

Agentic SQL Generation for Reliable Data Warehouse Query Automation

Authors

Ainsley R. Hartwell^{1*}, Edward J. Langford², Amelia S. Wren³

Affiliations

Department of Computer Science, School of Engineering, Faculty of Science and Engineering,
The University of Manchester, Kilburn Building, Oxford Road, Manchester M13 9PL, United Kingdom

*Corresponding author: a.r.hartwell@manchester.ac.uk

Abstract

Natural-language query generation has become important for enterprise data warehouses, but generated SQL often contains incorrect joins, missing filters, aggregation errors, inefficient execution plans, and schema hallucinations. This study investigates reliable SQL generation through agentic query validation and optimization. We propose SQLAgent-Opt, a multi-agent model consisting of a schema interpretation agent, a query drafting agent, an execution validation agent, a semantic checking agent, and a cost optimization agent. The schema interpretation agent extracts table relationships, primary keys, foreign keys, column types, and business constraints from database metadata. The query drafting agent converts natural-language questions into executable SQL. The execution validation agent checks syntax, runtime errors, and empty-result anomalies. The semantic checking agent verifies whether selected columns, joins, grouping rules, and filtering conditions match the original question. The cost optimization agent rewrites queries using index-aware predicates, join reordering, subquery simplification, and aggregation pruning. Experiments were conducted on 1,540 natural-language query tasks from finance, healthcare, retail, and logistics data warehouses, covering 326 relational tables, 4,870 columns, and 18.6 million records. Compared with a single-agent SQL generator, SQLAgent-Opt improved exact-match accuracy from 58.9% to 73.6% and execution accuracy from 66.4% to 82.1%. The rate of incorrect join paths decreased from 19.7% to 7.8%, while aggregation-related errors decreased by 34.5%. On large tables containing more than one million records, the optimized queries reduced average execution time by 41.2% and temporary memory usage by 26.8%. These results show that agentic SQL generation can improve both semantic correctness and execution efficiency in data warehouse automation.

Keywords: SQL generation; data warehouse; query optimization; multi-agent systems; semantic validation; natural-language interface; database automation

1. Introduction

Enterprise data warehouses integrate large volumes of financial, clinical, retail, logistics, and operational records across many interconnected tables. Natural-language interfaces provide an attractive way for analysts and business users to access these data without manually writing complex SQL statements. However, generating executable SQL is only part of the problem. A query may pass syntax checking and run successfully while still producing an incorrect result. It may follow an inappropriate join path, omit a necessary business constraint, aggregate records at the wrong granularity, confuse two semantically similar columns, or apply a condition to the wrong stage of the query. These errors are particularly difficult to detect because the returned table may appear plausible even when its business meaning is incorrect. Earlier Text-to-SQL research reduced syntactic failures through constrained decoding, grammar-based generation, and structured parsing [1-3]. Subsequent studies placed greater emphasis on schema linking, compositional generalization, and evaluation under changed questions or database schemas [3]. Related research has also shown that learned models may rely on multiple shortcut signals rather than the underlying task-relevant structure, and that explicitly identifying and removing such shortcuts can improve robustness [4-6]. This issue is highly relevant to enterprise Text-to-SQL because superficial similarities in field names, table structures, or previously observed query patterns may encourage a model to generate SQL that looks appropriate but does not reflect the actual business intent. Robust enterprise query generation therefore requires semantic verification in addition to syntactic correctness.

Large language models have substantially expanded the capabilities of Text-to-SQL systems. SQL-PaLM, DIN-SQL, DAIL-SQL, and CodeS introduced prompting strategies, task decomposition, model adaptation, demonstration selection, and execution feedback to improve SQL generation [7-10]. These approaches showed that language models can reason over schemas, decompose complicated questions, and revise generated queries based on intermediate feedback. Research communities in China have also contributed extensively to schema matching, large-scale database benchmarks, open Text-to-SQL models, and practical database reasoning [7,8]. At the same time, evaluation settings have gradually moved beyond small academic databases. BIRD introduced database values, realistic schema information, and query-efficiency considerations into Text-to-SQL evaluation [9]. BEAVER collected queries from enterprise data-warehouse environments, while Spider 2.0 extended the task toward complex enterprise workflows, multiple SQL dialects, cloud database platforms, and large schemas [11-14]. These developments demonstrate that performance measured on conventional benchmarks does not necessarily transfer directly to enterprise warehouses. The difference arises from both schema complexity and business semantics. Public Text-to-SQL datasets commonly provide relatively stable schemas, explicit foreign-key relationships, concise questions, and well-defined table meanings. Enterprise databases are much less uniform. A warehouse may contain hundreds of tables created by different departments over many years, duplicated or abbreviated field names, historical tables,

