Discussion of "LAMBDA: Large Model Based Data Agent"

Sammi Tang, Xuewei Wang

Astellas Pharma

Keywords: Large Language Model, Data Exploration, AI, Agents

Introduction

Large language models (LLMs) are making waves for efficient and reproducible data analysis. We congratulate the authors on developing an impressive LLM-based data analysis system (Sun et al., 2025a) that makes statistical and machine learning tools more accessible for users across diverse backgrounds. LAMBDA offers a remarkable contribution to this space by well-designing a dual-agent and code-free architecture for interactive data analysis. In contrast to fully autonomous LLM agents, the open-source LAMBDA emphasizes flexibility, robustness, human intervention, and domain adaptability. It provides an excellent choice for exploratory analysis and model prototyping with both structured and unstructured data.

Domain knowledge integration

One of the most scientifically compelling features of LAMBDA is its novel approach to enabling human knowledge integration for in-context learning (ICL). In contrast to existing techniques such as retrieval-augmented generation (Fan et al., 2024) or custom APIs, LAMBDA proposed a key-value (KV) knowledgebase that embeds domain expertise directly into the analytic workflow. This knowledgebase is implemented as a curated code repository, linking executable codes/scripts with structured summary of metadata (including intended task descriptions, code functionalities, and parameter specifications). Using transformer-based embedding models (e.g., Sentence-BERT), the system encodes user instructions and matches them to the most relevant code snippets via cosine similarity. In comparison to six data analysis agents (including GPT4-Advanced Data Analytics, ChatGLM-Data Analysis, OpenCodeInterpreter, Data Interpreter, and Taskweaver), LAMBDA completed domain-specific computing tasks with more effective code execution and a higher level of flexibility to deal with the misalignment between human instruction and default specifications within analysis tools.

It is worth emphasizing that the performance of LAMBDA's knowledge integration is dependent upon several design factors, including the contents of KV knowledgebase, the choice of embedding functions, the similarity measures and associated threshold settings. In the experiments conducted by the authors, LAMBDA's knowledge integration performance was derived from a default setting (the highest-scoring code snippet with cosine similarity greater than a threshold of $\theta = 0.2$). While this initial implementation provides a useful proof of concept, it also presents opportunities for further methodological assessments to better understand the impacts of these components on the performance of knowledge integration within KV framework.

For instance, prior work has shown that different transformer-based embedding models yield varying degrees of semantic similarity (Yang et al., 2020), suggesting that a similar comparative evaluation would be informative in the context of LAMBDA. Additionally, exploring similarity thresholds could further assess the reliability of the KV framework, i.e., whether higher similarity

translates to improved downstream analysis outcomes. Moreover, cosine similarity is not universally optimal for measuring similarity between text embeddings (Steck et al., 2024). Alternative similarity metrics may offer advantages depending on the structure of the embedding space and the nature of user prompts. Given that user-supplied code descriptions may vary in clarity and structure, introducing quality controls or standardization mechanisms for knowledge entries in the knowledgebase may enhance the robustness of code retrieval. Lastly, the distinction between "Full" and "Core" integration modes also deserves closer study, particularly in relation to task complexity and model interpretability.

Human intervention and transparency

LAMBDA's architecture is particularly attractive for its support of human-in-the-loop interaction. A key strength of this architecture is that it allows users to verify intermediate steps in the analytical pipeline. By presenting intermediate analysis codes and results at each step, the platform facilitates transparency and user control. This not only empowers users to debug and iterate effectively but also builds trust in the system's outputs. Additionally, the transparent interaction allows domain experts to refine prompts or modify code proactively, which ensures relevance and accuracy in specialized settings such as clinical research or finance. As such, LAMBDA provides an excellent example of responsible data science, by involving the user throughout the analytical process.

Moreover, the option for automated report generation allows to capture the prompt history, outputs, and visualizations into structured reports. This functionality bridges the gap between exploratory analysis and formal documentation. In many settings, such as regulatory submission, grant reporting, or academic publishing, transparency in methods and traceable workflows are mandatory. LAMBDA's ability to automatically generate such records from interactive sessions significantly reduces the documentation burden while enhancing compliance and clarity. These reports can also be shared across teams, making collaboration more efficient and reducing redundant effort. Integrating this feature with version control systems or collaborative notebooks (e.g., GitHub, JupyterHub) could elevate its utility further and turn LAMBDA into a more holistic data science platform.

