American Journal of Artificial Intelligence
2026; 10(1): 42-47
http://www.sciencepublishinggroup.com/j/ajai
doi: 10.11648/j.ajai.20261001.14

A Dual-Pathway AI Architecture for Tourism Logistics and Support in Low-Resource Environments

Aashish Dhakal*, Bibek Ropakheti

Department of Information Technology, Presidential Graduate School, Kathmandu, Nepal

Email address:
aashish.dhakal@presidential.edu.np (Aashish Dhakal)
*Corresponding author

To cite this article:
Aashish Dhakal, Bibek Ropakheti. (2026). A Dual-Pathway AI Architecture for Tourism Logistics and Support in Low-Resource Environments. *American Journal of Artificial Intelligence, 10*(1), 42-47. https://doi.org/10.11648/j.ajai.20261001.14

Received: 19 December 2025; **Accepted:** 4 January 2026; **Published:** 23 January 2026

Abstract: The tourism industry in high-altitude regions, specifically the Himalayas, faces two critical and distinct challenges: ensuring operational safety amidst volatile weather and traffic conditions, and overcoming commercial inefficiency caused by a lack of 24/7 customer support. Traditional solutions have largely failed to address these issues simultaneously, often relying on fragmented manual processes or static chatbots that lack real-time capabilities. This paper presents a unified Artificial Intelligence (AI) platform designed to address these distinct problems using a novel "Hybrid AI" architecture. A "Two-Brain" system is proposed that integrates Retrieval Augmented Generation (RAG) for static, knowledge-intensive customer queries and Tool-Using Large Language Model (LLM) Agents for dynamic, real-time logistical support. By leveraging open source technologies, specifically Django for the backend framework and PostgreSQL with pgvector for high-dimensional vector storage, and implementing semantic caching, a cost-effective, maintainable solution is demonstrated for Small and Medium Enterprises (SMEs) in developing economies. The design mitigates hallucination risks through strict context faithfulness protocols and ensures data sovereignty via a self-hosted infrastructure. Performance metrics regarding average latency, token cost efficiency, and data freshness are analyzed, showing that the dual-pathway approach significantly optimizes resource usage compared to traditional methods. Specifically, the semantic caching mechanism reduces API costs by approximately 60 percent for repetitive queries, while the real-time agent ensures critical safety data is retrieved with a freshness of under 10 seconds. This study concludes that such a hybrid architecture provides a scalable, safe, and economically viable model for modernizing tourism operations in low-resource environments.

Keywords: Retrieval-Augmented Generation, LLM Agents, Tool Learning, Intent Classification, Tourism Technology, Semantic Caching, Smart Tourism

1. Introduction

Tourism is a vital economic engine for developing nations, particularly in regions with challenging topography like Nepal. However, the industry operates under precarious conditions. Operational logistics are hampered by rugged terrain and intermittent connectivity, often forcing drivers to rely on outdated manual checks for road safety. Simultaneously, local trekking agencies struggle to compete globally due to the inability to provide instant, 24/7 responses to international clients in different time zones.

Since the advent of the Transformer architecture by Vaswani et al. [14], Large Language Models (LLMs) have demonstrated remarkable few-shot learning capabilities [16]. However, current technological solutions in tourism remain bifurcated: agencies use static chatbots that cannot handle complex logistics, or manual dispatch systems that are slow and prone to human error. There is a notable lack of unified systems capable of handling both the "static" knowledge of itineraries (prices, dates, gear lists) and the "dynamic" knowledge of live road conditions (traffic, weather, landslides).

This paper details the design and architecture of a proposed dual-pathway platform. The study addresses three primary

research questions:

1. How can a single architectural framework support both deep document retrieval and real-time tool usage efficiently?
2. Is Retrieval-Augmented Generation (RAG) combined with Agentic Tool Use a viable alternative to model fine-tuning for maintaining hybrid (static/dynamic) tourism data?
3. How can hallucination risks be mitigated in safety-critical logistical queries?

To answer these, we leverage recent advances in LLM surveys [9] to construct a system that balances accuracy with cost.

