From Compression to Execution: What Helps Large Language Models Digest Urban Graphs for Spatial QA?

Rachid Oucheikh*, Ali Mansourian

Department of Earth and Environmental Sciences, Lund University, Sölvegatan 12, SE-223 62, Lund, Sweden - {rachid.oucheikh, ali.mansourian}@mgeo.lu.se

* Corresponding author: rachid.oucheikh@mgeo.lu.se

Keywords: Large Language Models, Smart cities, GeoAI, Urban graph reasoning, Spatial question answering, Language-guided urban analytics.

Abstract

Urban graphs are central to smart-city analytics, but they remain difficult for language-guided models, specifically LLMs, to use effectively in spatial question answering. This paper studies how urban graph structure can be exposed to such models through different graph-language interaction designs. We construct a controlled multi-task urban graph QA benchmark over three cities that covers adjacency count, adjacency binary, reachability, shortest path, and centrality, and compare seven mechanisms spanning compression, memory, retrieval, execution, and structural reasoning, alongside text-only and GNN-only baselines. Within this benchmark, methods using more explicit structural evidence generally outperform compression and memory-based approaches: Path Tokens reaches 78.1% average accuracy, Graph Tool 87.4%, and Counterfactual 71.7%, while centrality remains the most difficult task. These results should be interpreted as a comparison of graph-language interaction designs rather than a fully input-matched ablation, but they suggest that exposing task-relevant structural information is an important factor for urban spatial QA.

1. Introduction

Urban road and street networks are central to smart-city analytics because connectivity, reachability, route length, and structural importance are graph properties rather than purely textual facts (Barthélemy, 2011; Zheng et al., 2014). LLMs offer an attractive natural-language interface to these analyses, but an urban graph cannot simply be placed in a prompt without choices about selection, encoding, and execution. Naive serialization can exceed the context window, while aggressive compression can erase neighborhoods, paths, and bottlenecks. The scientific problem is therefore to determine which graph-language interface preserves the structural object required by a spatial question. Such language-based access is particularly relevant to human-centered urban analytics, where users may express spatial questions directly rather than through formal GIS or graph operators.

1.1 Related Work and Positioning

Work on urban networks is already extensive, but it has developed mainly around structural analysis, prediction, and representation learning, rather than language-guided reasoning over city graphs. Foundational studies in spatial and urban network science established that street systems are not generic graphs: their topology is constrained by geometry, scale, and urban morphology, which directly shape accessibility, centrality, and routing behavior (Barthélemy, 2011). In parallel, urban computing positioned mobility, traffic, and geospatial data as core substrates for city intelligence (Zheng et al., 2014), while more recent efforts have expanded this line toward graph-based traffic forecasting (Jiang and Luo, 2022), urban knowledge graphs for data integration (Liu et al., 2023), road-network representation learning (Wu et al., 2020), and large cross-city urban network datasets such as Urbanity (Yap and Biljecki, 2023). Taken together, these studies show that urban graphs are already central to smart-city analytics, but they are usually used for forecasting, embedding, optimization, and structural characterization—not for answering natural-language questions about connectivity, reachability, shortest paths, or structural importance over city-centered graph cases.

Generic graph-reasoning studies have established that LLM performance depends strongly on graph encoding, task type, and graph structure. NLGraph evaluates natural-language graph problems and reports increasing brittleness as operations become more complex (Wang et al., 2023). Talk Like a Graph systematically compares textual graph encodings (Fatemi et al., 2024), while GraphText converts graph-syntax trees into text for LLM-based graph learning (Zhao et al., 2023). These studies motivate careful representation design, but they primarily use generic or synthetic graph settings and do not compare a broad spectrum of learned, retrieved, executable, and perturbation-based interfaces on held-out urban networks.

Hybrid graph-LLM research provides a second line of evidence. G-Retriever selects compact subgraphs for textual graph QA (He et al., 2024), and GNN-RAG retrieves answer candidates and verbalized paths for knowledge-graph QA (Mavromatis and Karypis, 2024). More broadly, recent GIS–LLM systems have shown how natural-language interfaces and agent-based architectures can support geospatial analysis and execution, but their focus is general GIS workflow generation rather than controlled reasoning over urban graph structure (Mansourian and Oucheikh, 2024, 2026). In geospatial research, MapQA emphasizes geometry, topology, and multi-hop map relations (Li et al., 2025a), GeoGraphRAG integrates graph retrieval into automated geospatial modelling (Liang et al., 2025), and knowledge-graph-assisted geospatial QA has focused on data discovery and retrieval (Li et al., 2025b). Urban-computing surveys further identify robust spatial reasoning as an open problem (Li et al., 2025c). The present work is complementary: it isolates the interface question on city road graphs and asks how compression, memory, retrieval, execution, and structural perturbation compare across local, path-based, and global spatial QA tasks.

