

Spatial-RAG——开启空间智能问答的新时代

0. 背景简介

在人工智能领域，大型语言模型（LLMs）如GPT系列已经在文本生成和理解任务中展现了强大的能力。然而，当涉及到空间推理任务时，LLMs的表现却显得力不从心。空间推理不仅要求模型理解复杂的空间关系，还需要结合地理数据和语义信息，生成准确的回答。为了突破这一瓶颈，研究人员推出了Spatial Retrieval-Augmented Generation（Spatial-RAG）——一个革命性的框架，旨在增强LLMs在空间推理任务中的能力。

1. 论文信息

标题：Spatial-RAG: Spatial Retrieval Augmented Generation for Real-World Spatial Reasoning Questions

作者：Dazhou Yu^{1*}, Riyang Bao^{1*}, Gengchen Mai², Liang Zhao¹

¹埃默里大学 Emory universtiy ²德州大学奥斯汀分校 University of Texas-Austin * equal contribution

原文链接：https://www.researchgate.net/publication/388656403_Spatial-RAG_Spatial_Retrieval_Augmented_Generation_for_Real-World_Spatial_Reasoning_Questions

2. 挑战与解决方案

空间问答长期以来一直是一个基础领域，它包括各种空间问题，从识别最近的邻居到检测线与多边形的交集。然而，传统的空间问答系统依赖于专门的空间查询语言（比如GeoSPARQL, Spatial SQL），这些语言与人类语言大相径庭，使得普通用户难以使用。更重要的是，这些系统缺乏从人类文本的丰富上下文中推断复杂空间和语义关系的能力，限制了它们在现实世界问答场景中的适用性。

近年来，大型语言模型（LLMs）的进展已经在机器学习（ML）的许多领域带来了变革，特别是在理解和生成类人文本方面。这一进展激发了人们通过直接从LLMs中提取空间知识来弥合空间问答与自然语言之间的差距。这些努力涵盖了广泛的应用，包括地理百科全书问答、地理定位和自动高精度地图生成等。尽管取得了这些进展，最近的研究表明，LLMs在空间推理方面表现出显著的局限性，甚至在处理基本的空间任务时也遇到困难，例如地理解析和理解相对空间关系。这种差距在处理现实世界的空间推理任务时尤为明显，例如图1中所示的场景：



图1. 现实世界中空间推理问题示例。满足空间约束的区域以蓝色突出显示

挑战：空间与语义的双重需求

- 空间约束：**图1用户的问题涉及复杂的空间关系：“沿某条路线推荐餐厅”。传统的空间数据库可以高效处理这些空间查询，但它们无法理解用户的语义需求：“要求肉食”。
- 语义理解：**LLMs擅长理解自然语言中的语义信息，例如用户对餐厅类型、价格或评分的偏好。然而，LLMs缺乏直接处理空间数据的能力，无法执行复杂的空间计算（例如，计算两个点之间的距离或判断一个点是否位于某个区域内）。

为了解决这些挑战，增强LLMs的空间推理能力，该研究将 **检索增强生成 (RAG)** 扩展到空间信息检索和推理，弥合结构化空间数据库与非结构化文本推理之间的差距。

RAG在知识密集型任务（如问答）中已经展示了其有效性，通过检索特定领域的文档来增强LLM的响应。然而，现有的RAG系统主要专注于检索和生成文本内容，缺乏空间推理任务所需的空间智能，尤其是涉及理解和计算几何体（如点、多段线和多边形）之间复杂空间关系的任务。

如图1中的例子，回答问题需要LLM从用户的文本请求中提取并形式化问题为“找到靠近多段线的点”，并基于空间地图（数据库）解决该问题。然后，它还需要推断用户偏好，以选择空间和语义上更优的候选对象。因此，系统必须无缝集成结构化空间检索与非结构化文本推理，确保空间准确性和上下文理解。为了实现这一目标，研究人员引入了空间

检索增强生成 (Spatial-RAG) , 这是一个新颖的框架, 将文本引导的空间检索与空间感知的文本生成相结合。具体来说, 为了识别空间相关的候选答案, 研究人员提出了一种新颖的空间混合检索模块, 结合了稀疏和密集检索器。为了对候选答案进行排序并生成最终答案, 研究人员提出基于空间和语义联合排序策略的Pareto前沿检索结果来驱动生成器。该研究贡献总结如下:

1. 通用的Spatial-RAG框架: Spatial-RAG是第一个将RAG扩展到空间问答的框架, 能够处理广泛的空间推理任务, 如地理推荐、空间约束搜索和上下文路径规划。无缝集成了空间数据库、LLMs和基于检索的增强, 使得能够在LLMs的熟悉操作范式内有效处理复杂的空间推理问题。
2. 稀疏-密集空间混合检索器: 提出了一种混合检索机制, 结合了稀疏检索 (基于SQL的结构化查询) 和密集检索 (基于LLM的语义匹配) 。这种双重方法确保检索结果在空间和语义上与用户查询一致, 显著提高了空间上下文中的检索准确性。
3. 多目标引导的空间文本生成器: 为了处理空间问答任务中的空间约束和文本推理, 引入了一个多目标优化框架, 动态平衡空间和语义相关性之间的权衡。这确保了生成的响应既几何准确又语言连贯。
4. 真实世界评估: 在从旅游网站收集的真实世界数据集上评估了方法, 该数据集包含用户对不同空间实体的问题和评论。在该数据集上的实验揭示了处理现实世界空间推理问题的能力。

通过这些创新, Spatial-RAG显著增强了LLMs的空间推理能力, 弥合了结构化空间数据库与自然语言问答之间的差距。

3. 方法简介

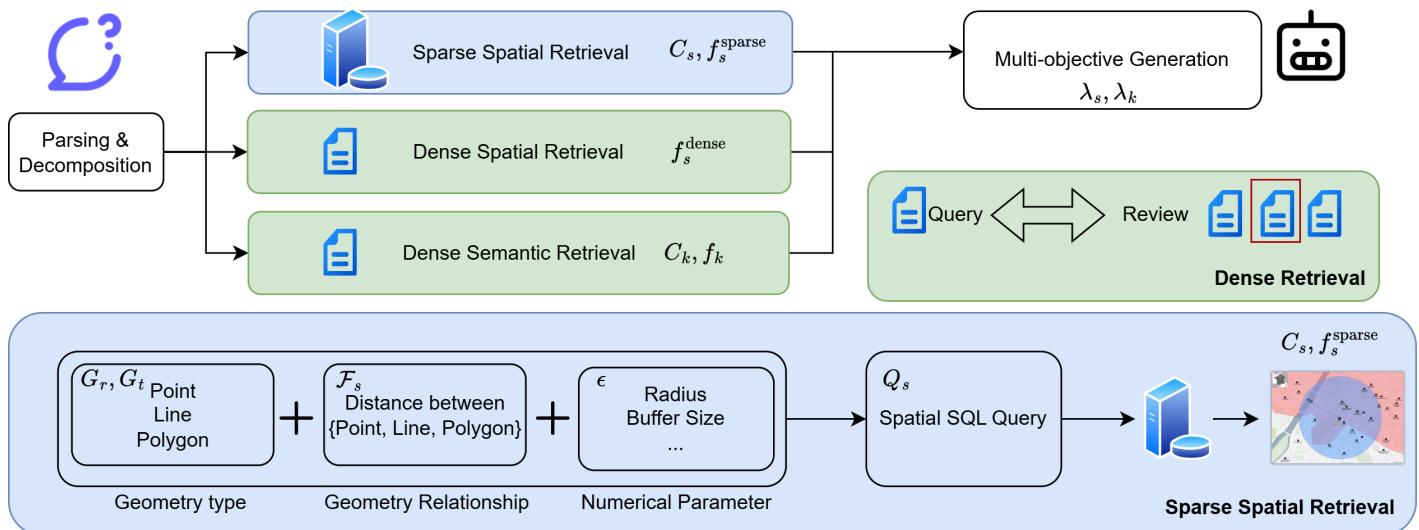


图2. Spatial-RAG 框架

3.1 概述

对于一个空间推理问题 q , Spatial-RAG 将生成答案 y , 形式上, 该研究定义:

$$y^* = \arg \max_y \lambda_s^T f_s(q, y) + \lambda_k^T f_k(q, y)$$

$$\text{s.t. } y \in C_s(q), \quad y \in C_k(q), \quad \lambda_s \geq 0, \quad \lambda_k \geq 0, \quad \mathbf{1}^T \lambda_s + \mathbf{1}^T \lambda_k = 1.$$