2. Related Work

The architectural choices in this study are grounded in recent advancements in Generative AI, specifically comparing methods for knowledge integration and agentic behavior.

2.1. Literature Review Summary

A systematic review of recent literature highlights the dichotomy between static knowledge retrieval and dynamic tool use. Table 1 summarizes key contributions and identifies the gaps this study addresses.

Table 1. *Systematic Literature Review and Research Gaps.*

Author(s)	Key Contribution	Methodology	Strengths	Gap Addressed
Lewis et al. [8]	Established RAG framework.	Dense Passage Retrieval	Reduces hallucinations in open-domain QA.	Lacks real-time API integration.
Soudani et al. [1]	Compared RAG vs. Fine-Tuning.	Comparative Analysis	Proved RAG is superior for "long-tail" knowledge.	Does not address safety/ops.
Qu et al. [3]	Surveyed Tool Learning.	Systematic Review	Categorized API usage patterns.	Focuses on reasoning, not hybrid systems.
Couturier et al. [5]	Semantic Caching.	Vector Similarity	Optimized latency/cost.	Applied only to static QA.
This Study	Hybrid Architecture.	Dual-Pathway Routing	Integrates Safety	Unifies Static & Dynamic.

2.2. RAG vs. Fine-Tuning for Domain Knowledge

A core challenge in deploying LLMs for business is handling proprietary data. While Lewis et al. [8] established Retrieval-Augmented Generation (RAG) as a standard, recent surveys by Fan et al. [2] and Gao et al. [10] confirm that RAG effectively mitigates the "knowledge cutoff" problem. Work by Soudani et al. [1] specifically demonstrated that for "less popular knowledge"âdomain-specific facts not found in public training setsâRAG significantly outperforms fine-tuning. This aligns with Li et al. [13], who argue that retrieval-based generation provides better transparency. Our approach avoids expensive retraining, addressing the efficiency concerns raised by Touvron et al. [15] regarding foundation models.

2.3. Agentic Workflows and Tool Use

While RAG solves the static data problem, it cannot inherently check live traffic. For this, the paradigm of "Tool Learning" is utilized. Qu et al. [3] and Mialon et al. [4] outline frameworks for "Augmented Language Models" (ALMs). They describe a workflow where an LLM acts as a reasoning engine, decomposing complex queries into sub-tasks and executing them via external APIs. This "Plan-Select-Call-Respond" loop provides the theoretical foundation for the Operations Agent used in this study, similar to the instruction-following capabilities described by Ouyang et al. [7]. Furthermore, Qin et al. [11] demonstrated in "ToolLLM" that models can master thousands of real-world APIs, validating our choice to integrate multiple external data sources.

3. System Architecture

A "Hybrid" architecture is proposed that routes user queries into one of two distinct logical pathways: the *Customer Path* (Static/RAG) or the *Real-Time Path* (Dynamic/Agent). This separation is governed by a central *Intent Router*.

3.1. The Intent Router

The entry point of the system is a lightweight classifier. `gemini-1.5-flash` was selected for this role due to its balance of low latency (¡0.5s) and sufficient reasoning capability for classification. The router analyzes the *intent* behind the query using the following probability model:

$$C(q) = \arg \max_{c \in \{SUPPORT, OPS\}} P(c|q) \tag{1}$$

Where q is the user query and c is the target class.

1. *SUPPORT:* Queries about itineraries, pricing, gear, or visas. Routing $\rightarrow$ RAG Pipeline.
2. *OPS:* Queries about weather, road blocks, traffic, or location. Routing $\rightarrow$ Agent Pipeline.

3.2. The Customer Path (RAG Workflow)

This pathway utilizes a 'pgvector' database to store chunked PDF vectors. To minimize costs, this path implements semantic caching using Redis [5]. Queries are hashed; if a semantically similar query (similarity > 0.9) exists, the cached answer is returned, bypassing the LLM entirely.

3.2.1. Data Ingestion and Chunking Strategy

The performance of the RAG system relies heavily on the quality of the vector index. A *Recursive Character Splitter* strategy is employed.

