# HoneyComb: A Flexible LLM-Based Agent System for Materials Science

**Huan Zhang[1,2], Yu Song[1,2], Ziyu Hou[4], Santiago Miret[3]*, Bang Liu[1,2,5]*†**
[1] DIRO & Institut Courtois, Université de Montréal
[2]Mila - Quebec AI Institute, [3]Intel Labs
[4]University of Waterloo, [5]Canada CIFAR AI Chair
{huan.zhang, yu.song, bang.liu}@umontreal.ca
{z26hou}@uwaterloo.ca, {santiago.miret}@intel.com

## Abstract

The emergence of specialized large language models (LLMs) has shown promise in addressing complex tasks in materials science. Many LLMs, however, often struggle with the distinct complexities of materials science tasks, such as computational challenges, and rely heavily on outdated implicit knowledge, leading to inaccuracies and hallucinations. To address these challenges, we introduce *HoneyComb*, the first LLM-based agent system specifically designed for materials science. HoneyComb leverages a reliable, high-quality materials science knowledge base (MatSciKB) and a sophisticated tool hub (ToolHub) tailored specifically for materials science to enhance its reasoning and computational capabilities. MatSciKB is a curated, structured knowledge collection based on reliable literature, while ToolHub employs an Inductive Tool Construction method to generate, decompose, and refine API tools for materials science. Additionally, HoneyComb leverages a retriever module that adaptively selects the appropriate knowledge source or tools for specific tasks, thereby ensuring accuracy and relevance. Our results demonstrate that HoneyComb significantly outperforms baseline models across various tasks in materials science, effectively bridging the gap between current LLM capabilities and the specialized needs of this domain. Furthermore, our adaptable framework can be easily extended to other scientific domains, highlighting its potential for broad applicability in advancing scientific research and applications. The code is available. [1]

## 1 Introduction

The emergence of large language models (LLMs) (OpenAI, 2024; Anthropic, 2024; Touvron et al., 2023b,a) in recent years has brought about the application of LLMs across a wide range of fields related to science and engineering (AI4Science and Quantum, 2023). This has resulted in a number of new benchmarks measuring the capabilities of language models to perform scientific tasks (Wang et al., 2023; Sun et al., 2024; Mirza et al., 2024; Song et al., 2023a) along with the development of custom LLMs and LLM-based systems for scientific domains, including chemistry (Bran et al., 2023; Boiko et al., 2023), biology (Madani et al., 2023) and materials science (Song et al., 2023b; Gupta et al., 2022; Walker et al., 2021).

While much progress has been made in adapting LLMs to common tasks in natural language processing (Song et al., 2023a,b), many more challenges remain in having LLMs be effective agents for real-world materials science tasks (Miret and Krishnan, 2024; Miret et al., 2024). As highlighted by Zaki et al. (2023), LLMs often fail in performing advanced tasks for materials science. Common mistakes by most LLMs include conceptual errors where models fail to retrieve correct concepts, equations, or facts relevant to the questions, and factual hallucinations where incorrect information is generated. An analysis by Miret and Krishnan (2024) also revealed that LLMs by themselves struggle to generate relevant and correct information pertaining to specialized materials science tasks. While Song et al. (2023b) showed that instruction fine-tuning can help in improving performance, the high costs of continuous model training and fine-tuning make retraining-based approaches challenging to scale. Moreover, this challenge is further compounded by the fact that relevant knowledge is continuously updated from diverse sources—including pre-print servers (e.g., arXiv and ChemRxiv), peer-reviewed literature, open encyclopedias like Wikipedia, and various relevant websites.

Furthermore, prior work has shown that utilizing external tools may be a more promising approach to solve complex scientific tasks rather instead of

*Equal advising.
†Corresponding author.
[1]https://github.com/BangLab-UdeM-Mila/NLP4MatSci-HoneyComb

relying entirely on an LLM's internal knowledge (Zheng et al., 2024; Buehler, 2024a). To jointly address these challenges, we propose transforming LLMs into LLM-based agents that access external knowledge and tools to boost their performance. This approach has already shown promise in adjacent domains, such as chemistry (Bran et al., 2023; Boiko et al., 2023), by enabling models to access real-time data and utilize both general and domain-specific tools. Altogether, the LLM-based agents showcase greater capabilities and performance compared to their native LLM counterparts.

In this paper, we present **HoneyComb**, the first, to the best of our knowledge, LLM-based agent system specifically designed for materials science. While there has been emerging research in LLMs for scientific domains, few studies have focused on developing comprehensive agent systems for materials science. Our work addresses two critical challenges: First, **MatSciKB** alleviates the challenge of obtaining reliable and relevant professional knowledge for materials science. As such, MatSciKB ensures the agent has access to the most current and accurate information, which is essential for effective performance. Second, **ToolHub** provides materials science-specific tools to augment the agent's capabilities. These tools enable the agent to perform specialized computational tasks and enhance its overall functionality. As detailed in Section 4, we observe that with the aid of MatSciKB and ToolHub, HoneyComb outperforms its native LLM counterparts in a more reliable way, given its ability to utilize up-to-date knowledge and tools.

## 2 Background

### 2.1 LLMs for Material Science

Advancements in text mining and information extraction from scientific publications have significantly benefited the application of LLMs for materials science (Kononova et al., 2021; Swain and Cole, 2016). Early work includes the development of specialized BERT models (Devlin et al., 2018), such as MatSciBERT (Gupta et al., 2022) and MatBERT (Walker et al., 2021). Song et al. (2023b) and Xie et al. (2023) leveraged instruction fine-tuning to develop a LLaMA-based model (Touvron et al., 2023a) tailored to materials science that matched the capabilities of commercial LLMs at the time of publication. The emergence of powerful commercial LLMs (OpenAI, 2024; Anthropic, 2024) has further expanded the possibility of applying LLMs to materials science. Yet, commercial LLMs remain expensive and opaque in their methodology, with consistent errors and shortcomings (Zaki et al., 2023; Miret and Krishnan, 2024), and open-source LLMs for materials science remain sparse. This motivates the need for a practical LLM-based system that is useful for real-world materials science tasks.

Given this need, we propose HoneyComb as an open-source system to augment the capabilities of diverse LLMs. HoneyComb integrates specialized tools as well as a dynamic retrieval system to enhance the functionality of any LLMs specifically for materials science. By leveraging relevant knowledge sources through MatSciKB and auxiliary tools through ToolHub, HoneyComb manages to improve the accuracy and relevance of the outputs of LLMs for materials science, while also addressing common challenges associated with static LLM applications in dynamic research fields.