derived indicators, undocumented relationships, and organization-specific naming conventions. The same concept may be represented differently across systems, while columns with similar names may encode different accounting or operational definitions [15-18]. Business questions may also contain implicit assumptions that are not directly stated in the schema. A request for “monthly active customers,” for instance, may depend on an internal definition of activity, customer status, cancellation handling, and reporting period. In this setting, schema linking cannot be reduced to lexical similarity between question terms and database fields. The system must infer which tables and attributes represent the intended business concept and determine how those concepts should be connected before SQL generation begins [19-22]. Enterprise environments introduce additional difficulties after the relevant schema has been identified. Join paths are a major source of semantic error because several technically valid paths may connect the same tables. A longer but semantically correct relationship can be replaced accidentally by a shorter path that changes the meaning or duplicates records [23-27]. Aggregation introduces another source of failure. Applying a filter before or after aggregation, using an inappropriate grouping key, or joining one-to-many tables before computing totals can substantially distort the result. Similar problems occur with temporal conditions, null handling, deduplication, nested queries, and derived business indicators. These failures are not always detected through execution-based validation because the database engine only verifies whether the SQL statement is executable [28-31]. A successful execution therefore provides weak evidence that the query correctly represents the natural-language request. Enterprise Text-to-SQL systems need validation mechanisms that consider the intended semantics of joins, filters, aggregation levels, and selected attributes rather than treating execution success as the final correctness criterion. Multi-agent approaches provide a promising way to introduce such validation and repair processes[32-35]. R³ uses agent discussion to revise candidate queries, SQLFixAgent focuses on identifying and correcting semantic SQL errors, and Tool-SQL targets queries that execute successfully but fail to satisfy the requested conditions [20]. MAG-SQL, CHESS, and CHASE-SQL further incorporate schema selection, decomposition, candidate generation, iterative refinement, and query evaluation into multi-stage workflows [21,22]. These systems indicate that separating generation from review can improve Text-to-SQL reliability, particularly for complex questions. Nevertheless, most existing approaches still evaluate success primarily through execution accuracy, exact matching, or selection among candidate queries. The internal causes of semantic errors are often not separated explicitly. Schema misunderstanding, incorrect join selection, inappropriate aggregation, missing filters, and business-rule violations may all lead to an incorrect answer, but they require different repair strategies[36-38]. A general retry prompt may correct some failures while introducing new ones elsewhere. Query efficiency creates another important requirement in enterprise environments. Large data warehouses often contain tables with millions or billions of records, so two semantically equivalent SQL statements can have substantially different execution costs. Poor join ordering, unnecessary scans, repeated subqueries, inappropriate intermediate

materialization, or missing filter pushdown may increase latency and temporary memory consumption even when the final answer is correct[39-41]. Database research has addressed performance optimization through learned query-planning and diagnosis systems. Bao uses learned guidance to select query-optimizer hints, while Panda applies language-model agents to database performance diagnosis. Recent studies have also shown that improvements in learned cost estimation do not necessarily translate directly into better final execution plans [42,14]. This distinction is important for Text-to-SQL. Query correctness and query efficiency are related but separate objectives. Optimizing execution cost before semantic correctness is established can preserve or even strengthen an incorrect query structure, whereas checking only correctness may produce SQL that is technically usable but unsuitable for repeated enterprise workloads. These limitations motivate an architecture in which schema understanding, SQL generation, semantic validation, and performance optimization are treated as connected but distinct stages. A system should determine the business meaning of the requested entities and measures before selecting tables and join relationships. Generated SQL should then be validated not only by successful execution but also by examining whether its joins, predicates, grouping structure, and returned attributes are consistent with the natural-language request. Performance tuning should begin only after the query has passed these semantic checks. Such separation provides clearer diagnostic signals: schema mismatches can be returned to the interpretation stage, incorrect SQL logic can be revised by a semantic repair component, and expensive but correct queries can be passed independently to a cost-optimization component. This structure also reduces the risk that one optimization objective interferes with another.