The novelty is deliberately scoped. We do not claim a new foundation model, a new graph algorithm, or universal superiority of one architecture. Instead, the paper contributes a controlled systems-level comparison in which the same city cases, questions, splits, graph encoder, language backbone, and training protocol are retained while the mode of graph exposure changes. This makes explicit what is newly enabled: an empirical map from task type to useful interface design, including the distinction between latent learning and access to exact graph procedures.

1.2 Research Questions and Contributions

The study addresses three research questions: RQ1, which graph-language interface families generalize best to a held-out city graph; RQ2, whether task-aligned structural evidence produces larger gains than latent graph summaries, particularly for reachability and shortest path; and RQ3, which tasks remain difficult even with stronger interfaces and what this reveals about interface-task alignment. The contributions are: (1) a multi-city benchmark with five urban graph QA tasks and leave-one-city-out evaluation; (2) a common taxonomy and implementation of seven interfaces spanning latent compression to executable and counterfactual evidence; and (3) a task-level analysis that separates reported performance from claims about what the LLM itself has learned.

2. Methodology

2.1 Problem Formulation and Urban Graph QA Setting

For each city c, the benchmark provides a graph $G_c = (V_c, E_c, X_c)$, where V_c denotes nodes, E_c denotes links, and X_c contains the 16-dimensional node features used by the shared graph encoder. A QA instance consists of a natural-language question q_i, a task identifier t_i, and a target class y_i. A mechanism m constructs a graph-conditioned representation $z_i^m = I_m(G_c, q_i)$. The predictor then estimates y_i from the question representation and z_i^m. The comparison therefore concerns the interface I_m through which graph structure becomes available to the language-facing predictor.

The three city cases differ in scale and connectivity: Sofia contains 1,096 nodes and 2,403 edges, Vienna 2,089 nodes and 7,325 edges, and Berlin 3,600 nodes and 9,140 edges. The city labels anchor the benchmark in urban-network structure while providing a controlled cross-city shift. The study does not claim that three graphs represent the full diversity of real road systems; the city-holdout protocol is used as a stricter test than a random within-city split.

Five tasks cover distinct structural scopes. Adjacency Count asks for the number of neighbors of a queried node. Adjacency Binary asks whether two queried nodes share an edge. Reachability tests bounded connectivity under the benchmark criterion. Shortest Path is a balanced distance-class task for a queried node pair. Centrality is a comparative structural-importance task under the benchmark target. Node identifiers are anonymized, binary and count variants are separated, shortest-path classes are balanced, and each city-task contains approximately 300–400 QA pairs. Together, the tasks move from local statistics to pairwise relations, path reasoning, and global comparison.

The five tasks also differ in the spatial extent of the graph information required to answer them. Adjacency count and adjacency binary depend primarily on the immediate neighborhood of one or two referenced nodes. Reachability extends this requirement to multi-step connectivity, while shortest-path questions require preservation of an ordered source–destination structure. Centrality is different again because the importance of a candidate node cannot generally be determined from its immediate neighborhood alone, but depends on its position relative to the wider network. The task suite therefore provides a progression from relatively local to increasingly non-local forms of graph reasoning, making it possible to examine whether different interfaces preserve different types of structural information.

Task	Scope	Structural object	Purpose
Adjacency Count	One node	Immediate neighborhood	Local degree / neighborhood size
Adjacency Binary	Node pair	Direct edge	Pairwise topology
Reachability	Node pair	Connected path under bound	Multi-hop connectivity
Shortest Path	Node pair	Distance / candidate path	Route-level reasoning
Centrality	Candidate nodes	Whole-graph comparison	Global structural importance

Table 1. Structural scope and motivation of the five QA tasks.

The suite was selected to prevent a single mechanism from being judged only on the type of structure it was designed to expose. This heterogeneity also makes task-level performance diagnostically important: two interfaces with similar overall accuracy may differ substantially in whether they preserve local neighborhoods, pairwise relations, multi-hop paths, or whole-network comparisons. Adjacency tasks test whether local information survives the interface; reachability and shortest path test whether multi-hop structure is preserved; and centrality tests whether the interface can support comparative reasoning over a substantially larger portion of the graph. The balanced shortest-path target and anonymized identifiers reduce simple frequency or lexical shortcuts, although the Text-Only results show that some residual task regularities may remain. The benchmark is therefore intended to be diagnostic: strong performance should primarily reflect the ability to exploit urban graph structure rather than recover answers from easily exploitable surface patterns. Separating adjacency count from binary adjacency, balancing the shortest-path target, anonymizing node identifiers, and holding out complete city graphs are all intended to strengthen this distinction.

2.2 Shared Architecture and Design Rationale

Figure 1 summarizes the experimental framework and the five graph–language interface families. Given an urban graph and spatial QA query, Compression exposes compact latent summaries, Memory exposes learned memory slots, Retrieval provides selected subgraphs or candidate paths, Execution supplies graph-derived procedural features, and Structural reasoning uses changes induced by graph perturbations. These mechanism-specific representations are then processed through the shared graph–language backbone to predict the answer. The figure therefore illustrates the progression from latent graph abstraction toward increasingly explicit task-relevant structural evidence.