### 2.2 Tool-Based LLM Agents for Scientific Applications

Prior work has shown success in expanding the capabilities of LLMs by augmenting them with diverse sets of tools (Qin et al., 2023b,a; Chern et al., 2023; Wang et al., 2024). Many works rely on pre-built integration frameworks, such as LangChain (Topsakal and Akinci, 2023), to build the relevant interfaces between the LLMs and the desired capabilities, such as search engine APIs. Wang et al. (2024) provides a recent survey of common approaches, challenges, and applications of tool-based LLMs in various technological and scientific fields.

One major application of tool-based LLMs is in query processing and optimization, where agents evaluate initial search results and iteratively refine queries to increase relevance and accuracy (Buehler, 2024a,b). This approach addresses the limitations of isolated LLMs, which may struggle to handle ambiguous query contexts. When generating structured datasets for solar cell materials, agents gather pertinent information from a vast array of scientific papers to automate data input and synthesis (Xie et al., 2024; Liu et al., 2024b). Furthermore, agents can utilize various tools to answer specific questions by tapping into external resources (Cheng et al., 2024). For example, ChemCrow, by Bran et al. (2023), integrates 18 expert-designed tools—such as literature search, molecule modification, and reaction execution—to

autonomously execute chemical syntheses. Tool augmentation has also been successful in other research within the chemistry domain, enabling real-world experiments using LLMs (Yoshikawa et al., 2023; Jablonka et al., 2023; Boiko et al., 2023). Coscientist, by Boiko et al. (2023), for example, relies on specialized tools to extend the capabilities of GPT-4, invoking domain-specific functionalities that are not inherently present within the LLM alone. The success of agent-based approaches in adjacent domains motivates the creation of HoneyComb, which extends the capabilities of LLMs specifically for materials science.

## 3 HoneyComb

In this work, we introduce HoneyComb, a specialized agent system designed to advance materials science research, as shown in Figure 1. It integrates three key components: 1) *MatSciKB*, a comprehensive knowledge base; 2) *ToolHub*, which includes general tools for broadly accessing up-to-date information and specialized tools developed through the Inductive Tool Construction method for targeted materials science queries; and 3) *Retriever*, which utilizes a hybrid approach for efficient and precise information retrieval.

### 3.1 MatSciKB

Our MatSciKB knowledge base integrates a diverse array of sources, as detailed in Table 1. This collection is meticulously curated to include materials science papers from arXiv, relevant Wikipedia entries, textbooks, comprehensive datasets, pertinent mathematical formulas, and concrete GPT-generated examples tailored to materials science. Each information source is thoroughly described in Appendix A.

MatSciKB is structured into 16 distinct categories relevant to materials science, as detailed in Appendix C and organized in a tree-like structure. MatSciKB supports efficient searching and CRUD (Create, Read, Update, Delete) operations (Giannaros et al., 2023), which are vital for both application and ongoing database maintenance. Given the continuously evolving and expanding body of knowledge in the materials science domain, efficient update and search capabilities based on real-time information are crucial for research and engineering applications. Additionally, our structured data approach enhances the integration of diverse data sources commonly encountered in materials science (Miret and Krishnan, 2024). This structure not only facilitates easy access and management but also allows for seamless extension to include additional data modalities.

| **MatSciKB** | |
|---|---|
| # Total Number of Data Entries | 38,469 |
| # Materials Science Papers on Arxiv | 20,384 |
| # Wikipedia for Material Science | 3,620 |
| # Materials Science Textbook | 1,930 |
| # Materials Science Dataset | 10,473 |
| # Materials Science Formula | 57 |
| # GPT-generated Examples | 2,005 |

Table 1: Summary of MatSciKB Data Sources and Entry Counts

### 3.2 ToolHub

The ToolHub in HoneyComb is bifurcated into *General Tools* and *Material Science Tools*. Both categories are organized through a unified interface that allows HoneyComb to make effective use of all available tools. **General Tools** provide researchers with access to the latest information, filling gaps not covered by the static entries in MatSciKB. **Material Science Tools** are specifically designed to handle complex calculations and in-depth analyses. The details of the unified interface are further elucidated in Appendix D.

**General Tools Construction**

In materials science, one of the persistent challenges is keeping research outputs aligned with the diverse and ever-evolving data modalities that describe complex material systems (Miret and Krishnan, 2024). The diversity of data sources and measurements leads to a rapid evolution of knowledge in this field, necessitating tools that can effectively access and integrate recent findings. Traditional static databases, while useful, often lag in capturing the newest research, creating gaps that can impede the currency and relevance of scientific analysis in real time (Brayne and Moffitt, 2022). Furthermore, the need to efficiently process complex and dynamic computational tasks within the research workflow remains inadequately addressed, often requiring manual intervention, which can introduce errors and inefficiencies. Thus, constructing tools that can handle varying data modalities and complexities, and that can adapt to the continual advancements in materials science, is essential for advancing the field.

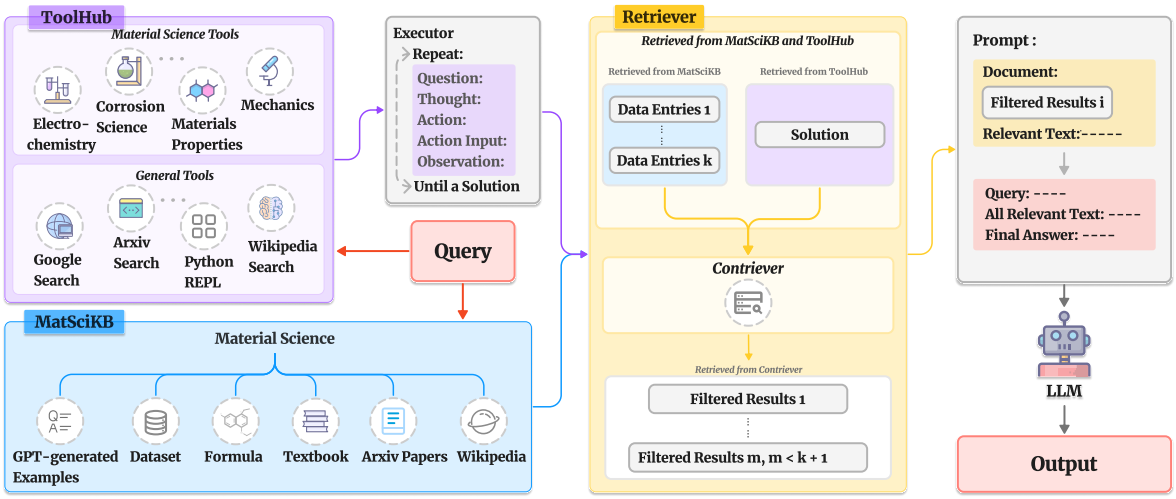


Figure 1: The overall architecture of HoneyComb. The model begins with a query input that activates the knowledge retrieval phase, where pertinent data entries and tools are extracted from the MatSciKB and ToolHub respectively. The Executor iteratively calls the relevant tools from the ToolHub, evaluating and refining these calls until a solution that adequately solves the query emerges. The preliminary solution generated by these tools is combined with relevant data entries and then undergoes further processing by the Retriever. Finally, the Retriever consolidates and filters this input, ultimately feeding it into the LLM for final answer generation.