To address these challenges, this study proposes SQLAgent-Opt, a multi-agent framework that integrates schema interpretation, SQL drafting, execution validation, semantic checking, error-specific repair, and query-cost optimization within a unified workflow. The framework begins by interpreting the natural-language request together with database metadata and identifying the tables, attributes, relationships, and business constraints that are most relevant to the task. A SQL-generation component then constructs an initial query using this structured interpretation. The generated statement is executed to detect syntax, reference, and runtime failures, after which a separate semantic-checking component evaluates whether the join paths, filters, grouping levels, aggregation functions, and selected fields are consistent with the intended question. Queries that fail these checks are repaired according to the identified error type rather than being regenerated through an undifferentiated retry process. Once semantic correctness has been established, a dedicated optimization stage evaluates execution behavior and revises the query to reduce unnecessary computation and resource consumption while preserving its result semantics[43]. SQLAgent-Opt is evaluated on 1,540 natural-language querying tasks drawn from four enterprise data-warehouse domains. The experiments measure not only overall SQL accuracy but also join-path errors, aggregation errors, execution time, and temporary memory consumption. This design allows the study to examine whether explicit schema interpretation reduces structural

query errors, whether independent semantic validation can detect executable but incorrect SQL, whether error-specific repair improves reliability over direct regeneration, and whether cost optimization can improve execution efficiency without changing query meaning. The purpose of the proposed framework is therefore broader than increasing benchmark execution accuracy. It seeks to establish a more reliable workflow for natural-language access to enterprise data in which correctness is evaluated at the level of business semantics and efficiency is evaluated after semantic validity has been confirmed. By separating interpretation, generation, validation, repair, and optimization, SQLAgent-Opt aims to reduce the risk of plausible but incorrect analytical results while also limiting unnecessary database resource consumption. This capability is particularly important in enterprise reporting, financial analysis, clinical data exploration, retail operations, and logistics management, where a syntactically valid query can directly influence downstream reports and decisions. The study consequently provides both a methodological contribution to multi-agent Text-to-SQL research and a practical framework for developing natural-language database systems that are more accurate, interpretable, and suitable for deployment in complex enterprise data environments.

2. Materials and Methods

2.1. Dataset Sample and Study Scope

This study used 1,540 natural-language SQL tasks from finance, healthcare, retail, and logistics data warehouses. The datasets contained 326 relational tables, 4,870 columns, and 18.6 million records. Each task included a user question, schema metadata, and an executable reference SQL query. Tasks were included when the question had a clear analytical target and the reference query ran successfully on a read-only database copy. The task set covered joins, filters, aggregation, grouping, sorting, subqueries, and date conditions. Data-changing statements, including INSERT, UPDATE, and DELETE, were excluded. Direct personal identifiers were not included in the test data.

2.2. Experimental Design and Control Setting

SQLAgent-Opt was evaluated as the experimental system. It included five agents for schema interpretation, SQL drafting, execution validation, semantic checking, and cost optimization. The control system used one SQL-generation agent. Both systems used the same language model, prompt content, schema metadata, token limit, database engine, and hardware setting. Two ablation settings were also tested. The first removed semantic checking, and the second removed cost optimization. Tasks from the same warehouse were assigned to the same data split. This prevented similar schemas and business rules from appearing in both development and test sets.

