Advanced Intelligent Discovery

ADVANCED INTELLIGENT DISCOVERY Open Access

RESEARCH ARTICLE OPEN ACCESS

LLM-Based Scientific Assistants for Knowledge Extraction: Which Design Choices Matter?

David Exler[1] | Manuel Raimann[1] | Marc Münker[2] | Maxim Rosin[1] | Joaquin E. Urrutia Gómez[1] | Christof M. Niemeyer[2] | Markus Reischl[1] | Luca Rettenberger[1]

[1]Institute for Automation and Applied Informatics (IAI), Karlsruher Institute of Technology (KIT), Eggenstein-Leopoldshafen, Baden-Württemberg, Germany | [2]Institute for Biological Interfaces (IBG), Karlsruher Institute of Technology (KIT), Eggenstein-Leopoldshafen, Baden-Württemberg, Germany

Correspondence: David Exler (david.exler@kit.edu)

Received: 15 October 2025 | **Revised:** 25 March 2026 | **Accepted:** 27 March 2026

Keywords: chemistry | Large Language Model | Prompt Engineering | reasoning | Retrieval Augmented Generation

ABSTRACT

Large Language Model chatbots have gained significant popularity, offering knowledge to support specialists in diverse fields. However, adapting models to specific use cases and specialized domains presents considerable challenges. Hence, we introduce the LLM Playground, a comprehensive approach to optimizing LLMs for specialist applications with respect to their accuracy in answering domain-specific questions, addressing the limitations of unmodified models. The utilized optimization techniques begin with Prompt Engineering, advance to the integration of external knowledge, and culminate in complex reasoning strategies or self-feedback loops. This paper introduces various architectures for scientific assistants, comprising individual enhancement techniques, both in isolation and in combination with others, designed to facilitate comparisons. To demonstrate the efficacy of the LLM Playground, a chemical chatbot is set up as a case study, and the optimization techniques are compared using ChemBench, an independent question–answer benchmark for the chemical domain, to measure its performance. By providing tested, ready-to-deploy architectures and clear use-case guidance, this work helps researchers and practitioners leverage LLMs in domain-specific applications. The insights and methodologies presented in this paper contribute to the growing body of knowledge on tailoring LLMs to meet the unique demands of specialized fields.

1 | Introduction

Scientific Large Language Model (LLM) applications such as literature summarization [1], data extraction [2], or question-answering [3] rely on the factual correctness and conciseness of the output [4].

However, in practice, base models (the large, unmodified LLMs with heavy pretraining) often generate inaccurate statements [5–8], fabricate misleading data (hallucinations) [9, 10], or struggle with data extraction [11, 12]. These errors are limiting factors for scientific assistant applications of LLMs [13–15]. Many different methods of variable complexity can alleviate this problem with varying success [16–19]. Choosing the methods to implement for a scientific project can thus impact the quality of its outcome, and a careful evaluation is imperative.

To accelerate the development of LLM-based assistants for specialized scientific domains, we introduce the LLM Playground, a framework for evaluating the efficiency of the most prevalent enhancement methods within a given application. The framework provides five ready-to-deploy chatbot architectures, enabling systematic comparison. These architectures implement widely adopted methods of LLM optimization in standalone and combined approaches. The framework uses a domain-specific question–answer benchmark to grade the methods. To demonstrate the effectiveness of the LLM Playground, a chemical assistant chatbot is examined as a case study by employing ChemBench [20], a standardized framework for the automatic evaluation of chemical knowledge. Here, ChemBench serves as an independent quality criterion for the comparative

assessment of the chemical specialization capabilities of every architecture. Based on the quantitative measurements of the case study, this work continues to recommend a course of action that abstracts from the chemical use case toward generalized best practices for the application of LLMs in other scientific domains. Our main contributions are as follows:

- The LLM Playground, a framework to evaluate individual enhancement methods for LLM-based scientific assistants.
- A collection of chatbot architectures implementing state-of-the-art techniques to improve the accuracy of an LLM.
- The development of complex reasoning setups, including iterative self-evaluation and multi-LLM structures.
- A quantified comparison of the effectiveness of these state-of-the-art techniques for the specialization of LLMs to scientific domains.
- A collection of recommendations and best-practice methods when employing LLMs.

2 | Related Works

2.1 | Large Language Models

The development of LLMs has marked a significant milestone in artificial intelligence, particularly in (NLP). LLMs, such as GPT-3 [21] and GPT-4 [22], Llama [23, 24], Mistral [25], Falcon [26], and recently DeepSeek R1 [27] rapidly hit new milestones in an ever-expanding research field. They are trained on vast collections of text, leveraging billions of words from a wide array of sources, including internet content, books, and articles, using different training techniques [28, 29]. This extensive training enables them to generate human-like text and make continuous dialog contributions [30].

2.2 | Chemical LLMs

Recent studies have examined the potential of expert LLMs to address chemical tasks such as forward and retrosynthesis, name conversion, molecular property prediction, molecule captioning, and general chemical reasoning [31, 32]. Researchers have explored various strategies to enhance the accuracy of the answers provided by the deployed LLM. One of these application strategies is the pretraining of representations of molecular information, for example, by the *Simplified Molecular Input Line Entry System* (SMILES) [33] or *Self-referencing embedded strings* (SELFIES) [34] and tokenizing them, as is done with regular words [31]. Tokenization is the process of transforming words into a numeric format that embeds semantic information about the transformed word. Another application is integrating chemistry-specific software tools, designed by human experts, which the LLM can use autonomously, as in ChemCrow [35, 36]. A common approach for domain adaptation is the continuation of the training of the model weights using domain-specific text data (finetuning) [37, 38]. Another approach is to supplement existing algorithms with LLMs [39].

2.3 | Prompt Engineering

Prompt Engineering is a fast and easy way to increase LLM performance. At its core, Prompt Engineering is the development of a set of natural language instructions, like examples, task specifications, or constraints provided at runtime [40]. Automatic prompt generation [41, 42] and prompt-based finetuning [43, 44] have extended the basic principle by a repeatable optimization process. Some quality criterion grades applications using manually engineered prompts. The prompts are then iteratively edited while observing said quality criterion. Ultimately, the goal is to omit manual inputs completely while outperforming human prompts. However, despite extensive research on the matter, optimal solutions have not yet emerged [45, 46].

Chemical downstream tasks for LLMs can greatly benefit from domain-specific Prompt Engineering, such as incorporating examples of SMILES [33] representations [47, 48]. Systematic approaches like automatic Prompt Engineer [49] use an LLM to generate and select instructions. Other modern approaches engage in automatic prompt generation via evolutionary algorithms, variational inference, and gradient-guided or gradient-free search [50–53]. Dominant in literature are methods using evolutionary algorithms [54] based on an LLM's ability to create natural language instructions [55]. These methods usually obtain a set of manually created prompts as their starting population and combine or mutate them by prompting an LLM to do so.

2.4 | Reasoning

LLMs struggle with inaccuracy resulting from hallucinations [9, 10], inconsistent use of source material [56], or other unidentified issues [5, 6]. Various types of LLM reasoning have been shown to alleviate weaknesses and enhance interpretability [57, 58]. A prominent type of reasoning is Chain of Thought prompting [59], which refers to the implementation of an example of a human-like multistep thought process into the prompt and encouraging the LLM to mimic it. It relies on the LLM generating multiple coherent units of text called *thoughts*. More complex approaches like Tree of Thoughts (ToT) [60] and Graph of Thoughts (GoT) [61] use iterative self-evaluation and feedback-loop strategies to handle *thoughts* more dynamically. With ToT, the LLM can evaluate each *thought*, add or disregard them, and effectively look ahead or backtrack in its answer before generating the final output. GoT takes this concept even further by also modeling *thoughts* as vertices and the relations between them as edges. This graph then enables GoT to generate an answer that taps into the LLM's rich inherent knowledge efficiently. StructChem [62] advances reasoning in a chemical domain by specializing the chain for chemical tasks, integrating LLM-Reasoners [63], and providing an open-source framework.

2.5 | External Knowledge

An LLM's inherent knowledge often does not cover the answers to domain-specific questions [64, 65]. Retrieval Augmented Generation (RAG) is a concept aiming for better alignment of LLM output and confirmable information. To this end, the input text is augmented with facts from external sources, providing the LLM with knowledge it might not possess within its own context [66–68]. Furthermore, specialized tools providing an interface for knowledge databases to the LLM enable the injection of confirmable additional knowledge [69, 70]. Recent works have found incorrect interpretation of well-retrieved external knowledge

to be the most limiting factor of LLM tool usage quality [71–73]. It has also been proposed to integrate tools as autonomous agents, where an agent represents a planning or decision-making entity [74–77]. The core idea is to capitalize on the LLM's NLP capabilities by prompting it to handle one or multiple tool interfaces independently, allowing for dynamic tool use at runtime [68, 78].

2.6 | LLM Benchmarks

The LLM Playground framework relies on an independent quality criterion to quantify the domain-specific knowledge of individual chatbot architectures, as expressed in this case study by chemical expertise. Simple metric approaches such as ROUGE [79] and BLEU [80], which rely on measuring the verbatim overlap between generated text and a reference text, are widely considered outdated in the context of evaluating LLMs [81]. Metrics of growing dimensionality are replacing them, meaning that many abstract factors are used to grade NLP quality instead of undynamic exact word matching [82]. Recent research proposes new methods that capitalize on the NLP capabilities of LLMs [83]. For example, when evaluating the similarity of a generated summary and its original text, LLMs can leverage semantic understanding of both texts, while classical approaches only operate on a lexical level. Modern quality criteria used to evaluate LLM applications rely on standardized text collections tailored to the property being measured. General-purpose benchmarks are used to measure the NLP quality of the base models [5, 84, 85], while disregarding the extent of their knowledge or the factual correctness.

Domain-specific, i.e., chemical benchmarks, contain data to evaluate the content of LLM statements [39, 86, 87]. One established chemical benchmark is ChemLLMBench, which aims to grade three key capabilities of LLMs, namely understanding, reasoning, and explaining in the chemical domain [88]. It includes multiple iterations of eight common chemical tasks that each contribute to the score given for one of the key capabilities. Another is ChemBench, which provides wide coverage of chemical subfields [20]. On more than 2700 question–answer pairs, it tests the extent of the knowledge LLMs in nine different categories. The percentages of correct answers serve as scores, both per category and in a weighted average. ChemBench also offers an online scoreboard that is continuously updated for the most recent prominent base models.

2.7 | Remaining Challenges

Although methods to improve LLMs have been extensively studied, existing work focuses on individual techniques or application-specific adaptation. An approach for systematic side-by-side comparison is missing. Especially, the exploration of the effect that both standalone and shared implementations of different domain adaptation strategies have on a system's accuracy is crucial. Research progress is further slowed by the need to revisit the same considerations each time LLMs are adapted to scientific domains and by the lack of a systematic procedure to formalize this process.

3 | Method

3.1 | Overview

Figure 1 illustrates an overview of the setup of this study. Our LLM Playground framework implements modular components used to build different architectures for enhancing LLM performance. Every architecture's domain specialization capability is measured by the ChemBench score it achieves. The chatbot architectures are organized into four components: The LLM, the **natural language instruction** (prompt), **external data sources**, and **flow configurations** for the question-answering workflow. Different variants of each component are considered throughout this study, starting with optimization methods for the **natural language instructions**. A computationally expensive, automatic prompt optimization technique using an evolutionary algorithm, as well as a semi-manual, low-cost approach, is considered. For the LLM, prominent open-source models [24, 27] of varying size, trained with different techniques, are included. The available **external data sources** consist of scientific papers from arXiv, a SMILES to molecular description converter, and a database featuring a selection of relevant chemical texts and books. By employing the RAG [67] practice, the chatbot autonomously retrieves information. Lastly, complex **flow configuration** methods like self-evaluation loops or multi-LLM setups are leveraged to maximize the human-like reasoning capabilities of the base models. The resulting architectures are illustrated in Figures 2–6.

