta\xi^{\gamma-1} F^{\gamma-1}(\xi) + \alpha\xi^\gamma F^{\gamma-2}(\xi) F'(\xi) - \xi^{2(\gamma-1)} F^{\gamma-1}(\xi)\Delta F(\xi) = 0.$$

In order to eliminate the t-dependence, we must have the following relations between α, β, and γ: (Correction: the relation given by GPT-4 is not right)

$$\beta + 1 = \gamma\beta,$$

$$\beta + \alpha = 2\alpha\gamma.$$

Solving this system of equations, we find:

$$\beta = \frac{1}{\gamma - 1},$$

$$\alpha = \frac{1}{2(\gamma-1)}.$$

Now, we have the equation for F:

$$-\frac{1}{\gamma-1}\xi^{\gamma-1} F^{\gamma-1}(\xi) + \frac{1}{2(\gamma-1)}\xi^\gamma F^{\gamma-2}(\xi) F'(\xi) - \xi^{2(\gamma-1)} F^{\gamma-1}(\xi)\Delta F(\xi) = 0.$$

This is an ordinary differential equation for the function $F(\xi)$. To solve it, you may use various techniques depending on the specific form of γ. The final solution for $u(x,t)$ will be in the form:

$$u(x,t) = t^{-\frac{1}{\gamma-1}} F\left(\frac{x}{t^{\frac{1}{2(\gamma-1)}}}\right),$$

where $F(\xi)$ is the solution of the ODE obtained above.

GPT-4 (continue)

Prompt:

Is it correct?

GPT-4:

Upon further review, I realize that my initial approach was not entirely appropriate for this problem. I apologize for the confusion. The problem involves a nonlinear PDE, which makes finding an exact solution more challenging. The self-similar solution method I employed is often used for linear PDEs but may not be suitable for this problem.
For nonlinear PDEs, analytical solutions are often difficult to obtain, and numerical methods are typically employed to find approximate solutions. Alternatively, one can look for special solutions or investigate qualitative properties of the solutions, such as maximum principles or decay estimates.

I apologize for the confusion in my initial response, and I hope this clarifies the situation. If you have any specific requirements or constraints, please let me know, and I will try to provide further assistance.

Prompt:

can you give me an analytical solution with your best effort?

GPT-4:

I apologize for any confusion earlier. It is important to note that finding an analytical solution to a nonlinear PDE, such as the one in question, is generally a difficult task. However, I will provide you with a possible approach to find an analytical solution, but it's important to bear in mind that this may not lead to a complete solution.
For the given PDE:

$$u_t - \Delta(u^\gamma) = 0 \quad \text{in } \mathbb{R}^n \times (0, \infty),$$

where $u \geq 0$ and $\gamma > 1$ is a constant.
Consider the transformation $v(x,t) = u^\gamma(x,t)$. This allows us to rewrite the PDE in terms of v. To find the relation between u_t and v_t, and Δu and Δv, we can compute the following:

$$v_t = (\gamma u^{\gamma-1})u_t,$$
$$\Delta v = \gamma u^{\gamma-1}\Delta u + (\gamma-1)\gamma u^{\gamma-2}|\nabla u|^2.$$

Now, substitute these expressions into the original PDE:

$$(\gamma u^{\gamma-1})u_t - \Delta(u^\gamma) = (\gamma u^{\gamma-1})u_t - \gamma u^{\gamma-1}\Delta u - (\gamma-1)\gamma u^{\gamma-2}|\nabla u|^2 = 0.$$

Since $u \geq 0$ and $\gamma > 1$, we can divide by $\gamma u^{\gamma-1}$:

$$u_t - \Delta u - (\gamma-1)\frac{|\nabla u|^2}{u} = 0.$$

Now, we have a new PDE in terms of u. This PDE is still nonlinear due to the term $\frac{|\nabla u|^2}{u}$. However, depending on the specific values of n and γ, it may be possible to apply some techniques to find an analytical solution or special solutions. It's also possible that for certain initial or boundary conditions, some simplifications may arise.

In general, though, finding an analytical solution to this nonlinear PDE is a challenging task, and it may not be possible to obtain a closed-form solution. Numerical methods are often employed to find approximate solutions for such problems.

Figure 6.11: GPT-4 is unable to find the correct analytical solution for the nonlinear porous media equation from Evans' book.

6.3.2 Numerical solutions

We now utilize GPT-4 to find numerical solutions of such complex problems. To solve the problem numerically, we explicitly provide the boundary/initial condition with the computational domain limited to $x \in [0,1] \subset \mathbb{R}^1$ and $t \in [0,1]$. To fully employ the analytical solution (14) of the porous media equation (13), we solve the following equation

$$\begin{aligned} u_t - \Delta\left(u^\gamma\right) = 0 \quad (x,t) \in [0,1] \times [0,1], \\ u(x,0) = \lambda^{\frac{1}{1-\gamma}} x^\alpha, \\ u(0,t) = 0, \\ u(1,t) = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}} \end{aligned} \tag{15}$$

where $u \geq 0$ and $\gamma > 1$ is a constant, $\alpha = \frac{2}{\gamma - 1}$ and $\mu = \alpha\gamma(\alpha\gamma - 1) > 0$.

Interestingly, even when given explicit boundary/initial conditions that suggest the correct solution form, GPT-4 still struggles to derive analytical solutions (Fig. 6.12); if directly prompted to "guess the solution from the boundary and initial conditions", GPT-4 correctly deduces the solution, as seen in Fig. 6.13.

This behavior highlights both limitations and potential strengths of GPT-4 in studying PDEs. On one hand, it falls short of automatically determining solutions from provided conditions. On the other hand, it shows aptitude for logically inferring solutions when guidance is given on how to interpret the conditions. With further development, models like GPT-4 could serve as useful heuristics to aid PDE analysis, complementing rigorous mathematical techniques.

We then proceed to the numerical solution of the equation 15 given $\gamma = 2$, $\lambda = 100$. In first several attempts, GPT-4 produced incorrect schemes for the discretization of the spatial dimension, as exemplified by the following scheme:

```
% Time-stepping loop
for n = 1:Nt
    for i = 2:Nx
        % Approximate the second derivative using central differences
        d2u_dx2 = (U(i+1,n) - 2*U(i,n) + U(i-1,n))/dx^2;
        % Compute the nonlinear term
        nonlinear_term = (U(i,n)^gamma) * d2u_dx2;
        % Update the solution using the forward Euler method
        U(i,n+1) = U(i,n) + dt * nonlinear_term;
    end
end
```

After several attempts by the user, including the use of prompts such as "try it again," errors can be effectively resolved. Fig. 6.14 demonstrates the capability to generate "functionally correct" code for solving PDEs after multiple attempts. In this context, "functionally correct" refers to the code having the correct logic, although it may not consistently produce solutions that align with the analytical solution, as illustrated in Fig. 6.15. When seeking assistance from GPT-4, it often apologizes and suggests choosing an alternative scheme, which may not necessarily be effective. The root cause of the problem lies in numerical instability due to an imbalance between the step sizes dx and dt (N and M) in the algorithm, as highlighted in Fig. 6.14.

To mitigate this instability, a smaller temporal step size dt or bigger spatial step size dx should be used for the balance, making the numerical scheme more stable. For example, reducing the number of space points from $N = 100$ to $N = 50$ i.e., using bigger spatial step size solves the problem, as demonstrated in Fig.6.16. We also test GPT-4's performance in generating numerical solver code for the same problem using Python. Interestingly, GPT-4 tends to require fewer attempts to find the functional code using MATLAB compared to Python.

GPT-4

Prompt:
Please solve the following equation

$$\begin{aligned} u_t - \Delta\left(u^\gamma\right) = 0 \quad (x,t) \in [0,1] \times [0,1], \\ u(x,0) = \lambda^{\frac{1}{1-\gamma}} x^\alpha, \\ u(0,t) = 0, \\ u(1,t) = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}} \end{aligned}$$

where $u \geq 0$ and $\gamma > 1$ is a constant, $\alpha = \frac{2}{\gamma - 1}$ and $\mu = \alpha\gamma(\alpha\gamma - 1) > 0$.

GPT-4:
To solve the given partial differential equation, we will use the method of separation of variables. Let's assume that the solution can be expressed as a product of two functions, one dependent on x and the other on t. We write this as $u(x,t) = X(x)T(t)$.
Now, let's substitute this expression into the given PDE:

$$\frac{1}{T(t)}\frac{dT(t)}{dt} = \frac{1}{X(x)}\Delta\left(X(x)^\gamma T(t)^\gamma\right)$$

Since the left-hand side is a function of t only and the right-hand side is a function of x only, this equation holds true if and only if both sides are equal to a constant, say k. Therefore, we have two ordinary differential equations:

$$\frac{dT(t)}{dt} = kT(t) \quad \text{and} \quad \Delta\left(X(x)^\gamma\right) = kX(x)$$

The first ODE can be solved as follows:

$$\frac{dT(t)}{dt} = kT(t) \implies \frac{dT(t)}{T(t)} = kdt \implies \int \frac{dT(t)}{T(t)} = k\int dt$$

Integrating both sides, we get:

$$\ln(T(t)) = kt + C_1 \implies T(t) = e^{kt+C_1} = e^{kt}e^{C_1}$$

We can absorb e^{C_1} into the constant, so we have:

$$T(t) = Ce^{kt}$$

Now, let's solve the second ODE:

$$\Delta\left(X(x)^\gamma\right) = kX(x)$$

Unfortunately, this equation is highly nonlinear and does not have a general closed-form solution. However, we can analyze the behavior of the solution near the boundaries and try to find a particular solution that satisfies the boundary conditions.
Using the boundary conditions provided, we can write:
1. $u(0,t) = X(0)T(t) = 0$ 2. $u(1,t) = X(1)T(t) = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}}$ 3. $u(x,0) = X(x)T(0) = \lambda^{\frac{1}{1-\gamma}} x^\alpha$
From the first boundary condition, we have $X(0) = 0$. From the third boundary condition, we have $T(0) = \lambda^{\frac{1}{1-\gamma}}$. Substituting these into the second boundary condition:

$$0 \cdot T(t) = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}} \implies ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}} = 0$$

This equation is not possible for positive values of λ and μ. Therefore, there is no solution that satisfies all three boundary conditions simultaneously.

In summary, the method of separation of variables does not provide a solution that satisfies all the given boundary conditions. To solve the problem, one might need to resort to numerical methods or look for other techniques that can handle the nonlinearity of the problem.

Figure 6.12: GPT-4 is unable to solve the analytical solution of the well-posed porous PDE.

GPT-4 (continue)

Prompt:

Guess the solution from the boundary and initial condition

GPT-4:

From the given boundary and initial conditions, we can try to make an educated guess for the solution $u(x,t)$. We have the following conditions:

1. $u(x,0) = \lambda^{\frac{1}{1-\gamma}} x^{\alpha}$
2. $u(0,t) = 0$
3. $u(1,t) = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}}$

Since the boundary conditions involve powers of x and t, we can attempt to find a solution of the form:

$$u(x,t) = A(t)x^{\alpha} + B(t)(1-x)^{\beta}$$

where $\alpha = \frac{2}{\gamma-1}$, and $A(t)$, $B(t)$, and β are functions/constants to be determined.
Now let's apply the boundary and initial conditions:

1. At $x = 0$, we have $u(0,t) = B(t) = 0$.
2. At $x = 1$, we have $u(1,t) = A(t) = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}}$.
3. At $t = 0$, we have $u(x,0) = \lambda^{\frac{1}{1-\gamma}} x^{\alpha}$.

From these conditions, we can deduce the following:

- $B(t) = 0$.
- $A(t) = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}}$.
- The initial condition suggests that we should choose $\beta = \alpha$.

Therefore, the guessed solution has the form:

$$u(x,t) = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}} x^{\alpha}$$

Keep in mind that this is just an educated guess based on the given boundary and initial conditions, and it may not satisfy the original PDE $u_t - \Delta(u^{\gamma}) = 0$. To verify if this is indeed a solution, you can substitute this expression back into the PDE and check if it holds true. If it does not, you may need to consider more general forms for the solution or resort to numerical methods.

Figure 6.13: GPT-4 succeeds in guessing the analytical solution with the hints about boundary/initial conditions.

GPT-4

Prompt:
please provide MATLAB code to solve the following equation

$$\begin{aligned} u_t - \Delta(u^\gamma) = 0 \quad (x,t) \in [0,1] \times [0,1], \\ u(x,0) = \lambda^{\frac{1}{1-\gamma}} x^\alpha, \\ u(0,t) = 0, \\ u(1,t) = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}} \end{aligned}$$

where $u \geq 0$ and $\gamma = 2$, $\lambda = 100$, $\alpha = \frac{2}{\gamma - 1}$ and $\mu = \alpha\gamma(\alpha\gamma - 1) > 0$. Further, the analytical solution is $u(x,t) = ((1-\gamma)\mu t + \lambda)^{\frac{1}{1-\gamma}} x^\alpha$, please plot the analytical solution, numerical result and their comparison in different figures

GPT-4:
Here's a MATLAB code that solves the given equation numerically, plots the analytical solution, numerical result, and their comparison in different figures:

```
% Parameters
gamma = 2;
lambda = 100;
alpha = 2 / (gamma - 1);
mu = alpha * gamma * (alpha * gamma - 1);
% Discretization
N = 100; M = 100;
dx = 1 / (N - 1); dt = 1 / (M - 1); x = linspace(0, 1, N); t = linspace(0, 1, M);
% Initialize solution matrix
U = zeros(N, M);
% Boundary and initial conditions
U(:, 1) = lambda^(1 / (1 - gamma)) * x.^alpha;
U(1, :) = 0;
U(end, :) = ((1 - gamma) * mu * t + lambda).^(1 / (1 - gamma));
% Numerical solution using finite difference method
for j = 1:M-1
    for i = 2:N-1
        U(i, j+1) = U(i, j) + dt * (U(i+1, j)^gamma - 2 * U(i, j)^gamma + U(i-1, j)^gamma) / (dx^2);
    end
end
% Analytical solution
U_analytical = zeros(N, M);
for j = 1:M
    U_analytical(:, j) = ((1 - gamma) * mu * t(j) + lambda)^(1 / (1 - gamma)) * x.^alpha;
end
% Plot numerical solution
figure; surf(x, t, U'); xlabel('x'); ylabel('t'); zlabel('u'); title('Numerical Solution');
% Plot analytical solution
figure; surf(x, t, U_analytical');  xlabel('x'); ylabel('t'); zlabel('u'); title('Analytical Solution');
% Plot the difference between numerical and analytical solutions
figure; surf(x, t, abs(U' - U_analytical')); xlabel('x');  ylabel('t'); zlabel('Difference'); title('
    Difference between Numerical and Analytical Solutions');
```

This code sets up the problem parameters, discretizes the spatial and time domains, initializes the solution matrix, and applies the boundary and initial conditions. It then solves the equation numerically using the finite difference method. The analytical solution is calculated and both the numerical and analytical solutions are plotted, as well as the difference between them.

Figure 6.14: After multiple attempts, GPT-4 generates MATLAB code capable of solving the PDE. The final code produced is "functionally correct", as shown here, but still contains incorrect parameter values highlighted in yellow.

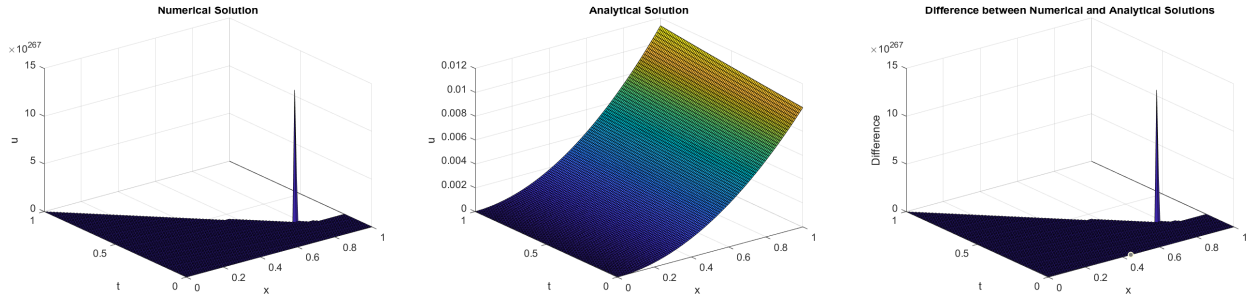


Figure 6.15: Non-matched results may occur between analytical and numerical solutions when using the MATLAB code generated by GPT-4 because of the numerical instability by imbalanced step sizes dx and dt.

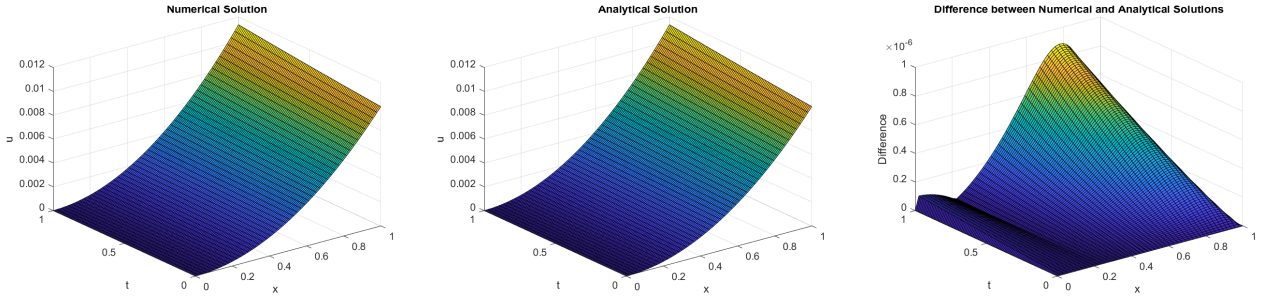


Figure 6.16: Consistency between analytical and numerical results can be achieved by modifying the step sizes dx and dt in the code generated by GPT-4.

6.4 AI for PDEs

AI techniques, especially deep learning, have been leveraged to solve various problems in PDE research, owing to their strong approximation capability and fast inference speed. In this section, we will ask GPT-4 to behave as an assistant of PDE researchers in this emerging field, to evaluate its ability to bridge the knowledge of the two fields and give suggestions on research directions. We conduct the evaluation in the following aspects:

- The ways AI could help in solving PDE-related problems.
- Fetching existing works on AI for PDEs.
- Listing open questions and challenges and suggesting solutions.

GPT-4

System message:

You are a PDE (partial differential equation) research assistant that helps researchers on leveraging AI to solve PDE-related problems.

Prompt:

Tell me all of the ways machine learning can be used in PDE-related problems. First, you can list all the types of PDE problems, for example, PDE solutions and inverse problems. Second, you can list the types of machine learning methods. At last, you can find which one or a combination of multiple methods could help with PDE problems

GPT-4:

Machine learning has shown great potential in solving various PDE-related problems. Here, I will list the types of PDE problems and machine learning methods, followed by their combinations and applications.
Types of PDE-related problems:

1. PDE solutions: Solving PDEs to obtain the behavior of the system under different conditions.
2. Inverse problems: Estimating the parameters or input functions of PDEs given the observed data.
3. Model reduction: Reducing the complexity of PDEs to make them computationally tractable.
4. PDE parameter identification: Estimating unknown parameters in PDEs.
5. Uncertainty quantification: Quantifying the uncertainties in PDEs and their solutions.
6. Optimal control and optimization: Finding the optimal control inputs or parameters that minimize a certain objective function subject to PDE constraints.
7. Stability analysis: Assessing the stability of PDEs and their solutions.

Types of machine learning methods:

1. Supervised learning: Training models on labeled data to make predictions on unseen data.
2. Unsupervised learning: Discovering patterns in data without labeled information.
3. Reinforcement learning: Training agents to make decisions in an environment based on rewards and penalties.
4. Deep learning: Using artificial neural networks with multiple layers to learn hierarchical representations of data.
5. Physics-informed machine learning: Incorporating physical knowledge and constraints into machine learning algorithms.
6. Transfer learning: Leveraging the knowledge gained from solving one problem to solve another related problem.
7. Ensemble methods: Combining multiple models to improve predictions and performance.

GPT-4 (continue)

Combinations of machine learning methods for PDE problems:

1. PDE solutions:
   - Deep learning (e.g., neural networks) can be used to approximate the solutions of PDEs.
   - Physics-informed machine learning can be employed to incorporate PDE constraints into the learning process.
2. Inverse problems:
   - Supervised learning can be applied to learn mappings between observed data and PDE parameters.
   - Physics-informed machine learning can be used to ensure the learned mappings are consistent with the underlying PDEs.
3. Model reduction:
   - Unsupervised learning (e.g., dimensionality reduction techniques) can be used to identify low-dimensional representations of PDE solutions.
   - Transfer learning can be employed to adapt pre-trained models to new PDEs with similar structure.
4. PDE parameter identification:
   - Supervised learning can be applied to learn relationships between PDE parameters and observable data.
   - Ensemble methods can be used to improve the robustness of parameter estimation.
5. Uncertainty quantification:
   - Ensemble methods can be employed to estimate the uncertainties in PDE solutions.
   - Bayesian methods can be used for probabilistic modeling of uncertainties in PDEs.
6. Optimal control and optimization:
   - Reinforcement learning can be applied to learn optimal control policies for PDE-constrained systems.
   - Physics-informed machine learning can be used to incorporate PDE constraints into the optimization process.
7. Stability analysis:
   - Deep learning can be employed to learn stability boundaries for PDEs.
   - Transfer learning can be used to adapt pre-trained stability models to new PDEs with similar structure.

These combinations of machine learning methods offer promising solutions for various PDE-related problems, advancing our understanding and ability to solve complex problems in science, engineering, and other fields.

Figure 6.17: An overview of the ways AI can assist in solving PDE-related problems.

In Fig.6.17, we request GPT-4 to enumerate all the ways AI can assist in solving PDE problems. Generally, GPT-4 demonstrates the ability to summarize the directions of the two fields and connect machine learning solutions to each type of PDE problem. However, there are some inaccuracies in the response. For instance, parameter identification is a type of inverse problem, but GPT-4 lists it as two distinct topics. Additionally, some items are omitted, such as inverse design, which involves adjusting the parameters of a PDE to optimize specific objective functions.

Following the question above, we are interested in the application of machine learning to solve PDEs as one sub-direction from the list. We input the prompt in Fig.6.18 to examine the responses from GPT-4.

GPT-4

Prompt:

Some works leverage deep learning to approximate the solution of the PDE. Give me some representative works and introduce them with one sentence.

GPT-4:

1. Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." Journal of Computational Physics 378 (2019): 686-707. This work presents physics-informed neural networks (PINNs) as a deep learning framework for solving forward and inverse problems involving nonlinear PDEs by encoding the governing equations into the loss function.