To address these challenges, HoneyComb has been designed with innovative solutions that markedly enhance research capabilities in materials science. First, we integrated General Tools that provide direct access to current publications and facilitate dynamic discussions, as shown in Table 2, effectively complementing the static MatSciKB. Second, recognizing the limitations of large language models (LLMs) in performing computational tasks, we implemented a Python REPL environment within HoneyComb. This environment is strategically utilized by the system when the agent, interacting with the ToolHub, identifies a need for basic numerical computations. The agent dynamically generates Python code for these tasks and executes it through the Python REPL, bypassing the LLM's computational limitations. This automation not only streamlines data processing but also enhances the precision and reliability of numerical analyses in research activities.

### Materials Science Tools Construction

Constructing domain-specific tool APIs presents significant challenges, requiring domain expert knowledge, and existing resources are limited. Additionally, many valuable data and tools are not open source, limiting their accessibility. Developing these tools is essential for effectively addressing the unique and complex queries inherent to materials science. The scarcity of pre-existing, specialized computational tools necessitates a methodical approach to tool construction and refinement.

| **General Tools** |
|---|
| Google Search |
| Google Scholar Search |
| Arxiv Search |
| Wikipedia Search |
| YouTube Search |
| Python REPL |

Table 2: ToolHub - General Tools

We propose the *Inductive Tool Construction* method, delineated in Algorithm 1, for constructing domain-specific tool APIs. It adopts a systematic approach to create and refine computational tools specifically designed for materials science queries. The process initiates by selecting a random subset of computational questions from dataset $D$, designated as $D_{train}$ for training, with the remaining questions forming $D_{test}$. For each question $q_i \in D_{train}$, a designated LLM, $M$ (such as GPT-4), is tasked with generating a Python function $f_i$ that addresses $q_i$. After creation, each function $f_i$ undergoes rigorous human verification to confirm its correctness.

However, the above procedures cannot ensure

**Algorithm 1** Inductive Tool Construction

**Require:** Train Set $D_{train}$, LLM $M$
**Ensure:** Set of atom tools $A$

```
A ← ∅ {Initialize the set of atom functions}
for each question q_i in D_train do
    f_i ← M(q_i) {Generate specific function for q_i}
    Human verifies f_i
    Decompose f_i into atom functions a_j, j ∈ {1, 2, ...}
    A ← A ∪ {a_j | j ∈ {1, 2, ...}} {Add atom functions to the set}
end for
return A
```

the generalizability of the constructed tool APIs. Thus, in the post-validation stage, we further use $M$ to decompose each $f_i$ into fundamental, reusable components known as atom functions $a_j$, which are crafted for extensive applicability across diverse queries. A detailed example is illustrated in Appendix E.

### 3.3 Agent-ToolHub Interactions

In HoneyComb, interactions between the agent and ToolHub are governed by a structured two-phase decision-making protocol. Our protocol emphasizes the critical selection and processing of data to ensure that only pertinent information influences the LLM's decisions. This approach is vital to prevent the degradation of model performance due to irrelevant or low-quality inputs (Liu et al., 2024a).

1. **Tool Assessor**: During the initial phase, the Assessor evaluates both the incoming query and the extensive suite of tools within the ToolHub. This evaluation aims to identify a manageable subset of the most relevant tools that are best suited to address the specific requirements of the query. By filtering out irrelevant tools at this stage, we ensure that the Executor is provided only with pertinent information, thereby optimizing the model's focus and enhancing its capacity to solve the problem accurately.

2. **Tool Executor**: As illustrated in Figure 2, the Executor receives the original query along with the subset of tools selected by the Assessor. Upon evaluating the selected tools and query, the Executor engages in a *thought* process to determine the most suitable tool for addressing the query. If the query's complexity exceeds the capacity of a single tool, the Executor recognizes the challenge and decomposes the query into smaller subquestions. This strategy allows for the sequential tackling of each part, starting with the selection of the optimal tool for the initial subquestion. It then initiates the *action* of executing the selected tool while inputting parameter values, termed an *action input*, derived from the query or subquestion. Upon execution, the tool generates a result termed an *observation*. Subsequently, the Executor engages in a reflective process to assess whether the observation adequately addresses the query. If the observation is adequate, it is finalized as the answer; if not, the process either reiterates with adjustments or progresses to the next subquestion, if the original query was segmented into multiple parts.

### 3.4 Retriever

In this section, we present the retriever in HoneyComb, which returns relevant texts or tools from MatSciKB and ToolHub when specific contexts are provided. The retriever integrates both the BM25 model (Trotman et al., 2014) and the Contriever model (Izacard et al., 2022), leveraging their respective strengths to achieve optimal information retrieval performance.

Specifically, the retriever employs a two-step strategy. Initially, BM25 utilizes efficient calculations of term frequency and inverse document frequency to rapidly process short text queries and keyword searches within long documents. The primary advantage of BM25 lies in its computational simplicity and rapid response time, allowing HoneyComb to extract the top $k$ most relevant knowledge points from an extensive materials science knowledge base, ensuring exceptional speed and efficiency. This approach provides basic relevance matching results within a minimal timeframe.