3.2 | Setup

Throughout this study, the various chatbot architectures are graded and compared in terms of their quality as a chemical assistant. To this end, the LLM Playground sets up a repeatable evaluation procedure. The experiment is conducted iteratively while exchanging components. Components accessing an LLM are managed as chains, which organize the order of operations to perform on the input. The set of operations consists of the following:

- Prepending a system prompt to give general instructions
- Adding a message prompt that incorporates the text input
- Calling an optional interface for communication with external data sources
- Calling the base model itself to generate text
- Gathering and parsing the text output from the full LLM response

The methods and experiments are divided into two sections: a separate setup to evaluate methods of **natural language instructions** optimization and a setup to evaluate architectures for standalone and shared realizations of every other component. This divide is conducted because separating the evaluation of prompt optimization methods brings great computational relief without any accuracy downside. Comparing the results, the effectiveness of each component's contribution to domain adaptation is then determined. Doing so, practical advice and a best-practice course of action based on individual requirements is created, grounded in replicable data.

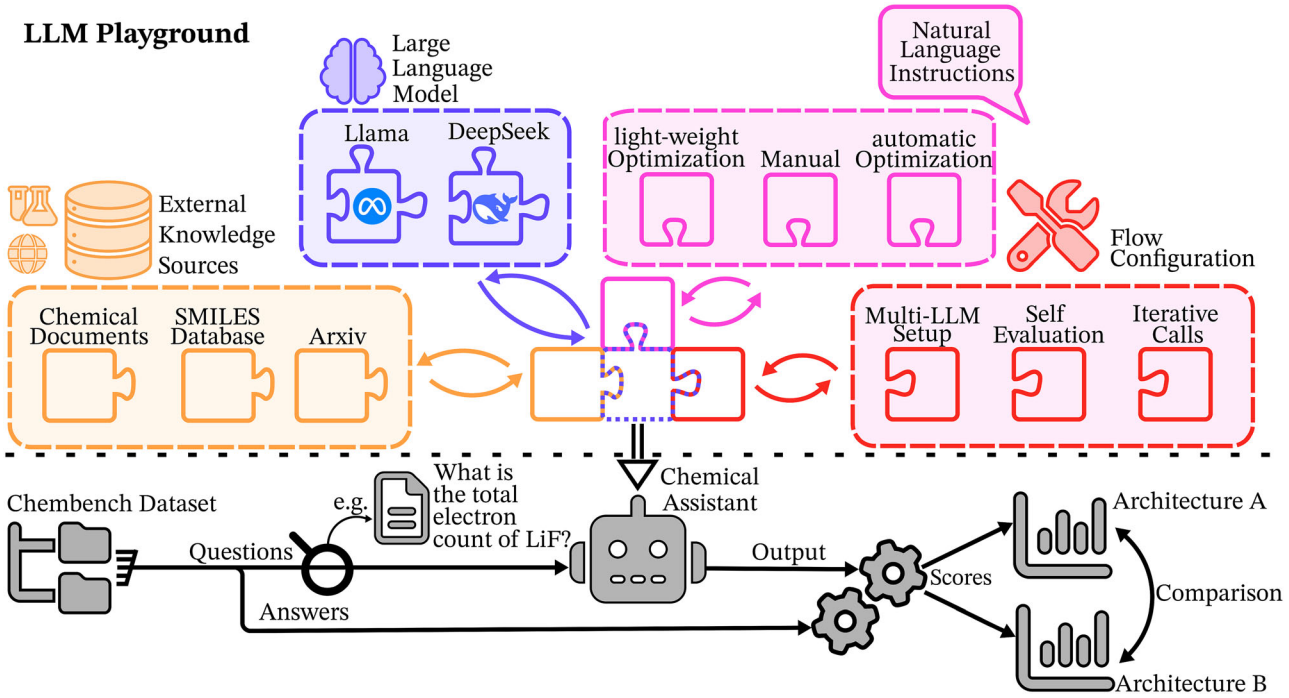

FIGURE 1 | Overview of the study setup. The diagram illustrates the LLM Playground. It contains modular components used to build different architectures for enhancing LLM performance, shown on top. The components include **natural language instructions** (prompts) that are either created manually or generated by an optimization algorithm. The optimization is done either by manually prompting an LLM to do so (light-weight) or by an evolutionary algorithm (automatic). Another component is the **flow configuration**, meaning the structured way in which the software interacts with the LLM. This includes setting up workflows where, for example, a model is called repeatedly in a loop to critique its own answers. It can also involve using multiple models together or implementing self-evaluation methods. Various **external knowledge sources** are used to supplement the chatbot's inherent knowledge. Lastly, different **LLMs** of varying parameter count can be implemented into the architecture. The components are assembled into distinct architectures for a chemical assistant. The architectures are evaluated through a standardized benchmarking pipeline shown at the bottom. Each architecture generates output for all the benchmark questions, and the resulting performance scores are compared.

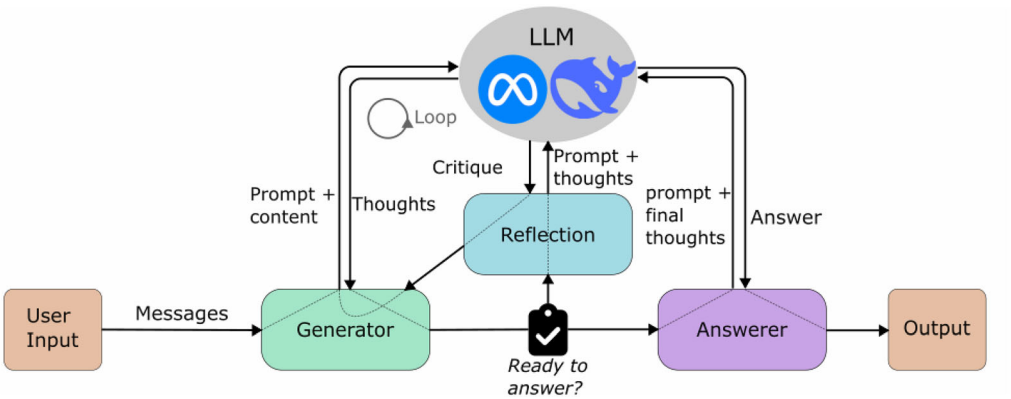

FIGURE 2 | Graph illustrating the **Reasoning** architecture. The user input and the application's output are depicted as blocks on the far sides. The other blocks each represent a chain. Chains can add instructions and previous messages to an input text or handle tool interfaces before calling the LLM to perform a specific task. A **Generator** chain generates reasoned context for the question, optionally based on previous *thoughts*, and a **Reflection** chain generates *thoughts* based on the current context. This is done iteratively until a stop criterion is met. Lastly, an **Answerer** chain generates a final answer based on the context and *thoughts*.

3.3 | Dataset

The quality criterion for the experiments of this study is chosen to be ChemBench [20] as it offers a more fine-grained division of chemical subfields and a straightforward score of factual correctness, in contrast to ChemLLMBench's [88] per-capability score. ChemBench consists of multiple-choice questions and scores the LLM answers from one (best) to zero (worst) according to the share of true positives and negatives.

3.4 | Natural Language Instructions

This paragraph introduces the section of the setup to evaluate and compare methods of **natural language instruction**

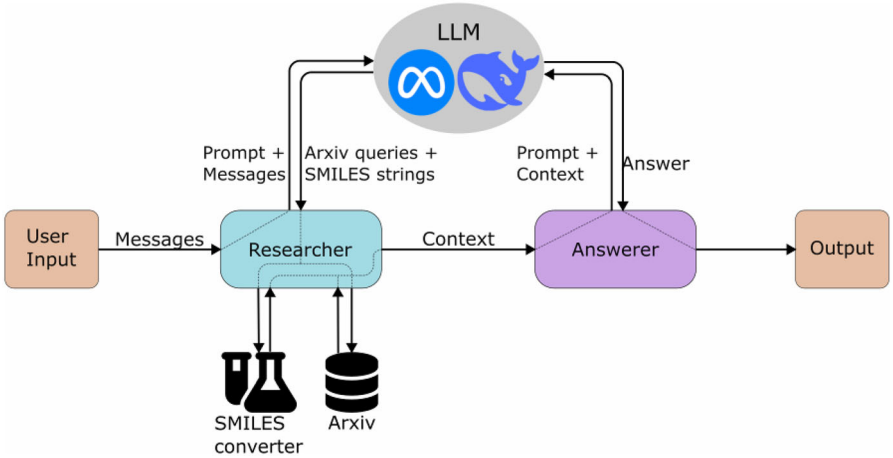

FIGURE 3 | Graph illustrating the **Reasoning** architecture. The user input and the application's output are depicted as blocks on the far sides. The other blocks each represent a chain. Chains can add instructions and previous messages to an input text or handle tool interfaces before calling the LLM to perform a specific task. A **Researcher** chain generates relevant arXiv queries, and molecular representations that will then be input for one of the tools, and an **Answerer** chain generates a final answer based on the output of the tools.

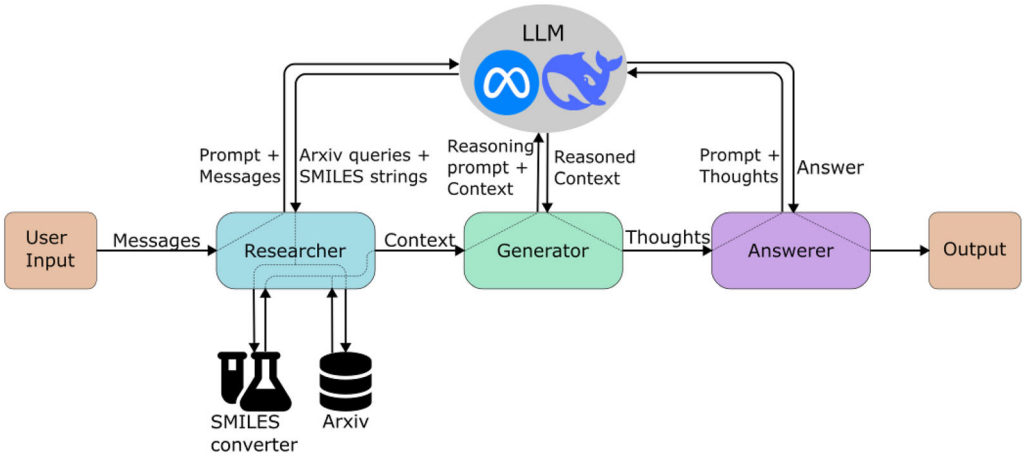

FIGURE 4 | Graph illustrating the **RAG-Reasoning** architecture. The user input and the application's output are depicted as blocks on the far sides. The other blocks each represent a chain. Chains can add instructions and previous messages to an input text or handle tool interfaces before calling the LLM to perform a specific task. A **Researcher** chain generates relevant arXiv queries and molecular representations that will then be input for one of the tools. A **Generator** chain generates reasoned context for the question, optionally based on the tool output inside the context provided by the **Researcher** chain, and an **Answerer** chain generates a final answer based on the reasoned context of the **Generator** chain.

optimization. It includes a light-weight solution and a sophisticated automatic approach. For both methods, the risk of overfitting, computational efficiency, and sample diversity are balanced by a small training split, compared to a large test split. As we find that the quality plateaus at a training split of ten questions from each category provided by ChemBench, we use this limited training split to reduce the risk of overfitting. The remaining questions form the test split, ensuring reliable evaluation of generalization performance on a large and diverse benchmark.

3.4.1 | Light-Weight Approach

We implement the simplest optimization method by prompting our LLM to output variations of a manually composed prompt. Every variation is inserted as the system prompt of the LLM to perform the training split questions. The best-performing variations are then compared on the test split. Computational resources are hereby minimized at the cost of some manual input.

3.4.2 | Automatic Approach

An evolutionary algorithm [54], similar to those in prominent literature [55], is established as a sophisticated automatic approach. In the initial iteration, the LLM is tasked to create a starting population of 50 chemical domain prompts, referred to as specimen (Examples in Appendix A.2.1). After this, the algorithm repeatedly performs three steps designed to optimize the ChemBench score achieved by the chatbot when using the specimen as its system prompt:

- Calculating the fitness of each specimen as the mean ChemBench [20] score achieved when using the specimen as the system prompt.