其中包含三个待解决的关键阶段:

1. **构建空间候选集 C_s** : 系统必须精确定义空间约束, 然后检索满足这些约束的空间对象。如图2 sparse spatial retrieval (稀疏空间检索) 所示, 通过将输入的自然语言问题解析为SQL查询来实现这一点, 该查询将在空间数据库上执行, 以高效地从数据库中检索相关空间对象。此过程在第4.2节中详细说明。

2. **计算空间相关性** $f_s(q, y)$: 为了在集成文本信息的同时有效计算空间相关性，研究人员提出了一种混合空间检索方案。如图2所示，该方法结合了来自数据库的稀疏空间相关性分数和来自文本嵌入的密集语义相似性分数。这使得系统能够根据输入问题的空间相关性对检索到的空间对象进行排序，详见第4.3节。
3. **多目标优化生成**: 在给定空间和语义约束的情况下，研究人员提出了一个多目标优化问题来平衡这些因素。系统计算候选答案的Pareto前沿，LLM动态在这些解决方案之间进行权衡，以生成最优响应。此步骤在第4.4节中详细介绍。

3.2 稀疏空间检索

空间推理问题的答案必须满足特定的空间约束。空间候选集 $C_s(q)$ 由满足一组空间约束 $\mathcal{C}_s(q)$ 的所有可能答案 y 组成。形式上，该研究定义：

$$C_s(q) = \{y \mid c_s(y, q) \leq 0, \forall c_s \in \mathcal{C}_s(q)\},$$

其中 $c_s(y, q)$ 表示编码空间条件的约束函数（例如，拓扑、方向或距离约束）， $\mathcal{C}_s(q)$ 是与问题 q 相关的所有空间约束的集合。例如，如果空间约束要求 y 与参考位置 l_q 的距离不超过 ϵ ，则可能的约束函数为：

$$c_s(y, q) = d(y, l_q) - \epsilon \leq 0.$$

这种公式确保只有空间上有效的答案才会被包含在 $C_s(q)$ 中。

处理空间约束需要在空间数据库中执行定义良好的空间SQL查询。此过程涉及识别适当的查询函数、参考空间对象、目标空间对象以及任何必要的数值参数。形式上，空间SQL查询可以表示为：

$$Q_s = \mathcal{F}_s(G_r, G_t, \epsilon)$$

其中 \mathcal{F}_s 是确定对象之间关系的空间查询函数。 G_r 表示从问题中提取的参考对象集合。 G_t 表示作为潜在答案的目标对象集合。 ϵ 是控制空间约束的数值参数。

鉴于这些约束的多样性和潜在的复杂性，大型语言模型（LLMs）通常难以直接从用户输入中构建完整且可执行的空间查询。为了弥合这一差距，研究人员逐步构建空间查询，允许LLM系统地填充所需的组件。

方法遵循三个关键步骤：

1. **几何识别**: 从用户输入中识别并提取参考空间对象 G_r 和候选目标空间对象 G_t ，并提取它们的空间几何体。
2. **查询函数选择**: 根据预期的空间关系（例如，包含、接近）确定适当的空间函数 \mathcal{F}_s 。
3. **参数估计**: 分配数值约束 ϵ 以确保精确的空间过滤（例如，缓冲区半径）。

通过形式化这一结构化过程，研究人员增强了LLM生成准确且可执行的空间SQL查询的能力，从而提高了系统处理复杂空间推理问题的能力。

3.2.1 几何识别

在空间推理任务中，准确识别空间对象并提取其空间几何体对于将问题解析为空间查询至关重要。空间对象 $g \in \mathcal{G}$ 通常可以分为三种基本类型：点、多段线和多边形。形式上，研究人员定义这些类别如下：