2.3. Measurement Methods and Quality Control

Generated SQL queries were executed on read-only database copies. Exact-match accuracy, execution accuracy, incorrect join paths, aggregation errors, execution time, and temporary memory use were recorded. SQL strings were standardized before exact comparison. This process normalized aliases, keyword format, and nonmeaningful spaces. Result sets were sorted before comparison. Numerical outputs were rounded to six decimal places when needed. Two database reviewers checked the reference SQL queries, table relations, and expected outputs before the experiments. Tasks with unclear conditions or unstable reference outputs were removed. The system saved the generated SQL, execution plans, runtime logs, error messages, and agent records for each task.

2.4. Data Processing and Evaluation Formulae

Schema metadata were converted into structured descriptions of tables, columns, primary keys, foreign keys, data types, indexes, and business rules. Physical table and column names were retained during SQL execution. Execution accuracy was defined as the proportion of generated queries that produced the same result set as the reference query.

$$EX = \frac{1}{N} \sum_{i=1}^N I(\text{Result}(\hat{q}_i) = \text{Result}(q_i^{\text{ref}}))$$

In this formula, N is the number of test tasks, $\hat{q}_i$ is the generated SQL query, and q_i^{ref} is the reference SQL query. The function $I(\cdot)$ equals 1 when the two result sets match and 0 otherwise. Runtime reduction was calculated for tasks involving tables with more than one million records.

$$TR = \frac{1}{N_L} \sum_{i=1}^{N_L} \left(\frac{t_i^{\text{base}} - t_i^{\text{opt}}}{t_i^{\text{base}}} \right) \times 100\%$$

Here, N_L is the number of large-table tasks, t_i^{base} is the runtime before optimization, and t_i^{opt} is the runtime after optimization.

2.5. Reproducibility and Statistical Analysis

Each valid query was executed five times under the same database and hardware settings. The first run was removed to reduce cold-cache effects. The median value from the remaining four runs was used for runtime and memory analysis. All systems were tested on the same tasks. Exact-match and execution outcomes were compared with paired tests. Runtime and temporary memory differences were assessed with bootstrap confidence intervals. Statistical significance was set at $p < 0.05$. Database versions, schema files, index definitions, prompts, model settings, random seeds, generated SQL queries, and execution logs were retained for later review.

3. Results and Discussion

3.1. Overall SQL Accuracy

SQLAgent-Opt achieved 73.6% exact-match accuracy and 82.1% execution accuracy. The single-agent generator achieved 58.9% and 66.4%, respectively. Exact-match accuracy increased by 14.7 percentage points, and execution accuracy increased by 15.7 percentage points. Execution accuracy was higher than exact-match accuracy in both systems. Some generated SQL queries did not have the same text as the reference query but returned the same records. This difference is common in Text-to-SQL evaluation [1, 3]. Fig. 1 shows a related comparison of execution accuracy across query difficulty levels. Methods such as SQL-PaLM and DIN-SQL improve SQL drafting and self-correction [25,26]. SQLAgent-Opt also checked schema relations, query meaning, and query cost.