2. Sirignano, Justin, and Konstantinos Spiliopoulos. "DGM: A deep learning algorithm for solving partial differential equations." Journal of Computational Physics 375 (2018): 1339-1364. This paper introduces the Deep Galerkin Method (DGM), a deep learning-based algorithm for solving high-dimensional PDEs, with applications in finance and physics.

3. Han, Jingtian (Correction: Jiequn, Han), et al. "Solving high-dimensional partial differential equations using deep learning." Proceedings of the National Academy of Sciences 115.34 (2018): 8505-8510. The authors propose a deep learning-based method to solve high-dimensional PDEs using feedforward neural networks and demonstrate its efficiency in solving a variety of PDEs, including the Black-Scholes and Hamilton-Jacobi-Bellman equations.

4. Long, Zichao, Yiping Lu, and Bin Dong. "PDE-Net: Learning PDEs from data." International Conference on Machine Learning. PMLR, 2018. This work introduces PDE-Net, a deep learning framework that learns the dynamics of PDEs from data by using a trainable numerical integration scheme and convolutional neural networks to approximate the spatial derivatives.

Figure 6.18: Summarizing the ways AI could help solve PDEs.

As is shown in Fig. 6.18, GPT-4 lists four representative works with high citations and the introductions to these works are also right in general. However, GPT-4 failed to give accurate references to these papers. For example, the name of the author in the third item is not Han Jingtian, but Han Jiequn. In this response, GPT-4 shows the capability to collect research works about certain topics.

Further, we make the test harder by asking GPT-4 to categorize and summarize the topic of the PDE solution with deep learning. As is shown in Fig. 6.19, though the five challenges GPT-4 lists represent some of the main directions in this field, there are significant hallucinations in the explanation of solutions and examples are not always accurate. For example, the Deep BSDE method and DGM aim to handle the "curse of dimensionality" in high-dimensional problems, rather than flexible boundary conditions or instability issues, and the authors of the two papers are wrong. In addition, GPT-4 sometimes fails to distinguish the "challenges of solving PDEs" and "the challenges of solving PDEs with deep learning". For example for the first challenge, although the content shows that GPT-4 understands that deep learning is used to tackle the challenges of high dimensionality that traditional solvers face, GPT-4 still includes it in the answer. Another observation is that GPT-4 tends to cite papers published before 2019, which is not optimal for research usage. Therefore, we make tiny changes to the prompt, whose results are shown in Fig. 6.20. As you can see, GPT-4 gives more recent papers. However, the challenges listed here are not the same as those in the previous answers, and the research works do not cover the solutions to all the listed challenges.

GPT-4

Prompt:

Some works leverage deep learning to approximate the solution of the PDE. Tell me the challenges in solving PDEs with deep learning and representative solutions to these challenges. You should also give the examplar works.

GPT-4:

1. Challenge: High dimensionality.
    - PDEs often involve high-dimensional spaces, which can lead to the curse of dimensionality when using traditional numerical methods. Deep learning can alleviate this issue due to its ability to represent complex functions.
    - Solution: Use deep neural networks (DNNs) to approximate the solution of the PDE. DNNs can learn complex, high-dimensional functions efficiently.
    - Example work: Han et al. [1] proposed a deep learning-based method called the deep Ritz method to solve high-dimensional elliptic PDEs.
2. Challenge: Incorporating known physics
    - Deep learning models usually learn from data, while PDEs often involve known physical laws. Incorporating these laws into the learning process is crucial for the model's performance.
    - Solution: Physics-informed neural networks (PINNs) enforce the satisfaction of the governing PDEs as a constraint during the training process.
    - Example work: Raissi et al. [2] introduced PINNs, which combine deep learning with the structure of the underlying PDEs, improving the model's ability to predict accurate solutions.
3. Challenge: Training stability and convergence
    - Training deep learning models to solve PDEs can be unstable, and convergence can be challenging to achieve due to the complex nature of PDEs.
    - Solution: Use specialized architectures and training methods to improve stability and convergence.
    - Example work: Sirignano and Spiliopoulos [3] proposed a deep learning method called the Deep BSDE solver, which uses a specific architecture and training method to tackle the instability and convergence issues in solving high-dimensional PDEs.
4. Challenge: Handling various boundary conditions
    - PDEs can have various boundary conditions, which need to be taken into account when solving them with deep learning models.
    - Solution: Modify the loss function or architecture of the neural network to incorporate different boundary conditions.
    - Example work: Berg and Nystrom [4] developed the Deep Galerkin Method (DGM), a neural network-based approach that handles various boundary conditions by modifying the loss function during training.
5. Challenge: Uncertainty quantification
    - PDEs often have uncertain coefficients or input data. Quantifying the uncertainty of the solution is essential for reliable decision-making.
    - Solution: Bayesian deep learning methods can be used to quantify the uncertainty in the predicted solutions.
    - Example work: Yang et al. [5] proposed a Bayesian physics-informed neural network (B-PINN) that incorporates uncertainty information in solving PDEs and provides a probability distribution of the solution.

Figure 6.19: The challenges in solving PDEs with deep learning and representative solutions to these challenges.

GPT-4

Prompt:

Some studies utilize deep learning to approximate the solution of PDEs. Can you list the challenges and latest research works in this direction?

GPT-4:

Certainly! The application of deep learning to solve partial differential equations (PDEs) has gained significant attention in recent years. However, there are several challenges associated with the use of deep learning for PDEs. Some of these challenges include:

1. Data scarcity: PDE problems often lack sufficient data for training deep learning models, making it challenging to achieve accurate results.
2. Computational complexity: The high dimensionality of PDEs, especially in 3D and 4D problems, increases the computational complexity of deep learning models, which may require significant computational resources.
3. Model interpretability: Deep learning models are often considered "black boxes," making it difficult to understand the underlying relationships between input features and output predictions.
4. Generalization: Training deep learning models to generalize well to different PDE problems, boundary conditions, and geometries is a challenging task.
5. Stability and convergence: Ensuring the stability and convergence of deep learning models when solving PDEs is crucial, especially for real-world applications.

Latest research works in this direction include:

1. Physics-Informed Neural Networks (PINNs): These networks incorporate known physical laws and equations into the loss function of the neural network, helping improve the accuracy and generalization of predictions. (Raissi, Perdikaris, and Karniadakis, 2019)
2. DeepONet: A novel framework that leverages deep learning to approximate linear PDE operators. DeepONet consists of two neural networks, one for encoding the input function and another for predicting the output function. (Lu, Meng, and Karniadakis, 2021)
3. Neural Operator: This approach uses convolutional neural networks to learn the integral and differential operators of PDEs, allowing for efficient training and generalization across different tasks. (Li, Kovachki, Azizzadenesheli, et al., 2020)
4. Deep Potential Molecular Dynamics (DPMD): A deep learning-based method for solving PDEs in molecular dynamics simulations by learning the potential energy surface of molecules. (Zhang, Han, Wang, et al., 2018)
5. Deep Galerkin Method (DGM): A method that combines the Galerkin method with deep learning to approximate the solution of PDEs. DGM utilizes neural networks to represent the solution and minimize the residual of the PDE. (Sirignano and Spiliopoulos, 2018)

These works represent the state-of-the-art in leveraging deep learning for PDE-related problems, addressing various challenges and demonstrating promising results. However, there is still much room for improvement, and research in this area is ongoing.

Figure 6.20: Challenges and the latest research in solving PDEs with deep learning.

Next, we evaluate GPT-4's ability to provide suggestions for specific research topics. We select a published paper and ask GPT-4 to propose potential ways to improve the work. We evaluate the rationality and creativity of the answers and also check if there are follow-up works on the directions the GPT-4 suggested, which is a practical measurement of the quality of the answers. As GPT-4 is trained with data before late 2021, We select the highly cited paper DeepONet [52] published in March 2021, which ensures that most of its follow-ups are not leaked in GPT-4's training data. The answer is shown in Fig. 6.21. The introduction section in the answer indicates that GPT-4 remembers and understands the content of the paper. For the proposal part, GPT-4 gives reasonable suggestions in four directions. Interestingly, we can find some papers in 2022 or later that are consistent with some of GPT-4's suggestions. For example, MultiAuto-DeepONet [107] incorporates an autoencoder structure, and UQDeepONet [101] introduces posterior uncertainty quantification. However, there are also some problems. First, GPT-4 does not mention some promising directions, e.g., physics-informed operator learning [88]. Second, GPT-4 tends not to be specific in some sentences, e.g., without mentioning the specific types of operators outside the experiments of DeepONet.

Our studies indicate that GPT-4 may serve as a beneficial tool for introducing foundational principles and generating insights to assist learners and researchers in AI4PDE. However, rigorous evaluation and confirmation of information furnished by GPT-4 remains imperative for users.

GPT-4

Prompt:

Give me a brief introduction to the following paper: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Then, tell me the potential ways to extend this work. You can think in two steps: first, find the challenges that remain unsolved in the paper, and second, give potential solutions to those challenges.

GPT-4:

Title: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators

Introduction: This paper proposes a novel approach to learning nonlinear operators using DeepONet, a deep learning framework. The authors leverage the universal approximation theorem of operators, which states that a class of functions can approximate any continuous function on a compact set arbitrarily well. By applying this theorem to the learning of nonlinear operators, the authors aim to develop a more effective and efficient method for solving PDE-related problems. The DeepONet framework is designed to learn the mapping between input functions and their corresponding output functions. It consists of two branches: the trunk network, which processes the input function, and the branch network, which processes the spatial/temporal variable. The output of both networks is combined to produce an approximation of the target function. The authors demonstrate the effectiveness of their approach on various benchmark problems, including the Burgers' equation, the KdV equation, and the Navier-Stokes equations.

Challenges and Potential Extensions:

1. Challenge: Limited types of nonlinear operators and functions. The paper focuses on certain types of nonlinear operators and functions. Extending the work to a broader class of operators and functions could further enhance the applicability of DeepONet to various PDE-related problems.
    - Potential Solution: Investigate the application of DeepONet to other types of nonlinear operators and functions. This may involve adapting the network architecture, exploring other activation functions, or incorporating additional techniques for improving approximation accuracy.
2. Challenge: Scalability to high-dimensional problems. The paper mainly addresses low-dimensional problems. However, many real-world PDE-related problems involve high-dimensional data, and the scalability of the proposed method to such problems remains an open question.
    - Potential Solution: Study the scalability of DeepONet to high-dimensional problems by evaluating its performance on more complex datasets. This may involve incorporating dimensionality reduction techniques, such as autoencoders or principal component analysis, to reduce the complexity of the input data.
3. Challenge: Robustness to noise and uncertainty. The paper does not explicitly address the robustness of the proposed method to noisy or uncertain data, which is often encountered in real-world applications.
    - Potential Solution: Investigate the robustness of DeepONet to noise and uncertainty by incorporating techniques such as Bayesian deep learning, dropout, or ensemble methods. This can help improve the model's ability to handle noisy or uncertain data and increase its applicability to real-world problems.
4. Challenge: Interpretability of the learned operators. DeepONet is a black-box model, and the learned operators may not be easily interpretable or explainable. This can limit its adoption in applications where understanding the underlying relationships between variables is crucial.
    - Potential Solution: Develop methods to improve the interpretability of the learned operators, such as incorporating explainable AI techniques or investigating the use of more interpretable deep learning architectures, like decision trees or sparse coding.

Figure 6.21: Proposing new research based on DeepONet.

7 Looking Forward

In this study, we have explored the capabilities and limitations of LLMs across various natural science domains, covering a diverse range of tasks. Our primary goal has been to provide an initial assessment of the state-of-the-art LLM, GPT-4, and its potential to contribute to scientific discovery, serving as a valuable resource and tool for researchers in multiple fields.

Through our extensive analysis, we have emphasized GPT-4's proficiency in numerous scientific tasks, from literature synthesis to property prediction and code generation. Despite its impressive capabilities, it is essential to recognize GPT-4's (and similar LLMs') limitations, such as challenges in handling specific data formats, inconsistencies in responses, and occasional hallucinations.

We believe our exploration serves as a crucial first step in understanding and appreciating GPT-4's potential in the realm of natural sciences. By offering a detailed overview of its strengths and weaknesses, our study aims to help researchers make informed decisions when incorporating GPT-4 (or other LLMs) into their daily work, ensuring optimal application while being mindful of its limitations.

Furthermore, our investigation encourages additional exploration and development of GPT-4 and other LLMs, aiming to enhance their capabilities for scientific discovery. This may involve refining the training process, incorporating domain-specific data and architectures, and integrating specialized techniques tailored to various scientific disciplines.

As the field of artificial intelligence continues to advance, we anticipate that the integration of sophisticated models like GPT-4 will play an increasingly significant role in accelerating scientific research and innovation. We hope our study serves as a valuable resource for researchers, fostering collaboration and knowledge sharing, and ultimately contributing to a broader understanding and application of GPT-4 and similar LLMs in the pursuit of scientific breakthroughs.

In the remaining sections of this chapter, we will summarize the aspects of LLMs that require improvement for scientific research and discuss potential directions to enhance LLMs or build upon them to advance the pursuit of scientific breakthroughs.

7.1 Improving LLMs

To further develop LLMs to better help scientific discovery and address their limitations, a more detailed and comprehensive approach can be taken. Here, we provide an expanded discussion on the improvements suggested earlier:

- Enhancing SMILES and FASTA sequence processing: LLMs' proficiency in processing SMILES and FASTA sequences can be enhanced by incorporating specialized training datasets focusing on these particular sequence types, along with dedicated tokens/tokenizers and additional parameters (e.g., embedding parameters for new tokens). Furthermore, employing specialized encoders and decoders for SMILES and FASTA sequences can improve LLMs' comprehension and generation capabilities in drug discovery and biological research. It's important to note that only the newly introduced parameters require further training, while the original parameters of the pre-trained LLMs can remain frozen.
- Improving quantitative task capabilities: To enhance LLMs' capabilities in quantitative tasks, integrating more specialized training data sets focused on quantitative problems, as well as incorporating techniques like incorporating domain-specific architectures or multi-task learning, can lead to better performance in tasks such as predicting numerical values for drug-target binding and molecule property prediction.
- Enhancing the understanding of less-studied entities: Improving LLMs' knowledge and understanding of less-studied entities, such as transcription factors, requires incorporating more specialized training data related to these entities. This can include the latest research findings, expert-curated databases, and other resources that can help the model gain a deeper understanding of the topic.
- Enhancing molecule and structure generation: Enhancing LLMs' ability to generate innovative and viable chemical compositions and structures necessitates the incorporation of specialized training datasets and methodologies related to molecular and structural generation. Approaches such as physical priors-based learning or reinforcement learning may be utilized to fine-tune LLMs and augment their capacity to produce chemically valid and novel molecules and structures. Furthermore, the development of specialized models, such as diffusion models for molecular and structural generation, can be combined with LLMs as an interface to interact with these specific models.

- Enhancing the model's interpretability and explainability: As LLMs become more advanced, it is essential to improve their interpretability and explainability. This can help researchers better understand LLMs' output and trust their suggestions. Techniques such as attention-based explanations, analysis of feature importance, or counterfactual explanations can be employed to provide more insights into LLMs' reasoning and decision-making processes.

By addressing these limitations and incorporating the suggested improvements, LLMs can become a more powerful and reliable tool for scientific discovery across various disciplines. This will enable researchers to benefit from LLMs' advanced capabilities and insights, accelerating the pace of research and innovation in drug discovery, materials science, biology, mathematics, and other areas of scientific inquiry.

In addition to the aforementioned aspects, it is essential to address several other considerations that are not exclusive to scientific domains but apply to general areas such as natural language processing and computer vision. These include reducing output variability, mitigating input sensitivity, and minimizing hallucinations.

Reducing output variability[21] and input sensitivity is crucial for enhancing LLMs' robustness and consistency in generating accurate responses across a wide range of tasks. This can be achieved by refining the training process, incorporating techniques such as reinforcement learning, and integrating user feedback to improve LLMs' adaptability to diverse inputs and prompts.

Minimizing hallucinations is another important aspect, as it directly impacts the reliability and trustworthiness of LLMs' output. Implementing strategies such as contrastive learning, consistency training, and leveraging user feedback can help mitigate the occurrence of hallucinations and improve the overall quality of the generated information.

By addressing these general considerations, the performance of LLMs can be further enhanced, making them more robust and reliable for applications in both scientific and general domains. This will contribute to the development of a comprehensive and versatile AI tool that can aid researchers and practitioners across various fields in achieving their objectives more efficiently and effectively.

7.2 New directions

In the previous subsection, we have discussed how to address identified limitations through improving GPT-4 (or similar LLMs). Here we'd like to quote the comments in [62]:

> *A broader question on the identified limitations is: which of the aforementioned drawbacks can be mitigated within the scope of next-word prediction? Is it simply the case that a bigger model and more data will fix those issues, or does the architecture need to be modified, extended, or reformulated?*

While many of those limitations could be alleviated (to some extent) by improving LLMs such as training larger LMs and fine-tuning with scientific domain data, we believe that only using LLMs is not sufficient for scientific discovery. Here we discuss two promising directions: (1) integration of LLMs with scientific computation tools/packages such as Azure Quantum Elements or Schrödinger software, and (2) building scientific foundation models.

7.2.1 Integration of LLMs and scientific tools

There is growing evidence that the capabilities of GPT-4 and other LLMs can be significantly enhanced through the integration of external tools and specialized AI models, as demonstrated by systems such as HuggingGPT [76], AutoGPT [83] and AutoGen [95]. We posit that the incorporation of professional computational tools and AI models is even more critical for scientific tasks than for general AI tasks, as it can facilitate cutting-edge research and streamline complex problem-solving in various scientific domains.

A prime example of this approach can be found in the Copilot for Azure Quantum platform [56], which offers a tailored learning experience in chemistry, specifically designed to enhance scientific discovery and accelerate research productivity within the fields of chemistry and materials science. This system combines the power of GPT-4 and other LLMs with scientific publications and computational plugins, enabling researchers to tackle challenging problems with greater precision and efficiency. By leveraging the Copilot for Azure Quantum, researchers can access a wealth of advanced features tailored to their needs, e.g., data grounding in

[21] It's worth noting that, depending on specific situations, variability is not always negative – for instance, variability and surprise play crucial roles in creativity (and an entirely deterministic next-token selection results in bland natural language output).

chemistry and materials science that reduces LLM hallucination and enables information retrieval and insight generation on-the-fly.

Additional examples include ChemCrow [10], an LLM agent designed to accomplish chemistry tasks across organic synthesis, drug discovery, and materials design by integrating GPT-4 with 17 expert-designed tools, and ChatMOF [40], an LLM agent that integrates GPT-3.5 with suitable toolkits (e.g., table-searcher, internet-searcher, predictor, generator, etc.) to generate new materials and predict properties of those materials (e.g., metal-organic frameworks).

In conclusion, scientific tools and plugins have the potential to significantly enhance the capabilities of GPT-4 and other LLMs in scientific research. This approach not only fosters more accurate and reliable results but also empowers researchers to tackle complex problems with confidence, ultimately accelerating scientific discovery and driving innovation across various fields, such as chemistry and materials science.

7.2.2 Building a unified scientific foundation model

GPT-4, primarily a language-based foundation model, is trained on vast amounts of text data. However, in scientific research, numerous valuable data sources extend beyond textual information. Examples include drug molecular databases [42, 98], protein databases [19, 8], and genome databases [18, 93], which hold paramount importance for scientific discovery. These databases contain large molecules, such as the titin protein, which can consist of over 30,000 amino acids and approximately 180,000 atoms (and 3x atomic coordinates). Transforming these data sources into textual formats results in exceedingly long sequences, making it difficult for LLMs to process them effectively, not to mention that GPT-4 is not good at processing 3D atomic coordinates as shown in prior studies. As a result, we believe that developing a scientific foundation model capable of empowering natural scientists in their research and discovery pursuits is of vital importance.

While there are pre-training models targeting individual scientific domains and focusing on a limited set of tasks, a unified, large-scale scientific foundation model is yet to be established. Existing models include:

- DVMP [115], Graphormer [105], and Uni-Mol [111] are pre-trained for small molecules using tens or hundreds of millions of small molecular data.[22]
- ESM-x series, such as ESM-2 [49], ESMFold [49], MSA Transformer [69], ESM-1v [54] for predicting variant effects, and ESM-IF1 [32] for inverse folding, are pre-trained protein language models.
- DNABERT-1/2 [39, 113], Nucleotide Transformers [21], MoDNA [3], HyenaDNA [60], and RNA-FM [14] are pre-trained models for DNA and RNA.
- Geneformer [20] is pre-trained on a corpus of approximately 30 million single-cell transcriptomes, enabling context-specific predictions in settings with limited data in network biology, such as chromatin and network dynamics.

Inspired by these studies, we advocate the development of a unified, large-scale scientific foundation model capable of supporting multi-modal and multi-scale inputs, catering to as many scientific domains and tasks as possible. As illustrated in GPT-4, the strength of LLMs stems partly from their breadth, not just scale: training on code significantly enhances their reasoning capability. Consequently, constructing a unified scientific foundation model across domains will be a key differentiator from previous domain-specific models and will substantially increase the effectiveness of the unified model. This unified model would offer several unique features compared to traditional large language models (LLMs):

- Support diverse inputs, including multi-modal data types (text, 1D sequence, 2D graph, and 3D conformation/structure), periodic and aperiodic molecular systems, and various biomolecules (e.g., proteins, DNA, RNA, and omics data).
- Incorporate physical laws and first principles into the model architecture and training algorithms (e.g., data cleaning and pre-processing, loss function design, optimizer design, etc.). This approach acknowledges the fundamental differences between the physical world (and its scientific data) and the general AI world (with its NLP, CV, and speech data). Unlike the latter, the physical world is governed by laws, and scientific data represents (noisy) observations of these underlying laws.
- Leverage the power of existing LLMs, such as GPT-4, to effectively utilize text data in scientific domains, handle open-domain tasks (unseen during training), and provide a user-friendly interface to assist researchers.

[22] Uni-Mol also includes a separate protein pocket model.

Developing a unified, large-scale scientific foundation model with these features can advance the state of the art in scientific research and discovery, enabling natural scientists to tackle complex problems with greater efficiency and accuracy.