Urban graphs are encoded by a two-layer graph attention network (GAT) operating on the 16-dimensional node features. A GAT is appropriate for sparse and irregular road graphs because it aggregates variable-size neighborhoods without imposing a raster structure. Two layers provide a controlled

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XII-4-W2-2026-155-2026

local receptive field and limit computational cost and over-smoothing; the experiments then test whether interface-specific evidence is needed when the question requires longer paths or global structure. Natural-language questions are tokenized with the native tokenizer of a frozen Llama 3.3 70B model. Freezing the language backbone reduces the risk that differences are caused mainly by unequal language-model fine-tuning rather than by the graph interface.

The graph and question representations are constructed separately and coupled by a mechanism-specific interface. This design holds the main encoders and optimization pipeline constant while varying the amount and type of structural evidence. It is not a fully matched ablation because some mechanisms receive richer, task-aligned inputs. That asymmetry is intentional: the practical research question is which interface design is useful, not whether all interfaces perform equally when deprived of the information they were designed to expose.

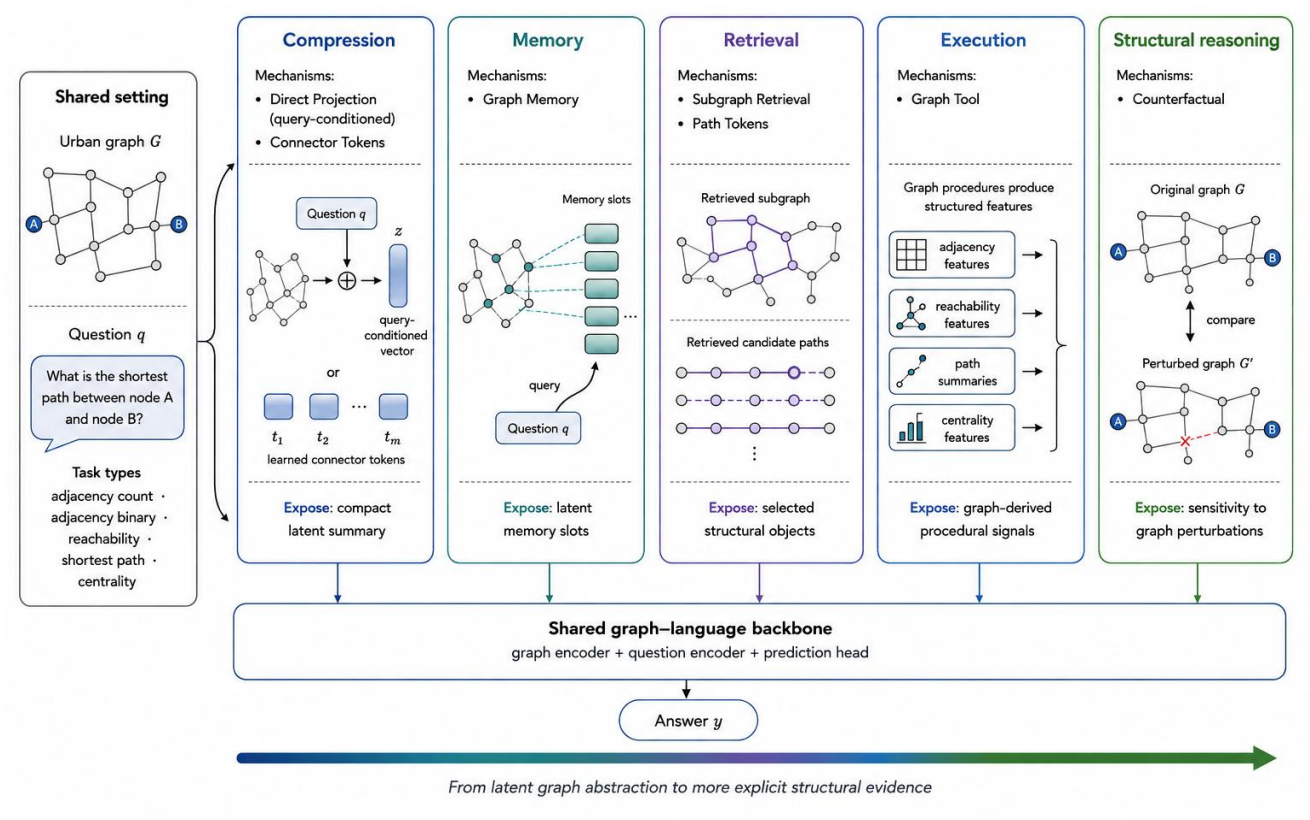

Figure 1. Graph–language interaction mechanisms compared in the urban spatial QA framework.