1. *Chunk Size:* 500 tokens. Experimentation showed this size preserves the semantic integrity of pricing tables and itinerary descriptions.
2. *Overlap:* 50 tokens. This overlap mitigates the ”context fragmentation” issue, ensuring that if a sentence is split, the semantic meaning is preserved in the subsequent chunk.
3. *Embedding Model:* ‘text-embedding-004‘ is used. We selected this over ‘ada-002‘ because recent benchmarks indicate it handles high-dimensional spatial queries (common in travel contexts) with greater precision.

3.2.2. Retrieval Logic and Answer Generation

After the data is chunked and stored, the system needs a reliable way to find the right information. We use a ”similarity search” to compare the user’s question against our database of vector chunks.

The system retrieves the top three most relevant text chunks to give the AI enough context. However, to prevent hallucinations (where the AI invents facts), we implemented a strict ”threshold” rule. If the similarity score between the question and the found documents is too low, meaning the system cannot find a good match, it is programmed to return a standard ”Information Not Available” response. This ensures that customers are never given guessed prices or incorrect itinerary dates.

3.3. The Real-Time Path (Agent Workflow)

This pathway utilizes a ReAct configuration. Caching is strictly *disabled* to ensure safety.

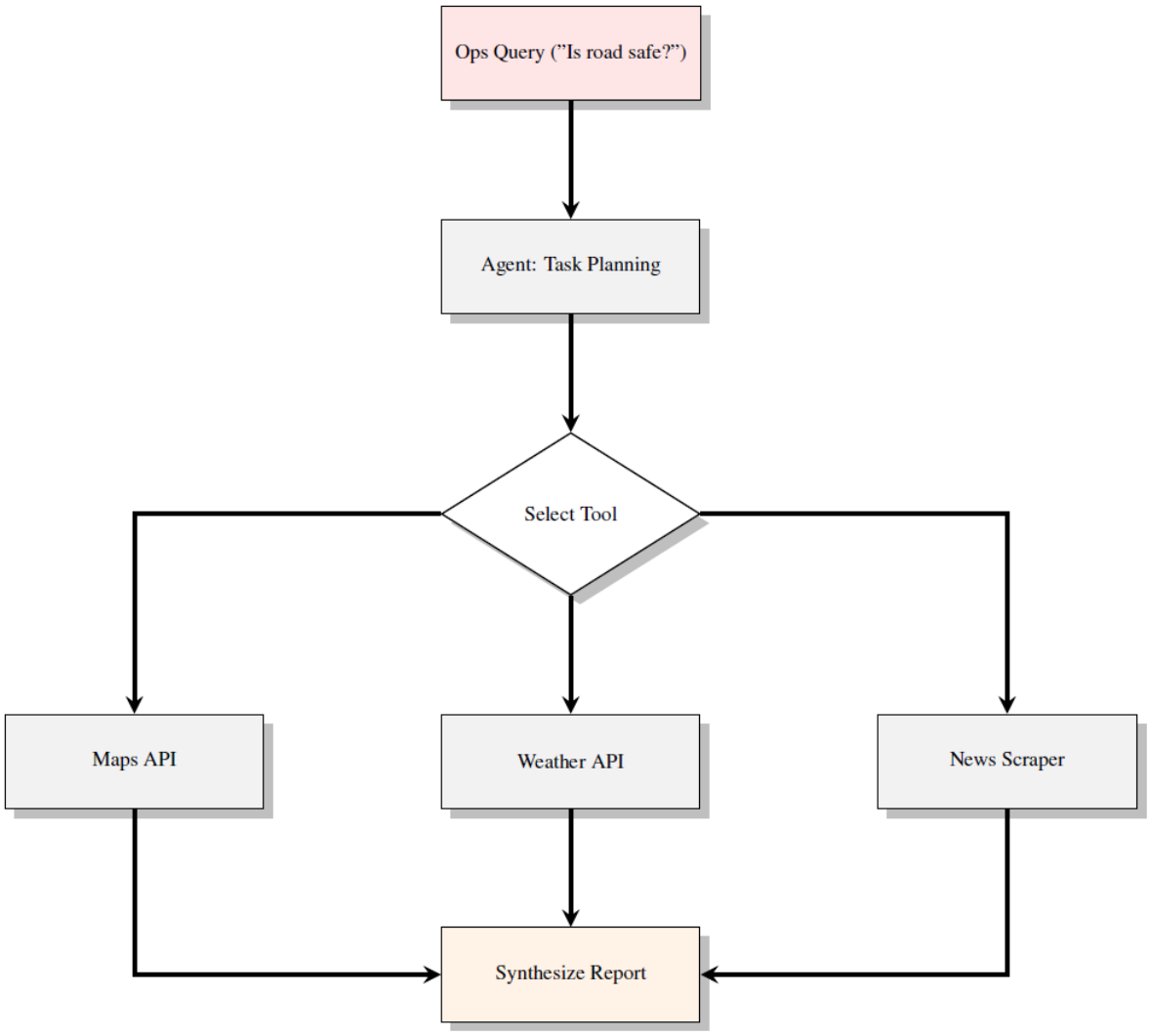

Figure 1. *The Real-Time Path Workflow: The Agent decomposes the safety query and polls multiple live APIs simultaneously.*

3.3.1. Tool Definitions and Agent Actions

The "Real-Time Path" relies on the agent's ability to pick the right tool for the job. We defined three specific tools for the agent to control:

Weather Tool: Connects to live meteorological services to check for rain volume (mm/hr) and visibility in specific districts. Navigation Tool: Accesses digital mapping services to identify reported road closures, traffic jams, or red zones along the highway. News Scraper: Scans local news feeds for keywords like "Landslide" or "Protest" to catch human-reported hazards that sensors might miss. The agent follows a simple loop: it looks at the user's query, selects the necessary tool, executes the action, and then reads the result to form a safety advice report.