These features effectively help users comprehend the data analysis process (from user inputs, data preprocessing steps, to model choices and results), reinforcing reproducibility and transparency. This is especially valuable in collaborative or educational settings, where traceability of analytical steps is crucial for peer review and didactic purposes. Future iterations of LAMBDA could further enhance the usability by including more explainable AI features or guided walkthroughs for less experienced users.

Engineering considerations for production

From an engineering standpoint, one of LAMBDA's notable innovations lies in its dual-agent architecture: a programmer that generates executable codes in response to user instruction, and an inspector that suggests corrections when errors arise during code execution. This self-correcting mechanism significantly improves system reliability. Specifically, the code execution success rate increased from 68.06% (with programmer alone) to 95.37% (with the dual-agent setting), which is decent to support its use for exploratory purpose. However, further efforts would be required to improve the system's reliability if one desires to make it production-ready.

First of consideration is to develop solutions to automatically address the remaining 4.63% failed execution, potentially through additional corrective agents. Second, successful code execution may not always guarantee intended analytical outcomes, given the known limitations and bugs associated with LLM-generated codes (Tambon et al., 2025). It's suggested to conduct additional reliability evaluations to assess the impact of such bugs, i.e., through pressure

test in edge-case scenarios. Moreover, extending the inspector's capabilities to verify analytical intent, such as appropriate use of variables or model assumptions, could further enhance system reliability, especially for less experienced users. This would move the inspector beyond syntactic correction toward semantic validation, which aligns with LAMBDA's goal of supporting domain-adaptable and human-guided workflows.

Performance benchmarking with domain-aware strategies

To assess LAMBDA's effectiveness, the authors benchmarked its performance across a diverse collection of datasets encompassing various analytic tasks (classification and regression), data modalities (tabular, imaging, text), and challenging scenarios such as high-dimensionality and missing data. For classification/regression tasks on classical tabular datasets, LAMBDA performs comparably to or outperforms manual baseline analyses conducted using R, highlighting its utility for routine applications. Moreover, LAMBDA demonstrates adaptability in handling more complex analysis scenarios, by the selection of proper statistical methods or machine learning models. Altogether, these experiments highlighted LAMBDA's versatility to deal with a broader range of analysis tasks and scenarios, therefore holding a great potential for general-purpose automated data exploration.

While these results are promising, further domain-specific validation would strengthen our understanding of LAMBDA's robustness in real-world scenarios. For example, all the three tested high-dimensional datasets come from transcriptomic studies (mRNA, microRNA or both), and LAMBDA consistently applied principal component analysis (PCA) for dimension reduction. The best model accuracies, ranging from 56.62% to 70.63% (in Table 4), are modest and may reflect the limitations of PCA in such contexts. In fact, PCA is often suboptimal for transcriptomic data (Xiang et al., 2021). Machine learning community has developed numerous methods to deal with the challenge of small n and large p problem (Guyon and Elisseeff, 2003), with some specific to analyze transcriptomic data (Wu and Ma, 2015). Consequently, the use of PCA here may not reflect best practices in the field. Moreover, PCA is an unsupervised method and often misaligned with supervised learning tasks common in transcriptomic analysis. Established alternatives such as sparse PCA, penalized regression-based selection, or PLS-Discriminant Analysis (Lee et al., 2018) could offer more informative feature sets. Incorporating such specialized methods, either through automated recommendations or domain-specific workflows, would greatly enhance LAMBDA's effectiveness in high-dimensional biomedical applications.

Like high-dimensional setting, various domains have developed specialized analysis strategies or methodologies to address their specific data or analysis challenges. Enabling such domain-specific strategies in LAMBDA or similar systems would enhance their applicability across disciplines. To achieve this, it is crucial to expand evaluation with a broader range of domain-specific datasets or use cases. Such rigorous validation would require cross-disciplinary collaboration and could be facilitated through community-driven benchmarking or crowdsourced testing initiatives.

Expansion with data and programming support

Data preparation remains a major component of the data science practice. Recent advances in generative AI are increasingly being applied to streamline data engineering tasks, i.e., extracting structured elements from internal data platforms (Yang et al., 2025) or publicly available scientific databases (Cinquin, 2024). The integration of such capabilities, through the development or integration of data engineering agents, would substantially extend LAMBDA's capability for end-to-end workflow from raw data to analytical insights.