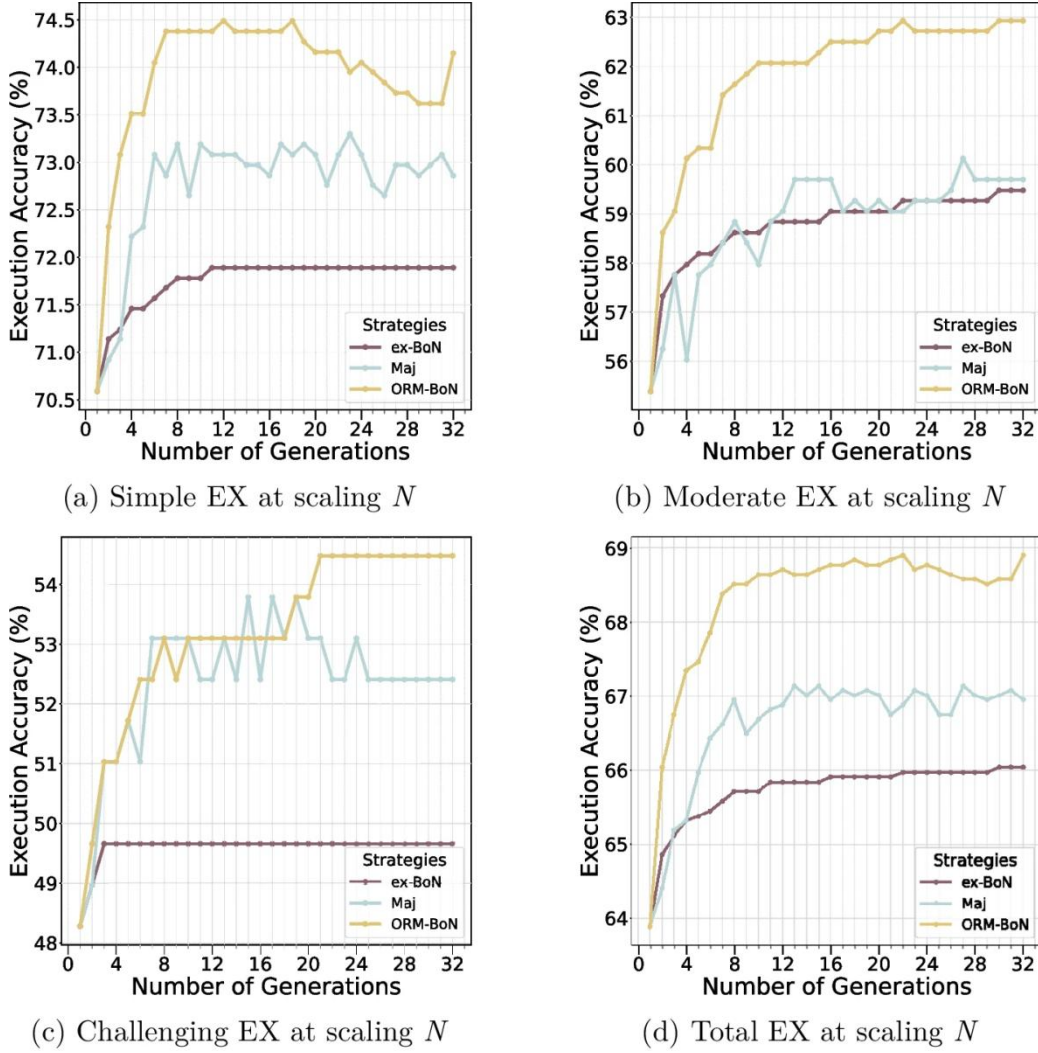


Fig. 1. Execution accuracy for simple, moderate, and difficult queries with different numbers of SQL candidates.

3.2. Join and Aggregation Errors

The incorrect join-path rate fell from 19.7% to 7.8%, a relative reduction of 60.4%. Aggregation errors decreased by 34.5%. Wrong joins and aggregation rules may still produce valid SQL statements. They can also return plausible results. Therefore, successful execution does not always mean that a query is correct. The schema agent checked table relations, primary keys, and foreign keys. The semantic agent checked selected columns, join conditions, grouping fields, and filters against the user question. Earlier agent-based methods also used query review and repair. However, their results cannot be directly compared with those reported here because the datasets and database settings were different [27,28].

3.3. Query Cost on Large Tables

For tasks containing tables with more than one million records, SQLAgent-Opt reduced average execution time by 41.2% and temporary memory use by 26.8%. The cost agent placed filters earlier, reordered joins, simplified subqueries, and removed unneeded aggregation steps. These changes were applied after the SQL statement passed semantic checking. Fig. 2 presents a related comparison of execution accuracy and query efficiency on the BIRD benchmark. Bao and Panda focused on plan selection and database performance diagnosis [29,30]. SQLAgent-Opt linked query optimization to checks of schema use and query meaning. Runtime results still depend on the database engine, indexes, hardware, and data size.