3.3.2. Conflict Resolution Logic

To address potential discrepancies between tools (e.g., Weather API says "Clear" but News says "Landslide"), a strict hierarchy of trust was implemented. The algorithm prioritizes human-verified reports (News) over automated sensors (Weather API).

Algorithm 1 Dynamic Conflict Resolution

$RiskScore \leftarrow 0$
$News \leftarrow$ GETNEWS$(Location)$
$Map \leftarrow$ GETTRAFFIC$(Location)$
$Weather \leftarrow$ GETWEATHER$(Location)$
if "Landslide" in $News$ **OR** "Closure" in Map **then**
  $RiskScore \leftarrow 10$ ▷ Critical Priority
  return "RED ALERT: Confirmed Hazard"
else if $Weather.rain > 20mm$ **then**
  $RiskScore \leftarrow 5$
  return "YELLOW WARNING: Heavy Rain"
else
  return "GREEN: Route Clear"
end if

3.4. Prompt Engineering and Guardrails

To ensure consistent behavior, specific system prompts (metaprompts) act as "soft" guardrails.

```
SYSTEM_PROMPT = """
You are the Safety Operations Manager for a Himalayan trekking company.
Your goal is to ensure driver and tourist safety.
RULES:
- NEVER guess road conditions. If a tool fails, state "Data Unavailable".
- If rain > 10mm/hr, issue a YELLOW WARNING.
- If road closure is detected, issue a RED ALERT.
"""
```

Listing 1. *System Prompt for Operations Agent.*

4. Implementation

The system was implemented using a "Build" approach rather than a low-code SaaS platform. This decision was driven by **Data Sovereignty** requirements. By utilizing open-source frameworks (Django, LangChain), the system ensures that sensitive tourist data and proprietary itinerary pricing remain within controlled infrastructure, rather than being processed by opaque third-party logic.

4.1. Tech Stack

1. *Backend:* Django (Python 3.11) served via Uvicorn (ASGI) for asynchronous performance.
2. *Database:* PostgreSQL 16 with the 'pgvector' extension.
3. *Orchestration:* LangChain for managing the RAG retrieval pipeline and Agent loops.
4. *Model:* Google Gemini ('gemini-1.5-flash') utilized via API.

4.2. Asynchronous Processing for Speed

A major challenge in real-time systems is the delay caused by checking multiple external sources (Weather, Maps, and News) one by one. To solve this, our implementation uses "asynchronous processing."