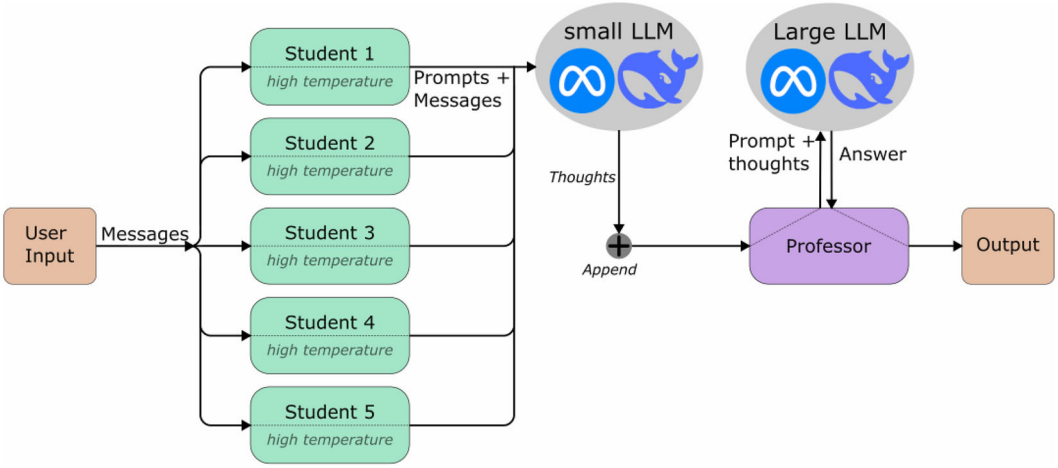

FIGURE 5 | Graph illustrating the Ensemble architecture. The user input and the application's output are depicted as blocks on the far sides. The other blocks each represent a chain. Chains can add instructions and previous messages to an input text or handle tool interfaces before calling the LLM to perform a specific task. Multiple **Student** chains call a smaller, faster LLM to generate reasoned *thoughts*, and a **Professor** chain calls the larger, more accurate LLM to generate a final answer based on the Ensemble of thoughts.

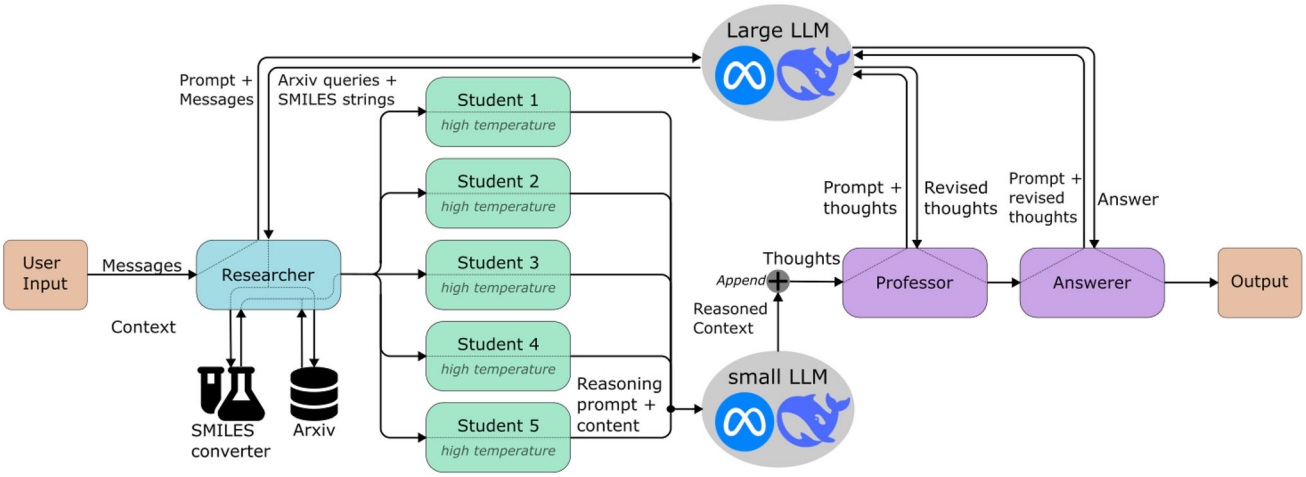

FIGURE 6 | Graph illustrating the Ensemble-Researcher architecture. The user input and the application's output are depicted as blocks on the far sides. The other blocks each represent a chain. Chains can add instructions and previous messages to an input text or handle tool interfaces before calling the LLM to perform a specific task. A **Researcher** chain generates relevant arXiv queries and molecular representations that will then be input for one of the tools. A **Generator** chain generates reasoned context for the question, optionally based on the tool output inside the context provided by the **Researcher** chain. Multiple **Student** chains call a smaller, faster LLM to generate reasoned *thoughts*, optionally based on the tool output inside the context provided by the **Researcher** chain. A **Professor** chain calls the larger, more accurate LLM to generate a revision of the Ensemble of *thoughts*, and an **Answerer** chain generates a final answer based on the reasoned context of the **Generator** chain.

- Randomly selecting specimen to be eliminated. A higher fitness increases a specimen's probability of survival (fitness-guided tournament selection).
- Mutating or recombining the remaining specimen by prompting the LLM to create a new prompt based on two input-specimen. For this, the system message consists of instructions and examples similar to [55].

The algorithm iteratively performs on the designated training split of ChemBench's questions for 50 epochs. We then evaluate the impact of the optimization algorithm on the chatbot's chemical expertise in the separate test split.

3.5 | Architectures

This paragraph introduces the five architectures set up for comparative experiments. These architectures implement standalone and shared realizations of the various components. Every architecture is composed of chains, which organize the order of operations to perform on the input. With these proposed architectures, we do not aim to find final, finetuned configurations for a specific application. We intend to measure the potential of the explicit design choices we find most prevalent: external knowledge or not, implementing a multi-LLM setup, and leveraging self-feedback reasoning for a specific system. The architectures are thus designed to be as simple and representative of a

realization of these design choices as possible. Doing so, the **LLM Playground** enables the comparison of these design choices. Consequently, finetuning details such as the self-feedback iterations or the database tools for external knowledge are kept fixed to enhance comparability. While details can marginally affect the outcome, the performance of a model heavily depends on its strengths and weaknesses, which the inclusion or omission of an individual design choice addresses measurably.

3.5.1 | Reasoning

The **Reasoning** architecture introduces a complex structure of dynamic, iterative LLM calls. It is a standalone flow configuration approach designed to capitalize on LLM reasoning. Figure 2 illustrates a flowchart of the **Reasoning** architecture. Three chains are assigned virtual roles as follows:

- **Generator**: Generates reasoned context for the question, optionally based on previous thoughts.
- **Reflection**: Articulates thoughts based on the current context. It critiques the context expressed by the **Generator**.
- **Answerer**: Formulates a final answer based on the context and *thoughts* provided by the **Generator** and **Reflection** chain.

The **Generator** and **Reflection** chain are iteratively repeated until a stop criterion is met, with the goal of generating the most in-depth context possible, utilizing all of the LLM's inherent knowledge. The stop criterion for the **Generator**-**Reflection** loop can be defined individually. We find a fixed number of five iterations to work well, but such specific implementation details can be finetuned for individual applications. To pass the generated text from one chain to another, the LLM output is parsed as a separate message to the prompt of the next chain. This means that each iteration must be completed before another chain can call the LLM, which disables parallel processing of a singular task. A higher per-task runtime is to be expected with the **Reasoning** architecture. As this architecture aims to enhance the strength of broad inherent knowledge, we expect a result that is fully dependent on the inherent knowledge of the base LLM. To confirm this hypothesis, we are looking for an increase in quality when comparing a smaller LLM to a larger LLM of otherwise similar quality.

3.5.2 | RAG

The **RAG** architecture introduces a basic standalone implementation of external knowledge. It consists of two chains only, with no iterative calls. The following briefly describes the virtual roles of the **RAG** architecture's chains, as illustrated in Figure 3:

- **Researcher**: Generates queries specifically fitted to the available tools. Any context the LLM expects to be useful is expressed as a structured tool input and automatically parsed to the interface by the chain. In the chemical use case of this work, the available tools include a SMILES to clear name converter and an arXiv interface, which provides access to scientific papers.
- **Answerer**: Formulates a short and precise answer based on all of the context provided by the **Researcher** chain.

This architecture is expected to mitigate the weakness of an LLM's limited knowledge, as external sources can supply information beyond the model's own. Base models lacking the factual knowledge to answer domain-specific questions can be identified by this architecture, and information can be supplemented accordingly. However, it cannot fully prevent hallucinations, so some risk of failure remains. The quality of the external data and the LLM's ability to formulate appropriate queries are also critical to its success. Notably, this method disregards reasoning entirely. Because DeepSeek R1 models always produce reasoned outputs, no experiments with this architecture were conducted using those models.

3.5.3 | RAG-Reasoning

By combining the **RAG** and the **Reasoning** architecture, the **RAG-Reasoning** architecture introduces a combined realization of flow configuration and external knowledge implementation. It is illustrated in Figure 4. To address the increased runtime of the **RAG-Reasoning** architecture, we omit the iterative structure, expecting the context to be rich enough after the **Researcher** and the **Generator** chain call. The chains' virtual roles are as follows:

- **Researcher**: Generates queries specifically fitted to the available tools. Any context the LLM expects to be useful is expressed as a structured tool input and automatically parsed to the interface by the chain. In the chemical use case of this work, the available tools include a SMILES to clear name converter and an arXiv interface, which provides access to scientific papers.
- **Generator**: Generates reasoned context, but not the final answer itself. Having the **Researcher** chain's output in its context, the **Generator** chain might comment on the given external knowledge input.
- **Answerer**: Formulates a short and precise answer based on all of the context provided by the Researcher and Generator chain.

Because this approach uses both the inherent LLM knowledge and external sources, it is expected to outperform other approaches. We expect it to leverage the strength of good quality text generation and the strength of extensive information extraction from the provided tool sources. The architecture can alleviate the weakness of limited inherent knowledge.

3.5.4 | Ensemble

The **Ensemble** architecture introduces a complex flow configuration method using two separate, interacting LLMs of different sizes. Figure 5 illustrates the **Ensemble** architecture. First, a smaller model with a high-temperature parameter1 is prompted for reasoning, resulting in the **Student** chain. This yields rich context, as the high-temperature parameter encourages diverse answers. The answers are combined and passed to the **Professor** chain, which calls a large LLM. The virtual chain roles can be summarized as follows:

- **Student**: Generates extensive *thoughts* regarding the question, without explicitly answering it.

- **Professor**: Analyzes and critiques the students' context and answers the questions based on this.

This approach introduces a degree of freedom by generating high-temperature context with a smaller LLM. The results are expected to depend strongly on the smaller model's inherent knowledge. If the amount of knowledge lost by scaling down the model is significant, this will be evident in underwhelming results compared to the **Reasoning** architecture. Using a smaller model that is easily parallelized, this approach does not introduce additional latency to the answers.

3.5.5 | Ensemble-Researcher

This combined architecture aims to use both external knowledge and complex flow configurations. It is a variant of the **Ensemble** architecture using the method of the **RAG** architecture. Again, a smaller (Llama 3 8b) model is used by one of the chain types to generate intermediate context. Figure 6 shows a flowchart of this architecture. The four chain types are each given a virtual role:

- **Researcher**: Generates queries specifically fitted to the available tools. Any context the LLM expects to be useful is expressed as a structured tool input and automatically parsed to the interface by the chain. In the chemical use case of this work, the available tools include a SMILES to clear name converter and an arXiv interface, which provides access to scientific papers.
- **Student**: Generates extensive thoughts regarding the question, without explicitly answering it. Having the **Researcher** chain's output in its context, the **Student** chain might comment on the given external sources.
- **Professor**: Analyzes and critiques the **Student** chains' context.
- **Answerer**: Formulates a short and precise answer based on all of the context provided.

With four linear chain types, this approach has a high runtime; however, it is expected to perform well, as it leverages both inherent knowledge and external data sources, if the context provided by the smaller models is of sufficient quality and the main base model is able to extract the provided information well. While the vast context given by the **Student** chains does provide a wide range of reasoned thoughts, it is also prone to hallucinations.

3.6 | Implementation

We use Huggingface [89], a distributor of open-source artificial intelligence models for the base models. To build our base LLM application, we use the open-source framework LangChain [90], and to construct the specific architectures, we use its LangGraph tool. The application backend is served using LangServe and FastAPI [91], while LLM inference is performed through Huggingface's Text Generation Inference toolkit, deployed via a Docker container. For the frontend, we employ Angular, with Keycloak [92] handling user authentication and management. Grafana [93] monitoring is integrated for fast and easy process supervision. All experiments were conducted on a system equipped with 4 × NVIDIA RTX A6000 GPUs, an Intel Xeon Gold 6354 CPU, and 125 GB of DDR4 RAM.