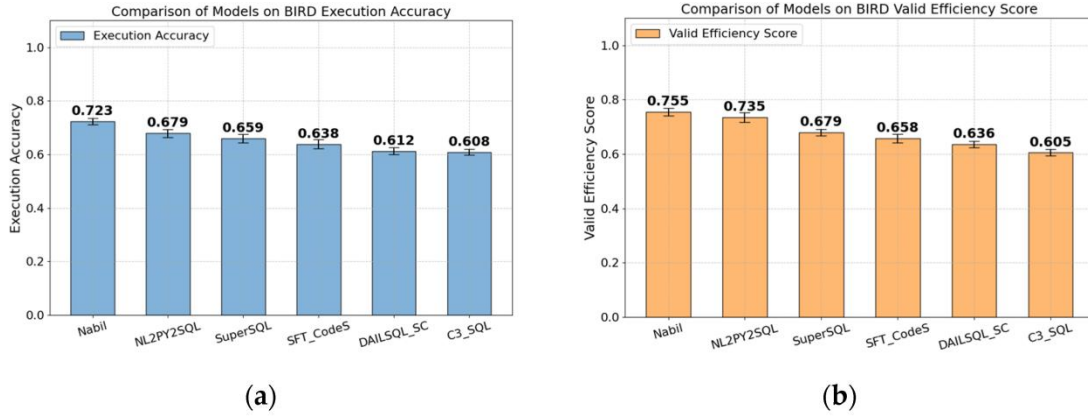


Fig. 2. Execution accuracy and valid efficiency scores of Text-to-SQL methods on the BIRD benchmark.

3.4. Practical Meaning and Study Limits

The results show that SQL drafting, query checking, and query optimization should be handled as separate steps. SQLAgent-Opt reduced both output errors and execution cost. However, the study used tasks with reference SQL queries and fixed database copies. It did not test schema changes, concurrent users, access-control rules, cache effects, or all SQL dialects. A query may also match the reference result while failing to answer an unclear business request. Enterprise datasets such as BEAVER and Spider 2.0 contain more complex schemas and workflows. Later

work should test the system on changing production-like databases and include expert checks of query meaning.

4. Conclusion

This study tested SQLAgent-Opt for natural-language SQL generation in data warehouses. In 1,540 tasks, exact-match accuracy increased from 58.9% to 73.6%, and execution accuracy increased from 66.4% to 82.1%. Incorrect join paths, aggregation errors, execution time, and temporary memory use also decreased. Unlike the single-agent control system, SQLAgent-Opt used separate steps to read schemas, write SQL, run queries, check query meaning, and reduce query cost. This structure reduced errors that may remain after a query runs successfully, such as wrong joins and grouping rules. The system may support reporting and data analysis in finance, healthcare, retail, and logistics. The study used fixed database copies and known reference queries. It did not assess changing schemas, simultaneous users, access controls, or all SQL dialects. Later studies should test the system on live enterprise databases and ask domain experts to check whether each query answers the intended business question.

References

1. Jiao, Y., Shi, T., Zhao, B., & Wang, A. (2026). Retrieval-Guided Structured Reasoning and Interpretable Representation Learning for Large-Scale Video–Language Models.
2. Baig, M. S., Sher, T., Rehman, A., & Sheikh, S. (2025). A systematic literature review of text-to-sql: Performance, challenges, and limitations. *ICCK Transactions on Advanced Computing and Systems*, 2(1), 1-24.
3. Liang, S., Hsu, K. H., & Liu, Z. (2026). Rolling Scheduling and Capacity Balancing for Server Assembly Under Engineering Change Disturbances.
4. Zheng, J., & Makar, M. (2022). Causally motivated multi-shortcut identification and removal. *Advances in Neural Information Processing Systems*, 35, 12800-12812.
5. Fan, Y., Miyazaki, T., Tang, Z., Wang, J., Huang, Y., & Omachi, S. (2025, November). Scene Text Reconstructor: A Contextual-Aware Masking Framework for Pre-training Text Detectors. In *International Conference on Neural Information Processing* (pp. 181-195). Singapore: Springer Nature Singapore.
6. Yuan, Y., Xu, T., Yin, J., & Huang, J. (2026). Modeling Loading Anomalies and Identifying Root Causes in Media Consumption Workflows on Large Social Media Platforms. Available at SSRN 7393318.
7. Wang, T., & Xia, Z. (2025). Stability of In-Context Learning: A Spectral Coverage Perspective. *arXiv preprint arXiv:2509.20677*.
8. Zhang, G., Xu, Y., Zhang, M., Wang, S., & Lin, N. (2021). The brain network in support of social semantic accumulation. *Social cognitive and affective neuroscience*, 16(4), 393-405.
9. Bao, Y., Qiu, Y., & Wang, H. (2026). FLARE: A Real-Time Framework for Detecting Malicious Account Activities at Internet Scale. Available at SSRN 7185659.

