DomAgent: Leveraging Knowledge Graphs and Case-Based Reasoning for Domain-Specific Code Generation

Extended Abstract

Shuai Wang
Chalmers University of Technology
Gothenburg, Sweden
shuaiwa@chalmers.se

Dhasarathy Parthasarathy
Volvo Group
Gothenburg, Sweden
dhasarathy.parthasarathy@volvo.com

Robert Feldt
Chalmers University of Technology
Gothenburg, Sweden
robert.feldt@chalmers.se

Yinan Yu
Chalmers University of Technology
Gothenburg, Sweden
yinan@chalmers.se

ABSTRACT

Large language models (LLMs) perform well on general code generation but often struggle with domain-specific software tasks due to limited specialized knowledge in their training data. We propose DomAgent, an autonomous coding agent that enables domain-adapted code generation through structured reasoning and targeted retrieval. Its core module, DomRetriever, combines knowledge-graph reasoning with case-based reasoning to iteratively retrieve and synthesize relevant domain knowledge and examples. Experiments on the DS-1000 benchmark and real-world Volvo truck software development tasks show that DomAgent significantly improves domain-specific code generation, allowing small open-source models to approach the performance of large proprietary LLMs. The code is publicly available at: https://github.com/Wangshuaiia/DomAgent.

KEYWORDS

LLMs; Domain-Specific Code Generation; Knowledge Graph

ACM Reference Format:
Shuai Wang, Dhasarathy Parthasarathy, Robert Feldt, and Yinan Yu. 2026. DomAgent: Leveraging Knowledge Graphs and Case-Based Reasoning for Domain-Specific Code Generation: Extended Abstract. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/HSTL5347

1 INTRODUCTION

Large language models (LLMs) have achieved strong performance in general code generation [10, 14, 15, 27] and are widely used in tools such as GitHub Copilot [25, 30, 39]. However, real-world software development is often domain-specific [4, 6, 16], requiring knowledge of specialized libraries [13, 40], system logic [9, 18], and workflow constraints [34]. Because most LLMs are trained on general public data, they frequently struggle with such tasks due to limited domain knowledge and weak context adaptation [2, 19, 23].

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/HSTL5347

Fine-tuning is one possible solution, but it is costly and difficult to maintain as domain libraries evolve [1, 5, 8, 36]. Retrieval-Augmented Generation (RAG) offers a more flexible alternative by providing external knowledge at inference time [33]. Existing retrieval methods mainly rely on textual similarity [11, 22, 29, 35, 42] or example matching [41]. While effective in simple settings, they often fail to precisely retrieve relevant packages [3, 37], functions [7, 43], or structured domain knowledge in complex software systems [24, 28].

In practice, domain-specific coding requires both concrete examples (bottom-up learning) [4, 12, 38] and structured conceptual knowledge (top-down reasoning) [31, 32]. Inspired by how humans learn, we propose an agent framework that integrates case-based retrieval with knowledge-graph-based retrieval. Our system uses knowledge graphs to represent domain packages and their relationships, and leverages this structure to guide diverse and representative case selection.

At the core of the framework is DomRetriever, a retrieval module that dynamically combines top-down knowledge reasoning and bottom-up example selection. Experiments on a data science benchmark and real-world truck software development tasks demonstrate that our approach significantly improves domain-specific code generation and enables smaller models to perform competitively in complex settings.

2 PROBLEM DEFINITION

We study domain-specific code generation with structured knowledge and reusable examples. Given a task q, a knowledge graph $\mathcal{G} = \langle e, r, e' \rangle$ encoding domain entities (e.g., packages and functions) and their relations, and a case base $\mathcal{B}$ of reference code examples, the objective is to retrieve relevant knowledge and representative cases, and synthesize the target code $\hat{y}$. Since constructing cases is costly, we aim to maintain a compact yet diverse case base that provides broad coverage with minimal redundancy.