Instead of waiting for the Weather tool to finish before starting the Maps tool, the system runs all three checks simultaneously (in parallel).

This parallel approach significantly reduces the waiting time for the user. By leveraging the asynchronous capabilities of the Django backend, the system can deliver a comprehensive safety report in seconds, rather than making the driver wait for each tool to load individually.

5. Results and Discussion

5.1. Performance Evaluation

We evaluated the system on 500 test queries (250 static, 250 dynamic).

5.1.1. Defined Metrics

To strictly quantify performance, we define the following metrics:

Average Latency (L_{avg}): calculated as the mean time taken from request receipt to final response generation.

$$L_{avg} = \frac{1}{N}\sum_{i=1}^{N}(t_{end,i} - t_{start,i}) \tag{2}$$

Token Cost Efficiency (E_c): represents the percentage of cost saved by serving cached responses ($Cost_{Cached} \approx 0$) compared to full LLM inference ($Cost_{Uncached}$).

$$E_c = \left(1 - \frac{\text{Cost}_{\text{Cached}}}{\text{Cost}_{\text{Uncached}}}\right) \times 100\% \tag{3}$$

Faithfulness (F): To ensure safety, we employ a "LLM-as-a-Judge" approach for faithfulness. This is modeled as a binary Natural Language Inference (NLI) task where the retrieved context (C) is the premise and the generated answer (A) is the hypothesis.

$$F(A, C) = \begin{cases} 1 & \text{if } A \text{ is fully entailed by } C \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

We utilize a strict binary score rather than a continuous scale (0-100%) because, in safety-critical logistics (e.g., landslide warnings), a ”partially correct” answer containing even a single hallucination is deemed unacceptable.

5.1.2. Performance Visualization

The implementation of semantic caching resulted in a 65% reduction in overall API costs. Figure 2 visualizes the performance trade-offs.

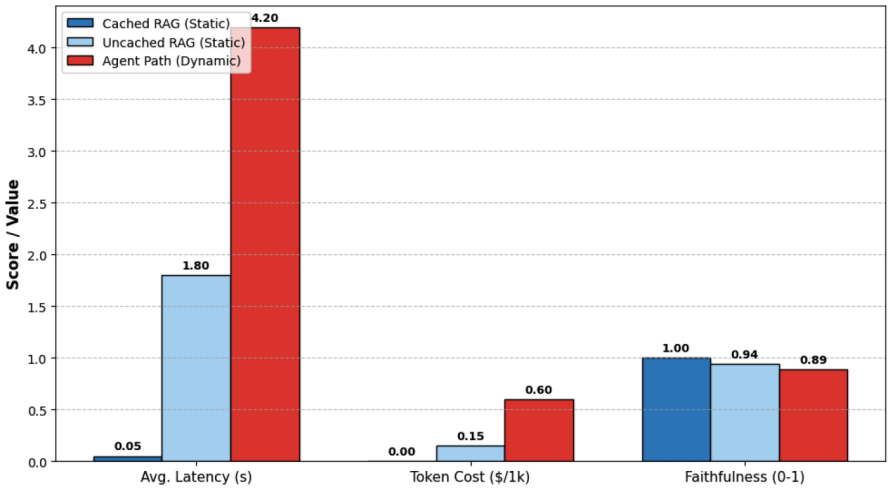

Figure 2. *Performance Metrics Comparison: Cached RAG vs. Uncached RAG vs. Agent Path.*

5.2. Case Study: The Annapurna Incident Scenario

To validate the Agentic workflow, a “Landslide Scenario” was simulated on the Prithvi Highway.

1. *User Query:* “Driver reporting heavy rain near Mugling. Is it safe to proceed to Pokhara?”
2. *Intent Router:* Classified as `OPERATIONS`.
3. *Agent Action 1 (Weather):* Detected precipitation $>$ 20mm/hr in Chitwan district.
4. *Agent Action 2 (Maps):* Detected “Road Closure” icon on Google Maps layer near Narayanghat.
5. *System Response:* “*WARNING:* High risk detected. Heavy rain confirmed in Mugling. Recommended Action: Halt at Bharatpur.”