- **点**: $\mathcal{G}_{\text{point}} = \{g \mid g \in \mathbb{R}^2, \dim(g) = 0\}$ 。此类别包括单个点和多点，表示面积可忽略的位置。例如，停车标志、地址点和用户的当前位置。在空间数据库中，这些实体通常表示为“点”几何类型。
- **多段线**: $\mathcal{G}_{\text{line}} = \{g \mid g \subseteq \mathbb{R}^2, \dim(g) = 1\}$ 。多段线（包括多段线组）表示宽度可忽略的线性一维对象。常见的例子包括街道、河流、公交路线和电力线。在空间数据库中，这些几何体抽象为“线串”类型。

- **多边形:** $\mathcal{G}_{\text{polygon}} = \{g \mid g \subseteq \mathbb{R}^2, \dim(g) = 2\}$ 。多边形（包括多边组）表示定义封闭区域的二维对象。这些几何体对于描绘区域（如人口普查区、地块、县、社区和分区区域）至关重要。

空间查询的复杂性取决于所涉及对象的类型。对于较简单的查询，例如“从给定位置找到最近的公交站”，只需要点几何体，空间候选集为：

$$C_s = \{g \mid g \in \mathcal{G}_{\text{point}}, d(g, g_{\text{point}}) < \epsilon\}$$

其中 $g_{\text{point}} \subseteq \mathcal{G}_{\text{point}}$ 表示点对象（例如，给定位置）， ϵ 是距离阈值。对于更复杂的查询，例如“我将从家沿着第7街和琼斯街步行到大学校园；请推荐一家我在步行途中购买早餐的咖啡馆。”，必须考虑多种几何类型，空间候选集为：

$$C_s = \{g \mid g \in \mathcal{G}_{\text{point}}, g \in B(g_{\text{polyline}}, \epsilon) \cup g_{\text{polygon}}\}$$

其中 $g_{\text{polyline}} \subseteq \mathcal{G}_{\text{polygon}}$ 表示多段线对象（例如，路线）， $g_{\text{polygon}} \subseteq \mathcal{G}_{\text{polygon}}$ 表示多边形区域（例如，大学校园）， B 是多段线周围的缓冲区， ϵ 是缓冲区大小。

通过以这种方式构建空间查询，研究人员确保了精确的几何表示，促进了强大的空间推理和查询执行。

3.2.2 查询函数识别和参数估计

在识别了空间查询中涉及的几何体之后，下一步是确定处理各种几何交互所需的适当空间查询函数 \mathcal{F}_s 。尽管几何体之间的交互不同，但它们可以通过距离函数 $d(g_r, g_t)$ 统一处理，该函数计算两个几何实体 $g_r, g_t \in \mathcal{G}$ 之间的最短距离。

形式上，给定参考几何体集合 $G_r \subseteq \mathcal{G}$ 和目标几何体集合 $G_t \subseteq \mathcal{G}$ ，空间候选集 C_s 可以定义为：

$$\begin{cases} \{g_t \in G_t \mid \exists g_r \in G_r, d(g_r, g_t) \leq \epsilon\}, & \text{如果 } d(g_r, g_t) > 0, \\ \{g_t \in G_t \mid \exists g_r \in G_r, g_r \cap g_t \neq \emptyset\}, & \text{如果 } d(g_r, g_t) = 0. \end{cases}$$

参数如搜索半径或缓冲区距离 ϵ 由LLM自主确定，通常基于上下文理解（例如，估计的步行距离或感兴趣区域）。参数 ϵ 可以表示为： $\epsilon = \phi(q)$ ，其中 ϕ 是将查询 q 的上下文映射到适当数值的函数。

一旦几何体 G_r, G_t 、函数 \mathcal{F}_s 和参数 ϵ 被确定，系统将构建精确的空间查询 Q_s 。这确保了从空间数据库中进行精确检索，保持结果的准确性和相关性。通过利用这些数学公式，系统有效地将空间推理任务转化为可执行的查询，促进了LLM框架内的强大空间智能。