3 METHODS

Our framework consists of three components: (1) coverage-aware case base construction guided by a knowledge graph (KG); (2) a unified retrieval module (DomRetriever) that integrates top-down

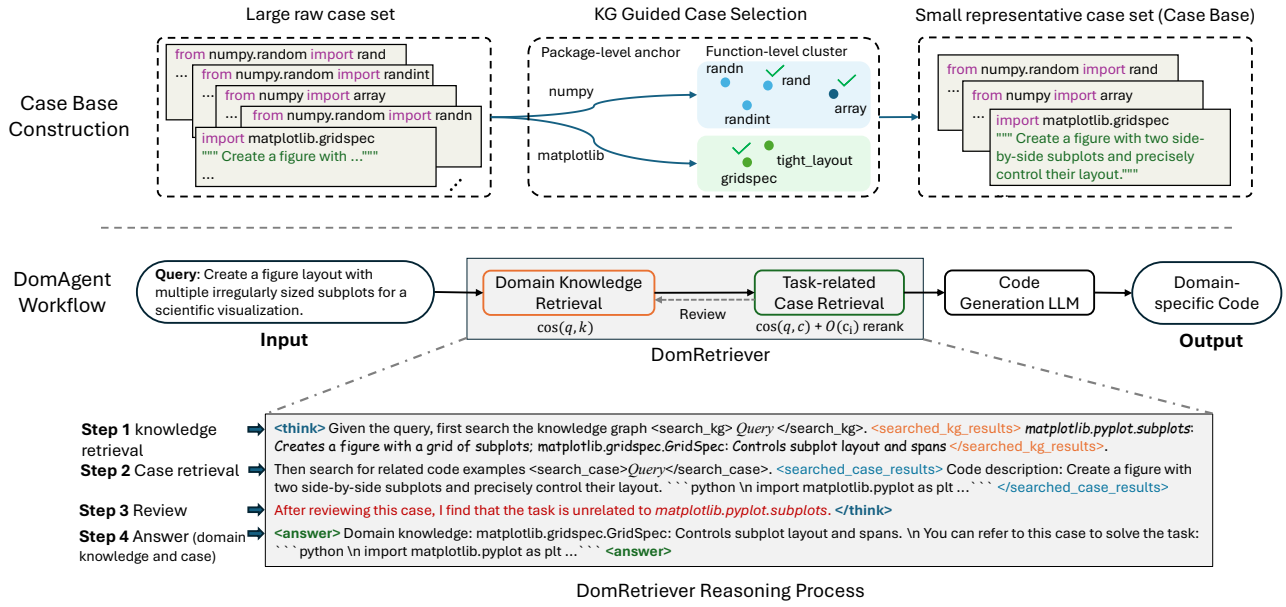

Figure 1: Case base construction (top) and the overall workflow of our DomAgent (bottom).

Table 1: Overall performance comparison (pass@1) on DS-1000.

Model	Size	Total
LLaMA3.1-8B	8B	30.4
GPT-4o	-	51.0
StarCoder [20]	15B	26.0*
WizardCoder [21]	15B	29.2*
CodeLLaMA-Python [26]	7B	28.0*
DomAgent (LLaMA3.1-8B; Ours)	8B	40.5
LLaMA3.1-8B + LLaMA3.3-70B (Ours)	70B	47.9
LLaMA3.1-8B + GPT-4o (Ours)	-	**58.6**

Table 2: Performance of GPT-4o-based models on Volvo truck CAN signal reading/writing across six domains.

Domain	Vanilla GPT-4o	+DomRetriever
Driver productivity	65.1	100
Connected systems	73.4	97.4
Energy	58.8	100
Vehicle system	71.1	97.1
Visibility	82.2	100
Dynamics	75.5	96.4
Total	**71.22**	**98.04**

knowledge retrieval and bottom-up case retrieval; and (3) reinforcement learning to train the agent to autonomously invoke retrieval tools.

Case Base Construction. We leverage the hierarchical structure of the KG to build a compact yet representative case base. For each package, we cluster its functions in embedding space to capture diverse usage patterns. We then iteratively select code cases that maximize package coverage and functional diversity, stopping when predefined coverage thresholds are met. Only executable and non-redundant cases are retained.

DomRetriever. Given a task q, we first perform top-down retrieval by identifying relevant packages and retrieving related KG nodes via embedding similarity. We then perform bottom-up retrieval by selecting semantically similar cases and re-ranking them based on overlap with retrieved packages. A reasoning LLM iteratively refines both knowledge and cases through tool calls (`SearchKG`, `SearchCase`). The refined knowledge and representative case are concatenated with q to generate the final code.

Agent Training. We fine-tune the LLM to learn tool invocation and apply reinforcement learning to optimize retrieval quality. A reward model evaluates the relevance between the task and retrieved knowledge/cases, and GRPO is used to update the agent policy.

4 RESULTS AND CONCLUSIONS

We evaluate our method on DS-1000 [17] and a real-world truck CAN signal task. DS-1000 includes 1,000 data science problems across seven libraries, supported by the DS-KG (505K triples); we build the case base from 300 tasks and test on the remaining 700. For industrial validation, we apply the system to 776 CAN signals across six domains in Volvo Truck.