10. Mini Han Wang , " AI-Powered Innovations in Ophthalmic Diagnosis and Treatment ", Bentham Science Publishers (2025). <https://doi.org/10.2174/97988988122871250101>.
11. Wang, S., Feng, Y., & Fang, X. (2026, May). A Large Language Model-Enabled Multi-Agent Collaboration Method for Complex Task Solving. In 2026 6th International Symposium on Computer Technology and Information Science (ISCTIS) (pp. 253-256). IEEE.
12. Yang, Q., & Du, Y. (2026). Predicting Rollback Risks and Controlling Gradual Rollout Traffic for Online Ranking Feature Deployments.
13. Zhang, Z. (2026). A Study on the Identification of Manipulative Design in Subscription and Payment Interfaces of Digital Consumer Platforms and Its Behavioral Effects. Available at SSRN 6734760.
14. Liu, W., Zhao, R., Sha, Z., Cui, Q., & Zhang, Y. (2026). Mapping the City Through the Lens of Language Models. arXiv preprint arXiv:2608.02971..
15. Zhang, Z. (2026). A Study on the Impact of Cost Presentation on Decision-making Bias and Cancellation Behavior in Online Subscriptions. Available at SSRN 7349839.
16. Cui, J., Zhang, G., Chen, Z., & Yu, N. (2022). Multi-homed abnormal behavior detection algorithm based on fuzzy particle swarm cluster in user and entity behavior analytics. *Scientific Reports*, 12(1), 22349.
17. Qi, C., & Qiao, X. (2026). Efficient Data Sampling and Feature Selection Algorithms for Scalable Machine Learning Pipelines.
18. Zhao, R., Liu, W., Sha, Z., Su, N., Zhang, Y., & Long, Y. (2026). Culturally uneven urban perception in large language models [Preprint]. arXiv. <https://doi.org/10.48550/arXiv.2604.20048v3>.
19. Xiong, W., Zeng, Y., & Guo, Y. (2026). A Study on the Characterization of Building MEP System States and the Learning of Behavioral Patterns Driven by Large-Scale Operational Data. Available at SSRN 7121883.
20. Hong, Z., Liang, S., Xu, T., & Chen, H. (2026). A Study on Failover Verification and Recovery Objective Prediction for Cross-Region Cloud Services.
21. Han, Z., Chen, W., Han, Y., Mao, R., & Qin, J. (2026). Fast Diversified Top-k Rule Discovery via User-Guided Embeddings. *IEEE Transactions on Knowledge and Data Engineering*.
22. Huang, J., Xu, T., Yang, J., & Yin, J. (2026). A Graph Neural Network Approach for Financial Data Validation and Risk Identification in Large-Scale Intercompany Transactions. Available at SSRN 7175058.
23. Zhao, Z., & Welsch, R. E. (2024). Hierarchical reinforced trader (hrt): A bi-level approach for optimizing stock selection and execution. arXiv preprint arXiv:2410.14927.
24. Quan, Q., Gao, Y., & Wang, Q. (2025). The impact of teacher emotional support on students' engagement in AI-mediated English learning environments: The mediating role of resilience and self-efficacy. *Acta Psychologica*, 260, 105766.