2.3 Mechanism Families

Compression. Direct Projection maps a query-conditioned graph representation to one token-like vector and combines it with the question through learned gating. Connector Tokens projects the graph to multiple learned tokens and uses cross-attention. Both test whether compact latent summaries can retain the structural information required by the QA task.

Memory. Graph Memory aggregates node embeddings into latent memory slots that are queried repeatedly by the question representation. It increases capacity relative to one or a few compressed tokens, but structural organization must still emerge internally without explicit paths or graph operations.

Retrieval. Subgraph Retrieval scores nodes for query relevance, builds a compact selected subgraph, and projects it to graph tokens. Path Tokens exposes candidate paths as path-level objects with padded node matrices and masks. Retrieval is selective rather than global: only the structural object judged relevant to the query is presented to the predictor.

Execution. Graph Tool supplies fixed-size signals produced by exact graph procedures, including adjacency, reachability, path-related, and centrality evidence. It should be interpreted as a tool-augmented system or execution-oriented upper bound, not as evidence that the LLM has internally learned these graph algorithms. Its role is to quantify the value of reliable procedural access through a language interface.

Structural reasoning. Counterfactual reasoning compares graph properties before and after controlled perturbations. Rather than describing only what the original graph contains, it exposes how answer-relevant properties change when selected elements of the network are modified or removed. The resulting differences provide evidence of structural sensitivity and are relevant to bottlenecks, rerouting, and important nodes.

Family	Exposed object	Expected strength	Main limitation
Compression	One / several latent tokens	Compact integration	Paths and global structure may be lost
Memory	Learned memory slots	Higher latent capacity	No explicit structural organization
Retrieval	Selected subgraph or paths	Task-aligned evidence	Depends on retrieval quality and object choice

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XII-4-W2-2026-155-2026

Family	Exposed object	Expected strength	Main limitation
Execution	Exact procedure-derived signals	Reliability and auditability	Not evidence of internally learned algorithms
Structural	Perturbation differences	Bottleneck / sensitivity reasoning	Depends on perturbation design

Table 2. Information exposed by each mechanism family and the principal interpretation caveat.

The mechanisms were chosen to represent distinct solutions to the same interface bottleneck rather than minor variants of a single architecture. Direct Projection and Connector Tokens test compression severity; Graph Memory tests whether additional latent capacity alone is enough; Subgraph Retrieval and Path Tokens test selective exposure at different structural granularities; Graph Tool tests executable access; and Counterfactual tests sensitivity to graph changes. This coverage supports a design-space comparison, but it also means that performance differences combine architecture, evidence selection, and evidence explicitness.

2.4 Baselines and Evaluation Metrics

The Text-Only baseline receives the natural-language question without graph evidence and therefore measures label priors, wording regularities, and any non-structural cues that survive anonymization. The GNN-Only baseline receives graph representations without the frozen language backbone and measures how far a graph encoder and classifier can solve the benchmark without a language-mediated interface. These baselines are complementary: Text-Only identifies tasks that may be partially predictable without topology, whereas GNN-Only identifies tasks for which graph encoding alone is sufficient but may be unstable across cities.

Accuracy is the proportion of correctly classified QA instances. Weighted F1 computes the class-wise F1 scores and weights them by class support, which is useful when the effective class distribution is not perfectly uniform after city holdout. We report both metrics because a high accuracy can conceal weak minority-class behavior, while weighted F1 remains comparable to the overall class distribution. The limited number of independent city folds does not support strong asymptotic significance claims; the mean and standard deviation are therefore interpreted descriptively, with emphasis on large and consistent gaps rather than small rank differences.

2.5 Experimental Protocol and Interpretation

All mechanisms use the same leave-one-city-out protocol. In each fold, one of Sofia, Vienna, or Berlin is held out for testing and the other cities are used for training and validation; results are averaged over three random seeds. Optimization uses AdamW, cosine annealing, cross-entropy loss, gradient clipping, and checkpoint selection by validation accuracy. Task-level accuracy and weighted F1 are reported as mean ± standard deviation over city-holdout folds and random seeds. Overall values are obtained by averaging across the five tasks; the accompanying variability summarizes the corresponding task-level standard deviations.

For mechanisms that require structured evidence, the interface is conditioned on the graph and the query. Depending on the mechanism, the relevant structural object may consist of a query-relevant subgraph, candidate paths, exact graph-procedure outputs, or properties obtained from controlled graph perturbations. The deterministic compiler represents these objects as fixed-size tool features, counterfactual features, or padded path-node matrices where required. The intended leakage-control rule is that the compiler operates only on the graph and question; ground-truth labels and test-set aggregate statistics are reserved for supervision and evaluation. Evidence is then reused across the common splits to avoid mechanism-specific sampling differences. Because the interfaces differ in evidence explicitness, the study supports conclusions about complete graph-language interaction designs, not causal claims that one neural component alone is superior.

