Weaver Of Tasks

Vincent Tu

Disclaimer

I apologize: unfortunately, I was unable to dedicate as much time as I would've liked to this Challenge. Because of this, my submission isn't up to par with my standards. Despite this, I believe/hope my submission offers a unique direction and insights for future work. In Discussion & Limitations, I will discuss current limitation and future directions.

Table of Contents

Preliminaries

Dataset: U.S. Wildfires (Tabular; SQLite)

Method Part 1: Develop a Python Data Analyst Agent

Method Part 2: Develop a SQL Agent

Method Part 3: Develop a Search-augmented Agent

Rules:

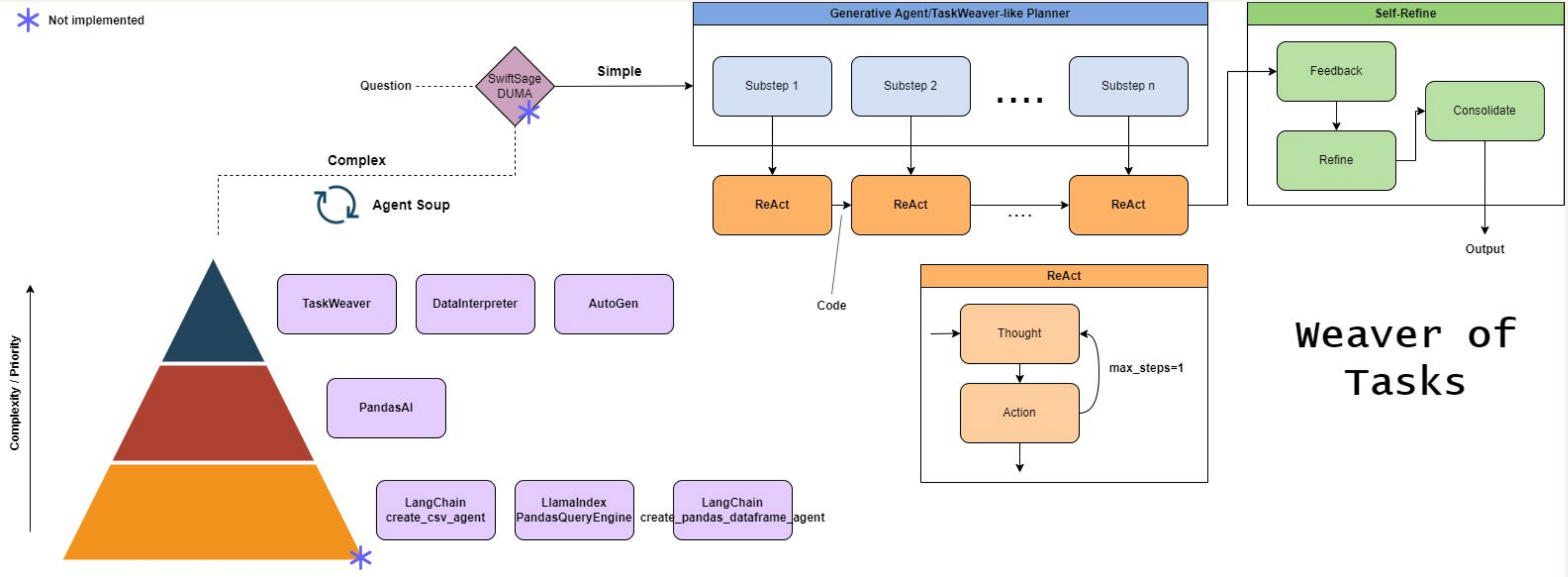
- Any LLM
- Any LLM Framework

Method: Part 1

Data Analyst Agent

- Similar to Generative Agents and TaskWeaver [1, 5], I use a question decomposition that breaks a question down into subtasks/planning steps
- I adopt the ReAct [2] structure for thinking/acting (reasoning)
- Inspired by Reflexion and Self-Refine [3, 4], I incorporate a refinement mechanism at the end of generation