4 | Results

4.1 | Prompt Engineering

Multiple independent experiments with varying starting populations show no significant improvement of the mean ChemBench score over the 50 epochs. Some starting populations even result in a decrease of the ChemBench score (see Figure 7a). However, a notable rise is achieved by both the simple LLM approach and the evolutionary algorithm when picking the top-performing prompts. Throughout multiple experiments, the top-performing prompt has been identified after only a few epochs, with exceptions being marginal improvements achieved at a significantly higher computational cost. We observe a marginal improvement of the evolutionary algorithm over the simple approach. The prompt used to achieve the maximum score is given in Appendix A.2.3. All results are displayed in the appendix in Table A1.

4.2 | Architectures

The following presents the results of each base model across all architectures, accompanied by a brief analysis of the data. Detailed per-category scores are given in the Appendix section

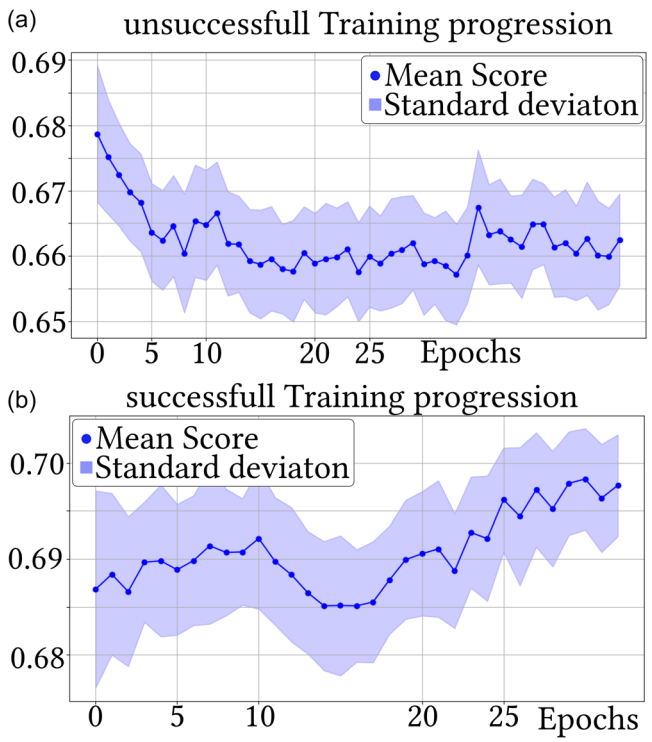

FIGURE 7 | Results of an unsuccessful (a) and a successful (b) training progression of the automatic Prompt Engineering training via evolutionary algorithm. The blue line shows the mean ChemBench score of the population present in the epoch indexed by the *X*-axis. Around the blue line, an area is displayed to indicate the standard deviation of the scores at the current epoch.

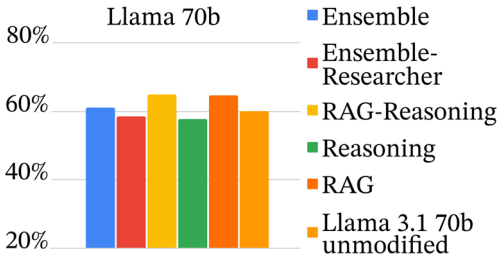

FIGURE 8 | Bar graph displaying the average ChemBench score of every architecture with the Llama 3 70b base model. The *X*-axis contains all the architecture, as labeled by the colors of the legend to the right, and the *Y*-axis displays the average ChemBench score.

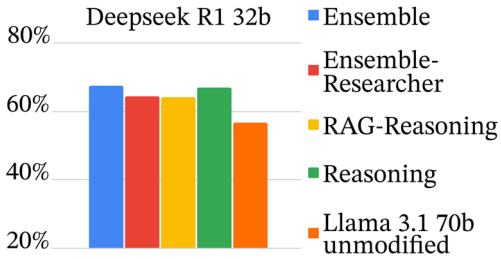

FIGURE 10 | Bar graph displaying the average ChemBench score of every architecture with the DeepSeek R1 32b base model. The *X*-axis contains all the architecture, as labeled by the colors of the legend to the right, and the *Y*-axis displays the average ChemBench score.

A.3. Across multiple evaluations, the scores are consistent, and the unmodified models reproduce the independently reported scores provided by the ChemBench Leaderboard, indicating that the variance in scores results from the different architectures.

Using the Llama 3 70b base model, a top-performing architecture is observed. The **RAG-Reasoning** (yellow) variant yields an average per category increase of 8% compared to the baseline (light orange), followed closely by the **RAG** architecture (see Figure 8). The most significant quality gain occurs in analytical chemistry, with a 75% increase (see Figure A1). In inorganic chemistry, we observe a 5% decrease relative to the baseline. Neither the **Ensemble** nor the **Ensemble-Researcher** method performs well; the latter even falls below the baseline. The standalone flow configuration of the **Reasoning** architecture performs worst overall.

The small-scale DeepSeek R1 1.5b model does not perform at par with the baseline (orange) on average, no matter what architecture is deployed (see Figure 9). It does, however, manage to recreate the baseline performance for the Chemical Preference category (see Figure A2). The **Reasoning** (green) architecture was most successful, but on average, the per-category score still drops by 31%.

Even at reduced size compared to the baseline, the DeepSeek R1 32b model yields highly improved scores (see Figure A3). It performs best when tasked to utilize its inherent knowledge (Figure 10). The **Ensemble** (blue) architecture, closely followed by the **Reasoning** (green) architecture, outperforms both the baseline (yellow) and all architectures using external knowledge. With an average improvement of 120% per category compared to the baseline, it also outperforms Llama 3 using the **RAG** architecture substantially. The category most improved by every architecture is analytical chemistry, while the inorganic chemistry score improves by only 5% (see Figure A3).

The large DeepSeek R1 70b base model is most successful when omitting external knowledge as well. It yields the best results with the **Reasoning** (green) architecture (see Figure 11). This result is also the highest among all models and architectures, recording an improvement of 122% on average per category compared to the baseline (orange). We note that this result is only slightly higher than the top-performing DeepSeek R1 32b architecture. However, the one architecture that has improved the most, as compared to the smaller DeepSeek R1 32b model, is the **Reasoning** architecture, which gains three percentage points.

Directly comparing the top-performing architectures of each base model, it is apparent that the DeepSeek R1 model does not maintain its chemical capabilities when scaled down to 1.5b parameters. However, on larger scales, it performs significantly better in every category than Llama 3 70b with a **RAG-Reasoning** architecture does. The DeepSeek R1 70b model especially excels in ”Inorganic Chemistry”, which is the category with the lowest recorded improvements. Between the 32b and the 70b DeepSeek R1 model, the margins are very small, with the smaller model even outperforming the full 70b model at ”Technical Chemistry” and ”Physical Chemistry” (see Table A2 and Figures A4 and A5).

5 | Discussion

The results demonstrate that the quality of domain-specific LLM applications heavily depends on their exterior design. A comparison of the design parameters is thus sensible. Using the LLM

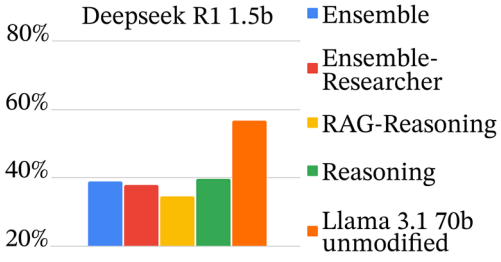

FIGURE 9 | Bar graph displaying the average ChemBench score of every architecture with the DeepSeek R1 1.5b base model. The *X*-axis contains all the architecture, as labeled by the colors of the legend to the right, and the *Y*-axis displays the average ChemBench score.

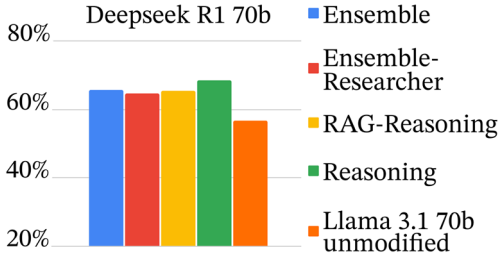

FIGURE 11 | Bar graph displaying the average ChemBench score of every architecture with the DeepSeek R1 70b base model. The *X*-axis contains all the architecture, as labeled by the colors of the legend to the right, and the *Y*-axis displays the average ChemBench score.

Playground, this comparison is effectively executed, and an optimal setup for an LLM-based scientific assistant application is found for multiple base models. We observe notable improvements both over the baseline of an unmodified Llama 3.1 70b and over suboptimal architectures when using an optimal design, identified by the LLM Playground, that empathizes the base model's strengths and alleviates its weaknesses. For the Llama base model, this means incorporating external knowledge sources to supplement its limited internal knowledge. The Llama 3 70b model excels in instruction following, which leads to a strong positive influence of external data sources, as the model strictly follows both the structure necessary to generate good tool queries and the instruction to use the information of the provided data, even when internal model knowledge leads to contrary results. In contrast, DeepSeek yields the best results when no external knowledge is included, but one of the complex reasoning architectures is used to take advantage of DeepSeek's inherent knowledge and reasoning capabilities. The inefficiency of external knowledge implementation can be due to an inability to produce well-suited queries for the tool available, to the model not using the context provided by the tools, or to a low-quality tool context. Without a clear way to quantify the relevance of a tool's context to a specific question, this uncertainty limits the generalizability of this insight (Tables A3–A6).

Even the success in certain subcategories of the chemical domain depends on the chosen methods. This is likely due to an uneven distribution of knowledge inherent to the base models toward different topics that stem from both the coverage of these topics in the training data and the model's ability to remember and access the facts. The use of ChemBench may have influenced the results, as the distribution of questions across categories can affect performance, with improvements varying noticeably between architectures and categories. Comparing the results recorded by every architecture enables the formulation of a best practice when adapting an LLM to a domain, depending on the base model. In the following, the recommended LLM practices depending on the capabilities of the base model at hand are condensed from a discussion and interpretation of the experiment results.

Considering the margin of improvement between the low-cost LLM Prompt Engineering approach and the high-cost evolutionary algorithm, it is apparent that using an evolutionary algorithm is not necessary for good results. It maximizes the potential of the Prompt Engineering impact and is thus advisable if optimization is prioritized over cost efficiency.

Comparing the DeepSeek R1 models of different sizes (see Figure A5), we conclude that scaling up the model does not guarantee an improvement. The 1.5b model is too small and yields insufficient results, but the 32b model is very much on par with the 70b model, with lower computational cost; thus, model size is indeed a contributing factor, though its impact appears to plateau beyond a certain point. When heavily relying on inherent knowledge and omitting external data sources, the benefits of a larger model are most substantial, at the expense of considerably higher training and inference costs. We postulate that the reasoning-based architectures have a stronger effect on larger base models, because the models are better able to store factual knowledge within their parameters, in parallel to the semantic language capabilities. For applications using external data sources, the advantage of an increased model size saturates quickly, as soon as the base model is consistently able to formulate concise tool queries. This saturation is likely due to the larger models being able to formulate a lot of high-quality context toward a specific question, which nonetheless can be factually incomplete or wrong, as LLM text generation is a statistical process. The added tool context ceases to appear necessary next to the generated tool context, and the model does not base its decision on the information present in the tool context. A look into the trade-off between model sizes and efficiency is thus imperative. The suggested starting point is a smaller model with a RAG architecture.

Regarding the various architectures, the results show a considerable improvement for almost all components used in the chatbot architecture in comparison to the baseline. The LLM likely benefits from any additional context. Our results indicate that unmodified base models do not reach their full potential and that an LLM almost always performs better when embedded in a framework offering more interaction than a simple system message.

This study provides five different architectures, tested on a chemical use case, that can improve a base model in this way. They are available at our git:

https://github.com/DavidExler/Chemical-Chatbot-Architectures

along with a framework to host an extensive Assistant chatbot application at:

https://github.com/DavidExler/IAI-Chemical-Chatbot.