Subsequently, we employ a pre-trained deep learning model (i.e., Contriever) to generate embedding vectors and compute their similarity, enabling a deeper understanding of complex linguistic structures and semantic information. The $k$ data entries from BM25, along with the final answer computed by the Executor—which employs the necessary materials science tools and general tools—are passed into Contriever for further re-

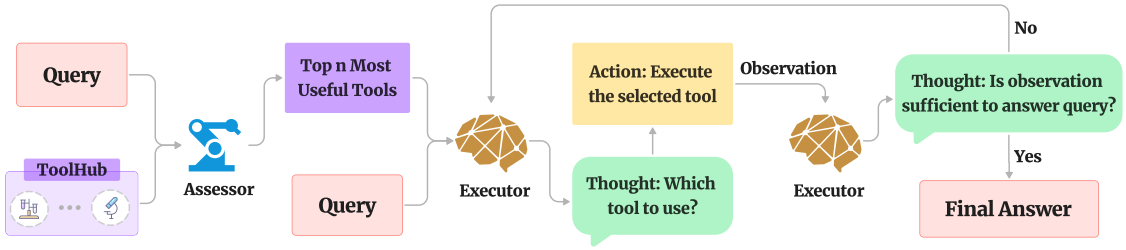


Figure 2: Interaction Workflow Between Tool Assessor and Executor in HoneyComb.

finement. Contriever retrieves the most relevant $m$ results, where $m < k + 1$. Although Contriever operates at a slower pace compared to BM25, its ability to deeply analyze semantic and contextual relationships ensures high precision and relevance for complex queries.

By combining BM25 and Contriever, our model adeptly responds to simple queries with speed while offering enhanced accuracy and relevance for complex queries. This hybrid approach ensures that the model is both efficient and capable of addressing sophisticated query requirements, thereby providing comprehensive, and precise information retrieval services.

## 4 Experiments

We conduct experiments on two question-answering datasets, namely MaScQA (Zaki et al., 2023) and SciQA (Johannes Welbl, 2017), to investigate the ability of HoneyComb in materials science tasks.

MaScQA, derived from the Graduate Aptitude Test in Engineering (GATE) in India, is tailored to reflect the real-world complexity and variety of issues encountered in materials science. This highly competitive examination assesses a comprehensive understanding of various undergraduate subjects (Indian Institute of Technology Kanpur, 2023; Zaki et al., 2023). With its 650 questions covering 14 domains, such as thermodynamics, atomic structure, and mechanical behavior, the dataset includes a wide range of question types: Multiple Choice Questions (MCQs), Numerical Answer Type (NUM), Matching Type (MATCH), and MCQs with numerical options (MCQN). Specifically designed for advanced problem-solving, this dataset is crucial for ensuring that our ToolHub functions effectively in real-world materials science research and applications. It demonstrates the efficacy and adaptability of the HoneyComb framework in tackling complex materials science issues within realistic scenarios.

The second dataset, SciQA, comprises 11,679 multiple-choice questions spanning the core disciplines of fundamental sciences from various crowd-sourced science exams (Johannes Welbl, 2017). This compilation not only emphasizes the dataset's comprehensive and interdisciplinary nature but also focuses on fostering a nuanced conceptual understanding. SciQA serves as a critical testbed to ascertain whether the HoneyComb framework can augment the LLM's capabilities beyond its initial programming. By integrating supplementary information, it aids in addressing intricate queries and unraveling complex scientific concepts that may have been overlooked during the initial training phase of the LLM. By bridging real-world complexities with rigorous academic standards, these datasets ensure that our MatSciKB and ToolHub are not only versatile but also remain at the forefront of technological and scientific application.

The choice of models for our experiments was driven by the need to evaluate the HoneyComb framework's enhancement capabilities across a spectrum of large language models known for their robust performance in diverse applications. We selected GPT-3.5, GPT-4 (OpenAI, 2024), LLaMA-2 (Touvron et al., 2023b), and LLaMA-3 (AI@Meta, 2024) due to their widespread use and proven effectiveness in handling complex language tasks. These models, with LLaMA-2 and LLaMA-3 having parameter sizes of 7 billion and 8 billion, respectively, represent the current state-of-the-art in generalized language understanding and provide a solid baseline for benchmarking. Additionally, we included HoneyBee (Song et al., 2023b), a specialized model with a parameter size of 7 billion, tailored specifically for materials science. The in-

clusion of both general-purpose and specialized models allows us to showcase how domain-specific adaptations through HoneyComb can elevate a model's functional scope beyond its original configuration, thus highlighting the adaptability and effectiveness of our framework.

### 4.1 HoneyComb Evaluation

We evaluated the performance of various models on MaScQA and SciQA, including HoneyBee, GPT-3.5, GPT-4, LLaMA-2, and LLaMA-3, and demonstrated the effects of using HoneyComb. The results are illustrated in Table 3.

The experimental results show that all models based on HoneyComb achieved significant improvements in accuracy on both MaScQA and SciQA. Specifically, on the MaScQA dataset, models such as HoneyBee and GPT-4 experienced substantial enhancements, with HoneyBee's accuracy increasing from 16.62% to 33.38%, and GPT-4 achieving an improvement from 58.46% to 79.07%. Other models also exhibited notable gains, with accuracy improvements ranging from 4.92% to 14.16%.

On the SciQA dataset, HoneyComb-based models showed even more dramatic performance gains. The HoneyBee model's accuracy rose significantly, from 33.96% to 79.69%, marking an improvement of 45.73%. For models based on GPT-3.5 and LLaMA-3, HoneyComb integration led to more modest increases, ranging between approximately 0.14% to 0.32%. In contrast, HoneyComb's integration with GPT-4 and LLaMA-2 brought considerable improvements of about 5.70% and 2.87%, respectively.

### 4.2 HoneyComb Evaluation Across the Material Categories of MaScQA

We assess the performance improvements when integrating the HoneyComb framework with various large language models (LLMs) across predefined topics within the MaScQA dataset, as shown in Table 4. The overall trend indicates that HoneyComb substantially enhances model performance across nearly all models and tasks. LLaMA-3 and HoneyBee exhibit particularly impressive gains, especially in Material Testing, where LLaMA-3 sees an improvement of 33.34%. These results demonstrate HoneyComb's capability to effectively augment models with its advanced ToolHub and extensive MatSciKB.

Despite having a higher baseline accuracy than LLaMA-3, LLaMA-2, and HoneyBee, GPT-3.5 exhibits a mixed response when integrated with HoneyComb. While some tasks show dips in performance, others benefit from the enhancements provided by HoneyComb, indicating that its performance does not uniformly decline across all topics. This variability could be due to the scope and depth of GPT-3.5's training data, which, although extensive, may not always align seamlessly with HoneyComb's highly specialized materials science knowledge. The sophisticated computational demands and dynamic nature of materials science queries might pose challenges in fully adapting GPT-3.5's pre-existing knowledge to the specific enhancements that HoneyComb offers.