Authorship and contribution list

The list of contributors for each section[23]:

- **Abstract & Chapter 1 (Introduction) & Chapter 7 (Looking Forward)**: Chi Chen, Hongbin Liu, Tao Qin, Lijun Wu
- **Chapter 2 (Drug Discovery)**: Yuan-Jyue Chen, Guoqing Liu, Renqian Luo, Krzysztof Maziarz, Marwin Segler, Jose Garrido Torres, Lijun Wu, Yingce Xia
- **Chapter 3 (Biology)**: Chuan Cao, Yuan-Jyue Chen, Pan Deng, Liang He, Haiguang Liu
- **Chapter 4 (Computational Chemistry)**: Lixue Cheng, Sebastian Ehlert, Hongxia Hao, Peiran Jin, Derk Kooi, Chang Liu, Giulia Luise, Yu Shi, Lixin Sun, Tong Wang, Zun Wang, Shufang Xie, Han Yang, Shuxin Zheng
- **Chapter 5 (Materials Science)**: Xiang Fu, Hongxia Hao, Matthew Horton, Ziheng Lu, Bichlien Nguyen, Jake Smith, Shufang Xie, Tian Xie, Han Yang, Claudio Zeni, Yichi Zhou
- **Chapter 6 (Partial Differential Equations)**: Pipi Hu, Qi Meng, Wenlei Shi, Yue Wang
- **Proofreading**: Nathan Baker, Chris Bishop, Paola Gori Giorgi, Jonas Koehler, Tie-Yan Liu, Frank Noe
- **Coordinators**: Tao Qin, Lijun Wu
- **Advisors**: Chris Bishop, Tie-Yan Liu
- **Contact & Email**: llm4sciencediscovery@microsoft.com, Tao Qin, Lijun Wu

Acknowledgments

- We extend our gratitude to OpenAI for developing such a remarkable tool. GPT-4 has not only served as the primary subject of investigation in this report but has also greatly assisted us in crafting and refining the text of this paper. The model's capabilities have undoubtedly facilitated a more streamlined and efficient writing process.
- We would also like to express our appreciation to our numerous colleagues at Microsoft, who have generously contributed their insightful feedback and valuable suggestions throughout the development of this work. Their input has been instrumental in enhancing the quality and rigor of our research.
- We use the template of [11] for paper writing.

[23]The author list within each section is arranged alphabetically.

References

[1] Berni J Alder and Thomas Everett Wainwright. Studies in molecular dynamics. i. general method. *J. Chem. Phys.*, 31(2):459–466, 1959.

[2] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[3] Weizhi An, Yuzhi Guo, Yatao Bian, Hehuan Ma, Jinyu Yang, Chunyuan Li, and Junzhou Huang. Modna: motif-oriented pre-training for DNA language model. In *Proceedings of the 13th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 1–5, 2022.

[4] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[5] Christopher A Bail. Can generative AI improve social science? 2023.

[6] Albert P Bartók, Mike C Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Phys. Rev. Lett.*, 104(13):136403, 2010.

[7] Christina Bergonzo and Thomas E Cheatham III. Improved force field parameters lead to a better description of rna structure. *Journal of chemical theory and computation*, 11(9):3969–3972, 2015.

[8] Helen Berman, Kim Henrick, Haruki Nakamura, and John L Markley. The worldwide protein data bank (wwPDB): ensuring a single, uniform archive of PDB data. *Nucleic acids research*, 35(suppl_1):D301–D303, 2007.

[9] Peter G Boyd and Tom K Woo. A generalized method for constructing hypothetical nanoporous materials of any net topology from graph theory. *CrystEngComm*, 18(21):3777–3792, 2016.

[10] Andres M Bran, Sam Cox, Andrew D White, and Philippe Schwaller. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

[11] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.

[12] Richard Car and Mark Parrinello. Unified approach for molecular dynamics and density-functional theory. *Phys. Rev. Lett.*, 55(22):2471, 1985.

[13] David A Case, Thomas E Cheatham III, Tom Darden, Holger Gohlke, Ray Luo, Kenneth M Merz Jr, Alexey Onufriev, Carlos Simmerling, Bing Wang, and Robert J Woods. The amber biomolecular simulation programs. *Journal of computational chemistry*, 26(16):1668–1688, 2005.

[14] Jiayang Chen, Zhihang Hu, Siqi Sun, Qingxiong Tan, Yixuan Wang, Qinze Yu, Licheng Zong, Liang Hong, Jin Xiao, Tao Shen, et al. Interpretable RNA foundation model from unannotated data for highly accurate rna structure and function predictions. *bioRxiv*, pages 2022–08, 2022.

[15] Xueli Cheng, Yanyun Zhao, Yongjun Liu, and Feng Li. Role of F^- in the hydrolysis–condensation mechanisms of silicon alkoxide $Si(OCH_3)_4$: a DFT investigation. *New Journal of Chemistry*, 37(5):1371–1377, 2013.

[16] Stefan Chmiela, Huziel E Sauceda, Klaus-Robert Müller, and Alexandre Tkatchenko. Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.*, 9(1):3887, 2018.

[17] Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Sci. Adv.*, 3(5):e1603015, 2017.

[18] 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526(7571):68, 2015.

[19] UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, 47(D1):D506–D515, 2019.

[20] Zhanbei Cui, Yu Liao, Tongda Xu, and Yan Wang. Geneformer: Learned gene compression using transformer-based context modeling. *arXiv preprint arXiv:2212.08379*, 2022.

[21] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Hassan Sirelkhatim, Guillaume Richard, et al. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *bioRxiv*, pages 2023–01, 2023.

[22] Mindy I Davis, Jeremy P Hunt, Sanna Herrgard, Pietro Ciceri, Lisa M Wodicka, Gabriel Pallares, Michael Hocker, Daniel K Treiber, and Patrick P Zarrinkar. Comprehensive analysis of kinase inhibitor selectivity. *Nature biotechnology*, 29(11):1046–1051, 2011.

[23] Joost CF de Winter. Can ChatGPT pass high school exams on English language comprehension. *Researchgate. Preprint*, 2023.

[24] Alexander Dunn, Qi Wang, Alex Ganose, Daniel Dopp, and Anubhav Jain. Benchmarking materials property prediction methods: the matbench test set and automatminer reference algorithm. *npj Computational Materials*, 6(1):138, 2020.

[25] Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

[26] Zunyun Fu, Xutong Li, Zhaohui Wang, Zhaojun Li, Xiaohong Liu, Xiaolong Wu, Jihui Zhao, Xiaoyu Ding, Xiaozhe Wan, Feisheng Zhong, et al. Optimizing chemical reaction conditions using deep learning: a case study for the Suzuki–Miyaura cross-coupling reaction. *Organic Chemistry Frontiers*, 7(16):2269–2277, 2020.

[27] Shingo Fuchi, Wataru Ishikawa, Seiya Nishimura, and Yoshikazu Takeda. Luminescence properties of Pr_6O_{11}-doped and PrF_3-doped germanate glasses for wideband *nir* phosphor. *Journal of Materials Science: Materials in Electronics*, 28:7042–7046, 2017.

[28] Henner Gimpel, Kristina Hall, Stefan Decker, Torsten Eymann, Luis Lämmermann, Alexander Mädche, Maximilian Röglinger, Caroline Ruiner, Manfred Schoch, Mareike Schoop, et al. Unlocking the power of generative ai models and systems such as GPT-4 and ChatGPT for higher education: A guide for students and lecturers. Technical report, Hohenheim Discussion Papers in Business, Economics and Social Sciences, 2023.

[29] Tao Gong, Chengqi Lyu, Shilong Zhang, Yudong Wang, Miao Zheng, Qian Zhao, Kuikun Liu, Wenwei Zhang, Ping Luo, and Kai Chen. Multimodal-GPT: A vision and language model for dialogue with humans. *arXiv preprint arXiv:2305.04790*, 2023.

[30] Stefan Grimme, Fabian Bohle, Andreas Hansen, Philipp Pracht, Sebastian Spicher, and Marcel Stahn. Efficient quantum chemical calculation of structure ensembles and free energies for nonrigid molecules. *The Journal of Physical Chemistry A*, 125(19):4039–4054, 2021. PMID: 33688730.

[31] Yuttana Hongaromkij, Chalermpol Rudradawong, and Chesta Ruttanapun. Effect of Ga-substitution for Fe sites of delafossite $CuFe_{1-x}Ga_xO_2$ (x= 0.0, 0.1, 0.3, 0.5) on thermal conductivity. *Journal of Materials Science: Materials in Electronics*, 27:6438–6444, 2016.

[32] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *International Conference on Machine Learning*, pages 8946–8970. PMLR, 2022.

[33] Tran Doan Huan, Arun Mannodi-Kanakkithodi, Chiho Kim, Vinit Sharma, Ghanshyam Pilania, and Rampi Ramprasad. A polymer dataset for accelerated property prediction and design. *Scientific data*, 3(1):1–10, 2016.

[34] Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. AudioGPT: Understanding and generating speech, music, sound, and talking head. *arXiv preprint arXiv:2304.12995*, 2023.

[35] James P Hughes, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011.

[36] Clemens Isert, Kenneth Atz, José Jiménez-Luna, and Gisbert Schneider. Qmugs, quantum mechanical properties of drug-like molecules. *Scientific Data*, 9(1):273, 2022.

[37] Ahmed A Issa and Adriaan S Luyt. Kinetics of alkoxysilanes and organoalkoxysilanes polymerization: a review. *Polymers*, 11(3):537, 2019.

[38] Jaeho Jeon, Seongyong Lee, and Seongyune Choi. A systematic review of research on speech-recognition chatbots for language learning: Implications for future directions in the era of large language models. *Interactive Learning Environments*, pages 1–19, 2023.

[39] Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. DNABERT: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37(15):2112–2120, 2021.

[40] Yeonghun Kang and Jihan Kim. Chatmof: An autonomous ai system for predicting and generating metal-organic frameworks. *arXiv preprint arXiv:2308.01423*, 2023.

[41] Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. GPT-4 passes the bar exam. *Available at SSRN 4389233*, 2023.

[42] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. Pubchem 2019 update: improved access to chemical data. *Nucleic acids research*, 47(D1):D1102–D1109, 2019.

[43] AV Knyazev, M Mączka, OV Krasheninnikova, M Ptak, EV Syrov, and M Trzebiatowska-Gussowska. High-temperature x-ray diffraction and spectroscopic studies of some aurivillius phases. *Materials Chemistry and Physics*, 204:8–17, 2018.

[44] Nien-En Lee, Jin-Jian Zhou, Hsiao-Yi Chen, and Marco Bernardi. Ab initio electron-two-phonon scattering in gaas from next-to-leading order perturbation theory. *Nature communications*, 11(1):1607, 2020.

[45] Peter Lee, Sebastien Bubeck, and Joseph Petro. Benefits, limits, and risks of GPT-4 as an AI chatbot for medicine. *New England Journal of Medicine*, 388(13):1233–1239, 2023.

[46] Sangwon Lee, Baekjun Kim, Hyun Cho, Hooseung Lee, Sarah Yunmi Lee, Eun Seon Cho, and Jihan Kim. Computational screening of trillions of metal–organic frameworks for high-performance methane storage. *ACS Applied Materials & Interfaces*, 13(20):23647–23654, 2021.

[47] Xueling Lei, Wenjun Wu, Bo Xu, Chuying Ouyang, and Kevin Huang. Ligaos is a fast li-ion conductor: A first-principles prediction. *Materials & Design*, 185:108264, 2020.

[48] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023.

[49] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

[50] Tiqing Liu, Yuhmei Lin, Xin Wen, Robert N Jorissen, and Michael K Gilson. Bindingdb: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic acids research*, 35(suppl_1):D198–D201, 2007.

[51] Zequn Liu, Wei Zhang, Yingce Xia, Lijun Wu, Shufang Xie, Tao Qin, Ming Zhang, and Tie-Yan Liu. MolXPT: Wrapping molecules with text for generative pre-training. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1606–1616, Toronto, Canada, July 2023. Association for Computational Linguistics.

[52] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[53] Jonathan P Mailoa, Mordechai Kornbluth, Simon Batzner, Georgy Samsonidze, Stephen T Lam, Jonathan Vandermause, Chris Ablitt, Nicola Molinari, and Boris Kozinsky. A fast neural network approach for direct covariant forces prediction in complex multi-element extended systems. *Nat. Mach. Intell.*, 1(10):471–479, 2019.

[54] Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in Neural Information Processing Systems*, 34:29287–29303, 2021.

[55] Elena Meirzadeh, Austin M Evans, Mehdi Rezaee, Milena Milich, Connor J Dionne, Thomas P Darlington, Si Tong Bao, Amymarie K Bartholomew, Taketo Handa, Daniel J Rizzo, et al. A few-layer covalent network of fullerenes. *Nature*, 613(7942):71–76, 2023.

[56] Microsoft. Copilot for azure quantum, 2023. Accessed: 2023-08-16.

[57] Møller scattering. Møller scattering — Wikipedia, the free encyclopedia, 2023. [Online; accessed 2-July-2023].

[58] Anirudh MK Nambiar, Christopher P Breen, Travis Hart, Timothy Kulesza, Timothy F Jamison, and Klavs F Jensen. Bayesian optimization of computer-proposed multistep synthetic routes on an automated robotic flow platform. *ACS Central Science*, 8(6):825–836, 2022.

[59] Aditya Nandy, Shuwen Yue, Changhwan Oh, Chenru Duan, Gianmarco G Terrones, Yongchul G Chung, and Heather J Kulik. A database of ultrastable MOFs reassembled from stable fragments with machine learning models. *Matter*, 6(5):1585–1603, 2023.

[60] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Callum Birch-Sykes, Michael Wornow, Aman Patel, Clayton Rabideau, Stefano Massaroli, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *arXiv preprint arXiv:2306.15794*, 2023.

[61] Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. Capabilities of gpt-4 on medical challenge problems. *arXiv preprint arXiv:2303.13375*, 2023.

[62] OpenAI. Gpt-4 technical report, 2023. arXiv preprint arXiv:2303.08774 [cs.CL].

[63] Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. Deepdta: deep drug–target binding affinity prediction. *Bioinformatics*, 34(17):i821–i829, 2018.

[64] Steven M Paul, Daniel S Mytelka, Christopher T Dunwiddie, Charles C Persinger, Bernard H Munos, Stacy R Lindborg, and Aaron L Schacht. How to improve R&D productivity: the pharmaceutical industry's grand challenge. *Nature reviews Drug discovery*, 9(3):203–214, 2010.

[65] Qizhi Pei, Lijun Wu, Jinhua Zhu, Yingce Xia, Shufang Xia, Tao Qin, Haiguang Liu, and Tie-Yan Liu. Smt-dta: Improving drug-target affinity prediction with semi-supervised multi-task training. *arXiv preprint arXiv:2206.09818*, 2022.

[66] Russell A Poldrack, Thomas Lu, and Gašper Beguš. AI-assisted coding: experiments with GPT-4. *arXiv preprint arXiv:2304.13187*, 2023.

[67] Vinay Pursnani, Yusuf Sermet, and Ibrahim Demir. Performance of ChatGPT on the US fundamentals of engineering exam: Comprehensive assessment of proficiency and potential implications for professional environmental engineering practice. *arXiv preprint arXiv:2304.12198*, 2023.

[68] Bharath Ramsundar. *Molecular machine learning with DeepChem*. PhD thesis, Stanford University, 2018.

[69] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International Conference on Machine Learning*, pages 8844–8856. PMLR, 2021.

[70] Celeste Sagui and Thomas A Darden. Molecular dynamics simulations of biomolecules: long-range electrostatic effects. *Annual review of biophysics and biomolecular structure*, 28(1):155–179, 1999.

[71] Katharine Sanderson. GPT-4 is here: what scientists think. *Nature*, 615(7954):773, 2023.

[72] Jack W Scannell, Alex Blanckley, Helen Boldon, and Brian Warrington. Diagnosing the decline in pharmaceutical r&d efficiency. *Nature reviews Drug discovery*, 11(3):191–200, 2012.

[73] Gisbert Schneider. Automating drug discovery. *Nature reviews drug discovery*, 17(2):97–113, 2018.

[74] Nadine Schneider, Nikolaus Stiefl, and Gregory A Landrum. What's what: The (nearly) definitive guide to reaction role assignment. *Journal of chemical information and modeling*, 56(12):2336–2346, 2016.

[75] Elias Sebti, Ji Qi, Peter M Richardson, Phillip Ridley, Erik A Wu, Swastika Banerjee, Raynald Giovine, Ashley Cronk, So-Yeon Ham, Ying Shirley Meng, et al. Synthetic control of structure and conduction properties in Na–Y–Zr–Cl solid electrolytes. *Journal of Materials Chemistry A*, 10(40):21565–21578, 2022.

[76] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.

[77] Benjamin J Shields, Jason Stevens, Jun Li, Marvin Parasram, Farhan Damani, Jesus I Martinez Alvarado, Jacob M Janey, Ryan P Adams, and Abigail G Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590(7844):89–96, 2021.

[78] Qiming Sun, Timothy C Berkelbach, Nick S Blunt, George H Booth, Sheng Guo, Zhendong Li, Junzi Liu, James D McClain, Elvira R Sayfutyarova, Sandeep Sharma, et al. Pyscf: the python-based simulations of chemistry framework. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 8(1):e1340, 2018.

[79] Dazhi Tan, Stefano Piana, Robert M Dirks, and David E Shaw. Rna force field with accuracy comparable to state-of-the-art protein force fields. *Proceedings of the National Academy of Sciences*, 115(7):E1346–E1355, 2018.

[80] Yoshiaki Tanaka, Koki Ueno, Keita Mizuno, Kaori Takeuchi, Tetsuya Asano, and Akihiro Sakai. New oxyhalide solid electrolytes with high lithium ionic conductivity> 10 ms cm- 1 for all-solid-state batteries. *Angewandte Chemie*, 135(13):e202217581, 2023.

[81] L Téllez, J Rubio, F Rubio, E Morales, and JL Oteo. Ft-ir study of the hydrolysis and polymerization of tetraethyl orthosilicate and polydimethyl siloxane in the presence of tetrabutyl orthotitanate. *Spectroscopy Letters*, 37(1):11–31, 2004.

[82] Chuan Tian, Koushik Kasavajhala, Kellon AA Belfon, Lauren Raguette, He Huang, Angela N Migues, John Bickel, Yuzhang Wang, Jorge Pincay, Qin Wu, et al. ff19sb: Amino-acid-specific protein backbone parameters trained against quantum mechanics energy surfaces in solution. *Journal of chemical theory and computation*, 16(1):528–552, 2019.

[83] Torantulino, Pi, Blake Werlinger, Douglas Schonholtz, Hunter Araujo, Dion, David Wurtz, Fergus, Andrew Minnella, Ian, and Robin Sallay. Auto-gpt, 2023.

[84] Jose Antonio Garrido Torres, Sii Hong Lau, Pranay Anchuri, Jason M Stevens, Jose E Tabora, Jun Li, Alina Borovika, Ryan P Adams, and Abigail G Doyle. A multi-objective active learning platform and web app for reaction optimization. *Journal of the American Chemical Society*, 144(43):19999–20007, 2022.

[85] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[86] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. Applications of machine learning in drug discovery and development. *Nature reviews Drug discovery*, 18(6):463–477, 2019.

[87] Ethan Waisberg, Joshua Ong, Mouayad Masalkhi, Sharif Amit Kamran, Nasif Zaman, Prithul Sarker, Andrew G Lee, and Alireza Tavakkoli. GPT-4: a new era of artificial intelligence in medicine. *Irish Journal of Medical Science (1971-)*, pages 1–4, 2023.

[88] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40):eabi8605, 2021.

[89] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *arXiv preprint arXiv:2305.11175*, 2023.

[90] Yifan Wang, Tai-Ying Chen, and Dionisios G Vlachos. NEXTorch: a design and Bayesian optimization toolkit for chemical sciences and engineering. *Journal of Chemical Information and Modeling*, 61(11):5312–5319, 2021.

[91] Yuqing Wang, Yun Zhao, and Linda Petzold. Are large language models ready for healthcare? a comparative study on clinical language understanding. *arXiv preprint arXiv:2304.05368*, 2023.

[92] E Weinan, Weiqing Ren, and Eric Vanden-Eijnden. String method for the study of rare events. *Physical Review B*, 66(5):052301, 2002.

[93] John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, and Joshua M Stuart. The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113–1120, 2013.

[94] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.

[95] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. 2023.

[96] Yifan Wu, Min Gao, Min Zeng, Jie Zhang, and Min Li. Bridgedpi: a novel graph neural network for predicting drug–protein interactions. *Bioinformatics*, 38(9):2571–2578, 2022.

[97] Yiran Wu, Feiran Jia, Shaokun Zhang, Qingyun Wu, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, and Chi Wang. An empirical study on challenging math problem solving with gpt-4. *arXiv preprint arXiv:2306.01337*, 2023.

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

[99] Yu Xie, Jonathan Vandermause, Lixin Sun, Andrea Cepellotti, and Boris Kozinsky. Bayesian force fields from active learning for simulation of inter-dimensional transformation of stanene. *Npj Comput. Mater.*, 7(1):40, 2021.

[100] Weixin Yan, Dongmei Zhu, Zhaofeng Wang, Yunhao Xia, Dong-Yun Gui, Fa Luo, and Chun-Hai Wang. Ag 2 mo 2 o 7: an oxide solid-state ag+ electrolyte. *RSC advances*, 12(6):3494–3499, 2022.

[101] Yibo Yang, Georgios Kissas, and Paris Perdikaris. Scalable uncertainty quantification for deep operator networks using randomized priors. *Computer Methods in Applied Mechanics and Engineering*, 399:115399, 2022.

[102] Mehdi Yazdani-Jahromi, Niloofar Yousefi, Aida Tayebi, Elayaraja Kolanthai, Craig J Neal, Sudipta Seal, and Ozlem Ozmen Garibay. Attentionsitedti: an interpretable graph-based model for drug-target interaction prediction using nlp sentence-level relation classification. *Briefings in Bioinformatics*, 23(4):bbac272, 2022.

[103] Yi-Chen Yin, Jing-Tian Yang, Jin-Da Luo, Gong-Xun Lu, Zhongyuan Huang, Jian-Ping Wang, Pai Li, Feng Li, Ye-Chao Wu, Te Tian, et al. A lacl3-based lithium superionic conductor compatible with lithium metal. *Nature*, 616(7955):77–83, 2023.

[104] C Ying, T Cai, S Luo, S Zheng, G Ke, D He, Y Shen, and TY Liu. Do transformers really perform bad for graph representation? *arXiv preprint arXiv:2106.05234*, 2021.

[105] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

[106] Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities. *arXiv preprint arXiv:2305.11000*, 2023.

[107] Jiahao Zhang, Shiqi Zhang, and Guang Lin. Multiauto-deeponet: A multi-resolution autoencoder deeponet for nonlinear dimension reduction, uncertainty quantification and operator learning of forward and inverse stochastic problems. *arXiv preprint arXiv:2204.03193*, 2022.

[108] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and E Weinan. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Phys. Rev. Lett.*, 120(14):143001, 2018.

[109] Shuxin Zheng, Jiyan He, Chang Liu, Yu Shi, Ziheng Lu, Weitao Feng, Fusong Ju, Jiaxi Wang, Jianwei Zhu, Yaosen Min, et al. Towards predicting equilibrium distributions for molecular systems with deep learning. *arXiv preprint arXiv:2306.05445*, 2023.