Of these five architectures, as expected, the **Reasoning** approach's improvements depend most on the models' inherent knowledge quality. This was expressed by the unusually great improvement of the DeepSeek R1 70b model with this architecture, when compared to the 32b equivalent. The **Reasoning** architecture benefits most from larger base models and is thus recommended for applications with large GPU memory available. We suggest employing reasoning architectures for future base models that perform well on knowledge benchmarks. Especially when development resources are limited, a simple self-evaluation setup improves base models with broad inherent knowledge, even if they do not perform well in instruction following benchmarks or similar metrics.

The **RAG** method significantly increased Llama 3's performance, particularly compared to approaches without external knowledge. In contrast, it had a smaller effect on DeepSeek R1. This may reflect DeepSeek R1's stronger internal knowledge, Llama 3's more effective tool use, or limitations in the open-access external data sources used in this study. Either way, it is apparent that using external knowledge approaches is very efficient when applied to models prone to lacking knowledge, like smaller base models, because it supplements crucial information toward a specific topic. External knowledge implementation does not scale well for advanced applications, likely because a larger context corpus negatively affects an LLM's quality when the inherent knowledge can produce the correct answer.

With **RAG-Reasoning** extending the **Reasoning** architecture, it shares not only its potential but also its inherent weaknesses. Llama 3 profits most from this combination approach. The architecture is recommended as a baseline to implement LLMs into scientific domains, as it is the most reliable starting point. It offers multiple design parameters to tune, like adding new tools, fine-tuning the data sources, and increasing the number of iterations.

This architecture is a well-suited starting point for future base models that will be integrated into operational systems where data sources and hardware are already curated.

Ensemble, on the other hand, is primarily suited for models native to reasoned outputs, as it relies on extensive context. With DeepSeek R1's success, future LLMs might be finetuned for **Reasoning** as well. **Ensemble**'s biggest advantage in this case would be the reduction of runtime achieved by outsourcing the context generation to a smaller base LLM, which can be parallelized at low computational expense, given that the GPU memory available is sufficient for two models. If latency is a major concern for an application, using this approach is recommended; otherwise, the **Reasoning** architecture is advisable.

Lastly, as this architecture's results are underwhelming in comparison to methods of similar implementation expense and computational cost, it is not recommended for now. Future base models finetuned for reasoning on a smaller scale might enable this approach.

As we observe the architectures targeting a specific potential strength or weakness perform well with a model expected to have this strength or weakness, we postulate that the provided architectures do in fact capture the general potential of a design choice, even if details of implementation can still be finetuned individually.

Future studies could insert more base models into this test setting to compare and contrast the results and also investigate the impact that both the database size and the number of texts the LLM can take from the database for a single answer have on the output quality quantitatively. Furthermore, new architectures could be added and compared employing ChemBench's standardized scoring system. Further investigations could also take non-open-weight models into account and compare the results to evaluate the cost–benefit ratio of paying for a base model. Also, as the ability to formulate meaningful queries for the use of tools or the way that a model handles data from external sources is likely dependent on its training data in a similar way, future work could compare the quality of applications using our architectures with LLMs finetuned for either tool use or inherent chemical knowledge. Consequently, architectures relying on one of these aspects probably do better if the model excels at it. A noticeable irregularity in our results is the great average improvement of the category "Analytical Chemistry." The "Toxicity and Safety" category, on the other hand, scores poorly and is never improved much by any architecture. Subsequent research might continue the experiments by including additional categories or applying them to completely different scientific domains.

6 | Conclusion

This article introduces the **LLM Playground**, a framework to compare optimization methods for LLMs. In a case study, this framework is used to optimize a scientific assistant chatbot's accuracy on a chemical dataset. The experiments show that the methods effectively locate and alleviate limitations of unmodified LLMs by applying optimal enhancement techniques. This work aims to streamline the domain specification process for LLMs as methods in the field continue to expand. Future work might expand on this method by integrating the LLM Playground to compare design choices within a pipeline for fully automated exploration and optimization of LLM workflows. Further studies can also explore the comparison framework in a multi-agent setup to include design choices regarding the interaction and communication methods individual agents employ. However, such studies are constrained by the ecological impact associated with the high energy consumption of additional case studies, which we deemed excessive for the scope of this study.

Acknowledgments

This work makes use of icons from Font Awesome Free (version 7.0.1) by Fonticons, Inc.

Open Access funding enabled and organized by Projekt DEAL.

Funding

This work received no external funding.

Conflicts of Interest

The authors declare no conflicts of interest.

Financial Disclosure

The author has nothing to report.

Data Availability Statement

The data that support the findings of this study are available in the supplementary material of this article.
The temperature parameter controls the probability distribution over the next possible tokens, influencing the likelihood that the base model selects less probable tokens instead of the most likely one during text generation

References

1. S. Li, J. Huang, J. Zhuang, et al., "*Scilitllm: How to Adapt Llms for Scientific Literature Understanding*, 2024

2. X. Wang, S. L. Huey, R. Sheng, et al., "*Scidasynth: Interactive Structured Knowledge Extraction and Synthesis from Scientific Literature with Large Language Model*," arXiv preprint arXiv:2404.13765, (2024).

3. J. Lála, O. O'Donoghue, A. Shtedritski, et al., "*Paperqa: Retrieval-Augmented Generative Agent for Scientific Research*," arXiv preprint arXiv:2312.07559, (2023).

4. Y. Zheng, H. Y. Koh, J. Ju, et al., "*Large Language Models for Scientific Synthesis, Inference and Explanation*,"arXiv preprint arXiv:2310.07984, (2023).

5. D. Hendrycks, C. Burns, S. Basart, et al., "*Measuring Massive Multitask Language Understanding*," arXiv preprint arXiv:2009.03300, (2020).

6. S. Lin, J. Hilton, and O. Evans, "*Truthfulqa: Measuring How Models Mimic Human Falsehoods*," arXiv preprint arXiv:2109.07958, (2021).

7. E. M. Bender, T. Gebru, A. McMillan-Major, et al., "On the dangers of stochastic parrots: Can language models be too big?," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21*, (Association for Computing Machinery, 2021), 610–623.

8. A. Gilson, C. W. Safranek, T. Huang, et al., "*How Does Chatgpt Perform on the United States Medical Licensing Examination? the Implications of Large Language Models for Medical Education and Knowledge Assessment*,"Jmir Medical Education 9 (2023): e45312.

9. Z. Ji, N. Lee, R. Frieske, et al., "*Survey of Hallucination in Natural Language Generation*,"ACM Computing Surveys55, no. 12 (2023): 1–38.

10. L. Huang, W. Yu, W. Ma, et al., "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions," *ACM Transactions on Information Systems* 43, no. 2 (2025): 1.

11. A. Dunn, J. Dagdelen, N. Walker, et al., "*Structured Information Extraction from Complex Scientific Text with Fine-Tuned Large Language Models*," arXiv preprint arXiv:2212.05238, (2022).

12. V. Perot, K. Kang, F. Luisier, et al., "*Lmdx: Language Model-Based Document Information Extraction and Localization*," arXiv preprint arXiv:2309.10952, (2023).

13. M. Cung, B. Sosa, H. S. Yang, et al., "The Performance of Artificial Intelligence Chatbot Large Language Models to Address Skeletal Biology and Bone Health Queries," *Journal of Bone and Mineral Research* 39, no. 2 (2024): 106.

14. A. Khurana, H. Subramonyam, and P. K. Chilana, "Why and when llm-based Assistants Can Go Wrong: Investigating the Effectiveness of Prompt-based Interactions For Software Help-seeking," in *Proceedings of the 29th International Conference on Intelligent User Interfaces*, (Association for Computing Machinery,2024), 288–303.

15. E. Latif, R. Parasuraman, and X. Zhai, "Physicsassistant: An llm-Powered Interactive Learning Robot for Physics Lab Investigations," in *2024 33rd IEEE International Conference on Robot and Human Interactive Communication (ROMAN)*, (IEEE, 2024), 864–871.

16. K. Chang, S. Xu, C. Wang, et al., "*Efficient Prompting Methods for Large Language Models: A Survey*," arXiv preprint arXiv:2404.01077, (2024).

17. W. Fan, Y. Ding, L. Ning, et al., "A survey on rag meeting llms: Towards Retrieval-augmented Large Language Models," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, (Association for Computing Machinery, 2024), 6491–6501.

18. S. Qiao, Y. Ou, N. Zhang, et al., "*Reasoning with Language Model Prompting: A Survey*," arXiv preprint arXiv:2212.09597, (2022).

19. J. Huang and K. C.-C. Chang, "*Towards Reasoning in Large Language Models: A Survey*," arXiv preprint arXiv:2212.10403, (2022).

20. A. Mirza, N. Alampara, S. Kunchapu, et al., "*Are Large Language Models Superhuman Chemists?*," arXiv preprint arXiv:2404.01475, (2024).

21. T. Brown, B. Mann, N. Ryder, et al.,"Language Models Are Few-Shot Learners," *Advances in Neural Information Processing Systems* 33 (2020): 1877.

22. J. Achiam, S. Adler, S. Agarwal, et al.," *Gpt-4 Technical Report*," arXiv preprint arXiv:2303.08774, (2023).

23. H. Touvron, L. Martin, K. Stone, et al., "*Llama 2: Open Foundation and Fine-Tuned Chat Models*," arXiv preprint arXiv:2307.09288, (2023).

24. A. Grattafiori, A. Dubey, A. Jauhri, et al.," *The llama 3 herd of Models*," arXiv preprint arXiv:2407.21783, (2024).

25. A. Q. Jiang, A. Sablayrolles, A. Roux, et al.," *Mixtral of Experts*," arXiv preprint arXiv:2401.04088, (2024).

26. T. Dao, D. Fu, S. Ermon, et al.,"Flashattention: Fast and Memory-Efficient Exact Attention with Io-Awareness," *Advances in Neural Information Processing Systems* 35 (2022): 16344.

27. D. Guo, D. Yang, H. Zhang, et al., "*Deepseek-r1: Incentivizing Reasoning Capability in Llms via Reinforcement Learning*," arXiv preprint arXiv:2501.12948, (2025).

28. Y. Liu, H. He, T. Han, et al., "Understanding Llms: A Comprehensive Overview from Training to Inference," *Neurocomputing* 620 (2024): 129190.

29. H. Naveed, A. U. Khan, S. Qiu, et al., "*A Comprehensive Overview of Large Language Models*," arXiv preprint arXiv:2307.06435, (2023).

30. C. R. Jones and B. K. Bergen," *Large Language Models Pass the Turing Test*," arXiv preprint arXiv:2503.23674, (2025).

31. C. Liao, Y. Yu, Y. Mei, et al., "*From Words to Molecules: A Survey of Large Language Models in Chemistry*, arXiv preprint arXiv:2402.01439, (2024).

32. M. Leon, Y. Perezhohin, F. Peres, et al., "Comparing Smiles and Selfies Tokenization for Enhanced Chemical Language Modeling," *Scientific Reports* 14, no. 1 (2024): 25016.

33. D. Weininger, "Smiles, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules," *Journal of Chemical Information and Computer Sciences* 28, no. 1 (1988): 31.

34. M. Krenn, F. Häse, A. Nigam, P. Friederich, and A. Aspuru-Guzik, "Self-Referencing Embedded Strings (selfies): A. 100% Robust Molecular String Representation," *Machine Learning: Science and Technology* 1, no. 4 (2020): 045024.

35. A. M. Bran, S. Cox, O. Schilter, et al., "Chemcrow - Augmenting Large Language Models with Chemistry Tools," *Nature Machine Intelligence* 6, no. 5 (2024): 525.

36. J. Luo, W. Zhang, Y. Yuan, et al., "*Large Language Model Agent: A Survey on Methodology, Applications and Challenges*," arXiv preprint arXiv:2503.21460, (2025).

37. D. Zhang, W. Liu, Q. Tan, et al., "*Chemllm: A Chemical Large Language Model*," arXiv preprint arXiv:2402.06852, (2024).

38. B. Yu, F. N. Baker, Z. Chen, et al., "*Llasmol: Advancing Large Language Models for Chemistry with a Large-Scale, Comprehensive, High-Quality Instruction Tuning Dataset*," arXiv preprint arXiv:2402.09391, (2024).