3.3 混合空间对象排序

空间相关性分数 f_s 由两个部分组成：一个来自空间数据库的稀疏空间检索分数，另一个来自基于问题和候选对象空间描述之间的文本相似性的密集空间检索分数。形式上，研究人员定义：

$$f_s = \lambda_s f_s^{\text{sparse}} + \lambda_d f_s^{\text{dense}},$$

其中 λ_s 和 λ_d 是控制每个分数贡献的权重系数。

3.3.1 稀疏空间相关性评分

稀疏空间相关性直接从空间数据库中使用显式空间关系计算。分数由空间查询函数 \mathcal{F}_s 确定，该函数计算参考对象和目标对象之间的距离。形式上，研究人员定义：

$$f_s^{\text{sparse}} = \begin{cases} \frac{1}{1+d(g_r, g_t)}, & \text{如果 } g_r \cap g_t = \emptyset \\ 1, & \text{如果 } g_r \cap g_t \neq \emptyset \end{cases}$$

其中 g_r 和 g_t 分别是参考和目标空间对象。 $d(g_r, g_t)$ 是测量空间数据库中接近度的距离函数。如果 g_t 与 g_r 重叠，分配一个完美的相关性分数1。

这确保了区域内的对象具有最大的相关性，而区域外的对象随着距离的增加，其分数逐渐衰减。

3.3.2 密集空间相关性评分

与稀疏评分不同，密集空间相关性是从与空间对象相关的文本描述中推断出来的。研究人员利用LLM从用户查询中提取关键空间属性，并将其与候选对象的描述进行比较。

提取空间需求：给定用户查询 q 和一组空间对象 G_t 的文本描述 d_t ，研究人员通过基于注意力的掩码函数提取相关的内容：

$$v_{q,s} = \mathcal{E}(\mathcal{M}_s(q)), \quad v_{t,s} = \mathcal{E}(\mathcal{M}_s(d_t)),$$

其中 $v_{q,s}$ 和 $v_{t,s}$ 是空间特征的密集向量表示， \mathcal{M} 是将输入文本映射到空间相关文本的提取函数， \mathcal{E} 是文本编码器。

通过余弦相似性排序：相关性分数通过余弦相似性计算：

$$f_s^{\text{dense}} = \frac{v_{q,s} \cdot v_{t,s}}{\|v_{q,s}\| \|v_{t,s}\|}.$$

3.3.3 混合排序作为广义模型

混合排序泛化了稀疏和密集排序方法：

- **仅稀疏情况：**如果 $\lambda_d = 0$ ，则 $f_s = \lambda_s f_s^{\text{sparse}}$ ，简化为纯基于距离的排序。
- **仅密集情况：**如果 $\lambda_s = 0$ ，则 $f_s = \lambda_d f_s^{\text{dense}}$ ，简化为纯基于语义的排序。
- **混合情况：**如果两个权重都非零，混合排序受益于显式空间约束和隐式语义相关性，从而形成更全面的排序机制。

这种公式确保混合排序通过捕捉空间接近度和语义对齐，优于任何单一排序方法。

3.4 多目标生成

语义候选集 C_k 和语义相关性分数 f_k 基于密集向量相似性计算。在获得所有分数和候选集后，问题变为多目标优化问题，因为每个视角（空间和语义）都独立贡献。

3.4.1 Pareto前沿计算

给定空间和语义相关性分数，目标是识别在空间和语义相关性之间实现最佳权衡的Pareto最优候选。一个候选 y 是Pareto最优的，如果没有其他候选在空间和语义相关性上都优于它。形式上，Pareto前沿 $P(q)$ 定义为：

$$P(q) = \{y \in C_s \cap C_k \mid \nexists y' \in C_s \cap C_k, f_s(q, y') \geq f_s(q, y) \text{ 且 } f_k(q, y') \geq f_k(q, y), \text{ 至少有一个严格不等式}\}.$$

这确保了 $P(q)$ 中的每个候选都是非支配的，意味着没有其他候选在空间和语义相关性上都严格优于它。

3.4.2 基于LLM的权衡决策

一旦确定了Pareto前沿 $P(q)$ ，研究人员使用LLM根据用户查询的上下文动态平衡空间约束和语义偏好之间的权衡。具体来说，LLM接收用户查询、稀疏空间相关性分数和空间对象描述作为输入：

$$I = \{q, (f_s^{\text{sparse}}(q, y), d_y), \forall y \in P(q)\}.$$

基于上下文信息的动态加权函数 $\lambda_1, \lambda_2 = h(I)$ 从输入中提取，调整空间与语义相关性的重要性，其中 h 是捕捉查询特定权衡的学习函数。