The evaluation unit is a complete QA instance rather than a graph-level summary. For each held-out city, every mechanism is evaluated on the same task instances, which keeps the comparison paired at the data level even when the interface inputs differ. Checkpoint selection is performed only on validation accuracy from the training cities. The held-out city is not used for model selection. This protocol is intended to test transfer across network size and morphology, not merely memorization of node identities or repeated within-city patterns.

Reproducibility requires more than the high-level architecture. A public release should include the graph-construction procedure, the meaning and normalization of all 16 node features, question templates, class-generation rules, the exact reachability bound and centrality target, evidence-compiler code, optimizer hyperparameters, and the three random seeds. These items are especially important because an apparently small change in candidate-path generation or tool features can materially change the amount of target-relevant evidence available to a mechanism.

2.6 End-to-End Inference Workflow and Evidence Boundary

At inference time, a QA instance first identifies the city graph, the natural-language question, and the task family. The question encoder produces a language representation, while the GAT produces node-level graph representations. The selected interface then determines what crosses the graph-language boundary. Compression methods pass pooled latent vectors; memory passes learned slots; retrieval passes a selected subgraph or path object; execution passes deterministic procedure outputs; and counterfactual reasoning passes differences between original and perturbed graph states. A common fusion and classification stage maps these inputs to the benchmark answer class.

This workflow separates three sources of capability that are often conflated in graph-LLM studies. The first is language understanding: identifying which entities, relation, and operation the question requests. The second is structural access: obtaining the neighborhood, path, metric, or perturbation needed to answer. The third is answer mapping: converting the combined representation into the required class. A strong score can arise from any combination of these stages. For example, Graph Tool can be strong because structural access is exact even if the language-to-class mapping remains learned; a compression mechanism can fail because the required path was lost before the classifier received it, even when the question was interpreted correctly.

The evidence boundary also determines auditability. A latent token can be inspected only indirectly through attribution or probing. A retrieved subgraph or path can be displayed to the user and checked against the network. A tool result can be accompanied by the operation, parameters, and supporting nodes or edges. Counterfactual evidence can report which perturbation changed the answer and by how much. For operational city systems, these differences matter because a

correct answer without inspectable support may still be unsuitable for route guidance, accessibility assessment, or emergency decisions. The present benchmark evaluates answer quality; future work should evaluate evidence faithfulness and user verification as first-class outcomes.

3. Results

3.1 Overall Performance

Table 3 reports the aggregate results. The compression and memory mechanisms remain close to the Text-Only baseline: their average accuracies range from 47.4% to 49.4%, compared with 48.1% for Text-Only. GNN-Only reaches 53.9% but has substantially larger variation. In contrast, Counterfactual reaches 71.7%, Path Tokens 78.1%, and Graph Tool 87.4%. Relative to Text-Only, the gains are 23.6, 30.0, and 39.3 percentage points, respectively. Weighted F1 follows the same ordering.

Mechanism	Family	Avg. Acc.	Avg. F1
Direct Projection	Compression	49.4 ± 1.7	45.1 ± 0.9
Connector Tokens	Compression	48.2 ± 2.4	36.4 ± 1.8
Graph Memory	Memory	47.4 ± 1.0	34.3 ± 0.8
Subgraph Retrieval	Retrieval	47.8 ± 2.8	35.6 ± 2.1
Path Tokens	Retrieval	78.1 ± 2.0	74.2 ± 1.7
Graph Tool	Execution	87.4 ± 1.1	83.1 ± 1.3
Counterfactual	Structural	71.7 ± 1.8	68.4 ± 1.5
Text-Only	Baseline	48.1 ± 2.2	44.4 ± 1.5
GNN-Only	Baseline	53.9 ± 8.3	45.9 ± 3.7

Table 3. Overall accuracy and weighted F1 (%). Values are averages across the five tasks; ± denotes the mean of the corresponding task-level standard deviations across city-holdout folds and random seeds.

3.2 Task-Level Results

The task-level results in Table 4 show where the hierarchy arises. Adjacency Count is relatively accessible to several graph-aware mechanisms, including the GNN-Only baseline. The strongest separation appears on Adjacency Binary, Reachability, and Shortest Path. Path Tokens reaches 92.6% on adjacency binary, 91.7% on reachability, and 96.0% on shortest path; Graph Tool reaches 93.1%, 93.5%, and 91.7% on the same tasks. Counterfactual is also strong on reachability and shortest path but does not improve adjacency binary.

Centrality is qualitatively harder. Most latent and retrieval interfaces remain close to the nominal four-class chance level, including Path Tokens at 24.7%. Graph Tool reaches 66.7% and Counterfactual 56.1%, indicating that global comparative reasoning benefits from procedure-derived or perturbation-based evidence. No single interface dominates every task: Path Tokens is best for shortest path, whereas Graph Tool is best for the other four tasks.