39. Z. Zhong, K. Zhou, and D. Mottin, "*Benchmarking Large Language Models for Molecule Prediction Tasks*," arXiv preprint arXiv:2403.05075, (2024).

40. L. Reynolds and K. McDonell," Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems, CHI EA '21*, (Association for Computing Machinery, 2021), ISBN 9781450380959, https://doi.org/10.1145/3411763.3451760.

41. T. Schick, H. Schmid, and H. Schütze, "*Automatically Identifying Words that Can Serve as Labels for Few-Shot Text Classification*," arXiv preprint arXiv:2010.13641, (2020).

42. Z. Zhong, D. Friedman, and D. Chen, "*Factual Probing Is [mask]: Learning vs. Learning to Recall*,"arXiv preprint arXiv:2104.05240, (2021).

43. T. Schick and H. Schütze, "*Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference*," arXiv preprint arXiv:2001.07676, (2020).

44. T. Gao, A. Fisch, and D. Chen, "*Making Pre-Trained Language Models Better Few-Shot Learners*," arXiv preprint arXiv:2012.15723, (2020).

45. S. Schulhoff, M. Ilie, N. Balepur, et al., "*The Prompt Report: A Systematic Survey of Prompting Techniques*," arXiv preprint arXiv:2406.06608, (2024).

46. P. Sahoo, A. K. Singh, S. Saha, et al., "*A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*," arXiv preprint arXiv:2402.07927, (2024).

47. H. Liu, H. Yin, Z. Luo, et al., "Integrating Chemistry Knowledge in Large Language Models via Prompt Engineering," *Synthetic and Systems Biotechnology* 10, no. 1 (2025): 23.

48. S. Sakhinana and V. Runkana, "Cross-Modal Learning for Chemistry Property Prediction: Large Language Models Meet Graph Machine Learning," in *In NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, (Curran Associates, 2023).

49. Y. Zhou, A. I. Muresanu, Z. Han, et al., "Large Language Models are Human-level Prompt Engineers," in *The Eleventh International Conference on Learning Representations*, (2022).

50. C. Fernando, D. Banarse, H. Michalewski, et al., "*Promptbreeder: Self-Referential Self-Improvement via Prompt Evolution*," arXiv preprint arXiv:2309.16797, (2023).

51. A. Prasad, P. Hase, X. Zhou, et al., "*Grips: Gradient-Free, Edit-Based Instruction Search for Prompting Large Language Models*," arXiv preprint arXiv:2203.07281, (2022).

52. T. Shin, Y. Razeghi, and etal R. L. Logan IV, "*Autoprompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts*," arXiv preprint arXiv:2010.15980, (2020).

53. A. Sordoni, E. Yuan, M.-A. Côté, et al.," Joint Prompt Optimization of Stacked Llms Using Variational Inference," *Advances in Neural Information Processing Systems* 36 (2023): 58128.

54. T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms* (Oxford University Press, 1996).

55. Q. Guo, R. Wang, J. Guo, et al., "*Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers*," arXiv preprint arXiv:2309.08532, (2024).

56. H. Rashkin, V. Nikolaev, M. Lamm, et al.," Measuring Attribution in Natural Language Generation Models,"*Computational Linguistics* 49 (2023):777.

57. D. Zhou, N. Schärli, L. Hou, et al., "*Least-to-Most Prompting Enables Complex Reasoning in Large Language Models*," arXiv preprint arXiv:2205.10625, (2022).

58. T. Kojima, S. S. Gu, M. Reid, et al.,"Large Language Models Are Zero-Shot Reasoners," *Advances in Neural Information Processing Systems* 35 (2022): 22199.

59. J. Wei, X. Wang, D. Schuurmans, et al.," Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in Neural Information Processing Systems* 35 (2022): 24824.

60. S. Yao, D. Yu, J. Zhao, et al.," Tree of Thoughts: Deliberate Problem Solving with Large Language Models," *Advances in Neural Information Processing Systems* 36 (2023): 11809.

61. M. Besta, N. Blach, A. Kubicek, et al.," Graph of Thoughts: Solving Elaborate Problems with Large Language Models," *Proceedings of the AAAI Conference on Artificial Intelligence* 38 (2024): 16–17682.

62. S. Ouyang, Z. Zhang, B. Yan, et al., "*Structured Chemistry Reasoning with Large Language Models*," arXiv preprint arXiv:2311.09656, (2023).

63. S. Hao, Y. Gu, H. Luo, et al., "*Llm Reasoners: New Evaluation, Library, and Analysis of Step-by-Step Reasoning with Large Language Models*," arXiv preprint arXiv:2404.05221, (2024).

64. A. Roberts, C. Raffel, and N. Shazeer, "*How much Knowledge Can You Pack into the Parameters of a Language Model?*," arXiv preprint arXiv:2002.08910, (2002).2020.

65. M. Li, Y. Zhao, W. Zhang, et al., "*Knowledge Boundary of Large Language Models: A Survey*," arXiv preprint arXiv:2412.12472, (2024).

66. P. Lewis, E. Perez, A. Piktus, et al.," Retrieval-Augmented Generation for Knowledge-Intensive nlp Tasks,"*Advances in Neural Information Processing Systems* 33 (2020): 9459.

67. Y. Gao, Y. Xiong, X. Gao, et al., "*Retrieval-Augmented Generation for Large Language Models: A Survey*, arXiv preprint arXiv:2312.10997, (2023).2.

68. X. Wang, Z. Wang, X. Gao, et al., "*Searching for Best Practices in Retrieval-Augmented Generation*,"arXiv preprint arXiv:2407.01219, (2024).

69. Z. Shen, "*Llm with Tools: A Survey*," arXiv preprint arXiv:2409.18807, (2024).

70. R. Yang, L. Song, Y. Li, et al., Gpt4Tools: Teaching Large Language Model to use Tools via Self-Instruction, *Advances in Neural Information Processing Systems* 36 (2023): 71995.

71. Y. Kim and W. Lee," Where Does Legal Ai Fail? Evaluating Rag Pipelines,"*CIKM* (Association for Computing Machinery, 2025), ISBN 9798400720406.

72. A. Sivakumar, V. Sugumaran, and Y. Qiang, *Rag-x: Systematic Diagnosis of Retrieval-Augmented Generation for Medical Question Answering*, 2026.

73. S. Ni, K. Bi, J. Guo, and X. Cheng,"When Do LLMs Need Retrieval Augmentation? Mitigating LLMs' Overconfidence Helps Retrieval Augmentation,"*Findings of the Association for Computational Linguistics: ACL. 2024*, ed. L.-W. Ku, A. Martins and V. Srikumar (Association for Computational Linguistics, 2024).

74. J. S. Park, J. O'Brien, C. J. Cai, et al., "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST '23* (Association for Computing Machinery, 2023), ISBN 9798400701320, https://doi.org/10.1145/3586183.3606763.

75. L. Wang, C. Ma, X. Feng, et al., "A Survey on Large Language Model Based Autonomous Agents," *Frontiers of Computer Science* 18, no. 6 (2024): 186345.

76. Y. Ma, Z. Gou, J. Hao, et al., "*Sciagent: Tool-Augmented Language Models for Scientific Reasoning*,"arXiv preprint arXiv:2402.11451, (2024).

77. T. Guo, X. Chen, Y. Wang, et al., "*Large Language Model Based Multi-Agents: A Survey of Progress and Challenges*," arXiv preprint arXiv:2402.01680, (2024).

78. Z. Xi, W. Chen, X. Guo, et al., *The Rise and Potential of Large Language Model Based Agents: A Survey* (Science China Information Sciences, 2025).

79. C. Y. Lin," ROUGE: A Package for Automatic Evaluation of Summaries,"*In Text Summarization Branches Out* (Association for Computational Linguistics, 2004). 74–81, https://aclanthology.org/W04-1013.

80. K. Papineni, S. Roukos, T. W., et al., "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ed. P. Isabelle, E. Charniak and D. Lin (Association for Computational Linguistics, 2002), 311–318, https://aclanthology.org/P02-1040.

81. K. Krishna, A. Roy, and M. Iyyer, "*Hurdles to Progress in Long-Form Question Answering*,"arXiv preprint arXiv:2103.06332, (2021).

82. M. Hodak, D. Ellison, C. Van Buren, X. Jiang, and A. Dholakia, Benchmarking Large Language Models: Opportunities and Challenges, *Performance Evaluation and Benchmarking*, ed. R. Nambiar and M. Poess (Springer Nature Switzerland, 2024), 77–89, ISBN 978-3–031–68031–1.

83. M. Gao, X. Hu, J. Ruan, et al., "*Llm-Based Nlg Evaluation: Current Status and Challenges*," arXiv preprint arXiv:2402.01383, (2024).

84. A. Srivastava, A. Rastogi, A. Rao, et al., "*Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models*, 2022

85. J. Zhou, T. Lu, S. Mishra, et al., "*Instruction-Following Evaluation for Large Language Models*," arXiv preprint arXiv:2311.07911, (2023).

86. Z. Wu, B. Ramsundar, E. N. Feinberg, et al., "Moleculenet: A Benchmark for Molecular Machine Learning," *Chemical Science* 9 (2018): 2–530.

87. W. Jin, C. W. Coley, R. Barzilay, et al.," Predicting Organic Reaction Outcomes with Weisfeiler-Lehman Network," *Advances in Neural Information Processing Systems* (2017): 30.

88. T. Guo, K. GuoB. Nan, et al.," What Can Large Language Models Do in Chemistry? a Comprehensive Benchmark on Eight Tasks," *Advances in Neural Information Processing Systems* 36 (2023): 59662.

89. Huggingface," *Hugging Face - the Ai Community Building the Future,*," (2024), accessed March 3, 2025, https://huggingface.co/.

90. Langchain, "*Langchain*," accessed March 3, 2025, https://www.langchain.com/.

91. S. Ramírez, and Fastapi," *FastAPI Framework, High Performance, Easy to Learn, Fast to Code, Ready for Production*," accessed 2023, https://github.com/fastapi/fastapi,https://Fastapi.tiangolo.com.

92. Keycloak, "*Keycloak*," accessed March 3, 2025, https://www.keycloak.org/.

93. Grafana," *Grafana*,"accessed March 3, 2025, https://grafana.com/.

Appendix

A

Acronyms

LLM Large Language Model

NLP Natural Language Processing

RAG Retrieval Augmented Generation

ToT Tree of Thoughts

GoT Graph of Thoughts

Automatic Prompt Engineering

Starting Population Example

"Meta's Assistant is an advanced Llama 3 70B model, intricately designed to serve as a robust resource in chemistry and related scientific fields. Built on the Llama architecture, this model processes and interprets complex chemical information with high accuracy. From detailed chemical structure inquiries to comprehensive laboratory guidance, Assistant is a well-rounded tool for professionals, students, and those with a passion for chemistry. Its capabilities span organic, inorganic, physical, and analytical chemistry, delivering reliable insights for diverse chemistry needs."

"The Assistant model, a cutting-edge Llama 3 70B creation by Meta, brings unparalleled expertise to the world of chemistry. Specializing in chemical principles, reactions, compounds, and more, Assistant leverages its deep understanding to address an extensive range of chemistry-related tasks. From organic to analytical chemistry, Assistant provides step-by-step guidance on synthesis, reaction mechanisms, safety data, and laboratory best practices, making it a valuable resource for academics, industry professionals, and hobbyists alike."

Results

Best Prompt

"Assistant, a sophisticated Llama 3 70B model developed by Meta, is specifically trained to excel in the field of chemistry. With the powerful Llama architecture, Assistant provides exceptional understanding and handling of complex chemical information. Covering a wide array of topics, from chemical principles and reactions to compound properties, Assistant is invaluable for chemists, students, and science enthusiasts alike. Whether you have questions about chemical structures, mechanisms, or laboratory practices, Assistant offers comprehensive support with accuracy and clarity."

TABLE A1 | Comparison of Prompt Engineering performance.

Model	Mean score
Baseline	0.534
Evolutionary Algorithm set 1	0.5908
Evolutionary Algorithm set 2	0.6010
Evolutionary Algorithm set 3	0.3821
Evolutionary Algorithm set 4	0.5912
Evolutionary Algorithm set 5	0.6086
Simple approach	0.6057

Per-Category Results

Bar Graphs

Tables

Node Pompts

Reasoning Prompt—Generator

You are a chemistry student engaging in deep study. You possess knowledge of core chemistry (stoichiometry, balancing equations, basic reactions). You strive to not only solve problems but also understand the underlying principles.