LLM选择排名最高的候选 y^* ：

$$y^* = \arg \max_{y \in P(q)} \lambda_s^T f_s(q, y) + \lambda_k^T f_k(q, y),$$

并生成自然语言响应。

系统适应不同的查询上下文，而不是使用固定的加权方案。通过将决策过程结构化为离散步骤（候选过滤 → Pareto选择 → 权衡平衡 → 响应生成），LLM避免了生成不可行或不合理的结果。这种结构化方法最大限度地提高了准确性和可用性，确保系统的最终响应与用户的原始意图紧密一致。

4. 实验部分

研究人员在 **纽约市** 和 **迈阿密** 的旅游数据集上对Spatial-RAG进行了评估，展示了其在处理真实世界空间推理问题上的显著优势。

数据集与评估指标

- **数据集：** 使用了来自TripAdvisor的用户问题和评论数据，涵盖纽约市的9,470个兴趣点（POIs）和迈阿密的2,640个POIs。
- **评估指标：**
 - i. **交付率：** 评估方法是否能够成功生成结果。
 - ii. **空间稀疏通过率：** 评估解析的空间查询是否正确。
 - iii. **空间密集通过率：** 评估答案是否满足问题中的空间相关语义约束。
 - iv. **语义通过率：** 评估答案是否符合问题中的语义约束。

对比方法

为了评估LLM（GPT-3.5-Turbo和GPT-4-Turbo）在此框架下的表现，研究团队对比了以下基线方法：

- **Sort-by-distance (SD)**：按照空间问题中的参考对象距离排序候选空间对象。
- **Text embedding (TE)**：基于文本描述的嵌入向量，计算目标对象与参考对象的向量距离，并选择最近的对象。
- **Spatial-text (ST)**：基于用户问题的嵌入向量计算与目标对象文本描述的相似度，并结合目标对象的距离得分进行加权求和后决策。
- **Naive RAG**：使用向量数据库存储所有空间对象描述，并基于向量相似性检索最相关的对象。
- **GeoLLM**：对空间对象进行编码，并通过添加附近对象的空间信息丰富上下文。

实验结果

City	Model	Delivery Rate	Spatial Sparse Pass Rate	Spatial Dense Pass Rate	Semantic Pass Rate
NYC	SD	100	-	57.1	32.1
	TE	99.6	-	48.6	<u>54.6</u>
	ST	100	-	53.6	55.9
	Naive RAG	<u>99.8</u>	-	52.0	54.5
	GeoLLM	<u>99.8</u>	-	<u>69.5</u>	42.8
	Spatial-RAG (GPT-3.5-Turbo)	87.2	67.0	64.4	47.4
Miami	Spatial-RAG (GPT-4-Turbo)	86.1	<u>65.0</u>	71.6	50.1
	SD	100	-	36.8	22.4
	TE	100	-	28.9	39.5
	ST	100	-	26.3	42.1
	Naive RAG	100	-	31.6	40.8
	GeoLLM	100	-	<u>52.6</u>	36.8
	Spatial-RAG (GPT-3.5-Turbo)	<u>86.8</u>	<u>75.8</u>	57.6	<u>43.9</u>
	Spatial-RAG (GPT-4-Turbo)	<u>86.8</u>	81.8	51.5	45.5

Table 1. Performance comparison of models in New York City (NYC) and Miami. The framework is deployed on GPT-4-Turbo and GPT-3.5-Turbo and compared its performance with several baseline models, including SD, TE, ST, Naive RAG, and GeoLLM.

Model	Delivery Rate	Spatial Sparse Pass Rate	Spatial Dense Pass Rate	Semantic Pass Rate
Spatial-RAG (GPT-4-Turbo)	86.1	<u>65.0</u>	<u>71.6</u>	<u>50.1</u>
w/o sparse spatial	98.9	-	53.3	78.4
w/o dense spatial	<u>89.3</u>	61.5	68.4	49.6
w/o dense semantic	85.9	72.8	75.9	34.8

Table 2. Different modules of the proposed framework are removed for ablation study.