Mechanism	GNN-Only	Text-Only	Counterfactual	Graph Tool	Path Tokens	Subgraph Retrieval	Graph Memory	Connector Tokens	Direct Projection
Adjacency count	86.2±2.1	70.6±3.5	78.1±3.4	92.3±1.1	85.7±2.7	83.4±4.9	85.9±2.5	85.3±1.4	78.2±3.5
Adjacency binary	60.3±13.2	52.2±2.1	50.1±0.2	93.1±0.1	92.6±5.0	50.0±0.0	50.0±0.0	50.0±0.0	51.6±1.7
Reachability	53.5±6.7	66.8±1.5	90.1±1.0	93.5±0.6	91.7±1.0	55.0±7.5	50.1±0.7	55.5±8.1	67.1±1.2
Shortest path	31.2±5.7	26.2±1.7	84.2±0.9	91.7±0.6	96.0±0.1	25.0±0.0	25.0±0.0	24.8±0.7	25.4±0.8
Centrality	38.5±13.8	24.6±2.0	56.1±3.8	66.7±3.2	24.7±1.4	25.7±1.6	25.8±1.7	25.8±1.7	25.5±1.4

Table 4. Task-level accuracy (%)

3.3 Interface–Task Alignment and Practical Implications

The descriptive result is clear: in this benchmark, generic latent compression is insufficient for the harder relational tasks, while interfaces that preserve the task-relevant structural object perform substantially better. The interpretation must

nevertheless distinguish access from learning. Graph Tool receives outputs from exact graph procedures, so its 87.4% average accuracy demonstrates the value of combining language with executable graph access; it does not demonstrate that the frozen LLM independently discovered adjacency, routing, or centrality algorithms. Path Tokens provides a narrower but still task-specific advantage by exposing candidate paths, which explains its particularly strong shortest-path result.

Figure 2 makes the contrast between average benefit and cross-task consistency more explicit. Graph Tool provides both the largest mean improvement over Text-Only (+39.4 percentage points) and a positive gain even on its least-improved task (+21.7 points). Path Tokens also achieves a large mean gain (+30.1 points), but its minimum gain is only +0.1 points because it offers essentially no improvement on centrality. Counterfactual shows a substantial mean benefit (+23.6 points) but remains slightly below Text-Only on adjacency binary. In contrast, the compression, memory, and Subgraph Retrieval interfaces have mean gains close to zero and negative minimum gains. Thus, methods above the horizontal zero line in Figure 2 improve on Text-Only across every task, whereas methods below it exhibit at least one task-specific degradation.

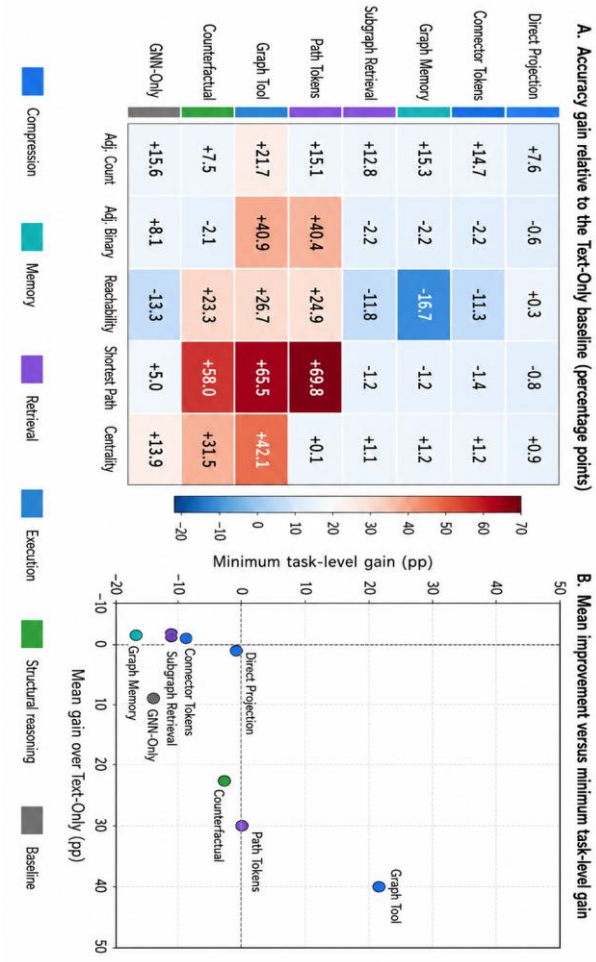

Figure 2. Accuracy gains of graph-aware interfaces relative to the Text-Only baseline. (A) Task-specific gains in percentage points. (B) Mean gain versus minimum task-level gain across the five tasks.

The mechanism ranking changes with the structural scope of the question. For adjacency count, even the GNN-Only baseline is strong, which is consistent with local neighborhood aggregation. On adjacency binary and shortest path, the weak latent methods collapse toward their nominal chance levels, while Path Tokens and Graph Tool remain strong. On reachability, Counterfactual also performs well, suggesting that perturbation-derived connectivity signals can preserve multi-hop structure. On centrality, path exposure alone is insufficient: a path is a local or pair-specific object, whereas comparative centrality depends on the relation between candidate nodes and the wider network.