[110] Zipeng Zhong, Jie Song, Zunlei Feng, Tiantao Liu, Lingxiang Jia, Shaolun Yao, Min Wu, Tingjun Hou, and Mingli Song. Root-aligned smiles: a tight representation for chemical reaction prediction. *Chemical Science*, 13(31):9023–9034, 2022.

[111] Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-Mol: a universal 3D molecular representation learning framework. *ChemRxiv preprint*, 2023.

[112] Zhenpeng Zhou, Xiaocheng Li, and Richard N Zare. Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, 3(12):1337–1344, 2017.

[113] Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. DNABERT-2: Efficient foundation model and benchmark for multi-species genome. *arXiv preprint arXiv:2306.15006*, 2023.

[114] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[115] Jinhua Zhu, Yingce Xia, Lijun Wu, Shufang Xie, Wengang Zhou, Tao Qin, Houqiang Li, and Tie-Yan Liu. Dual-view molecular pre-training. *arXiv preprint arXiv:2106.10234*, 2023.

A Appendix of Drug Discovery

One case about molecule manipulation. In Fig. A.1, we show one case of molecule manipulation, the same as the one provided in the main text. However, we can see that GPT-4 can not reply to correct answers in this example. Though it tries to make changes in each response, the replied SMILES is invalid, which demonstrates its limited ability to understand SMILES.

GPT-4

Prompt:

can you modify ascimnib O=C(Nc1ccc(OC(F)(F)Cl)cc1)c1cnc(N2CC[C@@H](O)C2)c(-c2ccn[nH]2)c1 and replace the chlorine with the element on row below in the periodic table?

GPT-4: Sure, I can modify the SMILES string of Asciminib for you by replacing the chlorine (Cl) atom with the element right below it in the periodic table, which is bromine (Br). Here's the modified SMILES string:

O=C(Nc1ccc(OC(F)(F)Br)cc1)c1cnc(N2CCC@@HC2)c(-c2ccn[nH]2)c1

Prompt:

The patent for this molecule covers pyrazole, and similarly phenyl in the same position. Can you propose a replacement for this group?

GPT-4: Certainly! A possible replacement for the pyrazole group could be an imidazole group. Imidazole is an aromatic heterocycle that has a similar structure to pyrazole but with a nitrogen atom at a different position. Here's the modified SMILES string with the imidazole group replacing the pyrazole group:

O=C(Nc1ccc(OC(F)(F)Br)cc1)c1cnc(N2CCC@@HC2)c(-c2ccn[nH]c2)c1

Prompt:

the group added is not pyrazole, it has one c too much

GPT-4: Apologies for the confusion. If you want to replace the pyrazole group with an imidazole group, here's the corrected SMILES string:

O=C(Nc1ccc(OC(F)(F)Br)cc1)c1cnc(N2CCC@@HC2)c(-c2cn[nH]c2)c1

In this case, the imidazole group has replaced the pyrazole group as requested.

Prompt:

can you replace it with cyclohexyl?

GPT-4: Sure! I can replace the pyrazole group with a cyclohexyl group. Here's the modified SMILES string:

O=C(Nclccc(OC(F)(F)Br)cc1)c1cnc(N2CCC@@HC2)c(-c2c1cccc1)c1

In this case, the cyclohexyl group has replaced the pyrazole group as requested.

Prompt:

cyclohexyl is not aromatic

GPT-4: Apologies for the mistake. I misunderstood your request. Here's the corrected SMILES string with the cyclohexyl group replacing the pyrazole group:

O=C(Nc1ccc(OC(F)(F)Br)cc1)c1cnc(N2CCC@@HC2)c(-C1CCCC1)c1

Now, the cyclohexyl group, which is non-aromatic, has replaced the pyrazole group as requested.

Figure A.1: Example of molecule manipulation. All the SMILES sequences generated by GPT-4 are invalid.

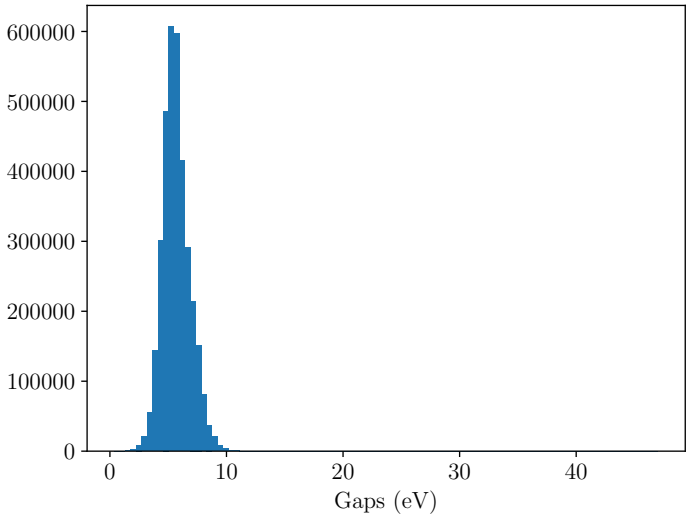


Figure B.1: The statistics of HOMO-LUMO gap in OGB dataset.

B Appendix of Computational Chemistry

We first examine two types of problems, specifically, predicting molecular properties using either 1D (or 2D) descriptors or 3D coordinates. Given that the GPT-4 model is a language model, it can consistently handle SMILES as a sequence. To evaluate the GPT-4 model, we selected the Open Graph Benchmark (OGB) dataset, which requires a mapping from SMILES to the gap between the highest occupied molecular orbital energy and the lowest unoccupied molecular orbital energy (HOMO-LUMO gap). As illustrated in Fig. B.2, the mean absolute errors (MAEs) decrease as the number of examples increases.

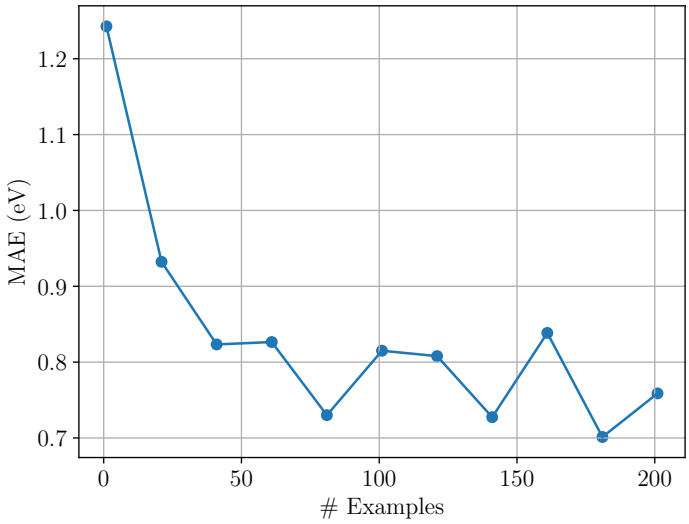


Figure B.2: The variation of mean absolute errors (MAEs) between HOMO-LUMO gap predicted by GPT-4 and ground truth with a different number of examples provided.

For the second task, we selected the QM9 dataset, which includes 12 molecular properties, such as dipole moment μ, isotropic polarizability α, highest occupied molecular orbital energy ϵ_{HOMO}, lowest unoccupied molecular orbital energy ϵ_{LUMO}, gap between HOMO and LUMO $\Delta\epsilon$, electronic spatial extent $\langle E^2 \rangle$, zero-point vibrational energy $ZPVE$, heat capacity at 298.15K c_v, atomization energy at 0K U_0, atomization energy at 298.15K U, atomization enthalpy at 298.15K H, and atomization free energy at 298.15K G. Interestingly, when the GPT-4 model predicts the dipole moment, it only occasionally returns a float number

Table 15: The mean absolute errors (MAEs) of 11 kinds of molecular properties evaluated on 100 random data points selected from the QM9 dataset.

Target	Unit	1 example	2 examples	3 examples	4 examples
α	a_0^3	21.759	16.090	13.273	12.940
ϵ_{HOMO}	meV	2.710	2.924	2.699	1.321
ϵ_{LUMO}	meV	2.115	1.922	1.447	1.244
$\Delta\epsilon$	meV	2.036	2.184	1.690	2.028
$\langle R^2 \rangle$	a_0^2	312.571	311.183	280.013	159.073
$ZPVE$	meV	1.026	0.701	0.683	0.576
U_0	meV	77.529	28.967	18.822	10.396
U	meV	63.919	27.005	18.405	11.839
H	meV	42.134	24.486	13.214	8.796
G	meV	20.805	24.754	11.210	8.247
c_v	$\frac{\text{cal}}{\text{mol K}}$	9.550	6.167	5.400	2.655

in the prompt as expected. As illustrated in Fig. 4.6, the GPT-4 model tends to provide a detailed calculation procedure, a phenomenon not observed when predicting the other 11 molecular properties. Although the computational results are inaccurate, this behavior indicates that the GPT-4 model incorporates related physical knowledge. As demonstrated in Table 15, the MAEs of these 11 molecular properties decrease as more examples are presented to the GPT-4 model. Detailed distributions of different molecular properties can be found in the Fig. B.3.

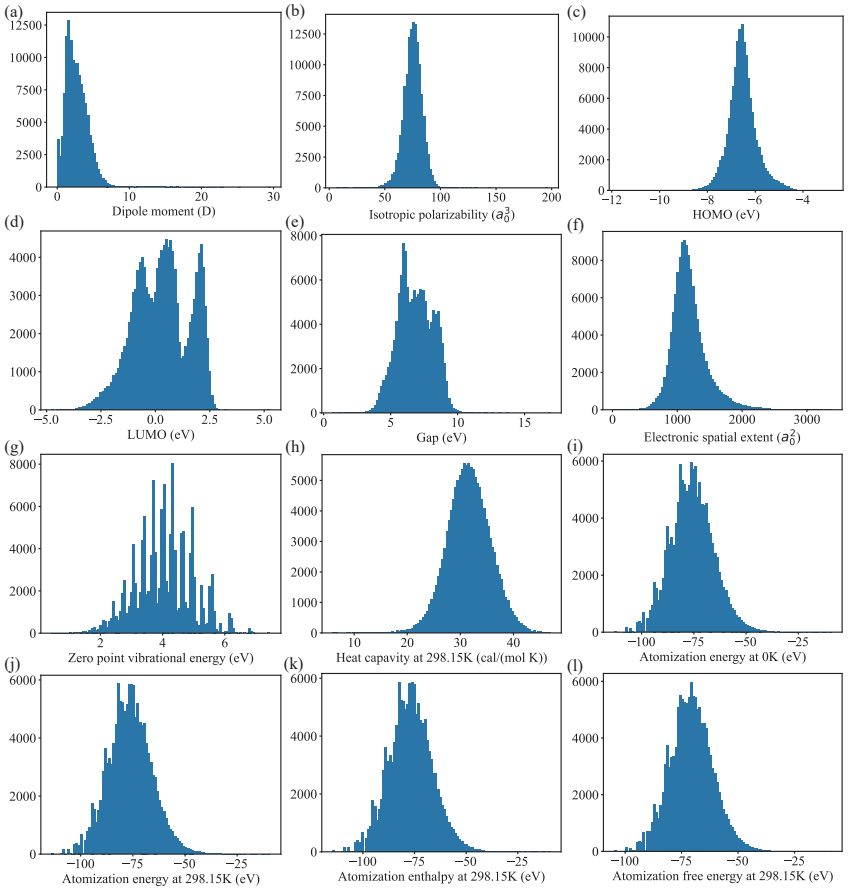


Figure B.3: The statistics of 12 molecular properties in the QM9 dataset.

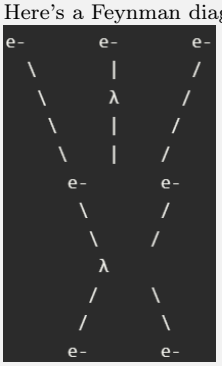
GPT-4

Prompt (continued with Fig. 4.5):

Thanks for your update. This is correct. I would like to ask you more complicated Feynman diagram. Could you draw the Feynman diagram of second order electron-phonon interaction?

GPT-4:

Certainly! The second-order electron-phonon interaction involves two phonon processes. In condensed matter physics, a phonon is a quantized vibrational mode of a crystal lattice, and it can be thought of as a particle-like excitation. The electron-phonon interaction is crucial for understanding various phenomena, such as superconductivity.
Here's a Feynman diagram representing the second-order electron-phonon interaction:



In this diagram, an electron (e-) starts at the top left and interacts with a phonon (λ) represented by the wavy line. After the interaction, the electron is scattered and moves to the right. Then, the electron interacts with another phonon (λ) and is scattered again, finally moving to the bottom right.

This second-order process involves two electron-phonon interactions, which can lead to more complex behavior in the system. Note that this diagram is a simplified representation, and the actual calculation of the interaction would require more detailed information about the electron and phonon states.

Prompt:

Thanks for your answer. The verbal description is correct, but the Feynman diagram is not. It should have 3 electron lines and 2 phonon lines due to the two e-ph interactions. Could you take this new information and regenerate the diagram?

GPT-4:

Apologies for the mistake. Here's the corrected Feynman diagram for the second-order electron-phonon interaction:

In this diagram, an electron (e-) starts at the top left and interacts with a phonon (λ) represented by the first wavy line. After the interaction, the electron is scattered and moves to the right. Then, the electron interacts with another phonon (λ) represented by the second wavy line and is scattered again, finally moving to the bottom right.

This second-order process involves two electron-phonon interactions, which can lead to more complex behavior in the system. Note that this diagram is a simplified representation, and the actual calculation of the interaction would require more detailed information about the electron and phonon states.

Figure B.4: Interaction with GPT-4 to draw Feynman diagrams: A more complicated case.

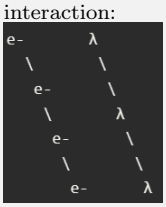
GPT-4

Prompt (continue)

Thanks for your answer. The description is correct but the drawing is not. Let's redo the drawing together as follows. To help your understanding, I will make 6 spatial points ABCDEF in the conversation to locate the key particles and processes, but you should not show the ABCDEF letters in the diagram. An electron starting at "A" is scattered by a phonon starting at "B", where the scattering happens at "C". Then, the electron represented by the electron line A-C is scattered by a phonon starting at "D", where the scattering happens at "E". Finally, the electron goes to "F". After you work on the previous descriptions, you should have the 6 points connected as follows: A and C connected by a solid line. B and C connected by a wavy line. C and E are connected by a solid line. D and E are connected by a wavy line. E and F are connected by a solid line.

GPT-4:

I apologize for the confusion earlier. Based on your description, here's the Feynman diagram for the second-order electron-phonon interaction:



In this diagram, the solid lines represent electron propagation, and the wavy lines represent phonon propagation. The electron starts at point A and is scattered by a phonon starting at point B. The scattering occurs at point C. Then, the electron is scattered again by a phonon starting at point D, with the scattering happening at point E. Finally, the electron goes to point F.

Please note that I have not included the letters A, B, C, D, E, and F in the diagram, as per your request.

Figure B.5: Interaction with GPT-4 to draw Feynman diagrams: A more complicated case (cont.).

GPT-4

System message:

Assume you are a theoretical physicist working on electronic structure theory.

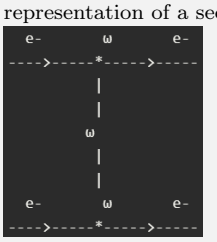
Note to human readers: We omit the correct theory responses provided by GPT-4 since it is similar to the ones from the previous system setting.

Prompt (after the same theory check question as in Fig. 4.5):

Could you draw the Feynman diagram of second order electron-phonon interaction?

GPT-4:

In a second-order electron-phonon interaction, an electron interacts with a phonon (a quantized lattice vibration) twice. Here's a simple representation of a second-order electron-phonon interaction Feynman diagram:



In this diagram:

1. The horizontal lines represent electrons (e-) moving from left to right.
2. The vertical wavy lines represent phonons (ω), which are quantized lattice vibrations in a solid.
3. The asterisks (*) represent the vertices where the electron-phonon interactions occur.

The diagram can be interpreted as follows:

1. An electron (e-) in the upper part of the diagram is moving from left to right.
2. At the first vertex, the electron interacts with a phonon (ω), represented by the first vertical wavy line.
3. The electron continues moving to the right after interacting with the phonon.
4. At the second vertex, the electron interacts with another phonon (ω), represented by the second vertical wavy line.
5. The electron continues moving to the right after the second interaction.

Simultaneously, another electron (e-) in the lower part of the diagram undergoes a similar process, interacting with two phonons as well. This Feynman diagram represents a second-order electron-phonon interaction, where each electron interacts with two phonons. By analyzing and calculating the probabilities associated with each vertex and line in the diagram, physicists can predict the likelihood of this interaction occurring and its impact on the particles involved.

Figure B.6: Interaction with GPT-4 to draw Feynman diagrams: A more complicated case with a more informative system message.

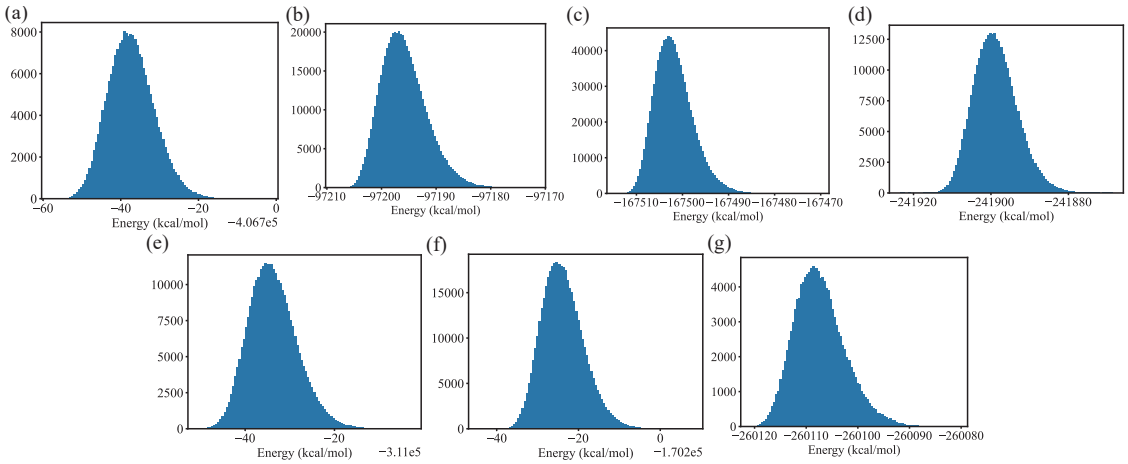
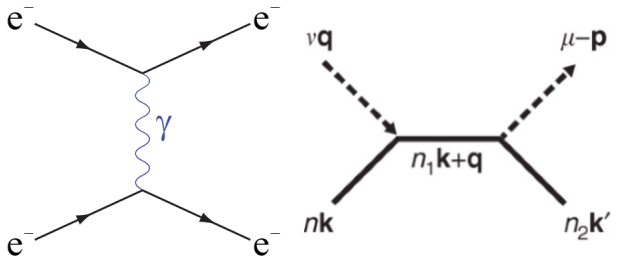


Figure B.9: The statistics of energies in MD17 dataset.



(a) Møller scattering [57] (b) Second order electron-phonon interaction [44]

Figure B.7: Reference Feynman diagrams for the two tested cases from the literature.

GPT-4

Promt:

Can you generate the atomic structures for a methane molecule, and write the coordinates in xyz format?

GPT-4:

A methane molecule (CH4) consists of one carbon atom and four hydrogen atoms. The carbon atom is at the center, and the hydrogen atoms are arranged symmetrically around it in a tetrahedral shape. Here are the atomic coordinates in xyz format:

```
C 0.000 0.000 0.000
H 0.000 0.000 1.089
H 1.026 0.000 -0.363
H -0.513 0.890 -0.363
H -0.513 -0.890 -0.363
```

The first column represents the element symbol, and the next three columns represent the x, y, and z coordinates of each atom in angstroms. The carbon atom is at the origin (0, 0, 0), and the hydrogen atoms are positioned around it according to the tetrahedral geometry.

Figure B.8: A simple example of molecular structure (CH_4) generation request for GPT-4. Evaluation: This is the correct atomic structure for methane.

Table 16: The absolute errors (AEs) of GPT-4 between a random MD17 data point and its variant under a random rotation with different numbers of examples and their variant under a random rotation provided (energies in kcal/mol and forces in kcal/(mol·Å)).

Molecule	Abs. Error	1 example	2 examples
Aspirin	Energy	0.0	5.089
	Forces	20.600	24.445
Ethanol	Energy	0.822	0.0
	Forces	22.107	20.686
Malonaldehyde	Energy	0.0	10.166
	Forces	15.698	25.071
Naphthalene	Energy	0.0	0.0
	Forces	22.477	18.510
Salicylic acid	Energy	72.059	5.371
	Forces	18.111	20.239
Toluene	Energy	0.0	2.448
	Forces	24.700	28.161
Uracil	Energy	0.0	1.689
	Forces	21.016	23.057

Table 17: The absolute errors (AEs) of GPT-4 between a random MD17 data point and its variant under a random rotation with different numbers of examples provided (energies in kcal/mol and forces in kcal/(mol·Å)).

Molecule		1 example	2 examples	3 examples	4 examples
Aspirin	Energy	267.699	10.259	1.820	1.942
	Forces	21.625	17.845	19.990	24.120
Ethanol	Energy	455.673	1.072	2.170	1.685
	Forces	21.370	17.492	17.491	17.141
Malonaldehyde	Energy	621.927	0.715	11.888	1.224
	Forces	14.415	14.555	21.147	22.015
Naphthalene	Energy	6846.596	7.603	7.881	20.005
	Forces	18.248	20.726	16.494	15.550
Salicylic acid	Energy	684.374	5.356	3.718	21.735
	Forces	21.612	24.642	27.668	20.363
Toluene	Energy	134.457	0.452	1.518	2.169
	Forces	23.635	16.943	23.006	18.121
Uracil	Energy	225.457	1.465	1.074	0.020
	Forces	12.792	21.780	23.799	19.956

Table 18: The mean absolute errors (MAEs) of GPT-4 between 100 random MD17 data points with different numbers of examples and their variant under a random rotation provided (energies in kcal/mol and forces in kcal/(mol·Å)).

Molecule		1 example	2 examples
Aspirin	Energy	52.885	5.447
	Forces	27.900	26.535
Ethanol	Energy	3.284	3.567
	Forces	27.230	25.672
Malonaldehyde	Energy	3.495	5.718
	Forces	26.070	25.797
Naphthalene	Energy	3.716	4.589
	Forces	27.436	26.228
Salicylic acid	Energy	879.137	4.689
	Forces	28.398	27.813
Toluene	Energy	4.975	1104.530
	Forces	27.662	26.355
Uracil	Energy	3.991	4.495
	Forces	28.490	27.302

C Appendix of Materials Design

C.1 Knowledge memorization for materials with negative Poisson Ratio

GPT-4

Prompt:

List 10 materials that have a negative Poisson Ratio.