### 4.3 Ablation Study

To understand the contribution of each component of HoneyComb, we conducted ablation studies by testing its performance when retrieving from only MatSciKB, only ToolHub, or both, as well as by evaluating the effect of excluding the retriever component.

The best performance for both MaScQA and SciQA is achieved when MatSciKB and ToolHub are combined with the retriever. For MaScQA, this combination leads to an accuracy of 79.07%, representing a substantial improvement of 17.69%. When used individually, MatSciKB contributes an increase of 16.93%, while ToolHub alone boosts accuracy by 11.85%. In SciQA, ToolHub alone improves accuracy by 5.5%, while the full integration of MatSciKB, ToolHub, and the retriever achieves the highest accuracy of 96.56%, a 5.72% improvement over the baseline. The results are summarized in Table 5.

## 5 Conclusion

In this work, we introduced HoneyComb, a pioneering LLM-based agent system tailored for materials science, consisting of three key components: a curated materials science knowledge base (MatSciKB), a dual-layered ToolHub encompassing both general and specialized computational tools, and a precision-focused Retriever module. These components are integrated to deliver accurate, real-time information and ensure reliable performance in advanced materials science tasks.

Experimental results demonstrate that HoneyComb outperforms contemporary general-purpose models (e.g., GPT and LLaMA series) and special-

| Dataset | HoneyBee | HoneyBee + | GPT-3.5 | GPT-3.5 + | GPT-4 | GPT-4 + | LLaMA2 | LLaMA2 + | LLaMA3 | LLaMA3 + |
|---|---|---|---|---|---|---|---|---|---|---|
| MaScQA | 16.62 | 33.38 | 33.54 | 38.46 | 58.46 | 79.07 | 22.15 | 36.31 | 24.62 | 47.23 |
| SciQA | 33.96 | 79.69 | 90.69 | 90.83 | 90.84 | 96.54 | 75.79 | 78.66 | 93.00 | 93.32 |

Table 3: HoneyComb evaluation with diverse LLMs including open-source LLMs (HoneyBee (Song et al., 2023b), LLaMA2 (Touvron et al., 2023b), LLaMA3 (AI@Meta, 2024)) and commercial LLMs (GPT3.5, GPT4 (OpenAI, 2024)). The results show that HoneyComb consistently improves the performance of all LLMs for SciQA and MaScQA.

| Category | HoneyBee | HoneyBee + | GPT-3.5 | GPT-3.5 + | GPT-4 | GPT-4 + | LLaMA2 | LLaMA2 + | LLaMA3 | LLaMA3 + |
|---|---|---|---|---|---|---|---|---|---|---|
| Atomic structure | 12.0 | 34.00 | 35.00 | 32.00 | 55.00 | 70.00 | 22.00 | 38.00 | 22.00 | 48.00 |
| Electrical | 20.0 | 38.89 | 30.56 | 41.67 | 41.67 | 75.00 | 25.00 | 36.11 | 30.56 | 44.44 |
| Fluid | 28.57 | 28.57 | 42.86 | 28.57 | 64.29 | 71.43 | 28.57 | 35.71 | 42.86 | 35.71 |
| Magnetism | 6.67 | 20.00 | 33.33 | 20.00 | 73.33 | 66.67 | 33.33 | 33.33 | 26.67 | 46.67 |
| Material Applications | 15.09 | 30.19 | 49.06 | 52.83 | 86.79 | 94.34 | 26.42 | 37.74 | 24.53 | 50.94 |
| Material characterization | 35.71 | 35.71 | 50.00 | 64.29 | 85.71 | 100.00 | 28.57 | 42.86 | 64.29 | 64.29 |
| Material manufacturing | 18.68 | 32.97 | 35.16 | 38.46 | 69.23 | 90.11 | 26.37 | 39.56 | 27.47 | 51.65 |
| Material processing | 11.43 | 28.57 | 42.86 | 40.00 | 68.57 | 94.29 | 14.29 | 45.71 | 40.00 | 57.14 |
| Material testing | 44.44 | 77.78 | 66.67 | 55.56 | 100.00 | 100.00 | 44.44 | 55.56 | 44.44 | 77.78 |
| Mechanical | 17.71 | 35.42 | 30.21 | 30.21 | 48.96 | 78.12 | 17.71 | 30.21 | 15.62 | 40.62 |
| Miscellaneous | 37.5 | 25.00 | 12.50 | 12.50 | 75.00 | 87.50 | 12.50 | 25.00 | 25.00 | 37.50 |
| Phase transition | 14.63 | 31.71 | 26.83 | 48.78 | 60.98 | 87.80 | 19.51 | 43.90 | 29.27 | 51.22 |
| Thermodynamics | 14.29 | 32.46 | 25.44 | 40.35 | 41.23 | 64.91 | 19.30 | 28.07 | 14.91 | 42.11 |
| Transport phenomena | 16.67 | 33.33 | 20.83 | 37.50 | 45.83 | 70.83 | 20.83 | 45.83 | 25.00 | 41.67 |

Table 4: Improvements of various LLMs integrated with HoneyComb compared to relevant baseline LLMs for different materials science tasks. With few exceptions, HoneyComb improves the performance of all LLMs across all tasks showing the utility of tool augmentation.

| Benchmark | MatSciKB | ToolHub | Retriever | Accuracy |
|---|---|---|---|---|
| MaScQA | | | | 61.38 |
| | | ✓ | ✓ | 73.23 |
| | ✓ | | ✓ | 78.31 |
| | ✓ | ✓ | ✓ | 79.07 |
| SciQA | | | | 90.84 |
| | | ✓ | ✓ | 96.34 |
| | ✓ | | ✓ | 85.57 |
| | ✓ | ✓ | ✓ | 96.56 |

Table 5: Ablation Study Results for MaScQA and SciQA based on GPT-4

ized models (e.g., HoneyBee) in materials science QA tasks. HoneyComb effectively bridges the gap between advanced large language models and the specific needs of materials science research, exemplifying how specialized agent systems can advance scientific research and serve as a blueprint for future developments in other knowledge-intensive fields.

## Limitations

While HoneyComb significantly enhances the performance of current state-of-the-art models across various materials science QA tasks, its generalizability and applicability may be limited beyond the specific datasets and tasks it was trained on. Given the diversity and complexity of materials science, it is uncertain how effectively HoneyComb would perform on tasks outside of the MaScQA and SciQA benchmarks, especially when facing more complex and novel challenges, such as designing synthesis recipes for new materials or predicting material properties.