This pattern suggests that 'more graph information' is not a precise design rule. The useful object depends on the query. A compact subgraph may contain many relevant nodes but still fail to preserve the particular path or comparison needed by the classifier. Conversely, a small set of task-aligned path tokens can outperform a higher-capacity memory because its inductive bias matches the question. The results therefore support interface selection by structural scope: neighborhoods for local counts, explicit edges for adjacency, paths or execution for routing, and global procedures or perturbations for centrality-like comparisons.

For practical urban QA systems, exact topological questions should preferentially call validated graph procedures and return the supporting edge, path, or metric as auditable evidence. Path retrieval is attractive when users need a route object that can be inspected or visualized. Latent interfaces may still be useful as compact learned summaries for ranking, approximate search, or soft semantic matching, but the present results do not support them as the sole source of truth for exact path and global-structure questions. A hybrid system can therefore use the LLM for intent interpretation and explanation while delegating deterministic graph operations to a controlled tool layer.

3.4 Limitations and Perspectives

Several limitations constrain the claims. First, the benchmark contains only three city graphs and controlled classification tasks. A city-holdout split is stronger than a random node split, but three networks cannot represent the full range of grid, radial, polycentric, informal, sparse, or highly constrained urban morphologies. The reported averages should therefore be read as evidence within the benchmark, not as population-level estimates for all cities.

Second, the mechanisms do not receive equally informative inputs. Graph Tool is deliberately supplied with exact procedure-derived signals and Path Tokens receives candidate paths, whereas compression and memory must infer useful structure from latent summaries. The comparison is consequently fair at the level of complete systems evaluated on common questions, but not at the level of equal information content. A future matched study could control token budget, evidence entropy, parameter count, and computational cost, or progressively reveal the same evidence through different encoders.

Third, the task generator and feature semantics are central to validity. The precise reachability bound, shortest-path bins, centrality definition, negative sampling, candidate construction, and 16-dimensional feature vector must be documented. Text-Only performance above nominal chance on some tasks suggests that wording or class priors may remain. Template diversification, adversarial paraphrases, permutation tests, and label-balanced city-specific evaluation would help quantify such shortcuts.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XII-4-W2-2026-155-2026

Fourth, the present paper reports predictive quality but not efficiency, calibration, or explanation faithfulness. Tool calls may improve correctness while adding latency; path retrieval may reduce context size but require candidate generation; and counterfactual features may be computationally expensive on large graphs. Future evaluation should therefore report token count, wall-clock time, memory use, tool-call success, calibration error, and whether the returned evidence actually supports the predicted answer.

Finally, operational urban networks are directed, weighted, semantic, and time dependent. Future work should incorporate road class, one-way restrictions, turn costs, travel modes, accessibility constraints, public-transport links, congestion, closures, and hazard exposure. Such extensions would support questions about service-specific accessibility and emergency routing without reducing every affected edge to a binary block. They would also enable safety margins and uncertainty-aware costs when the network changes during an event.

3.5 Design Guidance for Urban Graph QA Systems

The experimental results can be translated into a preliminary decision matrix for system design (Table 5). The matrix does not claim universal optimality; it records the interface that was most effective in the present benchmark and the evidence object that makes the answer inspectable. This distinction is important because production systems may trade accuracy against latency, compute cost, or the need to explain a result.

Question type	Preferred interface	Evidence to retain	Reason
Local count	Graph Tool / GNN	Queried node and neighbors	Local structure is directly available
Edge existence	Graph Tool / Path Tokens	Queried edge or one-hop path	Exact pairwise relation
Reachability	Graph Tool / Path / Counterfactual	Supporting path or connectivity signal	Multi-hop evidence is required
Shortest path	Path Tokens / Graph Tool	Candidate and selected path	Task-aligned path object
Centrality	Graph Tool / Counterfactual	Metric values or perturbation effect	Requires global comparison

Table 5. Result-derived guidance for selecting an interface and retaining auditable evidence.

A practical architecture can use routing rather than a single universal mechanism. The LLM first classifies the requested operation and extracts graph entities. Local descriptive questions can use a lightweight GNN or neighborhood lookup. Exact connectivity and routing questions should invoke deterministic graph procedures, with path retrieval used to package the result into a compact, inspectable context. Global comparative questions should use explicit metrics or carefully defined perturbations. The language model then explains the result, resolves ambiguity, and presents the supporting graph object rather than replacing the graph algorithm.