25. Yang, Z., Alexandrova, A. N., & Sautet, P. (2026). Modeling CO₂ Hydrogenation to Methanol on an Ensemble of Inverse ZrO₂ on Cu Catalytic Sites: Mechanism, Reactivity, and Deactivation. *Angewandte Chemie*, e5448247.
26. Jiang, C. (2026). Information doesn't work? Testing experiential drivers of VR application in heritage tourism by affective-cognitive pathways. *Journal of Hospitality and Tourism Technology*, 1-28.
27. Gu, X. (2026). Identifying Causal Effects and Analyzing Heterogeneity of User Growth Interventions on Digital Platforms: Evidence from Large-Scale Behavioral Data. Available at SSRN 6809181.
28. Wang, Z., Yu, J., Liu, H., Zheng, Z., Jin, Y., Li, S., ... & Zhang, K. (2025, July). Generative Music Models' Alignment with Professional and Amateur Users' Expectations. In *Findings of the Association for Computational Linguistics: ACL 2025* (pp. 6909-6920).
29. Zhao, J., Fan, J., & Li, L. (2026). A Study on an Explainable Causal-Enhanced LLM Agent for Predicting the Forming Quality of Automotive Component Materials.
30. You, S. (2026). Verifiable Audit Mechanisms in AI Compliance Automation: Scalability. Available at SSRN 6547458.
31. Xu, Y., Sun, Y., Zhai, B., Li, M., Liang, W., Li, Y., & Du, S. (2025, April). Zero-shot video moment retrieval via off-the-shelf multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 9, pp. 8978-8986).
32. Yuan, Y., Yin, J., Huang, J., & Xu, T. (2026). Weakly Supervised Anomaly Detection and Privacy Risk Scoring in Community Message Threads.
33. Wu, Y., & Su, D. (2026). An Evaluation of Parental Question Characteristics and the Quality of AI Recommendations in Family Counseling for Children with Autism.
34. Su, J., Cai, C., Zhu, F., He, C., Xu, X., Guan, D., & Si, C. (2024, September). Momentum auxiliary network for supervised local learning. In *European Conference on Computer Vision* (pp. 276-292). Cham: Springer Nature Switzerland.
35. Zheng, X., Chen, X., Gong, S., Griffin, X., & Slabaugh, G. (2025, September). Xfmamba: Cross-fusion mamba for multi-view medical image classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 672-682). Cham: Springer Nature Switzerland.
36. Su, J., Zhu, F., Shi, H., Han, T., Qiu, Y., Luo, J., ... & Gao, J. (2025). MAN++: Scaling Momentum Auxiliary Network for Supervised Local Learning in Vision Tasks. *arXiv preprint arXiv:2507.16279*.
37. Li, P., Zhang, H., Li, W., Huang, D., Guo, Z., Chen, J., ... & Koshizuka, N. (2025). GeoAvatar: A big mobile phone positioning data-driven method for individualized pseudo personal mobility data generation. *Computers, Environment and Urban Systems*, 119, 102252.
38. Lyu, Y., & Ding, R. (2025). Discovering Self-Regulated Learning Patterns in Chatbot-Powered Education Environment. *arXiv preprint arXiv:2510.01275*.

39. Xi, X., Chen, P., Qiu, Y., Luo, R., Tong, P., & Liang, J. Video-BCI: Bayesian Cognitive Integration of Self-Prior Hypotheses for Video Understanding. In Forty-third International Conference on Machine Learning.
40. Liang, J., Xiao, Y., Yang, H., Yu, Z., Liu, J. N., & Yang, K. (2026, April). C-HyPOD: Causal Hyperbolic Representation Learning with Prototype Orthogonal Disentanglement for Graph Out-of-Distribution Recommendation. In Proceedings of the ACM Web Conference 2026 (pp. 6541-6550).
41. Xu, Z., Zhang, X., Li, R., Tang, Z., Huang, Q., & Zhang, J. (2024). Fakeshield: Explainable image forgery detection and localization via multi-modal large language models. arXiv preprint arXiv:2410.02761.
42. Jia, Z., You, K., He, W., Tian, Y., Feng, Y., Wang, Y., ... & Zhang, Z. (2023). Event-based semantic segmentation with posterior attention. IEEE Transactions on Image Processing, 32, 1829-1842.
43. C. Huang, X. Chen, G. Chen, P. Xiao, G. Ye Li and W. Huang, "Deep Reinforcement Learning-Based Resource Allocation for Hybrid Bit and Generative Semantic Communications in Space-Air-Ground Integrated Networks," in IEEE Journal on Selected Areas in Communications, vol. 43, no. 12, pp. 3942-3954, Dec. 2025, doi: 10.1109/JSAC.2025.3623157.