LAMBDA currently operates with Python-based kernel support. While this choice ensures broad compatibility with growing Python-based data science ecosystem, the platform's utility

could be significantly enhanced through multi-language support. Particularly, incorporating R (widely adopted in statistics and closely aligned fields), would expand accessibility and promote adoption within a much broader community. Ongoing efforts to develop R-specific analysis agents, such as LLMAgentR (Boakye, 2025) and mergen (Jansen et al., 2025), provide promising avenues for such extensions. Integration of these tools into the LAMBDA framework would further its mission of democratizing data analysis and supporting reproducible, domain-adaptable research workflows.

While supporting multiple programming languages independently is feasible, another practical scenario in real-world application involves the development of hybrid programming workflows. For example, allowing data preprocessing in Python followed by statistical modeling in R, or vice versa. Such hybrid programming can help expand LAMBDA's practical relevance and could be achieved by a modular backend and agent-specific language selection, especially as tools like LLMAgentR continue to mature.

Summary

LAMBDA is well-engineered and represents a promising advancement in LLM-assisted data analysis, offering an architecture that emphasizes robustness, transparency, and domain adaptability. Its dual-agent design and KV-based knowledge integration distinguish it from other automated agent systems by supporting greater human oversight and iterative refinement. LAMBDA's flexibility to interact not only supports user comprehension of overall process, but also contributes to reproducibility and educational purposes.

While its current design facilitates effective human-in-the-loop exploration, future development around analytic intent verification, domain-specific validation, and hybrid programming support will be critical. The system holds great potential for future extensions, given its customizable knowledgebase and modular architecture. By integrating broader domain knowledge, expanding language capability and improving reliability through rigorous validation, it could evolve into a platform adaptable to specialized domains and serve as a strong example for trustworthy AI-driven data analytics platforms.

Its architecture also allows collaborative development across disciplines. Building such intelligent analysis systems at scale will require methodological refinement, cross-disciplinary benchmarking, and community-driven innovation, presenting an exciting frontier for next-generation data science.

References

Boakye, K. D. N. O. (2025). Llmagentr: Language model agents in r for ai workflows and research. *Review*.

Cinquin, O. (2024). Steering veridical large language model analyses by correcting and enriching generated database queries: first steps toward chatgpt bioinformatics. *Briefings in Bioinformatics*, 26:bbaf045.

Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., and Li, Q. (2024). 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*, pages 6491–6501, New York, NY, USA. Association for Computing Machinery.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

Jansen, J. A., Manukyan, A., Al Khoury, N., and Akalin, A. (2025). Leveraging large language models for data analysis automation. *PLOS ONE*, 20:e0317084.

Lee, L. C., Liong, C.-Y., and Jemain, A. A. (2018). Partial least squares-discriminant analysis (pls-da) for classification of high-dimensional (hd) data: a review of contemporary practice strategies and knowledge gaps. *Analyst*, 143:3526–3539.

Steck, H., Ekanadham, C., and Kallus, N. (2024). Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM Web Conference 2024*, pages 887–890.

Sun, M., Han, R., Jiang, B., Qi, H., Sun, D., Yuan, Y., and Huang, J. (2025a). Lambda: A large model based data agent. *Journal of the American Statistical Association*, pages 1–13.

Sun, M., Han, R., Jiang, B., Qi, H., Sun, D., Yuan, Y., and Huang, J. (2025b). A survey on large language model-based agents for statistics and data science. *The American Statistician*, pages 1–14.

Tambon, F., Moradi-Dakhel, A., Nikanjam, A., Khomh, F., Desmarais, M. C., and Antoniol, G. (2025). Bugs in large language models generated code: an empirical study. *Empirical Software Engineering*, 30:65.

Wu, C. and Ma, S. (2015). A selective review of robust variable selection with applications in bioinformatics. *Briefings in Bioinformatics*, 16:873–883.

Xiang, R., Wang, W., Yang, L., Wang, S., Xu, C., and Chen, X. (2021). A comparison for dimensionality reduction methods of single-cell rna-seq data. *Frontiers in Genetics*, 12.

Yang, X., He, X., Zhang, H., Ma, Y., Bian, J., and Wu, Y. (2020). Measurement of semantic textual similarity in clinical texts: Comparison of transformer-based models. *JMIR Medical Informatics*, 8:e19735.

Yang, Z., Xu, S.-S., Liu, X., Xu, N., Chen, Y., Wang, S., Miao, M.-Y., Hou, M., Liu, S., Zhou, Y.-M., Zhou, J.-X., and Zhang, L. (2025). Large language model-based critical care big data deployment and extraction: Descriptive analysis. *JMIR Medical Informatics*, 13:e63216.