Such routing also provides a clearer failure policy. If entity grounding is uncertain, the system can ask for clarification rather than executing an operation on the wrong nodes. If no path exists, it can return the disconnected components or the constraints causing failure. If a tool result conflicts with a learned prediction, the deterministic operation should normally take precedence for exact topology, while the discrepancy is logged for model diagnosis. This arrangement treats the LLM as an interface and orchestrator around verifiable spatial computation, which is more defensible than treating a high-capacity language model as an implicit graph database.

4. Conclusion

This paper compared seven ways of exposing urban graph structure to a language-guided spatial QA model. Across three city graphs and five tasks, latent compression and memory remained close to the text-only baseline, whereas path retrieval, graph execution, and counterfactual evidence produced much stronger results. Graph Tool achieved the best average accuracy and led four tasks; Path Tokens was strongest on shortest path. Centrality remained the principal challenge. The main conclusion is therefore scoped but actionable: graph-language performance depends strongly on whether the interface preserves or executes the structural object required by the question. For exact urban topology, tool access is more reliable than asking an LLM to infer graph operations from compressed latent summaries. Future studies should test this principle on more diverse, weighted, semantic, and dynamically changing urban networks.

References

Barthélemy, M., 2011. Spatial networks. *Physics Reports*, 499(1-3), 1-101. doi.org/10.1016/j.physrep.2010.11.002.

Fatemi, B., Halcrow, J., Perozzi, B., 2024. Talk like a graph: Encoding graphs for large language models. *International Conference on Learning Representations (ICLR)*. arXiv:2310.04560.

He, X., Tian, Y., Sun, Y., Chawla, N.V., Laurent, T., LeCun, Y., Bresson, X., Hooi, B., 2024. G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37, 132876-132907. doi.org/10.52202/079017-4224.

Jiang, W., Luo, J., 2022. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207, 117921. doi.org/10.1016/j.eswa.2022.117921.

Li, H., Yue, P., Wu, H., Teng, B., Zhao, Y., Liu, C., 2025b. A question-answering framework for geospatial data retrieval enhanced by a knowledge graph and large language models. *International Journal of Digital Earth*, 18(1), 2510566. doi.org/10.1080/17538947.2025.2510566.

Li, Z., Grossman, M., Qasemi, E., Kulkarni, M., Chen, M., Chiang, Y.-Y., 2025a. MapQA: Open-domain geospatial question answering on map data. arXiv preprint arXiv:2503.07871. doi.org/10.48550/arXiv.2503.07871.

Li, Z., Xia, L., Ren, X., Tang, J., Chen, T., Xu, Y., Huang, C., 2025c. Urban computing in the era of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(6), 1-43. doi.org/10.1145/3768163.

Liang, J., Hou, S., Jiao, H., Qing, Y., Zhao, A., Shen, Z., Xiang, L., Wu, H., 2025. GeoGraphRAG: A graph-based retrieval-augmented generation approach for empowering large language models in automated geospatial modeling. *International Journal of Applied Earth Observation and Geoinformation*, 142, 104712. doi.org/10.1016/j.jag.2025.104712.

Liu, Y., Ding, J., Fu, Y., Li, Y., 2023. UrbanKG: An urban knowledge graph system. *ACM Transactions on Intelligent Systems and Technology*, 14(4), Article 60. doi.org/10.1145/3588577.

Mansourian, A., Oucheikh, R., 2024. ChatGeoAI: Enabling geospatial analysis for public through natural language, with large language models. *ISPRS International Journal of Geo-Information*, 13(10), 348. doi.org/10.3390/ijgi13100348.

Mansourian, A., Oucheikh, R., 2026. Bridging natural language and GIS: A multi-agent framework for LLM-driven autonomous geospatial analysis. *International Journal of Digital Earth*, 19(1), 2633849. doi.org/10.1080/17538947.2026.2633849.

Mavromatis, C., Karypis, G., 2024. GNN-RAG: Graph neural retrieval for large language model reasoning. arXiv preprint arXiv:2405.20139. doi.org/10.48550/arXiv.2405.20139.

Wang, H., Feng, S., He, T., Tan, Z., Han, X., Tsvetkov, Y., 2023. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36, 30840-30861. doi.org/10.52202/075280-1345.

Wu, N., Wang, X., Li, F., Zhao, B.Y., Zheng, H., 2020. Learning effective road network representation with hierarchical graph neural networks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 6-14. doi.org/10.1145/3394486.3403043.

Yap, W., Biljecki, F., 2023. A global feature-rich network dataset of cities and dashboard for comprehensive urban analyses. *Scientific Data*, 10(1), 667. doi.org/10.1038/s41597-023-02578-1.

Zhao, J., Zhuo, L., Shen, Y., Qu, M., Liu, K., Bronstein, M., Zhu, Z., Tang, J., 2023. GraphText: Graph reasoning in text space. arXiv preprint arXiv:2310.01089. doi.org/10.48550/arXiv.2310.01089.

Zheng, Y., Capra, L., Wolfson, O., Yang, H., 2014. Urban computing: Concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology*, 5(3), Article 38. doi.org/10.1145/2629592.