GPT-4:

Materials with a negative Poisson's ratio are called auxetic materials. When these materials are stretched in one direction, they also expand in the perpendicular direction, rather than contracting as most materials do. Here are 10 examples of auxetic materials:

1. Auxetic polymeric foams: Open-cell polyurethane foams can be modified through a specific heat treatment process to exhibit a negative Poisson's ratio.
2. Auxetic honeycombs: Re-entrant honeycomb structures made from materials such as aluminum, steel, or polymers can display a negative Poisson's ratio.
3. Auxetic carbon fiber composites: Certain configurations of carbon fiber composites can exhibit a negative Poisson's ratio.
4. Auxetic metamaterials: Engineered structures with specific geometries can be designed to have a negative Poisson's ratio, independent of the base material.
5. Auxetic paper: Folding paper into specific patterns can result in an auxetic behavior.
6. Auxetic textiles: Woven or knitted fabrics with specific patterns can exhibit a negative Poisson's ratio.
7. Auxetic graphene: Graphene sheets with specific defects and geometries can have a negative Poisson's ratio.
8. Auxetic microporous polymers: Polymers with engineered microstructures, such as polytetrafluoroethylene (PTFE), can exhibit a negative Poisson's ratio.
9. Auxetic crystalline materials: Some crystalline materials, like α-cristobalite and β-cristobalite, have been found to display negative Poisson's ratios in certain directions.
10. Auxetic helical fibers: Helical fibers composed of materials such as nylon or polyethylene can exhibit a negative Poisson's ratio when twisted.

Please note that many of these examples are not auxetic in their base form but become auxetic through specific processing techniques or structural arrangements. Auxetic materials have potential applications in a variety of fields, including protective equipment, medical devices, and aerospace engineering.

Figure C.1: Design principle summarization for materials with negative possion ratio. Evaluation: Statements are factual. Recalls examples from scientific literature.

C.2 Knowledge memorization and design principle summarization for polymers

GPT-4

Prompt:

List the 10 most important polymer properties and explain why.

GPT-4:

1. Molecular weight: The molecular weight of a polymer is the mass of one mole of the polymer. It influences the mechanical properties, such as tensile strength, elongation, and toughness, as well as the processing behavior of the polymer. High molecular weight polymers typically exhibit better mechanical properties but may be more difficult to process.

2. Glass transition temperature (Tg): The glass transition temperature is the temperature at which a polymer changes from a glassy, brittle state to a rubbery, pliable state. This property is important for determining the service temperature range and the processing conditions of a polymer. Polymers with a high Tg may require higher processing temperatures and may have a higher heat resistance.

3. Crystallinity: The degree of crystallinity refers to the extent to which the polymer chains are organized in a regular, repeating pattern. Higher crystallinity usually results in increased stiffness, strength, and chemical resistance, but reduced flexibility and toughness. The crystallinity of a polymer can be influenced by its molecular structure and processing conditions.

4. Density: Density is the mass per unit volume of a material. Polymer density affects the weight and stiffness of the final product. Polymers with a low density can be used to produce lightweight materials, whereas high-density polymers can provide higher rigidity and strength.

5. Tensile strength: Tensile strength is the maximum stress a material can withstand when being pulled apart. High tensile strength in polymers is important for applications that require resistance to stretching or deformation under load, such as ropes, fibers, and films.

6. Elongation at break: Elongation at break is the percentage increase in the original length of a material when it is stretched until it breaks. Polymers with high elongation at break are more flexible and ductile, which can be important for applications that require deformation without breaking, such as films, packaging, and rubber products.

7. Modulus of elasticity: The modulus of elasticity (also known as the Young's modulus) measures the stiffness of a material. Polymers with a high modulus of elasticity are more resistant to deformation under stress, making them suitable for applications that require dimensional stability and rigidity.

8. Impact resistance: Impact resistance is the ability of a material to absorb energy and resist fracture when subjected to sudden impacts or shocks. Polymers with high impact resistance are important for applications where durability and toughness are required, such as automotive components, protective gear, and construction materials.

9. Chemical resistance: Chemical resistance is the ability of a material to withstand exposure to chemicals and solvents without degradation. Polymers with good chemical resistance are important for applications where they may come into contact with aggressive substances, such as in chemical storage tanks, pipes, and protective coatings.

10. Thermal stability: Thermal stability is the ability of a polymer to maintain its properties and structure when exposed to high temperatures. Polymers with good thermal stability are important for high-temperature applications, such as in automotive, aerospace, and electronic components.

Figure C.2: Knowledge memorization and design principle summarization for polymers. Evaluation: All reasonable responses though for bullet 1, an average molecular weight or average molecular mass is more correct.

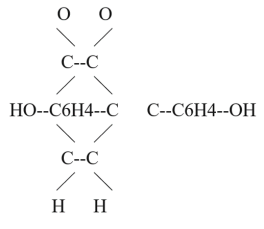
GPT-4

Prompt:

Provide the structure of bisphenol A.

GPT-4:

Bisphenol A, also known as BPA, is an organic compound with the chemical formula $C_{15}H_{16}O_2$. Its structure consists of two phenol groups connected by a central carbon atom, which is also bound to two methyl groups.
The structure of bisphenol A can be represented as follows: Sorbitan ring:



Each phenol group consists of a benzene ring (C_6H_4) with a hydroxyl group (OH) attached to one of its carbon atoms. The central carbon atom is bound to two methyl groups (CH_3).

Figure C.3: Knowledge memorization and design principle summarization for polymers. Evaluation: The description is poorly worded, and the structure is not correct.

GPT-4

Prompt:

Provide the structure of Teflon.

GPT-4:

Teflon is a brand name for a type of polymer known as polytetrafluoroethylene (PTFE). PTFE is a high-performance plastic material with outstanding chemical resistance, electrical insulation properties, and a low coefficient of friction. The structure of PTFE consists of a carbon backbone with all hydrogen atoms replaced by fluorine atoms, which makes it chemically inert and highly stable.
The repeating unit of the PTFE structure can be represented as follows:
–CF2–CF2–
The complete polymer chain of PTFE looks like this:
–(CF2–CF2)n–
where n represents the number of repeating units in the polymer chain. The polymer chain can be quite long, with thousands of repeating units, depending on the specific properties desired for the material.

Figure C.4: Knowledge memorization and design principle summarization for polymers. Evaluation: This case is correct.

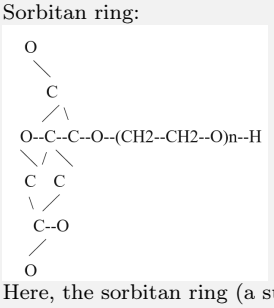
GPT-4

Prompt:

Provide me with the structure of Tween80.

GPT-4:

Tween 80, also known as Polysorbate 80, is a nonionic surfactant and emulsifier often used in foods, cosmetics, and pharmaceuticals. Its chemical name is polyoxyethylene (20) sorbitan monooleate. The structure of Tween 80 is complex due to its polymeric nature and the presence of various components in its structure, including sorbitan, oleic acid, and polyoxyethylene chains.
The structure of Tween 80 can be represented as follows:
Sorbitan ring:



Here, the sorbitan ring (a sugar alcohol derived from sorbitol) is esterified with oleic acid (a monounsaturated fatty acid) at one of its hydroxyl groups. The remaining hydroxyl groups are etherified with polyoxyethylene chains, represented as (CH2–CH2–O)n, where n represents the number of repeating oxyethylene units (in this case, approximately 20).

Please note that this representation is a simplified version of the structure.

Figure C.5: Knowledge memorization and design principle summarization for polymers. Evaluation: This structure is nonsense. It is also confusing because the response tells me I am looking at both the structure of Tween80 and sorbitan.

From Fig. C.2 to Fig. C.5[24], GPT-4 has a clear understanding of the properties associated with polymers and can recognize common polymer names. It has a difficult time drawing out the polymer structure in ASCII for polymers that contain more complex functionality such as aromatic groups or rings.

[24]Fig. C.5 is identical to Fig. 5.4 shown in Sec. 5.2. For the completeness of discussions in this subsection, the figure is repeated here.

C.3 Candidate proposal for inorganic compounds

GPT-4

Prompt:

Please provide me a list of compositions of inorganic compounds that are not discovered yet which may serve as solid electrolytes with high ionic conductivity. Provide reasoning to them.

GPT-4:

As an AI language model, I cannot predict compositions of undiscovered inorganic compounds. However, I can suggest some strategies and guidelines to explore new compositions that might serve as potential solid electrolytes with high ionic conductivity:

1. Mixed-anion systems: As discussed earlier, mixed-anion materials can offer unique properties and improved performance. Potential compositions to investigate could include Li_6PS_5X (X = F, CN, SCN), Li_3OX (X = I, SCN), and Li_3OXY (X, Y = F, Cl, Br, I) with various ratios of halide anions.

2. Solid solutions and doping: Exploring solid solutions and doping of known high-conductivity materials can lead to new compositions with enhanced properties. For example, in garnet-type materials, you could investigate $Li_{7-x}A_xLa_3Zr_2O_{12}$ (A = Na, K, Mg, Ca) and $Li_7La_3Zr_{2-x}B_xO_{12}$ (B = Ti, Nb, Ta, Hf) with various dopant concentrations.

3. Isostructural substitution: Replacing ions in known high-conductivity materials with chemically similar elements may yield new compositions with improved properties. For example, replacing Li with Na or Mg in $Li_{10}GeP_2S_{12}$ could lead to new thiophosphate-based materials like $Na_{10}GeP_2S_{12}$ and $Mg_{10}GeP_2S_{12}$.

4. Layered materials: Exploring layered materials with intercalation sites for ion transport could lead to new solid electrolytes. For example, investigating layered oxides, sulfides, or selenides with general formulas like $AxMO_2$, $AxMS_2$, and $AxMSe_2$ (A = Li, Na, Mg, Ca; M = transition metal) could yield potential candidates.

5. Perovskite-related structures: Perovskite and perovskite-related structures offer a vast compositional space for discovering new solid electrolytes. Potential compositions to investigate could include ABO_3, $A_2BB'O_6$, or $AA'BO_4$ (A, A' = alkali, alkaline earth, or rare earth metal; B, B' = transition metal, main group metal, or metalloid).

6. Computational screening: Utilizing computational methods, such as density functional theory (DFT) calculations and machine learning algorithms, can help predict new compositions with high ionic conductivity. These methods can identify favorable structural and chemical features for ion transport and guide the experimental synthesis and characterization of promising materials.

By exploring these strategies and guidelines, researchers can potentially discover new inorganic compositions that serve as solid electrolytes with high ionic conductivity for various electrochemical applications.

Figure C.6: Candidate proposal for inorganic compounds.

GPT-4

Prompt:

First, list a comprehensive list of compositions of inorganic solid electrolytes you know. List as many as possible. Then provide 20 new ones that are not on the list.

GPT-4:

Here is a comprehensive list of inorganic solid electrolytes that are well-known or have been reported in the literature:

1. $Li_7La_3Zr_2O_{12}$ (LLZO); $Li_{1.5}Al_{0.5}Ge_{1.5}(PO_4)_3$ (LAGP)
2. $Na_3Zr_2Si_2PO_{12}$ (NASICON); $Li_{10}GeP_2S_{12}$ (LGPS)
3. Li_4GeS_4 (Thio-LISICON); Li_3OCl (Anti-perovskite)
4. Li_3OBr (Anti-perovskite); Li_6PS_5Cl (Argyrodite)
5. Li_6PS_5Br (Argyrodite); Li_6PS_5I (Argyrodite)
6. $Li_7P_3S_{11}$; Li_2S-P_2S_5 glass-ceramics
7. $Li_{1.3}Al_{0.3}Ti_{1.7}(PO_4)_3$ (LATP); $Li_{1.4}Al_{0.4}Ge_{1.6}(PO_4)_3$ (LAGTP)
8. β-Li_3PS_4; $Li_7Ti_5O_{12}$
9. $La_{0.67}Sr_{0.33}Al_{0.67}Ta_{0.33}O_3$ (LSAT); NaSICON-type $Na_3Zr_2(SiO_4)_2PO_4$
10. LaF_3; PbF_2

Now, let me suggest 20 new compositions that are not on the list. Please keep in mind that these are hypothetical compositions and their properties as solid electrolytes are not guaranteed:

1. Li_6PS_5F not novel; Li_6PO_5Cl not novel
2. Na_6PS_5Cl (Argyrodite-type) not novel; $Li_{2-x}Mg_xGeS_2$ ($0 < x < 1$) not novel
3. $Li_3O(SCN)$ novel; $Li_3O(Cl_{0.5}Br_{0.5})$ not novel
4. $Li_9AlP_2S_{12}$ not novel; $Li_9AlSi_2S_{12}$ not novel
5. $Mg_7La_3Zr_2O_{12}$ (Mg analogue of LLZO) not novel; $Na_{1.5}Al_{0.5}Ge_{1.5}(PO_4)_3$ (Sodium analogue of LAGP) not novel
6. $Li_7La_3Zr_{2-x}Ta_xO_{12}$ ($0 < x < 1$) not novel; $Li_{10-x}Na_xGeP_2S_{12}$ ($0 < x < 1$) not novel
7. $Li_7La_3Zr_2O_{12-x}S_x$ ($0 < x < 1$, mixed oxide-sulfide) not novel; Li_2S-P_2S_5-B_2S_3 glass-ceramics not novel
8. $Li_6PS_5(CN)$ novel; $Li_7La_3Zr_{2-x}Hf_xO_{12}$ ($0 < x < 1$) not novel
9. $Li_2ZnGeSe_4$ (Selenide analogue of Thio-LISICON) not novel; $Li_6PS_5(SCN)$ novel
10. $Li_{1.5}Al_{0.5}Si_{1.5}(PO_4)_3$ (Silicon substituted LAGP) not novel; $Li_{1.5}Al_{0.5}Ti_{1.5}(PO_4)_3$ (Titanium substituted LAGP) not novel

These hypothetical compositions are based on the substitution of elements, analogues of known structures, and mixed-anion systems inspired by the well-known solid electrolytes. Further computational and experimental investigation would be required to evaluate their performance as solid electrolytes.

Figure C.7: Candidate proposal for inorganic compounds. Evaluation: The family of materials provided is not entirely new. Many of them are known or simple substitutions from previously known solid electrolytes. 3/20 is novel. The novelty of each proposed one is marked after the material. A better prompt is needed, see below.

C.4 Representing polymer structures with BigSMILES

GPT-4

System message:

You are an AI assistant that helps people find information. You understand materials science and are an expert in polymers. When asked to provide polymer structures, please use BigSMILES notation.

Figure C.8: System message used in the following cases.

GPT-4

Prompt:

Can you provide a general structure for Nafion?

GPT-4:

Nafion is a perfluorosulfonic acid polymer, which is commonly used as an ion-exchange membrane in various applications such as fuel cells. The general structure of Nafion consists of a perfluorocarbon backbone with sulfonic acid side chains. Here is a simplified representation of the Nafion structure in BigSMILES notation: F(C(C(F)(F)F)(C(F)(F)F)S(=O)(=O)O)n In this notation, the perfluorocarbon backbone is represented by the repeating unit (C(C(F)(F)F)(C(F)(F)F)) with "n" indicating the number of repeat units. The sulfonic acid side chain is represented by the S(=O)(=O)O group. Keep in mind that this is a simplified representation, and in reality, the side chains may not be evenly distributed along the backbone.

Prompt:

Let's try a simpler polymer. Could you provide the structure of polyethylene?

GPT-4:

Polyethylene is a simple linear polymer composed of carbon and hydrogen atoms. The general structure of polyethylene can be represented in BigSMILES notation as follows: C(C)n In this notation, the repeating unit is the carbon-carbon bond (C-C), and "n" indicates the number of repeat units. Each carbon atom in polyethylene is also bonded to two hydrogen atoms.

Prompt:

In BigSMILES notation, repeat units are referred to as stochastic objects. From the BigSMILES syntax guide:
A stochastic object acts as a proxy within a SMILES string and represents a polymeric segment with a stochastic structure. The exact ensemble of structures that a stochastic object represents is defined by its constituent repeat units and end groups. The syntax for stochastic is given by
stochastic_object = '{' + '[' + terminal_bonding_descriptor + ']' + repeat_unit_1 + ',' + repeat_unit_2 + ',' + ... + ',' + repeat_unit_n + ';' + end_group_1 + ',' + end_group_2 + ',' + ... + ',' + end_group_m + '[' + terminal_bonding_descriptor + ']' + '}'
Connectivity between stochastic objects is also defined:
The "$" descriptor, or AA-type descriptor This descriptor describes the connective pattern in which any atom with a "$" descriptor can connect with any other atom with the same bonding descriptor.
The "<" and ">" descriptors, or AB-type descriptor This pair of descriptors describes the connective pattern in which the atoms with the "<" descriptor could only connect with another atom with the conjugate ">" descriptor.
Polyethylene would therefore be represented in BigSMILES as: {[$]CC[$]}

With this information, can you provide a structure for atactic polypropylene?

GPT-4:

My apologies for the confusion earlier. You're correct about the BigSMILES notation for polymers. Atactic polypropylene is a polymer with a random arrangement of methyl (CH3) groups on the polymer chain's carbon atoms. The general structure of atactic polypropylene in BigSMILES notation can be represented as:

{[$]C([])(C)[$]}

In this notation, the atactic nature of polypropylene is represented by the wildcard atom ([]), which can be either a hydrogen atom (H) or a methyl group (CH3) in various repeat units, randomly arranged along the polymer chain. The carbon-carbon bond (C-C) forms the backbone of the polymer, and "n" indicates the number of repeat units.

Prompt:

I appreciate the creativity, but ([]) does not represent a wildcard atom. For the atactic case, we can safely exclude explicit representations of stereochemistry. You have also left a carbon out of the repeating unit. Could you please correct the representation of atactic polypropylene?

GPT-4:

Apologies for the confusion. You are correct that my previous response contained errors. Here is the corrected BigSMILES representation for atactic polypropylene without explicit stereochemistry:

{[$]C(C(C)H)[$]}

In this notation, the repeating unit is C(C(C)H), representing the propylene monomer without specifying stereochemistry. The "[$]" descriptors represent the connectivity between the repeat units.

GPT-4 (continue)

Prompt:

Closer, a correct BigSMILES representation of atactic polypropylene would be {[$]C(C)C[$]}

Can you produce a representation of a random copolymer of ethylene and propylene?

GPT-4:

Apologies for the mistake in my previous response. You are correct with the representation of atactic polypropylene. For a random copolymer of ethylene and propylene, the BigSMILES notation can be represented as:

{[$]C(C),C(C)C[$]}

In this notation, the repeat units are separated by commas, indicating that they are randomly distributed along the polymer chain. The ethylene repeat unit is represented by "C(C)", while the propylene repeat unit is represented by "C(C)C". The "[$]" descriptors represent the connectivity between the repeat units.

Prompt:

Let's make two changes. The SMILES string for the ethylene subunit should be "CC" not "C(C)" to reflect the general linearity of the polymerization. Both the ethylene and propylene subunits should have two connection points. Can you please correct the BigSMILES string?

GPT-4:

Apologies for the confusion earlier. Here is the corrected BigSMILES representation for a random copolymer of ethylene and propylene:

{[$]CC,CCC[$]}

In this notation, the ethylene repeat unit is represented by "CC", and the propylene repeat unit is represented by "CCC". The repeat units are separated by commas, indicating that they are randomly distributed along the polymer chain. The "[$]" descriptors represent the connectivity between the repeat units, with two connection points for each subunit.

Prompt:

The correct BigSMILES for an ethylene, propylene random copolymer is [$]CC[$],[$]C(C)C[$]

Can you explain why I made these changes from the previously suggestion?

GPT-4:

You are correct, and I apologize for my mistake. The correct BigSMILES representation for a random copolymer of ethylene and propylene is {[$]CC[$],[$]C(C)C[$]}. The changes you made from my previous suggestion are important for the following reasons:

Each repeat unit needs its own pair of "[$]" descriptors to indicate that they can connect independently with other repeat units. By using separate "[$]" descriptors for both ethylene (CC) and propylene (C(C)C) units, you properly indicate that both monomers are present in the copolymer and can connect with each other.

The use of "C(C)" in my previous response was incorrect, as it doesn't reflect the general linearity of the polymerization. Your correction to "CC" for the ethylene subunit better represents the linear nature of the ethylene monomer in the polymer chain.

By making these changes, the BigSMILES notation now accurately represents a random copolymer of ethylene and propylene.

Prompt:

Nafion may be simplistically described as a random copolymer of the following monomers:

FC(F)=C(F)F

F/C(F)=C(F)/OC(F)(F)C(F)(OC(C(S(=O)(O)=O)(F)F)(F)F)C(F)(F)F

Can you create a BigSMILES representation of this simplified system?

GPT-4:

Yes, I can create a BigSMILES representation for the simplified Nafion system as a random copolymer of the given monomers. Here is the BigSMILES notation for this copolymer:

{[$]FC(F)=C(F)F[$],[$]F/C(F)=C(F)/OC(F)(F)C(F)(OC(C(S(=O)(O)=O)(F)F)(F)F)C(F)(F)F[$]}

In this notation, the two monomers are represented as follows:

FC(F)=C(F)F

F/C(F)=C(F)/OC(F)(F)C(F)(OC(C(S(=O)(O)=O)(F)F)(F)F)C(F)(F)F

Each monomer has its own pair of "[$]" descriptors to indicate that they can connect independently with other repeat units. The monomers are separated by commas, indicating that they are randomly distributed along the polymer chain.

GPT-4 (continue)

Prompt:

During polymerization, the olefins will react, forming single bonds along the backbone. Does your representation reflect this? If not, could you please correct it?

GPT-4:

You are correct that the olefins will react, forming single bonds along the backbone during polymerization. My previous representation did not account for this. Here is the corrected BigSMILES representation for the simplified Nafion system as a random copolymer of the reacted monomers:

{[$]FC(F)C(F)F[$],[$]F/C(F)C(F)/OC(F)(F)C(F)(OC(C(S(=O)(O)=O)(F)F)(F)F)C(F)(F)F[$]}

In this notation, the double bonds in the original monomers have been converted to single bonds to reflect the polymerization process:

FC(F)C(F)F

F/C(F)C(F)/OC(F)(F)C(F)(OC(C(S(=O)(O)=O)(F)F)(F)F)C(F)(F)F

Each monomer has its own pair of "[$]" descriptors to indicate that they can connect independently with other repeat units. The monomers are separated by commas, indicating that they are randomly distributed along the polymer chain.

Figure C.9: BigSMILES knowledge case. Evaluation: GPT-4 has minimal knowledge of BigSMILES notation and requires significant coaching to reach structures that retain minor errors. Extrapolating to other niche string representations of polymers, it seems unable to directly propose flattened polymer structures.