Moreover, HoneyComb's reliance on high-quality LLMs for its knowledge base, tool construction, and retrieval processes introduces an additional limitation. The performance of these components is dependent on the availability and capability of the underlying LLMs, which may have their own constraints. Additionally, as our work has been primarily focused on the materials science domain, further studies are needed to assess the applicability and effectiveness of HoneyComb in other scientific fields.

## Broader Impacts

By expanding the HoneyComb agent system, there is potential to accelerate scientific discovery and innovation, deepening our understanding of complex material systems. Such expansion could lead

to significant advancements in materials design, development, and application, while also promoting the discovery and optimization of new materials, which would benefit a wide range of industries. Additionally, the versatility and adaptability of HoneyComb allow it to address challenges across various scientific domains, broadening its potential scope and impact.

Our research does not present any major ethical concerns.

## Acknowlegments

This work is supported by the Mila internal funding - Program P2-V1: Industry Sponsored Academic Labs (project number: 10379), the Canada CIFAR AI Chair Program, and the Canada NSERC Discovery Grant (RGPIN-2021-03115).

## References

Microsoft Research AI4Science and Microsoft Azure Quantum. 2023. The Impact of Large Language Models on Scientific Discovery: a Preliminary Study using GPT-4. *arXiv preprint arXiv:2311.07361*.

AI@Meta. 2024. Llama 3 model card.

Anthropic. 2024. Calude3.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. 2023. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools. *Preprint*, arXiv:2304.05376.

Carol Brayne and Terrie E Moffitt. 2022. The limitations of large-scale volunteer databases to address inequalities and global challenges in health and aging. *Nature Aging*, 2(9):775–783.

Markus J. Buehler. 2024a. Generative retrieval-augmented ontologic graph and multiagent strategies for interpretive large language model-based materials design. *ACS Engineering Au*, 4(2):241–277.

Markus J Buehler. 2024b. Mechgpt, a language-based strategy for mechanics and materials modeling that connects knowledge across scales, disciplines, and modalities. *Applied Mechanics Reviews*, 76(2):021001.

Yuheng Cheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang, Zekai Wang, Feng Yin, Junhua Zhao, and Xiuqiang He. 2024. Exploring large language model based intelligent agents: Definitions, methods, and prospects. *Preprint*, arXiv:2401.03428.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. 2023. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*.

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

Anastasios Giannaros, Aristeidis Karras, Leonidas Theodorakopoulos, Christos Karras, Panagiotis Kranias, Nikolaos Schizas, Gerasimos Kalogeratos, and Dimitrios Tsolis. 2023. Autonomous vehicles: Sophisticated attacks, safety issues, challenges, open topics, blockchain, and future directions. *Journal of Cybersecurity and Privacy*, 3(3):493–543.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *Preprint*, arXiv:2203.05794.

Tanishq Gupta, Mohd Zaki, NM Krishnan, et al. 2022. Matscibert: A materials domain language model for text mining and information extraction. *npj Computational Materials*, 8(1):1–11.

Indian Institute of Technology Kanpur. 2023. Gate 2023: Graduate aptitude test in engineering. Accessed: 2024-06-14.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Kevin Maik Jablonka, Qianxiang Ai, Alexander Al-Feghali, Shruti Badhwar, Joshua D Bocarsly, Andres M Bran, Stefan Bringuier, L Catherine Brinson, Kamal Choudhary, Defne Circi, et al. 2023. 14 examples of how llms can transform materials science and chemistry: a reflection on a large language model hackathon. *Digital Discovery*, 2(5):1233–1250.

Matt Gardner Johannes Welbl, Nelson F. Liu. 2017. Crowdsourcing multiple choice science questions.

Olga Kononova, Tanjin He, Haoyan Huo, Amalie Trewartha, Elsa A Olivetti, and Gerbrand Ceder. 2021. Opportunities and challenges of text mining in materials research. *Iscience*, 24(3).

LangChain contributors. 2023. Langchain: Open-source library for building language-based agents. Online; accessed 17-June-2023.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yue Liu, Sin Kit Lo, Qinghua Lu, Liming Zhu, Dehai Zhao, Xiwei Xu, Stefan Harrer, and Jon Whittle. 2024b. Agent design pattern catalogue: A collection of architectural patterns for foundation model based agents. *Preprint*, arXiv:2405.10467.

Ali Madani, Ben Krause, Eric R. Greene, Subu Subramanian, Benjamin P. Mohr, James M. Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z. Sun, Richard Socher, James S. Fraser, and Nikhil Naik. 2023. Large language models generate functional protein sequences across diverse families. *Nat. Biotechnol.*, 41(8):1099–1106.

Santiago Miret and NM Krishnan. 2024. Are llms ready for real-world materials discovery? *arXiv preprint arXiv:2402.05200*.

Santiago Miret, NM Anoop Krishnan, Benjamin Sanchez-Lengeling, Marta Skreta, Vineeth Venugopal, and Jennifer N Wei. 2024. Perspective on ai for accelerated materials design at the ai4mat-2023 workshop at neurips 2023. *Digital Discovery*.

Adrian Mirza, Nawaf Alampara, Sreekanth Kunchapu, Benedict Emoekabu, Aswanth Krishnan, Mara Wilhelmi, Macjonathan Okereke, Juliane Eberhardt, Amir Mohammad Elahi, Maximilian Greiner, et al. 2024. Are large language models superhuman chemists? *arXiv preprint arXiv:2404.01475*.

OpenAI. 2024. Openai. Accessed: 2024-06-14.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023a. Tool learning with foundation models. *Preprint*, arXiv:2304.08354.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023b. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Yu Song, Santiago Miret, and Bang Liu. 2023a. MatSciNLP: Evaluating scientific language models on materials science language tasks using text-to-schema modeling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3621–3639, Toronto, Canada. Association for Computational Linguistics.

Yu Song, Santiago Miret, Huan Zhang, and Bang Liu. 2023b. Honeybee: Progressive instruction finetuning of large language models for materials science. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5724–5739.

Liangtai Sun, Yang Han, Zihan Zhao, Da Ma, Zhennan Shen, Baocai Chen, Lu Chen, and Kai Yu. 2024. Scieval: A multi-level large language model evaluation benchmark for scientific research. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19053–19061.

Matthew C Swain and Jacqueline M Cole. 2016. Chemdataextractor: a toolkit for automated extraction of chemical information from the scientific literature. *Journal of chemical information and modeling*, 56(10):1894–1904.