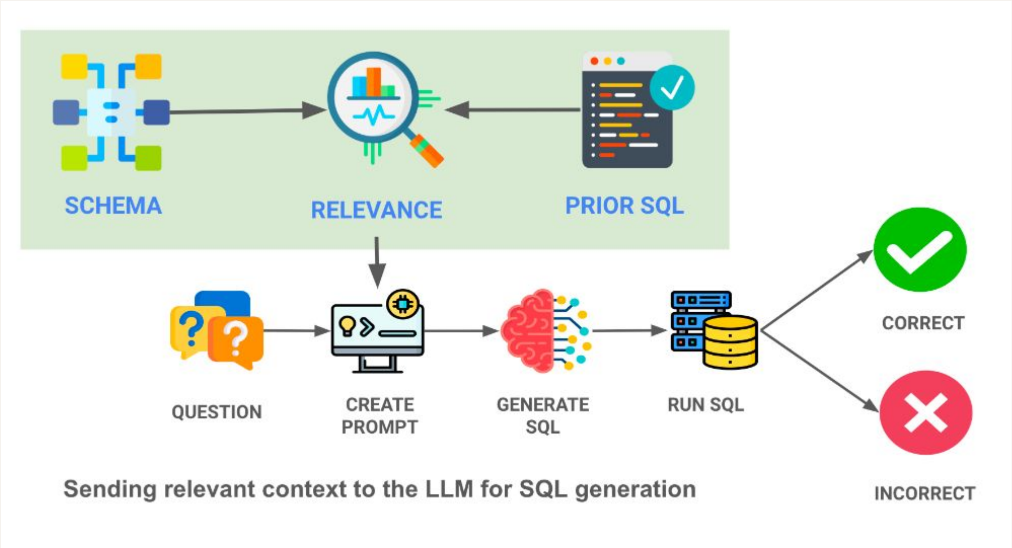
Data Analyst Agent



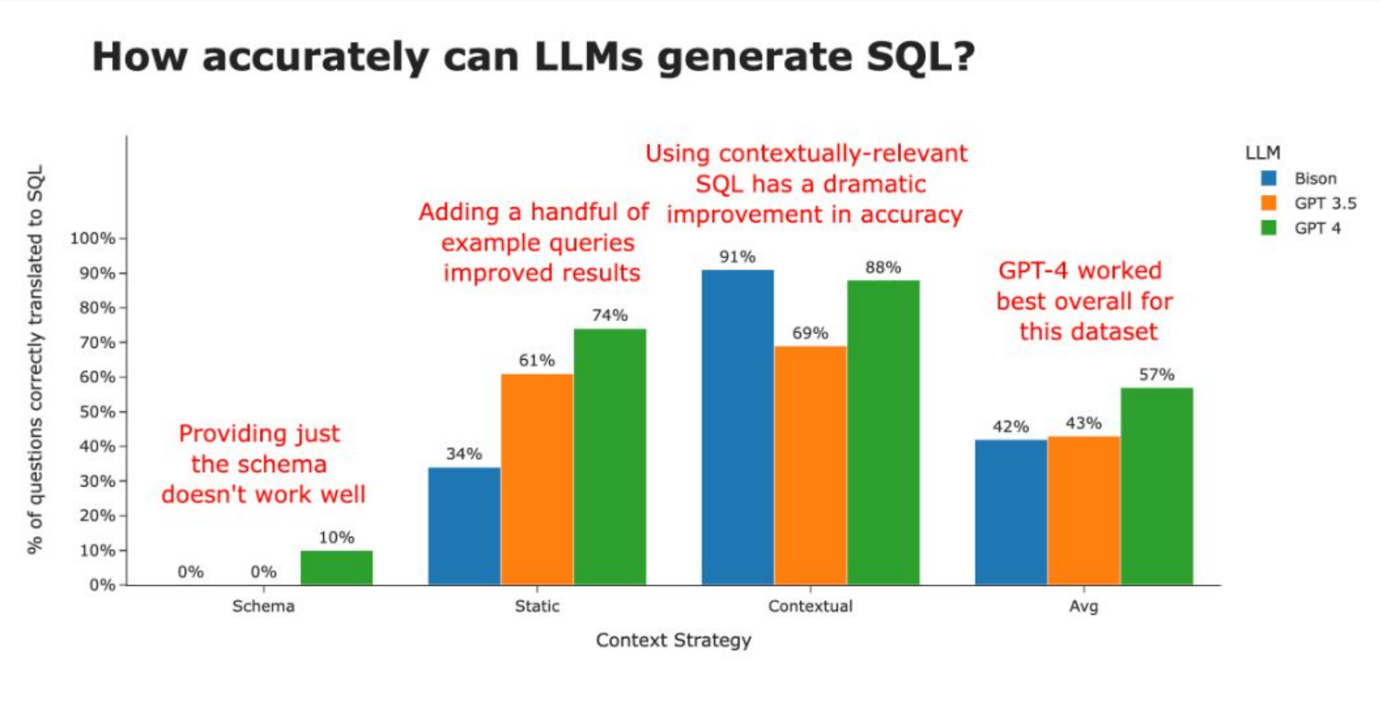
Method: Part 2

SQL Agent

- Due to time constraints, I used Vanna.AI's RAG-augmented, trainable text2SQL agent [7]



SQL Agent



How accurately can LLMs generate SQL?
% of questions correctly translated to SQL
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Providing just the schema doesn't work well
Adding a handful of example queries improved results