纽约 (NYC)

- Spatial-RAG (GPT-3.5-Turbo 和 GPT-4-Turbo) 在交付率上与其他基线方法存在一定差距，大约 86.1% 的问题被成功处理。
- 失败的 12.9% 的情况是由于无法从空间数据库检索到任何空间对象（可能由于多边形识别错误或SQL查询指定区域内无相关对象）。
- 另 0.9% 的情况是 LLM在重新排序 (reranking) 过程中未能正确排列检索结果。
- Spatial-RAG (GPT-4-Turbo) 比 GPT-3.5-Turbo 在 Spatial Dense Pass Rate 上高 7%，在 Semantic Pass Rate 上也稍有优势。
- SD 方法由于仅基于距离返回最近的对象，其 Spatial Dense Pass Rate 较高，但其他指标表现较差。
- GeoLLM 方法仅基于对象名称和距离，因此在 Spatial Dense Pass Rate 方面表现尚可。
- TE 和 ST 方法考虑了语义信息，在 Semantic Pass Rate 方面表现较优。
- Naive RAG 和 ST 共同优化了空间密集检索和用户语义检索，因此两者在这两个方面表现接近。

迈阿密 (Miami)

- Spatial-RAG 在迈阿密数据集上的表现也较好。
- 基线方法的表现模式与纽约数据集基本一致，但由于迈阿密的数据量较小 (QA 对数量仅为 133)，模型表现的稳定性可能受到影响。