Oguzhan Topsakal and Tahir Cetin Akinci. 2023. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to bm25 and language models examined. In *Proceedings of the 19th Australasian Document Computing Symposium*, pages 58–65.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Nicholas Walker, Amalie Trewartha, Haoyan Huo, Sanghoon Lee, Kevin Cruse, John Dagdelen, Alexander Dunn, Kristin Persson, Gerbrand Ceder, and Anubhav Jain. 2021. The impact of domain-specific

pre-training on named entity recognition tasks in materials science. *Available at SSRN 3950755*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. 2023. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*.

Tong Xie, Yuwei Wan, Wei Huang, Zhenyu Yin, Yixuan Liu, Shaozhou Wang, Qingyuan Linghu, Chunyu Kit, Clara Grazian, Wenjie Zhang, et al. 2023. Darwin series: Domain specific large language models for natural science. *arXiv preprint arXiv:2308.13565*.

Tong Xie, Yuwei Wan, Yufei Zhou, Wei Huang, Yixuan Liu, Qingyuan Linghu, Shaozhou Wang, Chunyu Kit, Clara Grazian, Wenjie Zhang, and Bram Hoex. 2024. Creation of a structured solar cell material dataset and performance prediction using large language models. *Patterns*, 5(5).

Naruki Yoshikawa, Marta Skreta, Kourosh Darvish, Sebastian Arellano-Rubach, Zhi Ji, Lasse Bjørn Kristensen, Andrew Zou Li, Yuchi Zhao, Haoping Xu, Artur Kuramshin, et al. 2023. Large language models for chemistry robotics. *Autonomous Robots*, 47(8):1057–1086.

Mohd Zaki, Jayadeva, Mausam, and N. M. Anoop Krishnan. 2023. Mascqa: A question answering dataset for investigating materials science knowledge of large language models. *Preprint*, arXiv:2308.09115.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. *Preprint*, arXiv:2310.06117.

# Appendix

## A MatSciKB Knowledge Source

- **ArXiv Papers**
  - Included all papers indexed under the “materials science” keyword on ArXiv.
  - Data entries were structured into key-value pairs: the key is the paper title, and the value is the abstract.
  - **Data Entries Count:** 20,384
- **Wikipedia Materials Science Concepts**
  - Scraped all 438 pages categorized under "Materials Science" on Wikipedia.
  - Each section within a page was separated as a distinct data entry.
  - Content was formulated into key-value pairs, with keys as section titles and values as content.
  - **Data Entries Count:** 3,620
- **Materials Science Textbooks**
  - Sourced 6 publicly available textbooks.
  - Converted each textbook PDF file into text documents.
  - Divided each textbook into data entries by each section within a chapter.
  - Formulated data entries into key-value pairs, with keys as section titles and values as content.
  - **Data Entries Count:** 1,930
- **Materials Science Dataset**
  - Utilized the multiple-choice dataset SciQA.
  - Extracted the "support" column from the dataset, which provides background knowledge for each question.
  - Each extracted "support" is treated as a data entry, with keys as the knowledge piece and values as empty strings, emphasizing their concise and standalone nature.
  - **Data Entries Count:** 10,473
- **Materials Science Formulas**
  - Formulas were collected from Wikipedia’s dedicated pages for materials science formulas.
  - Each formula is stored as a key-value pair in the database, where the key represents the name of the formula and the value contains the formula equation itself.
  - **Data Entries Count:** 57
- **GPT-Generated Examples**
  - Used a specific prompt to generate 50 materials science questions at a time, output in CSV format along with a confidence score. Please refer to Appendix B for the detailed prompt.

- Human reviewers then selected questions with higher confidence scores for inclusion in the dataset.
- Inspiration for question types was drawn from an external resource offering a wide range of materials science questions and answers.
- The key-value pairs were structured with questions as the keys and answers as the values.
- **Data Entries Count:** 2,005

## B Prompt for GPT-Generated Examples

Please generate 50 instances of materials science questions, specifically on atomic structure and interatomic bonding, in CSV format in the following order: question, answer, accuracy, confidence_score. - accuracy: For factual questions, please evaluate the answer by comparing it with known facts. This field should be a number between 0 and 1. - confidence_score: How confident are you in the answer? This field should be a number between 0 and 1. - Here are sample instances without accuracy and confidence_score: “In terms of which of the following properties, metals are better than ceramics?”,“ductility”; "In the wave-mechanical model of an atom, what do degenerate energy levels have?","equal energy"; "Which of the following molecules is diamagnetic?","CO". - Examples of generated instances: - "What is the valence electron configuration of carbon?","$2s^2 2p^2$",0.95,0.85 - "What type of crystal defect occurs when there is a line of irregularity in the lattice structure?","dislocation defect",0.96,0.91

## C Tree-Structure MatSciKB

MatSciKB is organized as a hierarchical tree with the parent node "Material Science" branching into 16 child nodes, each representing a specific domain within materials science. Below is a simplified representation of this structure:

```
  {
"Material Science": {
"Children": {
"Thermodynamics": {"Children": {"KB_1":
{}, "KB_2": {}, "KB_3": {}}},
"Atomic Structure": {"Children": {"KB_4":
{}, "KB_5": {}, "KB_6": {}}},
...
"Miscellaneous": {"Children": {"KB_7":
{}, "KB_n": {}, "KB_n+1": {}}}
} } }
```

Each child node encompasses knowledge base (KB) data entries relevant to its category. In the construction of MatSciKB, we predefined 16 topics that align with core areas in materials science. They are ’Miscellaneous’, ’Material Testing’, ’Fluid’, ’Material Characterization’, ’Magnetism’, ’Transport Phenomena’, ’Material Processing’, ’Electrical’, ’Phase Transition’, ’Material Applications’, ’Material Manufacturing’, ’Mechanical’, ’Atomic Structure’, ’Thermodynamics’, ’Formula’, and ’Fundamental_Science_Knowledge’.

To categorize the data entries within these nodes, we utilized BertTopic, a state-of-the-art topic modeling tool based on transformers and c-TF-IDF, which automatically identifies and clusters documents with high granularity and contextual relevance (Vaswani et al., 2023; Grootendorst, 2022). The integration of BertTopic allowed for the dynamic clustering of MatSciKB entries into 16 predetermined categories.

The process involved the following steps:

1. **Initial Clustering**: BertTopic was applied to cluster all data entries into a number of categories greater than the target number, based on the textual content of each entry.