Using contextually-relevant SQL has a dramatic improvement in accuracy
GPT-4 worked best overall for this dataset
0%
0%
10%
34%
61%
74%
91%
69%
88%
42%
43%
57%
Schema
Static
Contextual
Avg
Context Strategy
LLM
Bison
GPT 3.5
GPT 4

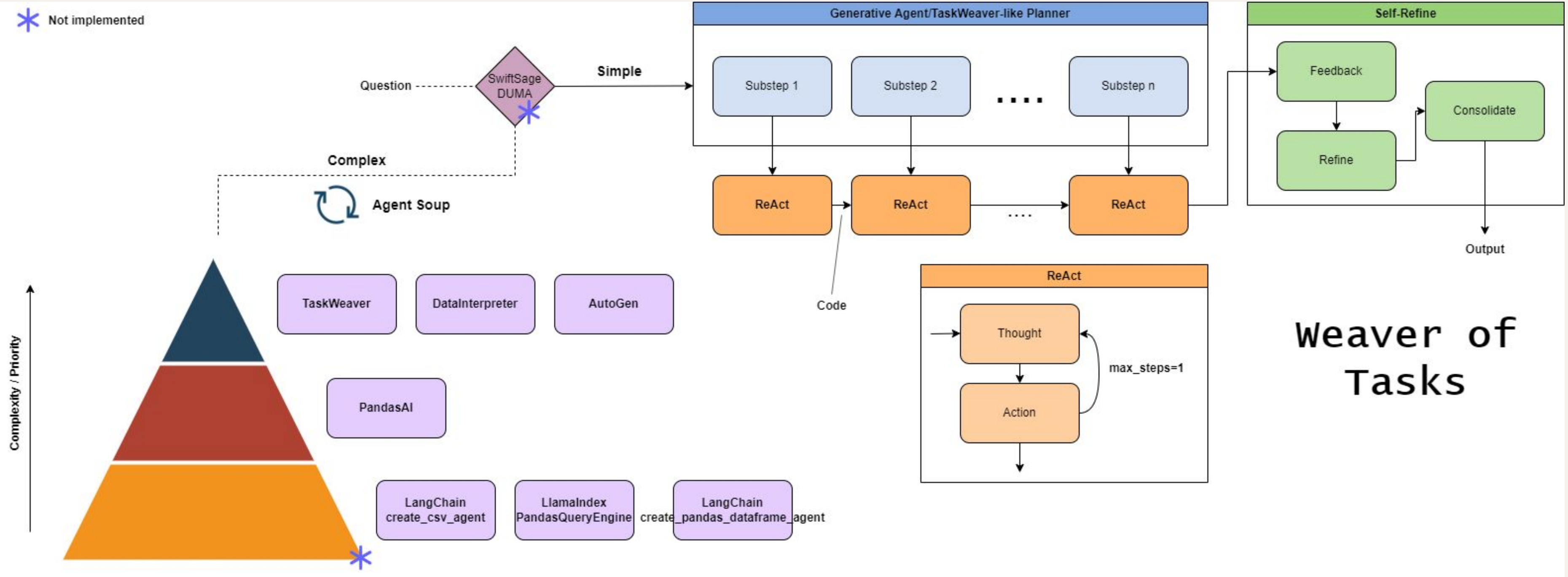
Method: Part 3

Search Agent

- Due to time constraints, I opted for LangChain's create_csv_agent augmented with You.com search and OpenWeatherMap APIs

Discussion & Limitations

Method: Part 1



Method: Part 1 Cont.