消融实验

研究人员通过移除稀疏空间模块、密集空间模块和密集语义模块进行了消融实验。结果表明：

- 移除稀疏空间模块后，交付率显著提高，但空间得分下降。
- 移除密集语义模块后，空间密集通过率最高，但语义通过率显著降低。

案例研究

图3和图4(b)展示了一个典型的多段线搜索案例。Spatial-RAG成功识别了用户意图，推荐了沿路线的餐厅，展示了其在复杂空间推理任务中的强大能力。

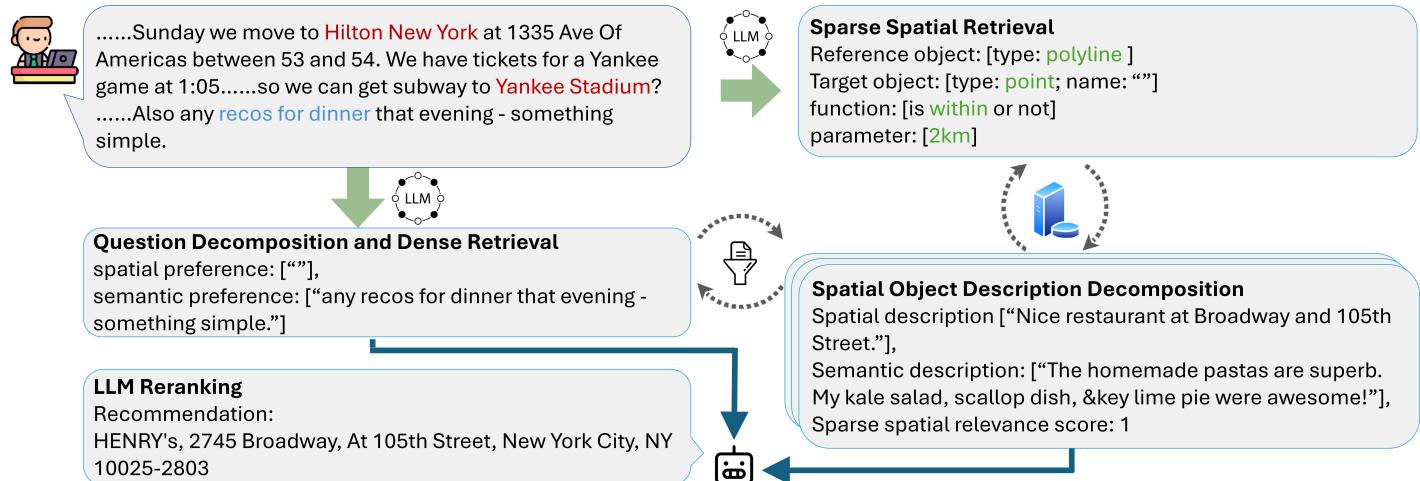


图3. Spatial-RAG 运行方式的示例：给定一个问题，1) 稀疏空间检索：LLM 将自然语言问题解析为空间数据库的空间 SQL 查询，检索满足空间约束和稀疏空间相关性分数的空间对象。2) 问题分解和密集检索：同时，Spatial-RAG 将问题分解为空间和语义组件，并将它们与空间对象的描述进行比较以执行密集检索，过滤掉不相关的内容。3) LLM 重新排序：语言代理平衡空间和语义方面以重新排序候选并生成最终答案。

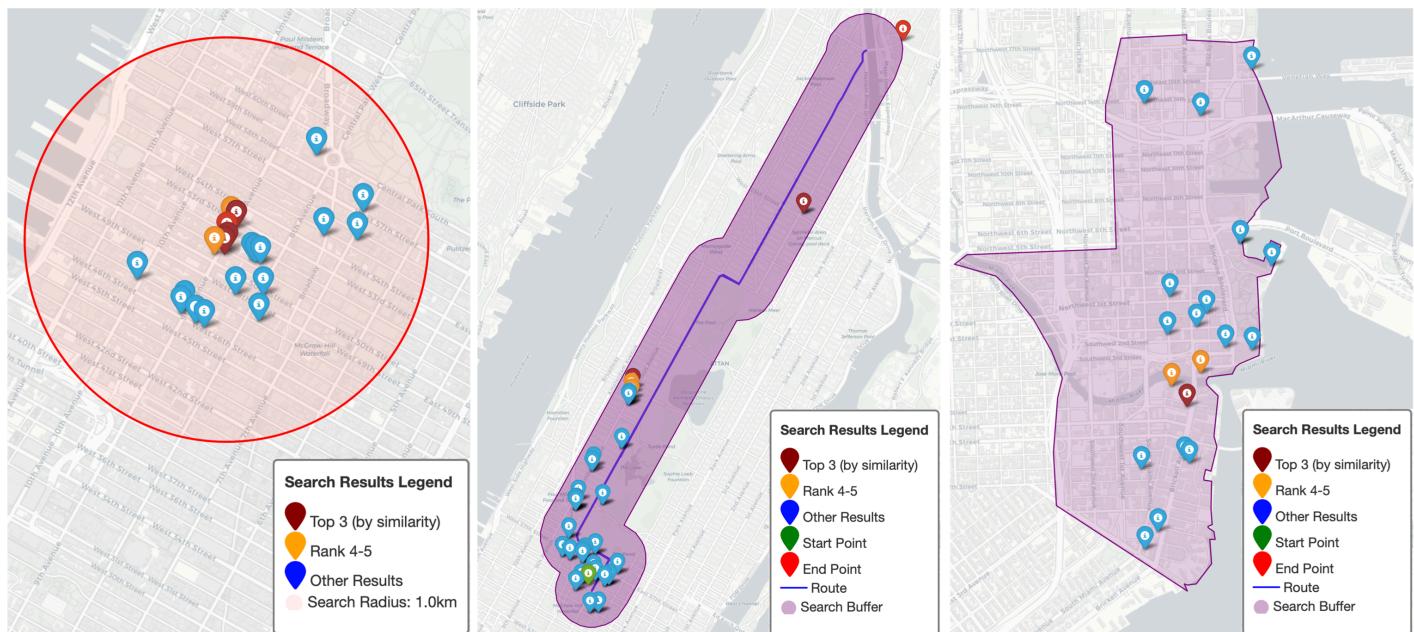


图4. (a) 查询 ϵ km半径内的点 (b) 查询路径周围的点 (c) 查询多边形内的点

5. 结论

Spatial-RAG通过结合空间数据库和LLM的语义理解能力，显著提升了空间推理任务的性能。实验表明，Spatial-RAG在真实世界数据集上表现优异，能够有效处理复杂的空间推理问题，为旅游推荐、路径规划等应用提供了强大的支持。