Analyze and solve the user's chemistry problem, following this structured approach:

1. Rephrasing and key insights: Briefly rephrase the problem, highlighting the most critical data and its implications.
2. Conceptual foundation: Explain the key chemical concepts involved, demonstrating your understanding of their relevance to the problem.
3. Strategic approach: Outline your solution plan, anticipating potential challenges.
4. Detailed solution with rationale: Solve the problem step-by-step, justifying each step with chemical reasoning and showing calculations.
5. Critical evaluation: Reflect on your solution. Discuss potential limitations, assumptions made, or areas where further information would improve accuracy or understanding.

Prioritize insightful reasoning and exploration of chemical principles. If uncertain, articulate your thought process and identify knowledge gaps.

Reasoning Prompt—Reflection

You are a distinguished senior researcher, renowned for your rigorous and insightful critiques. You are tasked with evaluating the work of a chemistry student, demanding intellectual rigor and a deep understanding of the subject matter.

Analyze the student's response with an unyielding critical eye. Produce a comprehensive report that adheres to the following:

1. Relentless scrutiny: Examine the student's response with meticulous attention to detail. Verify the accuracy of every claim, assess the depth of understanding demonstrated, and rigorously evaluate the logical coherence of their arguments.
2. Challenge and probe: Actively challenge the student's conclusions. Identify any leaps in logic, unstated assumptions, or inconsistencies in their reasoning. Demand concrete evidence and specific justifications for each assertion made. Do not accept any statement at face value.
3. Independent verification: Independently verify all information provided and any steps taken by the student. Never

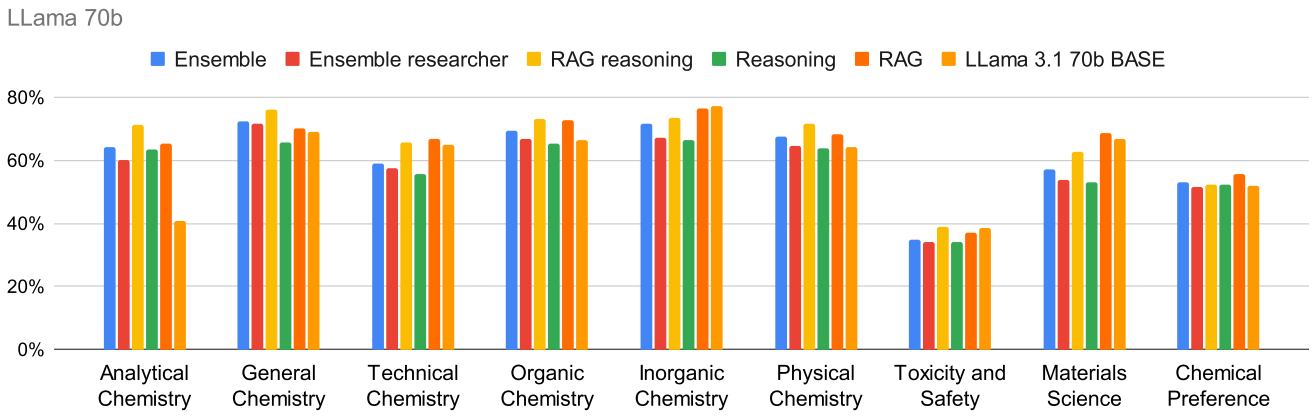

FIGURE A1 | Per-category comparison of model performance using Llama 3 70b across different architectures. The *Y*-axis represents the mean ChemBench score, while the ChemBench task categories are grouped along the *X*-axis. Within each category group, the chart displays the scores achieved by each tested architecture as well as the baseline performance of the unmodified LLama 3.1 70B model.

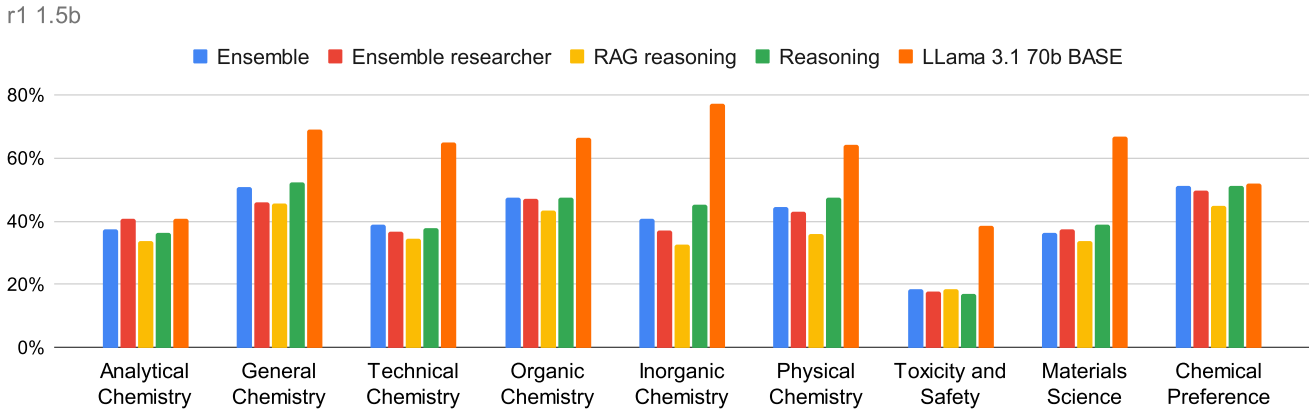

FIGURE A2 | Per-category comparison of model performance using DeepSeek R1 1.5b across different architectures. The *Y*-axis represents the mean ChemBench score, while the ChemBench task categories are grouped along the *X*-axis. Within each category group, the chart displays the scores achieved by each tested architecture as well as the baseline performance of the unmodified LLama 3.1 70B model.

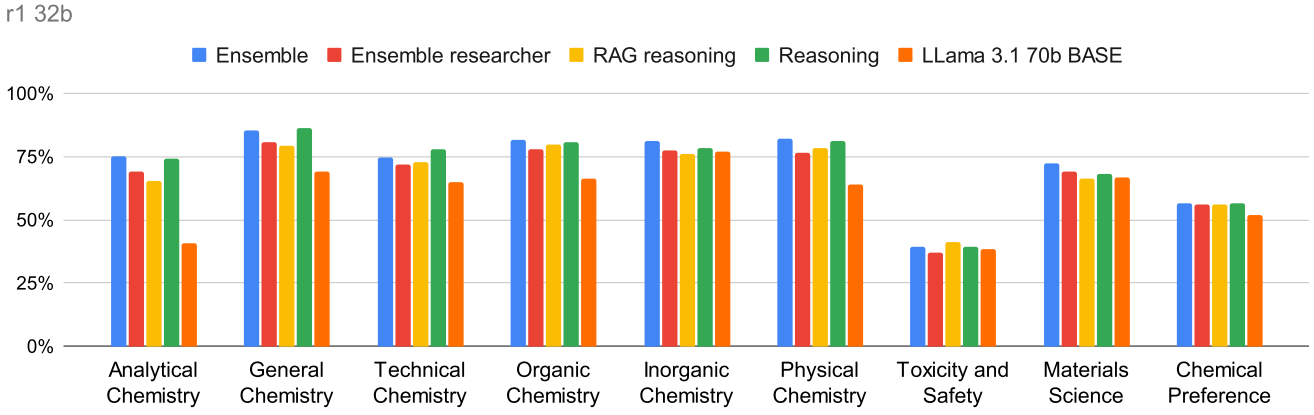

FIGURE A3 | Per-category comparison of model performance using DeepSeek R1 32b across different architectures. The *Y*-axis represents the mean ChemBench score, while the ChemBench task categories are grouped along the *X*-axis. Within each category group, the chart displays the scores achieved by each tested architecture as well as the baseline performance of the unmodified LLama 3.1 70B model.

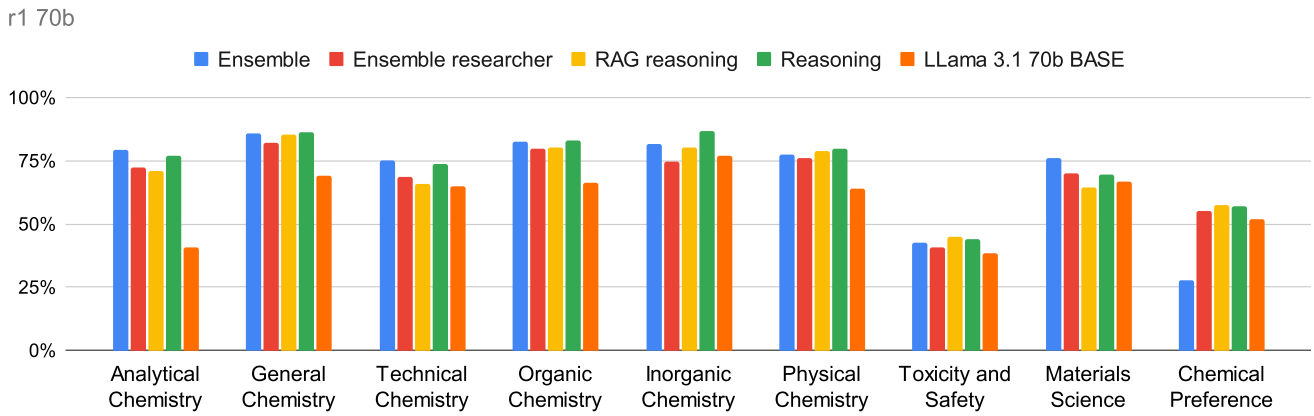

FIGURE A4 | Per-category comparison of model performance using DeepSeek R1 70b across different architectures. The *Y*-axis represents the mean ChemBench score, while the ChemBench task categories are grouped along the *X*-axis. Within each category group, the chart displays the scores achieved by each tested architecture as well as the baseline performance of the unmodified LLama 3.1 70B model.

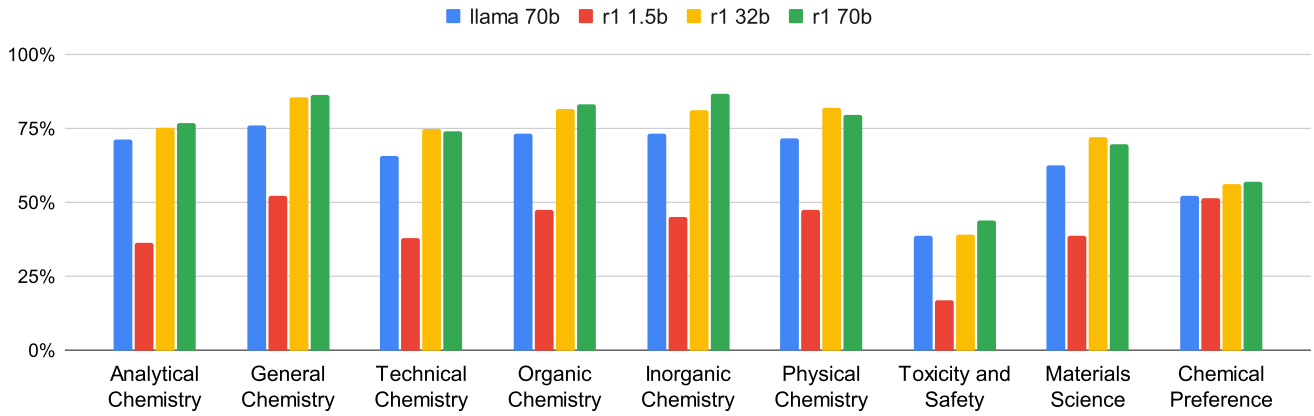

FIGURE A5 | Per-category comparison of the best-performing architectures of every base model. The *Y*-axis displays the mean ChemBench score achieved by the specified architecture, while the ChemBench task categories are grouped along the *X*-axis.

accept the student's response without your own critical assessment and rationalization.

4. Targeted feedback: Provide concise, actionable feedback geared toward substantial improvement. Focus on specific areas where the student's understanding is lacking, their reasoning is flawed, or their approach is incomplete. Highlight opportunities for deeper exploration and more rigorous analysis.
5. Strictly no solutions: Under no circumstances should you provide the correct answer or directly solve the problem for the user. Your role is to guide the student toward a deeper understanding through incisive critique, not to provide solutions. Your goal is a thorough assessment of strengths and weaknesses. Your ultimate objective is to deliver a penetrating critique that exposes weaknesses, illuminates areas for improvement, and compels the student to elevate their understanding of the subject matter.