Limitations

- Custom-crafted agent derived from a mix of literature
- Not benchmarked and tested
- Lacks complexity/performance of more complex agents like TaskWeaver, DataInterpreter, and AutoGen [5, 6, 10]

Future Directions

- Adopt SwiftSage/DUMA [8, 9] Fast-mind/Slow-mind architecture; Slow-mind utilizes a complex agent like TaskWeaver, DataInterpreter, and AutoGen [5, 6, 10] (refer to previous slide)
- Thoroughly benchmark agent on Python data analysis/code generation benchmarks like HumanEval, MBPP, DataScienceProblems (DSP) [12, 13, 14]
- Experiment with open-source LLMs fine-tuned on code generation/data analysis as fast-thinkers (compute-efficient) like StarCoder [15]
- Experiment with agent fine-tuning methods like AgentTuning, FireAct, etc [16, 17]

Method: Part 2

Limitations

- High-level, abstracted, and managed Text2SQL agent separate from Method: Part 1
- Not benchmarked and tested on SQL generation and SQL-instruction-following tasks

Future Directions

- Refer to recent works in LLM-based agent SQL systems like AutoGen [10], MAC-SQL [11], and [20]
- Thoroughly benchmark Text2SQL agent on SQL-instruction-following generation benchmarks like WikiSQL [18] and [19]
- Experiment with compute-efficient open-source alternatives like StarCoder [15] fine-tuned on instruction-following SQL generation
- Tinker with a monolithic agent system (Part 1 and Part 2) or a distributed system

Method: Part 3

Limitations

- High-level, abstracted, and managed search agent separate from Method: Part 1 and Part 2
- Not benchmarked and tested on tool-use and retrieval-based tasks

Future Directions

- Refer to recent works in LLM-based agent SQL systems like AutoGen [10]
- Thoroughly benchmark search aspects of the agent on tool-use and retrieval task benchmarks like RoTBench [21], ToolQA [22], and RGB [23]
- Experiment with compute-efficient open-source alternatives like StarCoder [15] fine-tuned on retrieval-related tasks and tool-use
- Tinker with a monolithic agent system (Part 1 and Part 2) or a distributed system

In summary…

- Thoroughly explore current novel methods along multiple dimensions (code generation, Text2SQL, tool-use, and retrieval)
- Comprehensively benchmark along all relevant dimensions with popular benchmarks
- Tinker with open source alternatives as a more compute-efficient strategy
- Along any particular dimension, ablate existing state-of-the-art methods

Conclusion

Ideas for Future Work

- Dynamic Plan Generation/Updating like in Data Interpreter [6]
- Code generation verification/refinement and extensible plugins like in TaskWeaver [5]
- Synthesize & implement findings from Vanna.AI
  - RAG-based fewshot examples
  - Experiment with performant domain-specific instruction-following LLMs
- Incorporate Experiential/Continual Learning like the ExpeL Agent
- Though cost-inefficient, it would be interesting to try an Agent Soup/Ensemble (like a Model Soup) like More Agents Is All You Need [25] possibly with smart ensembling (model merging methods?)
- Leveraging agentic fine-tuning methods to further optimize compute-efficient open source LLM alternatives like AgentTuning and FireAct [16, 17] and many more [26, 27, 28, 29, 30]
- Investigate the impacts of developing a multi-agent system vs. single agent systems
  - How does a multi-agent system compare to a single agent for tool-use, retrieval, and code generation?
  - Explore into different language models and alternative architectures like RWKV or Mamba [31, 32] (underexplored for agentic tasks)
- Develop a UI with Streamlit/Chainlit/Flask 😔

References

[1] Park, Joon Sung, et al. "Generative agents: Interactive simulacra of human behavior." *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 2023.

[2] Yao, Shunyu, et al. "React: Synergizing reasoning and acting in language models." *arXiv preprint arXiv:2210.03629* (2022).

[3] Madaan, Aman, et al. "Self-refine: Iterative refinement with self-feedback." *Advances in Neural Information Processing Systems* 36 (2024).

[4] Shinn, Noah, et al. "Reflexion: Language agents with verbal reinforcement learning." *Advances in Neural Information Processing Systems* 36 (2024).

[5] Qiao, Bo, et al. "Taskweaver: A code-first agent framework." *arXiv preprint arXiv:2311.17541* (2023).