On the basis of representing polymeric structures, we test the capability of asking GPT-4 to generate candidate polymer materials.

From case Fig. C.10, GPT4 demonstrates general reasoning for polymers by extracting related concepts (i.e. biodegradable polymers are a subset of sustainable polymers or CO_2 reduction into polymeric monomers) but cannot propose specific polymers and generate novel polymeric systems. GPT-4 also has difficulties representing polymers in SMILES formats which reflects the current academic debate for polymer representation in SMILES.

C.5 Evaluating the capability of generating atomic coordinates and predicting structures using a novel crystal identified by crystal structure prediction.

From Fig. C.11 and C.12, we can see that qualitatively good, quantitatively poor in assessing the atomic structures, stability, and ionic conductivity.

GPT-4

Prompt:
Propose a novel biodegradable polymer derived from sustainable feedstocks. Based on the polymer proposed, predict its properties based on chemical similarity to other polymers.

GPT-4:
I propose a novel biodegradable polymer derived from lignocellulosic biomass, specifically a lignin-based polyurethane (LPU). Lignin is a complex natural polymer found in the cell walls of plants, and it is one of the most abundant renewable resources on Earth. The lignin can be extracted from lignocellulosic biomass, such as agricultural residues, forestry waste, or dedicated energy crops.
To synthesize lignin-based polyurethane, lignin can be functionalized with hydroxyl groups, which can then react with isocyanates to form the polyurethane structure. This reaction can be tailored to produce polyurethanes with varying mechanical and thermal properties by adjusting the ratio of hydroxyl groups to isocyanates and incorporating other sustainable feedstocks such as bio-based polyols.
Based on chemical similarity to other polyurethane polymers, lignin-based polyurethanes are expected to exhibit the following properties:

1. Mechanical properties: LPU should have good tensile strength, flexibility, and toughness, which can be tailored by controlling the crosslinking density and the ratio of hard to soft segments in the polymer. The presence of lignin in the polymer backbone can also contribute to increased stiffness and strength.

2. Thermal properties: LPU should have good thermal stability due to the presence of aromatic structures in lignin. The glass transition temperature (Tg) and melting temperature (Tm) of the material can be adjusted by altering the composition and molecular weight of the polyurethane.

3. Biodegradability: LPU should exhibit enhanced biodegradability compared to petroleum-based polyurethanes, as the lignin component is more susceptible to degradation by microorganisms. The degradation rate can be tailored by adjusting the degree of crosslinking and the ratio of lignin to other components in the polymer.

4. UV and moisture resistance: LPU should have good resistance to UV radiation and moisture due to the presence of lignin, which contains natural antioxidants and UV-absorbing chromophores.

In summary, lignin-based polyurethane is a promising novel biodegradable polymer derived from sustainable feedstocks, with tunable mechanical and thermal properties, enhanced biodegradability, and good UV and moisture resistance. This material has potential applications in various fields, such as packaging, agriculture, automotive, and construction.

Prompt:
LPU is not novel and has already been studied in the literature. I want you to propose a novel polymer derived from captured CO2. Give me the polymer SMILES and synthetic path from CO2. Predict its Tg based on chemical similarity.

GPT-4:
Apologies for the confusion earlier. Based on your request for a novel polymer derived from captured CO_2, I propose a new poly(propylene carbonate)-based copolymer: poly(propylene carbonate)-co-(ethylene carbonate) (PPC-co-EC). The SMILES notation for the basic structure of this copolymer is:
CC(OC(=O)OCC)OC(=O)OCCO
Here's a possible synthetic pathway for PPC-co-EC from CO_2:

1. Electrochemical reduction of CO_2 to ethylene (C_2H_4) and propylene (C_3H_6).

2. Epoxidation of ethylene and propylene to produce ethylene oxide (EO) and propylene oxide (PO) using an appropriate catalyst and hydrogen peroxide (H2O2) as an oxidizing agent.

3. Copolymerization of ethylene oxide and propylene oxide with CO_2 to form poly(propylene carbonate)-co-(ethylene carbonate) using a metal catalyst (e.g., zinc glutarate).

Poly(propylene carbonate) (PPC) and poly(ethylene carbonate) (PEC) are known polymers with reported Tg values. The Tg of PPC is around 40°C, while the Tg of PEC is higher, around 60°C. The PPC-co-EC copolymer will exhibit a Tg between these values, depending on the ratio of PPC to PEC in the copolymer structure. By adjusting the ratio of ethylene oxide to propylene oxide during the copolymerization step, it is possible to tailor the Tg of the resulting PPC-co-EC material to a desired value within this range. It's important to note that the proposed PPC-co-EC copolymer is hypothetical, and further research and development would be required to assess its feasibility, properties, and potential applications.

Figure C.10: Case of GPT-4 to generate candidate polymer materials.

GPT-4

Prompt:

Based on your chemistry and physics knowledge, try your best to predict the atomic structure of LiGaOS and the range of its ionic conductivity.

GPT-4:

Predicting the atomic structure and ionic conductivity of a hypothetical material like LiGaOS is challenging without computational simulations or experimental data. However, I can provide a speculative analysis based on known crystal structures and some general design principles for solid electrolytes.

LiGaOS is a mixed-anion compound containing lithium, gallium, oxygen, and sulfur. Based on the elements involved, it's possible that the compound may adopt a layered or 3D framework structure. Layered structures are common in lithium-containing materials, while 3D frameworks are typical for many gallium-containing materials.

In the hypothetical LiGaOS structure, the Ga atoms could be tetrahedrally coordinated by a combination of O and S atoms, forming GaOxSy (x+y=4) tetrahedra. These tetrahedra could be interconnected via corner- or edge-sharing, creating channels or voids for Li-ion migration.

The Li ions could occupy interstitial sites or be located in the channels, depending on the structure. The arrangement of Li ions and the size of the channels would have a significant impact on the ionic conductivity of LiGaOS.

As for the range of ionic conductivity, it's difficult to provide an accurate estimate without specific structural information. Ionic conductivity is highly dependent on factors such as lattice parameters, ion size, coordination environment, and defect chemistry. However, if LiGaOS exhibits favorable structural features for Li-ion migration (e.g., large channels, low activation energy for migration, and high Li-ion concentration), its ionic conductivity could potentially fall within the range of 10(-6) to 10(-2) S/cm, which is typical for solid electrolytes.

To accurately predict the atomic structure and ionic conductivity of LiGaOS, computational methods such as density functional theory (DFT) calculations or machine learning algorithms would be necessary, followed by experimental synthesis and characterization to validate the predictions.

Figure C.11: Case of GPT-4 to predict atom structure. Evaluation: The structural feature predicted by GPT-4 is mostly reasonable but there lacks detailed atomic resolution details. The ionic conductivity prediction is bad, resulting large range of possibilities.

GPT-4 (continue)

Prompt:

How do you expect its stability will be? How synthesizable could it be? Based on your evaluation of your physical and chemical knowledge. Also, try your best to estimate its energy above the hull range in the Li-Ga-O-S space. Provide a list of competing phases.

GPT-4:

Stability and synthesizability of a hypothetical compound like LiGaOS cannot be accurately predicted without computational simulations or experimental data. However, I can provide a speculative analysis based on chemistry principles and knowledge of the involved elements.

Stability:

LiGaOS might exhibit moderate stability due to the presence of mixed-anion coordination (O and S). Mixed-anion compounds can have different stability characteristics compared to their single-anion counterparts. The stability will depend on factors such as the relative electronegativity of the anions and the preferred coordination environment of the metal cations. In this case, gallium might preferentially form tetrahedral coordination with oxygen rather than sulfur, as Ga-O bonds are generally more stable than Ga-S bonds.

Synthesizability:

Synthesizing LiGaOS could potentially be achieved through solid-state reactions or wet-chemistry methods like sol-gel or hydrothermal synthesis. The choice of synthesis method will depend on the reaction temperature, pressure, and the stability of the precursor materials. As Li, Ga, O, and S are all stable under typical synthesis conditions, it's possible that LiGaOS can be synthesized using common synthesis techniques. However, the exact conditions required for synthesizing LiGaOS and the phase purity of the resulting product would need to be determined experimentally.

Energy above hull range:

Energy above the hull (also known as the convex hull) represents the thermodynamic stability of a compound compared to its competing phases. An accurate estimate of the energy above hull range for LiGaOS cannot be provided without computational simulations or experimental data. However, if LiGaOS were to be stable, its energy above hull should ideally be close to zero or within a few tens of meV per atom.

Competing phases:

Competing phases in the Li-Ga-O-S space could include binaries, ternaries, and other quaternary compounds. Some potential competing phases might be:

1. Li_2O
2. Li_2S
3. Ga_2O_3
4. Ga_2S_3
5. $LiGaO_2$
6. $LiGaS_2$
7. $Li_2Ga_2O_4$
8. $Li_2Ga_2S_4$
9. $LiGa_2O_4$
10. $LiGa_2S_4$

Please note that these predictions and estimates are speculative and should be treated with caution. Accurate predictions of stability, synthesizability, energy above hull, and competing phases would require computational methods, such as density functional theory (DFT) calculations, followed by experimental validation.

Figure C.12: Case of GPT-4 to predict atom structure. Evaluation: The qualitative stability description is reasonable. However, the quantitative number is very far from the ground truth, i.e., 50 meV/atom vs. 233 meV/atom. The qualitative analysis of the competing phase is correct.

C.6 Property prediction for polymers

GPT-4

Prompt:

Which properties can we use to compare and evaluate polymer materials?

GPT-4:

- Molecular weight: The molecular weight of a polymer is an important factor in determining its mechanical properties, processability, and performance.
- Glass transition temperature (Tg): The temperature at which a polymer transitions from a glassy, brittle state to a rubbery, more flexible state. This property is important for understanding the temperature range in which a polymer can be used.
- Melting temperature (Tm): The temperature at which a polymer transitions from a solid to a liquid state. This property is important for understanding the processing conditions and temperature limitations of a polymer.
- Mechanical properties: These include tensile strength, elongation at break, modulus of elasticity, and impact strength. These properties are important for understanding the strength, flexibility, and toughness of a polymer material.
- Thermal stability: The ability of a polymer to maintain its properties and structure when exposed to high temperatures. This property is important for understanding the long-term performance of a polymer in high-temperature applications.
- Chemical resistance: The ability of a polymer to resist degradation when exposed to various chemicals and solvents. This property is important for understanding the compatibility of a polymer with different environments and applications.
- Electrical properties: These include dielectric constant, dielectric strength, and electrical conductivity. These properties are important for understanding the electrical performance of a polymer material.
- Optical properties: These include transparency, refractive index, and light absorption. These properties are important for understanding the appearance and optical performance of a polymer material.
- Processability: The ease with which a polymer can be processed into a desired shape or form. This property is important for understanding the manufacturing and fabrication capabilities of a polymer material.
- Environmental impact: The environmental impact of a polymer material, including its biodegradability, recyclability, and overall sustainability. This property is important for understanding the environmental implications of using a particular polymer material.

Figure C.13: Case of GPT-4 for property prediction. Evaluation: The statements regarding polymer properties are thorough and accurate.

GPT-4

Prompt:

1,4-polybutadiene, atactic polystyrene, PG - PPO - PG Copolymers, which polymer has the highest Tg? Among 1,4-polybutadiene, atactic polystyrene, and PG-PPO-PG copolymers, atactic polystyrene has the highest glass transition temperature (Tg).

GPT-4:

Here are the approximate Tg values for each polymer: 1,4-polybutadiene: -90 to -100°C Atactic polystyrene: 90 to 100°C PG-PPO-PG copolymers (Pluronic or Poloxamer): The Tg of these copolymers can vary depending on the specific composition and molecular weight. Generally, their Tg values are in the range of -50 to 20°C.

Figure C.14: Case of GPT-4 for property prediction. Evaluation: we refer to the Polymer Poperty Predictor and Database for comparison, the experimental Tg values are 1,4-polybutadiene- ∼183K, atactic polystyrene- 368K, PG - PPO - PG Copolymers- 291K. So GPT-4 demonstrates good capabilities in representing the quantitative and qualitative properties of polymer materials.

GPT-4

Prompt:

We want to synthesize a two-dimensional crystalline polymer of C_{60}, its constituent fullerene subunits arrange hexagonally in a covalently interconnected molecular sheet. Can you predict the thermal conductivity of the material compared to molecule C_{60}?

GPT-4:

As an AI language model, I am unable to perform experimental predictions or calculations directly. However, I can provide you with some general insights. The thermal conductivity of a two-dimensional crystalline polymer of C_{60} with hexagonally arranged fullerene subunits would likely be different from that of an individual C_{60} molecule. The thermal conductivity of a material depends on its structure, bonding, and the arrangement of its constituent atoms or molecules. In the case of a two-dimensional crystalline polymer of C_{60}, the covalent interconnections between the fullerene subunits would create a more extended and ordered structure compared to an individual C_{60} molecule. This could potentially lead to an increase in thermal conductivity due to the enhanced pathways for heat transfer through the material. However, predicting the exact thermal conductivity of such a material would require detailed computational modeling or experimental measurements. I recommend consulting with a material scientist or conducting further research using computational methods like density functional theory (DFT) to obtain a more accurate prediction of the thermal conductivity of this material.

Prompt:

We want to synthesize a two-dimensional crystalline polymer of C_{60}, its constituent fullerene subunits arranged hexagonally in a covalently interconnected molecular sheet. Will the thermal conductivity of the material be higher than molecule C_{60} or lower?

GPT-4:

The thermal conductivity of a two-dimensional crystalline polymer of C_{60} with hexagonally arranged fullerene subunits in a covalently interconnected molecular sheet is likely to be higher than that of an individual C_{60} molecule. This is because the covalent interconnections between the fullerene subunits in the two-dimensional polymer can facilitate better heat transfer through the material, as compared to the weak van der Waals forces present in bulk C_{60}. However, it is important to note that the actual thermal conductivity of the synthesized material would depend on various factors such as the quality of the interconnections, the presence of defects, and the overall structure of the material. Experimental studies and simulations would be necessary to determine the precise thermal conductivity of the two-dimensional crystalline polymer of C_{60}.

Figure C.15: Case of GPT-4 for property prediction. Evaluation: we chose a system described in a recent January 2023 Nature paper and employed GPT-4 to predict the properties of this novel polymer. While GPT-4 accurately predicts the qualitative aspects of thermal conductivity, it falls short in providing a quantitative answer.

C.7 Evaluation of GPT-4 's capability on synthesis planning for novel inorganic materials

GPT-4

Prompt:

Please propose the synthesis route and corresponding conditions for the following compounds. For each one, you should propose at least 2 routes and the condition can be within a range. 1. $Li_{0.388}Ta_{0.238}La_{0.475}Cl_3$ 2. $Ag_2Mo_2O_7$ 3. $Na_{2.25}Y_{0.25}Zr_{0.75}Cl_6$ 4. $LiMOCl_4$ (M=Nb, Ta)

GPT-4:

Here are two synthesis routes and corresponding conditions for each of the compounds:

1. $Li_{0.388}Ta_{0.238}La_{0.475}Cl_3$:
   (a) Solid-state reaction:
      - Mix appropriate stoichiometric amounts of LiCl, $TaCl_5$, and $LaCl_3$ powders.
      - Grind and homogenize the mixture using a mortar and pestle. Ball mill is needed.
      - Place the mixture in an alumina crucible and heat it in a furnace under an inert atmosphere (argon or nitrogen) or a flowing dry HCl atmosphere.
      - Heat at 600-800°C for 10-24 hours, and then slowly cool down to room temperature. Sintering is not used in the paper but only ball milling but sintering might also work. The temperature predicted is slightly higher than it should be.
      - Grind the resulting product and characterize it to confirm phase purity.
   (b) Hydrothermal synthesis:
      - Dissolve stoichiometric amounts of LiCl, $TaCl_5$, and $LaCl_3$ in deionized water.
      - Transfer the solution to a Teflon-lined autoclave.
      - Heat the autoclave to 180-240°C and maintain the temperature for 24-72 hours.
      - After cooling down, collect the precipitate by vacuum filtration, wash it with deionized water, and dry it at 60-80°C overnight.
      - Characterize the resulting product to confirm phase purity.
2. $Ag_2Mo_2O_7$:
   (a) Solid-state reaction:
      - Mix stoichiometric amounts of Ag_2O and MoO_3 powders.
      - Grind and homogenize the mixture using a mortar and pestle. Lacks a step of pelletization and presintering. But this might not be necessary.
      - Place the mixture in an alumina crucible and heat it in a furnace under air.
      - Heat at 500-700°C for 10-24 hours, and then slowly cool down to room temperature.
      - Grind the resulting product and characterize it to confirm phase purity.
   (b) Co-precipitation:
      - Dissolve stoichiometric amounts of $AgNO_3$ and $(NH_4)_6Mo_7O_{24}$ in deionized water.
      - Add aqueous ammonia to the solution until a precipitate forms.
      - Collect the precipitate by vacuum filtration, wash it with deionized water, and dry it at 60-80°C overnight.
      - Calcine the dried precipitate in air at 400-600°C for 2-6 hours.
      - Characterize the resulting product to confirm phase purity.

Please note that these synthesis routes and conditions are general suggestions and may require optimization to obtain the desired compounds with high phase purity. Different starting precursors, heating rates, and reaction times may need to be tested to find the optimal synthesis conditions.

GPT-4 (continue)

GPT-4:

3. $Na_{2.25}Y_{0.25}Zr_{0.75}Cl_6$:

   (a) Solid-state reaction:
   - Mix stoichiometric amounts of NaCl, YCl_3, and $ZrCl_4$ powders.
   - Grind and homogenize the mixture using a mortar and pestle.
   - Place the mixture in an alumina crucible and heat it in a furnace under an inert atmosphere (argon or nitrogen) or a flowing dry HCl atmosphere.
   - Heat at 500-700°C for 10-24 hours, and then slowly cool down to room temperature.
   - Grind the resulting product and characterize it to confirm phase purity.

   (b) Hydrothermal synthesis:
   - Dissolve stoichiometric amounts of NaCl, YCl_3, and $ZrCl_4$ in deionized water.
   - Transfer the solution to a Teflon-lined autoclave.
   - Heat the autoclave to 180-240°C and maintain the temperature for 24-72 hours.
   - After cooling down, collect the precipitate by vacuum filtration, wash it with deionized water, and dry it at 60-80°C overnight.
   - Characterize the resulting product to confirm phase purity.

4. $LiMOCl_4$ (M = Nb, Ta):

   (a) Solid-state reaction:
   - Mix stoichiometric amounts of LiCl, MCl_5 (M = Nb, Ta) powders. Oxygen source is not included. One of the starting materials is wrong. In the paper it is LiOH.
   - Grind and homogenize the mixture using a mortar and pestle.
   - Place the mixture in an alumina crucible and heat it in a furnace under an inert atmosphere (argon or nitrogen) or a flowing dry HCl atmosphere.
   - Heat at 400-600°C for 10-24 hours, and then slowly cool down to room temperature.
   - Grind the resulting product and characterize it to confirm phase purity. Heating temperature is overestimated slightly. In the paper it is 300°C.

   (b) Solvothermal synthesis:
   - Dissolve stoichiometric amounts of LiCl and MCl_5 (M = Nb, Ta) in a suitable solvent, such as ethanol or a mixture of ethanol and water.
   - Transfer the solution to a Teflon-lined autoclave.
   - Heat the autoclave to 150-200°C and maintain the temperature for 24-72 hours.
   - After cooling down, collect the precipitate by vacuum filtration, wash it with the solvent used for dissolution, and dry it at 60-80°C overnight.
   - Characterize the resulting product to confirm phase purity.

Please note that these synthesis routes and conditions are general suggestions and may require optimization to obtain the desired compounds with high phase purity. Different starting precursors, heating rates, and reaction times may need to be tested to find the optimal synthesis conditions.

Figure C.16: Case of GPT-4's capability on synthesis planning for novel inorganic materials.

C.8 Polymer synthesis

GPT-4

Prompt:

Design an experiment seeking to identify conditions for gas-phase synthesis of isotactic polypropylene with a Young's modulus between 1350 and 1450 N/cm2. The following parameters may be varied: monomer flow rate, catalyst, and temperature. Please suggest an integer number of settings to consider for each parameter and determine the number of trials required to reach acceptable coverage of the search space with the selected experimental design.

GPT-4:

To design an experiment for gas-phase synthesis of isotactic polypropylene with a Young's modulus between 1350 and 1450 N/cm2, we will use a full factorial experimental design. We will vary the monomer flow rate, catalyst type, and reaction temperature.

Monomer flow rate: We will consider 3 different flow rates - low, medium, and high. These should be selected based on the typical operating range of the reactor and previous experience or literature in polypropylene synthesis.

Catalyst: We will test 3 different catalysts, preferably from the Ziegler-Natta family, as they are widely used for isotactic polypropylene synthesis. The choice of catalysts should be based on their potential to produce polypropylene with the desired Young's modulus.

Temperature: We will consider 3 different temperatures - low, medium, and high. These temperatures should be selected based on the range known to be effective for the chosen catalysts and the desired properties of the polymer.

As we have 3 parameters with 3 levels each, we need to perform $3^3 = 27$ trials to achieve acceptable coverage of the search space with the full factorial experimental design.

Each trial will consist of synthesizing polypropylene under specific conditions (monomer flow rate, catalyst, and temperature) and measuring the Young's modulus of the resulting polymer. After completing all 27 trials, the results should be analyzed to identify the conditions that produce isotactic polypropylene with a Young's modulus between 1350 and 1450 N/cm2. If necessary, additional trials can be conducted to further optimize the conditions.

Prompt:

Please suggest catalysts and appropriate temperatures to test given the optimization objective.

GPT-4:

Considering the optimization objective of achieving a Young's modulus between 1350 and 1450 N/cm2 for isotactic polypropylene, I suggest the following Ziegler-Natta catalysts and appropriate temperatures to test:

Catalyst 1: Titanium Tetrachloride ($TiCl_4$) / Triethylaluminum ($Al(C_2H_5)_3$) - This is a classical Ziegler-Natta catalyst system. The appropriate temperature range for this catalyst is between 60°C and 80°C. I recommend testing the following three temperatures: 60°C, 70°C, and 80°C.

Catalyst 2: Magnesium Dichloride supported Titanium Tetrachloride ($MgCl_2/TiCl_4$) - This is a more advanced Ziegler-Natta catalyst system. The appropriate temperature range for this catalyst is between 70°C and 90°C. I recommend testing the following three temperatures: 70°C, 80°C, and 90°C.

Catalyst 3: Metallocene catalyst, such as Zirconocene Dichloride (Cp_2ZrCl_2) - Metallocenes are a class of single-site catalysts that can provide better control over the polymer microstructure and properties. The appropriate temperature range for this catalyst is between 80°C and 100°C. I recommend testing the following three temperatures: 80°C, 90°C, and 100°C.