The results are shown in Table 1 and 2 respectively. Across both benchmark and real-world industrial tasks, our approach substantially improves domain-specific code generation and enables small open-source models to approach or even match large proprietary systems. This demonstrates that structured retrieval, rather than large-scale fine-tuning alone, is key to practical domain adaptation.

REFERENCES

[1] DM Anisuzzaman, Jeffrey G Malins, Paul A Friedman, and Zachi I Attia. 2025. Fine-tuning large language models for specialized use cases. *Mayo Clinic Proceedings: Digital Health* 3, 1 (2025), 100184.

[2] Mihir Athale and Vishal Vaddina. 2025. Knowledge Graph Based Repository-Level Code Generation. In *2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*. IEEE, 169–176.

[3] Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, et al. 2025. Main-rag: Multi-agent filtering retrieval-augmented generation. *The 63rd Annual Meeting of the Association for Computational Linguistics (ACL)* (2025).

[4] Dustin Dannenhauer, Zohreh Dannenhauer, Despina Christou, and Kostas Hatalis. 2024. A case-based reasoning approach to dynamic few-shot prompting for code generation. In *ICML 2024 Workshop on LLMs and Cognition*.

[5] Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. How Abilities in Large Language Models are Affected by Supervised Fine-tuning Data Composition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 177–198.

[6] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. 2024. A Survey on In-context Learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 1107–1128.

[7] Xinyu Gao, Yun Xiong, Deze Wang, Zhenhan Guan, Zejian Shi, Haofen Wang, and Shanshan Li. 2024. Preference-guided refactored tuning for retrieval augmented code generation. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 65–77.

[8] Yubin Ge, Devamanyu Hazarika, Yang Liu, and Mahdi Namazifar. [n.d.]. Supervised Fine-Tuning of Large Language Models on Human Demonstrations Through the Lens of Memorization. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

[9] Xiaodong Gu, Meng Chen, Yalan Lin, Yuhan Hu, Hongyu Zhang, Chengcheng Wan, Zhao Wei, Yong Xu, and Juhong Wang. 2025. On the effectiveness of large language models in domain-specific code generation. *ACM Transactions on Software Engineering and Methodology* 34, 3 (2025), 1–22.

[10] Daya Guo, Qihao Zhu, Dejian Yang, and et al. 2024. DeepSeek-Coder: When the Large Language Model Meets Programming – The Rise of Code Intelligence. *arXiv preprint arXiv:2401.14196* (2024).

[11] Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. DS-Agent: Automated Data Science by Empowering Large Language Models with Case-Based Reasoning. In *ICML*.

[12] Kostas Hatalis, Despina Christou, and Vyshnavi Kondapalli. 2025. Review of case-based reasoning for LLM agents: theoretical foundations, architectural components, and cognitive integration. *arXiv preprint arXiv:2504.06943* (2025).

[13] Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2025. Next-generation database interfaces: A survey of llm-based text-to-sql. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[14] Bohan Hui et al. 2024. Qwen2.5-Coder Technical Report. *arXiv preprint arXiv:2409.12186* (2024).

[15] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, and et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).

[16] Sathvik Joel, Jie Wu, and Fatemeh Fard. 2024. A survey on llm-based code generation for low-resource and domain-specific programming languages. *ACM Transactions on Software Engineering and Methodology* (2024).

[17] Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. DS-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*. PMLR, 18319–18345.

[18] Chengwei Li, Zhenyu Xu, Bowen Wu, Xiang Li, Wenqiang Zhang, Ningyu Zhang, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2025. Retrieval-Augmented Code Generation for Universal Information Extraction. *arXiv preprint arXiv:2501.04702* (2025). https://arxiv.org/abs/2501.04702

[19] Jia Li, Ge Li, Xuanming Zhang, Yunfei Zhao, Yihong Dong, Zhi Jin, Binhua Li, Fei Huang, and Yongbin Li. 2024. Evocodebench: An evolving code generation benchmark with domain-specific evaluations. *Advances in Neural Information Processing Systems* 37 (2024), 57619–57641.

[20] Raymond Li, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia LI, Jenny Chim, Qian Liu, et al. [n.d.]. StarCoder: may the source be with you! *Transactions on Machine Learning Research* ([n. d.]).

[21] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2024. WizardCoder: Empowering Code Large Language Models with Evol-Instruct. In *ICLR*.