[6] Hong, Sirui, et al. "Data Interpreter: An LLM Agent For Data Science." *arXiv preprint arXiv:2402.18679* (2024).

[7] “How Accurate Can AI Generate SQL?” *Vanna.AI - Personalized AI SQL Agent*, Vanna.AI, 17 Aug. 2023, vanna.ai/blog/ai-sql-accuracy.html.

[8] Lin, Bill Yuchen, et al. "Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks." *Advances in Neural Information Processing Systems* 36 (2024).

[9] Tian, Xiaoyu, et al. "DUMA: a Dual-Mind Conversational Agent with Fast and Slow Thinking." *arXiv preprint arXiv:2310.18075* (2023).

[10] Wu, Qingyun, et al. "Autogen: Enabling next-gen llm applications via multi-agent conversation framework." *arXiv preprint arXiv:2308.08155* (2023).

[11] Wang, Bing et al. “MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL.” *ArXiv* abs/2312.11242 (2023): n. Pag.

[12] Chen, Mark, et al. "Evaluating large language models trained on code." *arXiv preprint arXiv:2107.03374* (2021).

[13] Austin, Jacob, et al. "Program synthesis with large language models." *arXiv preprint arXiv:2108.07732* (2021).

[14] Chandel, Shubham, et al. "Training and evaluating a jupyter notebook data science assistant." *arXiv preprint arXiv:2201.12901* (2022).

References

[15] Lozhkov, Anton, et al. "StarCoder 2 and The Stack v2: The Next Generation." *arXiv preprint arXiv:2402.19173* (2024).

[16] Chen, Baian, et al. "Fireact: Toward language agent fine-tuning." *arXiv preprint arXiv:2310.05915* (2023).

[17] Zeng, Aohan, et al. "Agenttuning: Enabling generalized agent abilities for llms." *arXiv preprint arXiv:2310.12823* (2023).

[18] Zhong, Victor, Caiming Xiong, and Richard Socher. "Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv 2017." *arXiv preprint arXiv:1709.00103* (2017).

[19] Nan, Linyong et al. “On Evaluating the Integration of Reasoning and Action in LLM Agents with Database Question Answering.” *ArXiv* abs/2311.09721 (2023): n. Pag.

[20] Chen, Ziru et al. “When is Tree Search Useful for LLM Planning? It Depends on the Discriminator.” *ArXiv* abs/2402.10890 (2024): n. Pag.

[21] Ye, Junjie et al. “RoTBench: A Multi-Level Benchmark for Evaluating the Robustness of Large Language Models in Tool Learning.” *ArXiv* abs/2401.08326 (2024): n. Pag.

[22] Zhuang, Yuchen et al. “ToolQA: A Dataset for LLM Question Answering with External Tools.” *ArXiv* abs/2306.13304 (2023): n. Pag.

[23] Chen, Jiawei et al. “Benchmarking Large Language Models in Retrieval-Augmented Generation.” *AAAI Conference on Artificial Intelligence* (2023).

[24] Zhao, Andrew et al. “ExpeL: LLM Agents Are Experiential Learners.” *ArXiv* abs/2308.10144 (2023): n. Pag.

[25] Li, Junyou et al. “More Agents Is All You Need.” *ArXiv* abs/2402.05120 (2024): n. Pag.

[26] Schick, Timo et al. “Toolformer: Language Models Can Teach Themselves to Use Tools.” *ArXiv* abs/2302.04761 (2023): n. Pag.

[27] Qin, Yujia et al. “ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs.” *ArXiv* abs/2307.16789 (2023): n. Pag.

[28] Gou, Zhibin et al. “ToRA: A Tool-Integrated Reasoning Agent for Mathematical Problem Solving.” *ArXiv* abs/2309.17452 (2023): n. Pag.

References

[29] Yin, Da et al. “Agent Lumos: Unified and Modular Training for Open-Source Language Agents.” (2023).

[30] Paul, Debjit et al. “REFINER: Reasoning Feedback on Intermediate Representations.” *Conference of the European Chapter of the Association for Computational Linguistics* (2023).

[31] Peng, Bo et al. “Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence.” (2024).

[32] Gu, Albert and Tri Dao. “Mamba: Linear-Time Sequence Modeling with Selective State Spaces.” *ArXiv* abs/2312.00752 (2023): n. pag.