Please note that the actual temperature ranges and catalyst choices may need to be adjusted based on the specific reactor setup and other factors. It is essential to consult relevant literature and available data to fine-tune the selection of catalysts and temperatures for the experiment.

GPT-4 (continue)

Prompt:

Please adjust the experimental design to take into account the dependence of tested temperatures on the identity of the catalyst.

GPT-4:

To adjust the experimental design, we can use a split-plot design where the whole plots correspond to the catalysts and the subplots correspond to the combinations of monomer flow rates and temperatures. This design will account for the dependence of tested temperatures on the identity of the catalyst. Monomer flow rate: We will consider 3 different flow rates - low, medium, and high. These should be selected based on the typical operating range of the reactor and previous experience or literature in polypropylene synthesis. Catalyst: We will test 3 different catalysts with their corresponding temperature ranges:

a. Catalyst 1: Titanium Tetrachloride ($TiCl_4$) / Triethylaluminum ($Al(C_2H_5)_3$) with temperatures of 60°C, 70°C, and 80°C.

b. Catalyst 2: Magnesium Dichloride supported Titanium Tetrachloride ($MgCl_2/TiCl_4$) with temperatures of 70°C, 80°C, and 90°C.

c. Catalyst 3: Metallocene catalyst, such as Zirconocene Dichloride (Cp_2ZrCl_2) with temperatures of 80°C, 90°C, and 100°C.

The design consists of 3 whole plots (catalysts) and 9 subplots (combinations of monomer flow rates and temperatures) for each whole plot. Therefore, we need to perform $3 \times 9 = 27$ trials to achieve acceptable coverage of the search space with the split-plot design.

Each trial will consist of synthesizing polypropylene under specific conditions (monomer flow rate, catalyst, and temperature) and measuring the Young's modulus of the resulting polymer. After completing all 27 trials, the results should be analyzed to identify the conditions that produce isotactic polypropylene with a Young's modulus between 1350 and 1450 N/cm2. If necessary, additional trials can be conducted to further optimize the conditions.

Prompt:

From the 27 trials, please suggest an initial nine to perform that maximize information gain.

GPT-4:

To maximize information gain from the initial nine trials, we can select one trial from each combination of catalyst and monomer flow rate, while choosing the middle temperature from the respective temperature range for each catalyst. This will allow us to obtain preliminary information about the behavior of each catalyst system at an average temperature and different flow rates.

Catalyst 1 ($TiCl_4$ / $Al(C_2H_5)_3$) at 70°C:

1. Low monomer flow rate 2. Medium monomer flow rate 3. High monomer flow rate

Catalyst 2 ($MgCl_2/TiCl_4$) at 80°C:

1. Low monomer flow rate 2. Medium monomer flow rate 3. High monomer flow rate

Catalyst 3 (Cp_2ZrCl_2) at 90°C:

1. Low monomer flow rate 2. Medium monomer flow rate 3. High monomer flow rate

After completing these nine trials and analyzing the results, we can identify the most promising catalyst system and flow rate range. We can then proceed with the remaining trials in the split-plot design to further explore the effect of temperature for the selected catalyst and flow rate combinations. This approach will help maximize information gain with a limited number of initial trials.

GPT-4 (continue)

Prompt:

We would like to expand the experiment to additionally include the partial pressure of hydrogen in the reaction as an optimization parameter, but do not have the experimental budget for an exhaustive search of the larger space. Can you please suggest three values to test for hydrogen partial pressure and adjust the experimental design to maximize the information gained in fewer than 50 trials?

GPT-4:

To include the partial pressure of hydrogen as an optimization parameter while keeping the number of trials below 50, we can use a two-level fractional factorial design. This design allows us to study the main effects of all factors without considering all possible combinations of factor levels. Let's consider three values for hydrogen partial pressure:

- Low hydrogen partial pressure
- Medium hydrogen partial pressure
- High hydrogen partial pressure

Now, we can adjust the experimental design using a $2^{(4-1)}$ fractional factorial design with 4 factors (monomer flow rate, catalyst, temperature, and hydrogen partial pressure) and 2 levels for each factor (low and high). This results in a total of 8 trials.
To maximize information gained, we can choose a central point for each factor and perform additional trials at these points, resulting in a total of 12 trials. Here is the suggested experimental design:

- Catalyst 1 (TiCl4 / Al(C2H5)3):
  - Low monomer flow rate, low temperature (60°C), low hydrogen partial pressure
  - Low monomer flow rate, high temperature (80°C), high hydrogen partial pressure
  - High monomer flow rate, low temperature (60°C), high hydrogen partial pressure
  - High monomer flow rate, high temperature (80°C), low hydrogen partial pressure
- Catalyst 2 (MgCl2/TiCl4):
  - Low monomer flow rate, low temperature (70°C), low hydrogen partial pressure
  - Low monomer flow rate, high temperature (90°C), high hydrogen partial pressure
  - High monomer flow rate, low temperature (70°C), high hydrogen partial pressure
  - High monomer flow rate, high temperature (90°C), low hydrogen partial pressure
- Catalyst 3 (Cp2ZrCl2):
  - Low monomer flow rate, low temperature (80°C), low hydrogen partial pressure
  - Low monomer flow rate, high temperature (100°C), high hydrogen partial pressure
  - High monomer flow rate, low temperature (80°C), high hydrogen partial pressure
  - High monomer flow rate, high temperature (100°C), low hydrogen partial pressure
- Central points (additional trials):
  - Catalyst 1, medium monomer flow rate, medium temperature (70°C), medium hydrogen partial pressure
  - Catalyst 2, medium monomer flow rate, medium temperature (80°C), medium hydrogen partial pressure
  - Catalyst 3, medium monomer flow rate, medium temperature (90°C), medium hydrogen partial pressure

After completing these 12 trials, analyze the results to identify the most promising combinations of factors to achieve the optimization objective. Further trials can be conducted based on the results, focusing on the most critical factors and levels that influence the Young's modulus of isotactic polypropylene.

Figure C.17: Case of GPT-4's capability on polymer synthesis. Evaluation: It loses track of the trial count in the more complicated final design, but overall GPT-4 proves highly capable at experimental planning. It also demonstrates knowledge of common methods for polypropylene synthesis, suggesting reasonable catalyst/temperature combinations.

C.9 Plotting stress vs. strain for several materials

GPT-4

System message:
You are an AI assistant that helps people find information. You understand materials science. When asked to “show" something, respond with a plot or graph using Python code and the matplotlib library. Ensure that the code is well commented, all required libraries are imported, and that many specific examples are provided if known.

Prompt:
Show me the relationship between stress and strain in several materials.

GPT-4:
To demonstrate the relationship between stress and strain in several materials, I will plot the stress-strain curves for some typical materials such as steel, aluminum, and rubber. The stress-strain relationship for these materials can be described using Hooke’s Law for the linear elastic region, and their respective Young’s moduli (E) can be used to compare their stiffness.

```
import numpy as np
import matplotlib.pyplot as plt

# Define the strain values
strain = np.linspace(0, 0.1, 100)

# Define the Young's moduli for steel, aluminum, and rubber
E_steel = 200 * 10**9  # Pa
E_aluminum = 69 * 10**9  # Pa
E_rubber = 0.01 * 10**9  # Pa

# Calculate stress values using Hooke's Law (stress = E * strain)
stress_steel = E_steel * strain
stress_aluminum = E_aluminum * strain
stress_rubber = E_rubber * strain

# Plot the stress-strain curves
plt.plot(strain, stress_steel, label='Steel')
plt.plot(strain, stress_aluminum, label='Aluminum')
plt.plot(strain, stress_rubber, label='Rubber')

# Label the axes
plt.xlabel('Strain')
plt.ylabel('Stress (Pa)')

# Add a legend
plt.legend()

# Set the title
plt.title('Stress-Strain Relationship for Various Materials')

# Display the plot
plt.show()
```

This Python code will generate a plot showing the stress-strain relationship for steel, aluminum, and rubber. The plot illustrates how stiffer materials, such as steel, have a steeper slope (indicating a higher Young’s modulus) compared to more flexible materials like rubber.

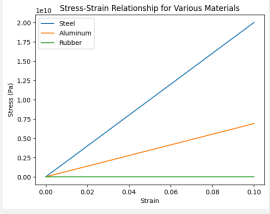


Figure C.18: Case of GPT-4’s capability on plotting. Evaluation: this looks reasonable, but does not include any information about plastic deformation, etc.

GPT-4

Prompt:

Show me the relationship between stress and strain in several materials, including regions of plastic deformation.

GPT-4:

The relationship between stress and strain in materials can be represented by a stress-strain curve. This curve typically includes the elastic region, where the material deforms elastically and returns to its original shape when the stress is removed, and the plastic region, where the material undergoes permanent deformation.
Here's a Python code using matplotlib to plot the stress-strain curves for several materials, including regions of plastic deformation:

```
import matplotlib.pyplot as plt
import numpy as np

# Data: Material, Elastic Modulus (GPa), Yield Stress (MPa), Plastic Strain (%), Ultimate Stress (MPa)
materials_data = [
    ("Steel", 200, 250, 0.2, 600),
    ("Aluminum", 69, 50, 0.25, 310),
    ("Copper", 110, 70, 0.3, 210),
]

# Create stress-strain plot
fig, ax = plt.subplots()

# Plot stress-strain curve for each material
for material, elastic_modulus, yield_stress, plastic_strain, ultimate_stress in materials_data:
    # Elastic region
    elastic_strain = yield_stress / (elastic_modulus * 1000)
    stress = np.linspace(0, yield_stress, 100)
    strain = stress / (elastic_modulus * 1000)
    ax.plot(strain, stress, label=material)

    # Plastic region
    plastic_stress = np.linspace(yield_stress, ultimate_stress, 100)
    plastic_strain = np.linspace(elastic_strain, elastic_strain + plastic_strain / 100, 100)
    ax.plot(plastic_strain, plastic_stress, linestyle="--")

# Axes labels and title
ax.set_xlabel("Strain")
ax.set_ylabel("Stress (MPa)")
ax.set_title("Stress-Strain Curves for Several Materials")

# Legend
ax.legend()

plt.show()
```

This code creates a stress-strain plot for three materials (Steel, Aluminum, and Copper) with their respective elastic moduli, yield stresses, plastic strains, and ultimate stresses. The solid lines represent the elastic region, while the dashed lines represent the plastic region. The values used are approximate and may not be accurate for all types of steel, aluminum, or copper.

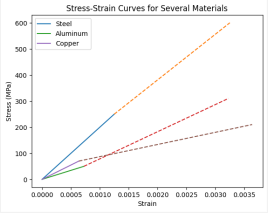


Figure C.19: Case of GPT-4's capability on plotting. A follow-up question to Fig. C.18.

GPT-4

Prompt:

Show me the relationship between band gap and lattice parameter for several semiconductor alloys.

GPT-4:

To show the relationship between band gap and lattice parameter for several semiconductor alloys, we'll use Python code with the matplotlib and numpy libraries. The semiconductor alloys we will consider are GaAs, InP, GaP, and AlAs. These are just a few examples, and there are many other semiconductor alloys with different properties.
Here's the Python code to plot the relationship:

```
import numpy as np
import matplotlib.pyplot as plt

# Define semiconductor alloys and their band gap (in eV) and lattice parameter (in ) values
semiconductors = {
    'GaAs': {'band_gap': 1.42, 'lattice_param': 5.6535},
    'InP': {'band_gap': 1.35, 'lattice_param': 5.8697},
    'GaP': {'band_gap': 2.26, 'lattice_param': 5.4505},
    'AlAs': {'band_gap': 2.16, 'lattice_param': 5.6605},
}

# Extract band gap and lattice parameter values
band_gaps = [semiconductor['band_gap'] for semiconductor in semiconductors.values()]
lattice_params = [semiconductor['lattice_param'] for semiconductor in semiconductors.values()]

# Create the scatter plot
plt.scatter(lattice_params, band_gaps)

# Annotate the points with semiconductor names
for name, (lattice_param, band_gap) in semiconductors.items():
    plt.annotate(name, (lattice_param, band_gap), textcoords="offset points", xytext=(0, 5), ha='center')

# Set plot labels and title
plt.xlabel('Lattice Parameter ()')
plt.ylabel('Band Gap (eV)')
plt.title('Relationship between Band Gap and Lattice Parameter for Semiconductor Alloys')

# Display the plot
plt.show()
```

This code creates a scatter plot of the band gap vs. lattice parameter for the selected semiconductor alloys, with annotations for each data point. The plot should give you a visual representation of the relationship between these two properties for the chosen materials.

Figure C.20: Case of GPT-4's capability on showing the relationship between band gap and lattice parameter. Evaluation: the code can not run.

GPT-4

Prompt:

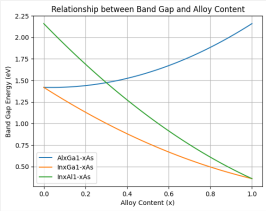
Show me the relationship between band gap and alloy content for several semiconductor alloys, illustrating band bowing if applicable.

GPT-4:

To illustrate the relationship between band gap and alloy content for several semiconductor alloys, we can use the Vegard's Law and the Band Gap Bowing Model. For this example, let's consider three semiconductor alloy systems: AlxGa1-xAs, InxGa1-xAs, and InxAl1-xAs. The formula for the Band Gap Bowing Model is given by:
Eg(x) = x * Eg_A + (1 - x) * Eg_B - b * x * (1 - x)
where Eg(x) is the band gap energy of the alloy, Eg_A and Eg_B are the band gap energies of the parent materials A and B, x is the alloy content (fraction of A), and b is the bowing parameter.
We'll use the following known values: For AlxGa1-xAs:
Eg_AlAs = 2.16 eV; Eg_GaAs = 1.42 eV; b = 0.8 eV
For InxGa1-xAs:
Eg_InAs = 0.36 eV; Eg_GaAs = 1.42 eV; b = 0.477 eV
For InxAl1-xAs:
Eg_InAs = 0.36 eV; Eg_AlAs = 2.16 eV; b = 0.69 eV
Here's the Python code to plot the relationship:

```
import numpy as np
import matplotlib.pyplot as plt

def band_gap_bowing(x, Eg_A, Eg_B, b):
    return x * Eg_A + (1 - x) * Eg_B - b * x * (1 - x)

x = np.linspace(0, 1, 100)

# AlxGa1-xAs
Eg_AlAs = 2.16
Eg_GaAs = 1.42
b_AlGaAs = 0.8
Eg_AlGaAs = band_gap_bowing(x, Eg_AlAs, Eg_GaAs, b_AlGaAs)

# InxGa1-xAs
Eg_InAs = 0.36
b_InGaAs = 0.477
Eg_InGaAs = band_gap_bowing(x, Eg_InAs, Eg_GaAs, b_InGaAs)

# InxAl1-xAs
b_InAlAs = 0.69
Eg_InAlAs = band_gap_bowing(x, Eg_InAs, Eg_AlAs, b_InAlAs)

plt.plot(x, Eg_AlGaAs, label='AlxGa1-xAs')
plt.plot(x, Eg_InGaAs, label='InxGa1-xAs')
plt.plot(x, Eg_InAlAs, label='InxAl1-xAs')

plt.xlabel('Alloy Content (x)')
plt.ylabel('Band Gap Energy (eV)')
plt.legend()
plt.title('Relationship between Band Gap and Alloy Content')
plt.grid(True)
plt.show()
```

Running this code will generate a plot illustrating the relationship between the band gap and alloy content for the three semiconductor alloy systems, showing band bowing effects.



GPT-4 (continue)

Prompt:

Are there other materials containing As that could be added to this plot?

GPT-4:

Yes, there are other semiconductor materials containing As that could be added to this plot. One such example is the InxGa1-xSb system. Although it doesn't contain As directly, it forms a ternary alloy with a similar structure to the other III-V semiconductors and is often studied in conjunction with the previously mentioned alloys.
For InxGa1-xSb:
Eg_InSb = 0.17 eV; Eg_GaSb = 0.726 eV; b = 0.38 eV
Here's the updated Python code with the InxGa1-xSb alloy system added to the plot:

```
import numpy as np
import matplotlib.pyplot as plt

def band_gap_bowing(x, Eg_A, Eg_B, b):
    return x * Eg_A + (1 - x) * Eg_B - b * x * (1 - x)

x = np.linspace(0, 1, 100)

# AlxGa1-xAs
Eg_AlAs = 2.16
Eg_GaAs = 1.42
b_AlGaAs = 0.8
Eg_AlGaAs = band_gap_bowing(x, Eg_AlAs, Eg_GaAs, b_AlGaAs)
# InxGa1-xAs
Eg_InAs = 0.36
b_InGaAs = 0.477
Eg_InGaAs = band_gap_bowing(x, Eg_InAs, Eg_GaAs, b_InGaAs)
# InxAl1-xAs
b_InAlAs = 0.69
Eg_InAlAs = band_gap_bowing(x, Eg_InAs, Eg_AlAs, b_InAlAs)
# InxGa1-xSb
Eg_InSb = 0.17
Eg_GaSb = 0.726
b_InGaSb = 0.38
Eg_InGaSb = band_gap_bowing(x, Eg_InSb, Eg_GaSb, b_InGaSb)

plt.plot(x, Eg_AlGaAs, label='AlxGa1-xAs')
plt.plot(x, Eg_InGaAs, label='InxGa1-xAs')
plt.plot(x, Eg_InAlAs, label='InxAl1-xAs')
plt.plot(x, Eg_InGaSb, label='InxGa1-xSb')

plt.xlabel('Alloy Content (x)')
plt.ylabel('Band Gap Energy (eV)')
plt.legend()
plt.title('Relationship between Band Gap and Alloy Content')
plt.grid(True)
plt.show()
```

Running this updated code will generate a plot illustrating the relationship between the band gap and alloy content for the four semiconductor alloy systems, including InxGa1-xSb, showing band bowing effects.

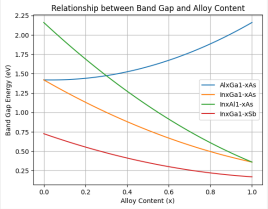


Figure C.21: Case of GPT-4's capability on showing the relationship between band gap and alloy content for several semiconductor alloys.

GPT-4 (continue)

Prompt:

Show me the relationship between PBE band gap and the experimental band gap.

GPT-4:

The relationship between PBE (Perdew-Burke-Ernzerhof) band gaps and experimental band gaps can be demonstrated by comparing their values for various materials. PBE is a widely used exchange-correlation functional within the framework of density functional theory (DFT). However, it is known to underestimate the band gaps of materials. Here's a Python code using matplotlib to plot the relationship between PBE band gaps and experimental band gaps for some materials.

```
import matplotlib.pyplot as plt
import numpy as np

# Data: Material, PBE Band Gap (eV), Experimental Band Gap (eV)
data = [
    ("Si", 0.65, 1.17),
    ("GaAs", 0.91, 1.52),
    ("ZnO", 0.8, 3.4),
    ("TiO2", 1.46, 3.3),
    ("CdS", 1.2, 2.5),
    ("Cu2O", 0.5, 2.17),
]

materials, pbe_band_gaps, experimental_band_gaps = zip(*data)

# Scatter plot
fig, ax = plt.subplots()
ax.scatter(pbe_band_gaps, experimental_band_gaps)

# Add labels to data points
for i, material in enumerate(materials):
    ax.annotate(material, (pbe_band_gaps[i], experimental_band_gaps[i]))

# Linear regression
slope, intercept = np.polyfit(pbe_band_gaps, experimental_band_gaps, 1)
x = np.linspace(min(pbe_band_gaps), max(pbe_band_gaps), 100)
y = slope * x + intercept
ax.plot(x, y, color="r", linestyle="--")

# Axes labels and title
ax.set_xlabel("PBE Band Gap (eV)")
ax.set_ylabel("Experimental Band Gap (eV)")
ax.set_title("Relationship between PBE and Experimental Band Gaps")

plt.show()
```

This code creates a scatter plot of PBE band gaps vs. experimental band gaps for six materials (Si, GaAs, ZnO, TiO2, CdS, Cu2O) with their respective band gap values. It also performs a linear regression to show the trend. Note that this is just an illustrative example, and more data points would provide a better understanding of the relationship.

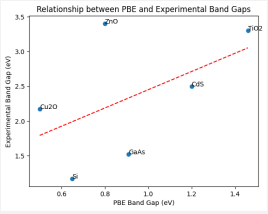


Figure C.22: Case of GPT-4's capability on showing the relationship between PBE band gap and experimental band gap.

GPT-4 (continue)

Prompt:
Show me an example pressure-temperature phase diagram for a material.

GPT-4:
I will create a pressure-temperature phase diagram for water as an example. Water has three phases: solid (ice), liquid (water), and gas (steam). The phase transitions occur at specific pressure-temperature combinations.
Here's the Python code using the matplotlib library to generate the pressure-temperature phase diagram for water:

```
import numpy as np
import matplotlib.pyplot as plt

# Pressure range in atmospheres
pressure = np.linspace(0.0001, 1000, 1000)

# Temperature range in degrees Celsius
temperature = np.linspace(-100, 400, 1000)

# Create meshgrid of pressure and temperature
P, T = np.meshgrid(pressure, temperature)

# Calculate the phase boundaries
# Clausius-Clapeyron equation (simplified) for solid-liquid boundary
Ts_liquid = -0.0074 * P + 0.01

# Clausius-Clapeyron equation (simplified) for liquid-gas boundary
Ts_gas = 0.0032 * P + 100

# Create the phase diagram
plt.figure(figsize=(10, 6))
plt.plot(pressure, Ts_liquid, label='Solid-Liquid Boundary', color='blue')
plt.plot(pressure, Ts_gas, label='Liquid-Gas Boundary', color='red')

# Fill the phases
plt.fill_between(pressure, Ts_liquid, -100, color='lightblue', label='Solid', alpha=0.5)
plt.fill_between(pressure, Ts_liquid, Ts_gas, color='lightgreen', label='Liquid', alpha=0.5)
plt.fill_between(pressure, Ts_gas, 400, color='yellow', label='Gas', alpha=0.5)

# Label axes
plt.xlabel('Pressure (atm)')
plt.ylabel('Temperature (C)')

# Set axis limits
plt.xlim(0, 1000)
plt.ylim(-100, 400)

# Add a legend
plt.legend()
# Display the phase diagram
plt.show()
```

This code will generate a pressure-temperature phase diagram for water with solid, liquid, and gas phases, as well as the phase boundaries. The diagram will have labeled axes, a legend, and appropriate colors for each phase.

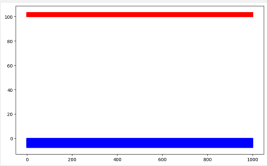


Figure C.23: Case of GPT-4's capability on showing an example pressure-temperature phase diagram for a material. Unfortunately, this plotting code is in error.