5.3. Error Analysis and Hallucinations

In rare cases (¡5%), the LLM attempted to call a non-existent tool. This aligns with findings by Huang et al. [12] regarding hallucination in tool-using models. We mitigated this by lowering the model temperature to 0.2 and utilizing strict prompt schemas. Furthermore, Zhang and Zhang [6] suggest retrieval-augmented verification can reduce hallucinations; similarly, we implemented a ”Faithfulness Check” that flags responses unsupported by the retrieved vector context.

6. Conclusion

This paper presented a dual-pathway AI architecture capable of solving the distinct challenges of operational safety and commercial efficiency in tourism. By hybridizing RAG and Agentic workflows, the study successfully addressed the challenges of traversing the Himalayas. The platform improves physical safety by providing drivers with real-time intelligence and enhances economic resilience.

Future work will focus on integrating multimodal capabilities, allowing drivers to upload photos of road conditions for automated severity analysis via Vision-Language Models. The authors also plan to release the “Himalayan-RAG” dataset to encourage further research into low-resource tourism technologies.

ORCID

0009-0008-8367-2502 (Aashish Dhakal)
0009-0009-8395-7417 (Bibek Ropakheti)

Abbreviations

AI	Artificial Intelligence
LLM	Large Language Model
RAG	Retrieval-Augmented Generation
API	Application Programming Interface
SME	Small and Medium Enterprise
POI	Point of Interest
PII	Personally Identifiable Information
QA	Question Answering

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] H. Soudani, E. Kanoulas, and F. Hasibi, "Fine Tuning vs. Retrieval Augmented Generation for Less Popular Knowledge," in *Proceedings of the 2nd International ACM SIGIR Conference on Information Retrieval in the Asia Pacific*, 2024. https://doi.org/ 10.1145/3637528.3671470

[2] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, and Q. Li, "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*, 2024, pp. 6491-6501. https://doi.org/10.1145/3637528.3671470

[3] C. Qu, S. Dai, X. Wei, H. Cai, S. Wang, D. Yin, J. Xu, and J. Wen, "Tool learning with large language models: A survey," *Frontiers of Computer Science*, pp. 1-33, 2024.

[4] G. Mialon, R. Dessi, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Roziere, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, E. Grave, Y. LeCun, and T. Scialom, "Augmented language models: A survey, " *Transactions on Machine Learning Research*, 2023.

[5] C. Couturier, S. Mastorakis, H. Shen, S. Rajmohan, and V. Rühle, "Semantic caching of contextual summaries for efficient question-answering with language models," *arXiv preprint arXiv: 2505.11271*, 2025. https://doi.org/10.48550/arXiv.2505.11271

[6] W. Zhang and J. Zhang, "Hallucination mitigation for retrieval-augmented large language models: A review," *Mathematics*, vol. 13, no. 5, p. 856, 2025. https://doi.org/10.3390/math13050856

[7] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 27730-27744.

[8] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktsächel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459-9474.

[9] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al., "A survey of large language models," *arXiv preprint arXiv: 2303.18223*, 2023. https://doi.org/10.48550/arXiv.2303.18223

[10] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv: 2312.10997*, 2023. https://doi.org/10.48550/arXiv.2312.10997

[11] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, et al., "Toolllm: Facilitating large language models to master 16000+ real-world apis," *arXiv preprint arXiv: 2307.16789*, 2023. https://doi.org/10.48550/arXiv.2307.16789

[12] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al., "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *arXiv preprint arXiv: 2311.05232*, 2023. https://doi.org/10.48550/arXiv.2311.05232

[13] H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu, "A survey on retrieval-augmented text generation," *arXiv preprint arXiv: 2202.01110*, 2022. https://doi.org/10.48550/arXiv.2202.01110

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, vol. 30, 2017.

[15] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., "Llama: Open and efficient foundation language models," *arXiv preprint arXiv: 2302.13971*, 2023. https://doi.org/ 10.48550/arXiv.2302.13971

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language models are few-shot learners," in *Advances in neural information processing systems*, vol. 33, 2020, pp. 1877-1901.