2. **Cluster Analysis and Selection**: Human reviewers analyzed each cluster, identifying those whose common keywords and themes closely aligned with one of the predefined 16 topics.

3. **Category Assignment**: Entries from clusters that aligned well with a predefined topic were assigned to that category and then removed from the dataset.

4. **Iterative Refinement**: The remaining entries underwent subsequent rounds of clustering and analysis. This process was repeated until no entries were left unclassified.

## D Tools Unified Interface Using LangChain

LangChain is an advanced framework designed to enhance applications that utilize LLM by offering standardized interfaces for various modules (LangChain contributors, 2023). This framework facilities the seamless integration and efficient management of LLM with external tools and systems.

Utilizing LangChain, HoneyComb has developed a unified interface that standardizes the integration of a wide array of tools.

In HoneyComb, the unified interface provided by LangChain ensures that all tools, regardless of their specific function, are treated as standardized LangChain objects. This standardization is achieved by defining each tool with a consistent set of attributes:

1. **Function Signature**: Each tool is defined with a clear function signature that specifies input and output types,

2. **Metadata Description**: Each tool is accompanied by metadata that describes its purpose, suitable use cases, parameters description.

Examples of function signatures and metadata descriptions in HoneyComb are:

- **Google Search**
  - **Function Signature**: Google_Search(query: str, timeout: Optional[int] = 30) -> str
  - **Metadata Description**: General web search for up-to-date information across various topics.
- **Wikipedia Search**
  - **Function Signature**: Wikipedia_Search(topic: str, summarize: bool = True) -> str
  - **Metadata Description**: Retrieves and optionally summarizes detailed Wikipedia articles, particularly useful for quick reference checks.
- **A Sample Mass Flow Rate Tool**
  - **Function Signature**: calculate_initial_mass_flow_rate(args: str) -> float
  - **Metadata Description**: See figure 3.

## E Examples of Inductive Tool Construction

See figure 4 for a detailed example illustrating how inductive tool construction work.

*Calculate the initial mass flow rate of liquid metal draining from a cylindrical vessel through a nozzle.*

*Parameters:*
*args (str): A string containing the required parameters separated by "|" in the following order:*
*- density (float): Density of the liquid metal in kg/m^3*
*- nozzle_diameter (float): Diameter of the nozzle in mm*
*- discharge_coefficient (float): Discharge coefficient of the nozzle (dimensionless)*
*- height (float): Height of the liquid metal column in the vessel in meters*

*Returns:*
*float: Initial mass flow rate in kg/s.*

Figure 3: Metadata Description of a Sample Mass Flow Rate Tool

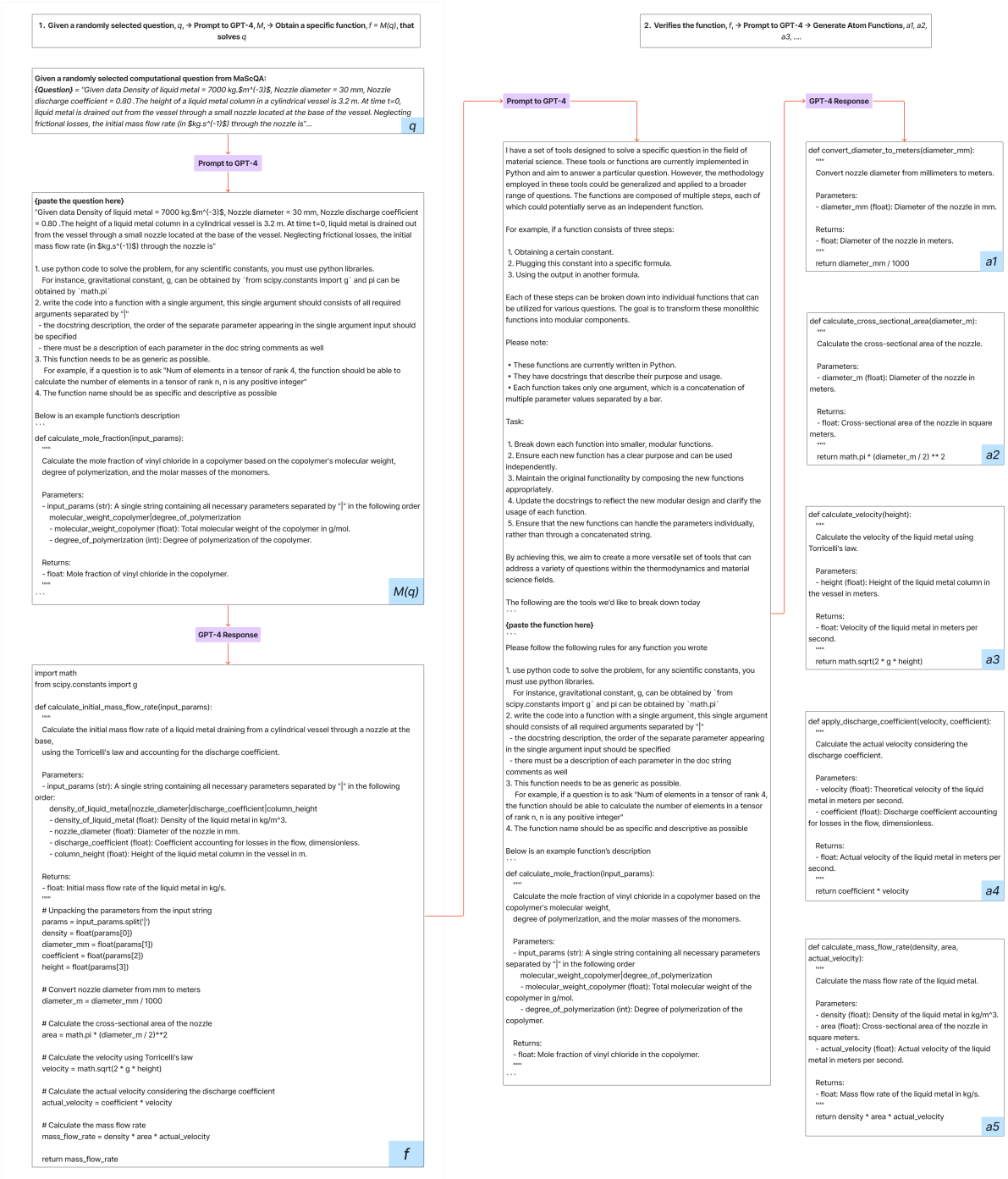


Figure 4: An example of inductive tool construction