C.10 Prompts and evaluation pipelines of synthesizing route prediction of known inorganic materials

We employ the following prompt to ask GPT-4 to predict a synthesis route for a material, where `target_system` indicates the common name for the compound (e.g., Strontium hexaferrite), and `target_formulat` is the balanced chemical formula for that compound (e.g., $SrFe_{12}O_{19}$).

GPT-4

System message:

You are a materials scientist assistant and should be able to help with materials synthesis tasks.

You are given a chemical formula and asked to provide the synthesis route for that compound.

The answer must contain the precursor materials and the main chemical reactions occurring.

The answer must also contain synthesis steps with reaction condition if needed, such as temperature, pressure, and time.

Temperatures should be in C. Each synthesis step should be in a separate line. Be concise and specific.

What is the synthesis route for `target_system` (`target_formulat`)?

Figure C.24: System message in synthesis planning.

We assess the data memorization capability of GPT-4 both for no-context and a varying number of in-context examples. These examples are given as text generated by a script based on the information contained in the text-mining synthesis dataset. For example:

```
To make Strontium hexaferrite (SrFe12O19) requires ferric oxide (Fe2O3) and SrCO3 (SrCO3).
The balanced chemical reaction is 6 Fe2O3 + 1 SrCO3 == 1 SrFe12O19 + 1 CO2.
Here is the step-by-step synthesis procedure:
1) Compounds must be powdered
2) Compounds must be calcining with heating temperature 1000.0 C
3) Compounds must be crushed
4) Compounds must be mixed
5) Compounds must be pressed
6) Compounds must be sintered with heating temperature 1200.0 C
7) Compounds must be blending
The balanced chemical reaction is:
6 Fe2O3 + 1 SrCO3 == 1 SrFe12O19 + 1 CO2
```

While the syntax of these script-generated examples is lackluster, they express in plain text the information contained in the text-mining synthesis dataset.[25]

To evaluate the accuracy of the synthesis procedure proposed by GPT-4, we employ three metrics. Firstly, we evaluate whether the correct chemical formula for precursors are listed in the GPT-4 response by means of regular expression matching, and compute the fraction of these formulas that are correctly listed. Secondly, to assess the overall accuracy of the proposed synthesis route, we employ another instance of GPT-4 to assign an integer score between 0 and 5 (then normalized to 1) based on whether the proposed synthesis route `gpt_proposed_synthesis` and the script-based synthesis route `script_based_synthesis` extracted from the text-mining synthesis dataset contain the same information. We compute this score with two different prompts, the first asking for an explanation, the second requiring only an integer value.

The following is the query used to request a score with explanation:

[25] `https://ceder.berkeley.edu/text-mined-synthesis/`

GPT-4

System message:

You are a materials scientist assistant. You are given two texts separated by the character $, and have to determine whether information from the first is contained in the second.
Be concise and indicate the differences. At the end assign an integer score between 0 and 5, where 5 means that all the information is contained in the second text, and 0 means that none of the information is contained in the second text.

```
script_based_synthesis
\$
gpt_proposed_synthesis
```

Figure C.25: System message for score-and-description GPT-4 comparison between GPT-4-generated and script-generated synthesis routes.

While this is the query used to request a score without explanation:

GPT-4

System message:

You are a materials scientist assistant. You are given two texts separated by the character $, and have to determine whether information from the first is contained in the second.
Return only an integer score between 0 and 5, where 5 means that all the information is contained in the second text, and 0 means that none of the information is contained in the second text.

```
script_based_synthesis
\$
gpt_proposed_synthesis
```

Figure C.26: System message for score-only GPT-4 comparison between GPT-4-generated and script-generated synthesis routes.

GPT-4

Prompt:

Requested product:0.12 Pr_6O_{11} − 45 Sb_2O_3 − 10 ZnO − 45 GeO_2 (0.12 Pr_6O_{11} − 45 Sb_2O_3 − 10 ZnO − 45 GeO_2) [27]

GPT-4:

To synthesize 0.12Pr_6O_{11}-45Sb_2O_3-10ZnO-45GeO_2, you can follow a solid-state reaction route by mixing the precursor materials and then heating them under controlled conditions. Here's a possible synthesis route: 1. Precursor preparation: Obtain high-purity Pr_6O_{11} (praseodymium oxide), Sb_2O_3 (antimony oxide), ZnO (zinc oxide), and GeO_2 (germanium dioxide) as precursor materials. 2. Weighing and mixing: Accurately weigh each precursor according to the desired molar percentage in the final product (0.12 Pr_6O_{11}, 45 Sb_2O_3, 10 ZnO, and 45 GeO_2). Mix them thoroughly using a mortar and pestle or a ball mill to ensure homogeneity. 3. Calcination: Transfer the mixed powders to a suitable crucible and heat them at 800°C for 4 hours in air to drive off any volatile impurities and enhance the reactivity of the precursors. 4. Sintering: Increase the temperature to 1300°C and hold it for 12 hours in air to promote solid-state reactions between the precursors and the formation of the desired compound. This step may need to be optimized depending on the specific properties desired. 5. Cooling: Allow the sample to cool down to room temperature naturally. The final product should be a dense and homogeneous bulk material of the desired composition. The overall balanced chemical reaction for this synthesis can be represented as: 0.12 Pr_6O_{11} + 45 Sb_2O_3 + 10 ZnO + 45 GeO_2 → 0.12 Pr_6O_{11} − 45 Sb_2O_3 − 10 ZnO − 45 GeO_2

script-generated synthesis route:

To make 0.12 Pr_6O_{11} − 45 Sb_2O_3 − 10 ZnO − 45 GeO_2 requires Pr_6O_{11} and GeO_2 and Sb_2O_3 and ZnO (ZnO). The balanced chemical reaction is 0.12 Pr_6O_{11} + 45 GeO_2 + 45 Sb_2O_3 + 10 ZnO ⟶ 1 0.12 Pr_6O_{11} − 45 Sb_2O_3 − 10 ZnO − 45 GeO_2. Here is the step-by-step synthesis procedure:

The balanced chemical reaction is: 0.12 Pr_6O_{11} + 45 GeO_2 + 45 Sb_2O_3 + 10 ZnO ⟶ 1 0.12 Pr_6O_{11} − 45 Sb_2O_3 − 10 ZnO − 45 GeO_2

GPT-4 score (with explanation):

Information from the first text is mostly contained in the second text. The second text explains the synthesis procedure in more detail, including the precursor preparation, weighing and mixing, calcination, sintering, and cooling steps. The balanced chemical reaction is also mentioned in both texts. However, the second text does not mention the specific requirement of Pr_6O_{11}, GeO_2, Sb_2O_3, and ZnO as in the first text.

Score: 4

GPT-4 score (value only, avg):

5.0

Figure C.27: GPT-4 prompt, proposed synthesis route, script-based reference route, GPT-4-score with explanation, and GPT-4-score

GPT-4 correctly predicts the precursors required to form the compound and the balanced reaction, which are in this case trivially deducible by the product itself. The GPT-assigned scores reflect the correctness of the proposed precursors and reactions. GPT-4 also proposes synthesis steps that are absent in the reference dataset.

While at first glance the proposed synthesis route appears correct, the preparation steps do not correspond to the ones present in the paper this reaction is taken from. Indeed, the paper [27] indicates a melt-quenching synthesis method, where the powders are mixed and melted to a liquit state at 1250 °C for 15 minutes, while GPT-4 proposes a solid-state sintering solution (a common synthesis pathway for this family of compounds) at 1300 °C for 12 hours. This error is not spotted by the GPT-4-assigned score by virtue of the reference database entry being incomplete.

GPT-4

Prompt:

Requested product: Bi_2MoO_6 (Bi_2MoO_6)

GPT-4:

To synthesize Bi_2MoO_6 (Bismuth Molybdate), we can use a solid-state reaction method involving the precursor materials: Bismuth(III) oxide (Bi_2MoO_6) and Molybdenum(VI) oxide (MoO_3).

Synthesis Steps:

1. Preparation of precursor materials:

- Obtain high purity Bismuth(III) oxide (Bi_2O_3) and Molybdenum(VI) oxide (MoO_3) powders.

2. Mixing and grinding:

- Mix stoichiometric amounts of Bi_2O_3 and MoO_3 powders (2 moles of Bi_2O_3 for each mole of MoO_3).
- Grind the mixture in a mortar and pestle or a ball mill to ensure homogeneity and increase the contact surface area between the precursors.

3. Calcination:

- Place the mixture in a furnace and heat it at a temperature of 700°C.
- Maintain the temperature for 4 hours to promote the solid-state reaction between the precursors and allow the formation of the Bi2MoO6 compound.
- After the calcination, the product should be cooled down to room temperature.

4. Optional: For further improvement in the crystallinity and particle size, a second calcination step can be performed at a higher temperature (e.g., 800°C) for a few hours.

The balanced chemical reaction producing the target material from the precursors is:

$Bi_2O_3 + MoO_3 \rightarrow Bi_2MoO_6$

Script-generated synthesis route:

To make Bi_2MoO_6 (Bi_2MoO_6) requires MoO_3 (MoO_3) and $Bi(NO_3)_3{\cdot}5H_2O$ ($Bi(NO_3)_3{\cdot}5H_2O$).

The balanced chemical reaction is 1 MoO_3 + 2 $Bi(NO_3)_3{\cdot}5H_2O$ → 1 Bi_2MoO_6 + 10 H_2O + 6 NO_2 + 1.5 O_2. Here is the step-by-step synthesis procedure:

1) Compounds must be homogenized with mixing media water and mixing device agate

2) Compounds must be calcined with heating temperature 499.85 C

3) Compounds must be annealing

The balanced chemical reaction is:

1 MoO_3 + 2 $Bi(NO_3)_3{\cdot}5H_2O$ → 1 Bi_2MoO_6 + 10 H_2O + 6 NO_2 + 1.5 O_2

GPT-4 score (with explanation):

Information from the first text is not fully contained in the second text. The second text uses Bi_2O_3 instead of $Bi(NO_3)_3{\cdot}5H_2O$ as a precursor, and the calcination temperature is different (700 °C instead of 499.85 °C). The balanced chemical reaction is also different. The second text includes optional second calcination step not mentioned in the first text. Score: 2

GPT-4 score (value only, avg.):

2.0

Figure C.28: GPT-4 prompt, reply, script-based reference, GPT-4-score with explanation, and GPT-4-score for Bi_2MoO_6 synthesis.

GPT-4 correctly identifies only one of the two precursors, and therefore also proposes an incorrect balanced reaction. The sintering temperature (773 °C) is in line with the one proposed in the paper [43] (700 °C), but does not match the one (mistakenly) reported in the reference dataset entry. Moreover, GPT-4 proposes an additional calcination step at higher temperature, which the paper also reports. The GPT-based score

correctly identifies the presence of a wrong precursor and the difference in sintering temperatures, and assigns a score that is in line with the differences. Interestingly, GPT-4 's proposed synthesis is more accurate than the one present in the reference dataset.

C.11 Evaluating candidate proposal for Metal-Organic frameworks (MOFs)

Metal-organic frameworks are a promising class of materials in crucial applications such as carbon capture and storage. Rule-based approaches [9, 46] that combine building blocks with topology templates have played a key role in designing novel functional MOFs.

Task1: Our first task evaluates GPT4's capability in recognizing whether a reasonable MOF can be assembled given a set of building blocks and a topology. This task requires spatial understanding of the building block and topology 3D structures and reasoning about their compatibility. This study is based on the PORMAKE method proposed in [46], which offers a database of RCSR topologies and building blocks, along with an MOF assembling algorithm.

For a preliminary study, we investigate the RCSR (Reticular Chemistry Structure Resource) topologies of two well-studied MOFs: topology 'tbo' for HKUST-1 and opology 'pcu' for MOF-5. The 'tbo' topology can be described as a 3,4-coordinated net, while the 'pcu' topology is a 6-coordinated net. Given the topology, we need to propose 2 and 1 node building block to assemble a MOF for the 'tbo' and 'pcu' topologies, respectively.

For 'pcu', we randomly sample a metal node building block with 6 connection points from the PORMAKE building block database. For 'tbo', we randomly sample two node building blocks with 3 and 4 connection points from the PORMAKE database while ensuring at least one building block contains metal. We then propose the chemical formulas of the sampled node building blocks and the topology to GPT4 ask it to decide whether the selected node building block is compatible with the topology. We get reference labels by using PORMAKE to assemble the proposal. Following previous works [46, 59], we accept the structure if the RMSD between each node building block and its local node structure in the topology is below 0.3 Angstrom. For both 'tbo' and 'pcu', we repeat the experiment 100 times with uniformly randomly sampled building blocks.

GPT-4

System message::
You are a materials scientist assistant.
You are given the chemical formulas of one or more node building blocks and a topology.
Decide whether the building blocks and the topology can be assembled to get a reasonable MOF structure.
In the node building block formula, an X represents a connection point.
Explain your step-by-step reasoning. In the last line, return "yes" or "no".

Query:

node building blocks:

topology:

Figure C.29: System message in candidate proposal for MOFs.

GPT-4

Prompt:

Query: node building blocks: $C_{12}H_{12}Cd_3Cl_6N_{18}X_6$

topology: pcu

GPT-4:

Step 1: Analyze the node building block formula.

The given node building block formula is C12H12Cd3Cl6N18X6. This formula indicates that there are 6 connection points (X6) in the building block. Step 2: Analyze the topology.

The given topology is pcu (primitive cubic). The pcu topology requires a 4-connected node, which means that each node should have 4 connection points to connect with other nodes in the MOF structure.

Step 3: Compare the node building block and topology requirements.

The node building block has 6 connection points, while the pcu topology requires a 4-connected node. The number of connection points in the node building block does not match the requirement of the pcu topology.

Conclusion: No, the building blocks and the topology cannot be assembled to get a reasonable MOF structure.

Figure C.30: Case of candidate proposal for MOFs. One case of correct reject. GPT-4 tries to reason about the number of connection points and claims pcu only admits 4-connected nodes which is incorrect. It got the right answer but the reasoning is wrong.

GPT-4

Prompt:

Query:

node building blocks: $C_6CrN_6X_6$

topology: pcu

GPT-4:

Step 1: Identify the coordination number of the node building block In the given chemical formula, $C_6CrN_6X_6$, there are 6 connection points (X_6). This means the coordination number of the node building block is 6.

Step 2: Check if the coordination number matches the coordination number required by the given topology The pcu topology has a coordination number of 6, which means it requires node building blocks with 6 connection points.

Step 3: Compare the coordination numbers The coordination number of the node building block (6) matches the coordination number required by the pcu topology (6). Since the coordination numbers match, the building blocks and the topology can be assembled to get a reasonable MOF structure.

Answer: yes

Prompt:

Query:

node building blocks: $C_{18}H_{12}Cu_3N_{15}OX_6$

topology: pcu

GPT-4:

Step 1: Identify the number of connection points in the node building block formula. In the given formula, C18H12Cu3N15OX6, there are 6 X connection points.

Step 2: Determine the coordination number of the topology. The pcu (Primitive cubic) topology has a coordination number of 6.

Step 3: Compare the number of connection points in the node building block formula with the coordination number of the topology. In this case, the number of connection points in the node building block formula (6) matches the coordination number of the pcu topology (6).

Step 4: Determine if the building blocks and the topology can be assembled to get a reasonable MOF structure. Since the number of connection points in the node building block formula matches the coordination number of the pcu topology, they can be assembled to get a reasonable MOF structure.

Answer: Yes

Figure C.31: Case of candidate proposal for MOFs. Again, GPT4 tries to argue from the number of connection points. these two times it gets the number of connection points right. However, this is not a sufficient condition for the compatibility between the building block and the topology.

GPT-4

Prompt:

Query: node building blocks: $C_{19}H_{15}SiX_3$, $C_4CuN_4X_4$

topology: tbo

GPT-4:

Step 1: Analyze the node building blocks - $C_{19}H_{15}SiX_3$: This building block has 3 connection points (X). - $C_4CuN_4X_4$: This building block has 4 connection points (X).

Step 2: Analyze the topology - tbo: This topology is a four-connected (4-c) net. Each node in the structure has four connections.

Step 3: Assess the compatibility of node building blocks with the topology - The $C_4CuN_4X_4$ building block is compatible with the tbo topology since it has the required 4 connection points. - The $C_{19}H_{15}SiX_3$ building block, however, is incompatible with the tbo topology since it only has 3 connection points.

Step 4: Decide if a reasonable MOF structure can be assembled - Since only one of the node building blocks ($C_4CuN_4X_4$) is compatible with the tbo topology, a reasonable MOF structure cannot be assembled with both building blocks.

Answer: no

Figure C.32: Case of candidate proposal for MOFs. One case of correct reject. Reasoning is still from the number of connection points. Although the answer is correct, the reasoning is wrong as it mistakenly believes the tbo topology has all nodes having 4 connection points.

Task 2: design MOFs with desired properties. We focus on the pcu topology, and the three most compatible metal nodes in terms of the RMSD between the building block and the topology node local structure (all with RMSD < 0.03 Å). They are N16 ($C_6O_{13}X_6Zn_4$), N180 ($C_{16}H_{12}Co_2N_2O_8X_6$), and N295 ($C_{14}H_8N_2Ni_2O_8X_6$) in the PORMAKE database. They contain 23, 34, and 40 atoms excluding connection points. There are 219 2-c linkers in the PORMAKE database. For each experiment, we randomly sample 5 linker molecules. The results in a design space of 15 MOFs. We ask GPT-4 to suggest a linker-metal node combination to maximize the pore limiting diameter (PLD). We assemble all the MOFs using PORMAKE and compute the PLD using Zeo++ to evaluate the suggestion of GPT-4.

This is a challenging task that requires spatial understanding of the building blocks and the pcu topology, as well as the concept of pore limiting diameter. GPT-4 failed to give the highest PLD MOF on all 5 experiments. GPT-4 selects the 3rd, 6th, 15th, 3rd, 11th/12th GPT-4 proposed two answers in the last experiment) MOF when sorted from high to low PLD. Our task asks GPT-4 to maximize PLD. In all 5 examples, GPT-4 chose the metal node $C_{16}H_{12}Co_2N_2O_8X_6$ which contains the most atoms. However, N16 (with the fewest number of atoms) consistently gives the highest PLD in all 5 experiments. In two out of five experiments (experiment 1 and 4), GPT-4 picked the linker molecule that gives the highest PLD. Overall, GPT-4 shows limited capability in reasoning about the geometry of complex metal nodes and linker molecules in metal organic frameworks to propose MOFs with certain properties.

GPT-4

System message:

Show me an example pressure-temperature phase diagram for a material.

GPT-4:

You are a materials scientist assistant. You are given a list of metal nodes and a list of linker molecules.

Your task is to select one linker and one metal node such that the linker-node combination under the pcu topology can make a MOF with the highest pore limiting diameter (PLD).

In the building block formula, X represents a connection point.

In the last line, return two integers: the index of the metal node and the index of the linker.

Figure C.33: System message in design MOFs.

GPT-4

Prompt:

Metal nodes:

1. $C_{14}H_8N_2Ni_2O_8X_6$
2. $C_6O_{13}X_6Zn_4$
3. $C_{16}H_{12}Co_2N_2O_8X_6$

Linker molecules:

1. $C_8H_8O_2X_2$
2. $C_{14}H_{10}X_2$
3. $C_5H_7N_2X_2$
4. $C_6H_2F_2X2$
5. $C_{10}H_6X_2$

GPT-4:

To determine the MOF with the highest pore limiting diameter (PLD), we need to consider the size and geometry of both the metal nodes and linker molecules. Larger metal nodes and linkers with longer and more flexible chains generally result in higher PLDs.
In this case, the metal nodes and linker molecules can be ranked by size as follows:
Metal nodes:

1. $C_{16}H_{12}Co_2N_2O_8X6$ (largest)
2. $C_{14}H_8N_2Ni_2O_8X_6$
3. $C_6O_{13}X_6Zn_4$ (smallest)

Linker molecules:

1. $C_{14}H_{10}X_2$ (largest)
2. $C_{10}H_6X_2$
3. $C_8H_8O_2X_2$
4. $C_6H_2F_2X_2$
5. $C_5H_7N_2X_2$ (smallest)

Combining the largest metal node ($C_{16}H_{12}Co_2N_2O_8X_6$) with the largest linker molecule ($C_{14}H_{10}X_2$) should result in the MOF with the highest PLD in the pcu topology.

Hence, the index of the metal node is 3, and the index of the linker is 1.

GPT-4 (continue)

Prompt:
Metal nodes:

1. $C_{14}H_8N_2Ni_2O_8X_6$
2. $C_6O_{13}X_6Zn_4$
3. $C_{16}H_{12}Co_2N_2O_8X_6$

Linker molecules:

1. $C_{12}H_{10}X_2$
2. $C_6H_{12}X_2$
3. $C_{10}H_8Cl_3InN_2X_2$
4. $C_6H_3NO_2X_2$
5. $C_8H_6N_2O_2X_2$

GPT-4:
To select the best linker-node combination for the highest PLD, we need to consider the size and geometry of both the metal nodes and the linker molecules.
In general, larger metal nodes and larger linkers will result in larger PLDs. Based on the chemical formulae, we can make the following observations:
Metal nodes:

1. $C_{14}H_8N_2Ni_2O_8X_6$ (Ni-based node) - large
2. $C_6O_{13}X_6Zn_4$ (Zn-based node) - small
3. $C_{16}H_{12}Co_2N_2O_8X_6$ (Co-based node) - large

Linker molecules:

1. $C_{12}H_{10}X_2$ (large linker)
2. $C_6H_{12}X_2$ (small linker)
3. $C_{10}H_8Cl_3InN_2X_2$ (large linker with In, Cl)
4. $C_6H_3NO_2X_2$ (small linker)
5. $C_8H_6N_2O_2X_2$ (medium linker)

Based on these observations, we can make the following possible combinations:

1. Large metal node (Ni-based or Co-based) + large linker ($C_{12}H_{10}X_2$ or $C_{10}H_8Cl_3InN_2X_2$)
2. Large metal node (Ni-based or Co-based) + medium linker ($C_8H_6N_2O_2X_2$)

We can then narrow down the options by considering any specific interactions between the metal nodes and linker molecules. However, since no additional information is given, we can assume that the larger linker would result in a larger PLD.
Therefore, the best combination is a large metal node (Ni-based or Co-based) with the large linker C12H10X2.
The two integers representing the index of the metal node and the index of the linker are:
Metal node index: 1 (Ni-based node) or 3 (Co-based node) Linker index: 1
Hence, the best combinations are either (1, 1) or (3, 1) with similar PLD values.

Figure C.34: Case of candidate proposal for MOFs. In both examples, GPT4 attempts to consider the sizes of the building blocks using the number of atoms. It does not try to reason about the geometry and reached the wrong conclusion.