[22] Noor Nashid, Mifta Sintaha, and Ali Mesbah. 2023. Retrieval-based prompt selection for code-related few-shot learning. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2450–2462.

[23] Shuyin Ouyang, Jie Zhang, Zeyu Sun, and Albert Merono Penuela. 2025. Knowledge-Enhanced Program Repair for Data Science Code. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 782–782.

[24] Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 237–250.

[25] Bo Qiao, Liqun Li, Xu Zhang, Shilin He, Yu Kang, Chaoyun Zhang, Fangkai Yang, Hang Dong, Jue Zhang, Lu Wang, et al. 2023. Taskweaver: A code-first agent framework. *arXiv preprint arXiv:2311.17541* (2023).

[26] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023).

[27] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, and et al. 2023. Code Llama: Open Foundation Models for Code. *arXiv preprint arXiv:2308.12950* (2023).

[28] Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. 2024. Fine tuning vs. retrieval augmented generation for less popular knowledge. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. 12–22.

[29] Hanzhuo Tan, Qi Luo, Ling Jiang, Zizheng Zhan, Jing Li, Haotian Zhang, and Yuqun Zhang. 2024. Prompt-based code completion via multi-retrieval augmented generation. *ACM Transactions on Software Engineering and Methodology* (2024).

[30] Tim Tully, Joff Redfern, and Derek Xiao. [n.d.]. *2024: The State of Generative AI in the Enterprise*. https://menlovc.com/2024-the-state-of-generative-ai-in-the-enterprise/

[31] Shuai Wang, Wenji Mao, Penghui Wei, and Daniel D Zeng. 2022. Knowledge structure driven prototype learning and verification for fact checking. *Knowledge-Based Systems* 238 (2022), 107910.

[32] Shuai Wang, Penghui Wei, Qingchao Kong, and Wenji Mao. 2024. A knowledge enhanced learning and semantic composition model for multi-claim fact checking. *Knowledge-Based Systems* 304 (2024), 112439.

[33] Shuai Wang and Yinan Yu. 2025. iQUEST: An Iterative Question-Guided Framework for Knowledge Base Question Answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 15616–15628. https://doi.org/10.18653/v1/2025.acl-long.760

[34] Shuai Wang, Yinan Yu, Robert Feldt, and Dhasarathy Parthasarathy. 2025. Automating a Complete Software Test Process Using LLMs: An Automotive Case Study. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE, 373–384.

[35] Xixi Wang, Miguel Costa, Jordanka Kovaceva, Shuai Wang, and Francisco C Pereira. 2025. Plugging Schema Graph into Multi-Table QA: A Human-Guided Framework for Reducing LLM Reliance. *arXiv preprint arXiv:2506.04427* (2025).

[36] Xixi Wang, Jordanka Kovaceva, Miguel Costa, Shuai Wang, Francisco Camara Pereira, and Robert Thomson. 2025. Domain-Adapted Pre-trained Language Models for Implicit Information Extraction in Crash Narratives. *arXiv preprint arXiv:2510.09434* (2025).

[37] Zora Zhiruo Wang, Akari Asai, Xinyan Velocity Yu, Frank F. Xu, Yiqing Xie, Graham Neubig, and Daniel Fried. 2025. CodeRAG-Bench: Can Retrieval Augment Code Generation?. In *Findings of the Association for Computational Linguistics: NAACL 2025*, Luis Chiruzzo, Alan Ritter, and Lu Wang (Eds.). 3199–3214.

[38] Ian Watson and Farhi Marir. 1994. Case-based reasoning: A review. *The knowledge engineering review* 9, 4 (1994), 327–354.

[39] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. [n.d.]. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

[40] Zezhou Yang, Sirong Chen, Cuiyun Gao, Zhenhao Li, Xing Hu, Kui Liu, and Xin Xia. 2025. An empirical study of retrieval-augmented code generation: Challenges and opportunities. *ACM Transactions on Software Engineering and Methodology* (2025).

[41] Zhixiong Zeng, Shuai Wang, Nan Xu, and Wenji Mao. 2021. Pan: Prototype-based adaptive network for robust cross-modal retrieval. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 1125–1134.

[42] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473* (2024).

[43] Hao Zhu, Yiming Zhang, Shikun Feng, Zhiqiang Chen, Pengjun Qian, Min Zhang, and Jie Zhou. 2023. AceCoder: Utilizing Existing Code to Enhance Code Generation. *arXiv preprint arXiv:2312.03618* (2023). https://arxiv.org/abs/2312.03618