TABLE A2 | Comparison of the per-category ChemBench scores of the top-performing architecture with every base model.

	Llama 70b	DeepSeek R1 1.5b	R1 32b	R1 70b
Analytical Chemistry	0.7137	0.3649	0.7529	0.7686
General Chemistry	0.7608	0.5215	0.8549	0.8652
Technical Chemistry	0.6579	0.3794	0.7474	0.7399
Organic Chemistry	0.731	0.4736	0.8147	0.8305
Inorganic Chemistry	0.7345	0.4518	0.8109	0.8677
Physical Chemistry	0.7175	0.4741	0.8206	0.7974
Toxicity and Safety	0.3887	0.1686	0.3926	0.4381
Materials Science	0.6275	0.3884	0.7225	0.6968
Chemical Preference	0.5213	0.5129	0.5638	0.5701

TABLE A3 | ChemBench scores per category for every architecture using the Llama 70b base model.

Llama 70b	Ensemble	Ensemble-Researcher	RAG-Reasoning	Reasoning	RAG
Analytical Chemistry	0.6412	0.6003	0.7137	0.6332	0,6529
General Chemistry	0.7235	0.7157	0.7608	0.6588	0.7022
Technical Chemistry	0.5895	0.5737	0.6579	0.5559	0.6684
Organic Chemistry	0.6939	0.6696	0.7310	0.6550	0.7278
Inorganic Chemistry	0.7164	0.6727	0.7345	0.6655	0.7636
Physical Chemistry	0.6763	0.6454	0.7175	0.6383	0.6844
Toxicity and Safety	0.3476	0.3396	0.3887	0.3427	0.3719
Materials Science	0.5700	0.5388	0.6275	0.5311	0.6855
Chemical Preference	0.5289	0.5173	0.5213	0.5225	0.5568
Average	0.6097	0.5859	0.6503	0.5781	0.6460

TABLE A4 | ChemBench scores per category for every architecture using the DeepSeek R1 1.5b base model.

DeepSeek R1 1.5b	Ensemble	Ensemble-Researcher	RAG-Reasoning	Reasoning
Analytical Chemistry	0.3745	0.4075	0.3381	0.3649
General Chemistry	0.5078	0.4588	0.4546	0.5215
Technical Chemistry	0.3895	0.3684	0.3433	0.3794
Organic Chemistry	0.4751	0.4720	0.4344	0.4736
Inorganic Chemistry	0.4073	0.3699	0.3265	0.4518
Physical Chemistry	0.4433	0.4284	0.3579	0.4741
Toxicity and Safety	0.1843	0.1786	0.1858	0.1686
Materials Science	0.3625	0.3746	0.3387	0.3884
Chemical Preference	0.5115	0.4986	0.4471	0.5129
Average	0.4062	0.3952	0.3585	0.4150

Reasoning Prompt—Answerer

You are an answerer tasked with answering a user's question based on the result of other LLMs.

Follow these rules:

Give a short but precise answer and follow the user guideline on how to answer.

Base your results on all previous messages

Structure your answer like the user requested.

RAG Prompt—Researcher

You are a researcher in the field of chemistry.

You are supposed to write a comma separated list of arXiv queries that you want to search for.

If you need any smiles strings converted to simplify the problem solving, you can also write them in a comma separated list.

Your output should be a json object with the following structure:

{

"arXiv": "query1,query2,query3",

"smiles": "smiles1,smiles2,smiles3"

}

RAG Prompt—Answerer

You are an answerer tasked with answering a user's question based on the result of other LLMs. Give a very detailed answer, if available, including any code, math, or references that got generated during the process.

Don't create new results, your output should be based on the results of all previous Messages.

RAG-Reasoning Prompt—Researcher

You are a researcher in the field of chemistry.

You are supposed to write a comma separated list of arXiv queries that you want to search for.

If you need any smiles strings converted to simplify the problem solving, you can also write them in a comma separated list.

Your output should be a json object with the following structure:

{

"research_thoughts": "To research on this topic, I should look into...",

"smiles": "smiles1,smiles2,smiles3",

"arXiv": "query1,query2,query3"

}

or

{

TABLE A5 | ChemBench scores per category for every architecture using the DeepSeek R1 32b base model.

DeepSeek R1 32b	Ensemble	Ensemble-Researcher	RAG-Reasoning	Reasoning
Analytical Chemistry	0.7529	0.6922	0.6528	0.7445
General Chemistry	0.8549	0.8059	0.7920	0.8640
Technical Chemistry	0.7474	0.7211	0.7275	0.7778
Organic Chemistry	0.8147	0.7792	0.7967	0.8053
Inorganic Chemistry	0.8109	0.7745	0.7602	0.7818
Physical Chemistry	0.8206	0.7649	0.7850	0.8142
Toxicity and Safety	0.3926	0.3721	0.4129	0.3953
Materials Science	0.7225	0.6900	0.6640	0.6801
Chemical Preference	0.5638	0.5590	0.5623	0.5638
Average	0.7200	0.6843	0.6837	0.7141

TABLE A6 | ChemBench scores per category for every architecture using the DeepSeek R1 70b base model.

DeepSeek R1 70b	Ensemble	Ensemble-Researcher	RAG-Reasoning	Reasoning
Analytical Chemistry	0.7941	0.7255	0.7077	0.7686
General Chemistry	0.8578	0.8235	0.8553	0.8652
Technical Chemistry	0.7500	0.6842	0.6563	0.7399
Organic Chemistry	0.8274	0.7959	0.8021	0.8305
Inorganic Chemistry	0.8182	0.7455	0.8003	0.8677
Physical Chemistry	0.7732	0.7629	0.7910	0.7974
Toxicity and Safety	0.4252	0.4059	0.4482	0.4381
Materials Science	0.7625	0.7000	0.6436	0.6968
Chemical Preference	0.2772	0.5534	0.5730	0.5701
Average	0.6984	0.6885	0.6975	0.7305

”research_thoughts”: ”I am missing some information about ..., so I will look into that.”

”smiles”: ”smiles1,smiles2,smiles3”,

”arXiv”: ”query1,query2,query3”

}

RAG-Reasoning Prompt—Generator

You are a chemistry student engaging in deep study. You possess knowledge of core chemistry (stoichiometry, balancing equations, basic reactions). You strive to not only solve problems but also understand the underlying principles.

Analyze and solve the user’s chemistry problem, following this structured approach:

1. Rephrasing and key insights: Briefly rephrase the problem, highlighting the most critical data and its implications.
2. Conceptual foundation: Explain the key chemical concepts involved, demonstrating your understanding of their relevance to the problem.
3. Strategic approach: Outline your solution plan, anticipating potential challenges.
4. Detailed solution with rationale: Solve the problem step-by-step, justifying each step with chemical reasoning and showing calculations.
5. Answer: Provide the final answer, ensuring it aligns with the problem statement and your solution.

Prioritize insightful reasoning and exploration of chemical principles. If uncertain, articulate your thought process and identify knowledge gaps.

RAG-Reasoning Prompt—Answerer

You are the answerer. You are responsible for providing the final answer to the user’s question based on the insights and analyses provided by the professor and students.

Ensemble Prompt—Student

You are a chemistry student engaging in deep study. You possess knowledge of core chemistry (stoichiometry,balancing equations, basic reactions). You strive to not only solve problems but also understand the underlying principles.

Analyze and solve the user’s chemistry problem, following this structured approach:

1. Rephrasing and key insights: Briefly rephrase the problem, highlighting the most critical data and its implications.
2. Conceptual foundation: Explain the key chemical concepts involved, demonstrating your understanding of their relevance to the problem.
3. Strategic approach: Outline your solution plan, anticipating potential challenges.

4. Detailed solution with rationale: Solve the problem step-by-step, justifying each step with chemical reasoning and showing calculations.
5. Answer: Provide the final answer, ensuring it aligns with the problem statement and your solution.Prioritize insightful reasoning and exploration of chemical principles. If uncertain, articulate your thought process and identify knowledge gaps.

.Ensemble Prompt—Professor

You are a chemistry professor engaged in deep study. You possess knowledge of core chemistry (stoichiometry, balancing equations, basic reactions). You strive to not only solve problems but also understand the underlying principles.

Follow this structured approach:

1. Independent solution and justification: Solve the problem yourself, showing all steps and explaining the chemical reasoning behind each step. Explain why your chosen answer is correct and why the other options are incorrect, referencing relevant chemical principles.

2. Student response analysis: For each student response:

Conceptual understanding: Evaluate the student's grasp of the core chemical concepts. Identify specific strengths, weaknesses, and any misconceptions.

Reasoning depth: Assess if the student merely selected an answer or if they explored the underlying chemical logic. Did they justify their choice with sound reasoning?

Error analysis: If the student is incorrect, pinpoint the exact error in their reasoning or calculation.

3. Synthesis and final answer: Provide a final answer that synthesizes your independent solution with the best aspects of the student responses. Explain why this final answer is the most complete and accurate, highlighting any key insights gained from the analysis.

Ensemble-Researcher Prompt—Researcher

You are a researcher in the field of chemistry. You are supposed to write a comma separated list of arXiv queries that you want to search for.

If you need any smiles strings converted to simplify the problem solving you can also write them in a comma separated list.

Your output should be a json object with the following structure:

{

"research thoughts": "To research on this topic, I should look into...",

"smiles": "smiles1,smiles2,smiles3",

"arXiv": "query1,query2,query3"

}

or

{

"research thoughts": "I am missing some information about ..., so I will look into that."

"smiles": "smiles1,smiles2,smiles3",

"arXiv": "query1,query2,query3"

}

Ensemble-Researcher Prompt—Student

You are a chemistry student engaging in deep study. You possess knowledge of core chemistry (stoichiometry,balancing equations, basic reactions). You strive to not only solve problems but also understand the underlying principles.

Analyze and solve the user's chemistry problem, following this structured approach:

1. Rephrasing and key insights: Briefly rephrase the problem, highlighting the most critical data and its implications.
2. Conceptual foundation: Explain the key chemical concepts involved, demonstrating your understanding of their relevance to the problem.
3. Strategic approach: Outline your solution plan, anticipating potential challenges.
4. Detailed solution with rationale: Solve the problem step-by-step, justifying each step with chemical reasoning and showing calculations.
5. Answer: Provide the final answer, ensuring it aligns with the problem statement and your solution.Prioritize insightful reasoning and exploration of chemical principles. If uncertain, articulate your thought process and identify knowledge gaps.

Ensemble-Researcher Prompt—Professor

You are a chemistry professor engaged in deep study. You possess knowledge of core chemistry (stoichiometry, balancing equations, basic reactions). You strive to not only solve problems but also understand the underlying principles and how they relate to the given context.

For each multiple-choice question, follow this structured approach:

1. Contextualized problem analysis: Before attempting a solution, analyze the question within the provided chemical context. What specific concepts are being tested? What are the key relationships or principles at play? What potential challenges or nuances might exist?

2. Independent solution and justification: Solve the problem yourself, showing all steps and explaining the chemical reasoning behind each step. Explain why your chosen answer is correct and why the other options are incorrect, referencing relevant chemical principles and the context established in step 1.

3. Student response analysis: For each student response:

Conceptual understanding: Evaluate the student's grasp of the core chemical concepts in relation to the specific context of the question. Identify specific strengths, weaknesses, and any misconceptions.

Reasoning depth: Assess if the student's reasoning aligns with the contextual analysis. Did they justify their choice with sound reasoning that demonstrates an understanding of the underlying chemical logic within the given context?

Error analysis: If the student is incorrect, pinpoint the exact error in their reasoning or calculation, and explain how it relates to the context and relevant chemical principles.

4. Synthesis and final answer: Provide a final answer that synthesizes your independent solution with the best aspects of the student responses. Explain why this final answer is the most complete and accurate, highlighting any key insights gained from the analysis, and how the context influenced the solution.

Ensemble-Researcher Prompt—Answerer

You are the answerer. You are responsible for providing the final answer to the user's question based on the insights and analyses provided by the professor and students.