SpaRAGi: Spatial Inference using Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

The advent of large language models (LLMs) has enabled powerful applications across several domains such as science, healthcare, finance, and law. However, LLMs are challenged when asked domain-specific questions. In particular, the spatial knowlege and spatial inference capabilities of LLMs are limited. Our goal is to enhance their accuracy for queries that reason about spatial data. To this end, we leverage the emerging Retrieval Augmented Generation (RAG) paradigm via which LLMs can enrich their context using external data, during inference. We present a framework that i) extracts context from a geospatial database regarding the spatial relations between entities, and ii) retrieves the relevant context to a query at inference time, forwarding it to the LLM to enhance its accuracy. Overall, our framework sets the ground for the use of spatial knowledge retrieval techniques for improving the effectiveness of LLMs.

1 Introduction

Retrieval augmented generation (RAG) (Lewis et al., 2020) improves the performance of generative models, such as large language models (LLMs) by retrieving relevant information from external sources. RAG has been especially useful when we need to generate responses based on large and complex sources of knowledge that have not been used in the model training process. The success of RAG has brought opportunities for new research in data management and information retrieval toward improving LLM effectiveness (Fan et al., 2024).

Spatial data collections are typically in structured format and stored in database systems such as PostgreSQL[1] and Oracle Spatial[2], or GIS like QGIS[3]. The relations between all pairs of spatial data entities on a map are not explicitly stored or used in the training process of a foundation model, so existing models are not trained with such knowledge. Hence, LLMs underperform when it comes to questions that reason about spatial entities.

To fill this gap, we propose SpaRAGi, a framework that enriches model generation through RAG, with spatially enriched context to help infer spatial knowledge without re-training or fine-tuning the model. The main challenge is that this knowledge is not explicitly stored in natural language, which would make it comprehensible natively by LLMs, but in record format which carries additional processing costs. To alleviate this, SpaRAGi pre-processes spatial data sources to generate text snippets that succinctly capture *non-trivial spatial relations* between entities on a map.[4] These documents are then encoded and employed by a RAG mechanism to retrieve knowledge that can boost the accuracy of small, open-source LLMs, such as Llama (Touvron et al., 2023) and Mistral (Jiang et al., 2023), in spatial reasoning tasks.

The number of pairwise spatial relations between entities on a map is quadratic, making their generation and encoding a challenging task. In SpaRAGi, we address this scalability challenge by i) dividing the map into numerous local partitions, ii) computing non-trivial spatial relations between all pairs of entities within each local region, and iii) structuring the computed relations in a clear and comprehensive textual format to enhance the model's ability to infer non-local relations.

We focus on generation tasks involving spatial relations and implement and test our framework using spatial data. Nonetheless, our approach can be generalized to assist any RAG approach that involves complex relations between objects and can be supported by inference rules. An example

[1]https://www.postgresql.org/
[2]https://www.oracle.com/database/spatial/
[3]https://www.qgis.org/

[4]The fact that two entities are *topologically disjoint* is trivial and can be inferred if no other explicit topological relation is known for these entities.

```
> Using SpaRAGi
> Give a query (type 'exit' to quit): Does Stanton County Nebraska contain Zipcode 68779?
> Prompt: Does Stanton County Nebraska contain Zipcode 68779?
> Context: Stanton County Nebraska contains Zipcode 68779 entirely. This means that Zipcode 68779 lies completely inside
of Stanton County Nebraska's area and their borders do not intersect at all. Hence, Stanton County Nebraska's area covers
more square kilometers than the area of Zipcode 68779. Specifically, Stanton County Nebraska covers about 712.59 square
kilometers whilst Zipcode 68779 has an area of 263.44 square kilometers.
> Response: Yes, Stanton County Nebraska contains Zipcode 68779.
```

Figure 1: Example spatial query interaction with SpaRAGi.

usage scenario is illustrated in Figure 1, where SpaRAGi helps Llama-3.1-8B-Instruct to respond correctly to the query by enriching the original prompt with the necessary context for an accurate response. In particular, SpaRAGi retrieves encoded context which is similar to the prompt, accesses the corresponding text, and combines it with the prompt before feeding it to the model. The model would otherwise hallucinate on the answer, based on the general knowledge it might possess.

Existing literature indicates that LLMs primarily utilize spatial data indirectly, relying on external tools (Manvi et al., 2024; Singh et al., 2024; Zhang et al., 2023) rather than incorporating it directly during the prompting process. This approach arises due to the inherently complex nature of spatial data and its significant differences from text. But since LLMs are tailored to handle text data, the research question asked in this study is *why are there not any spatial datasets in text form?* To address this research question, we decompose it into the following. First, *how can spatial text be generated?* In §3.1, we introduce a synthetic spatial text generator designed to extract key spatial information from spatial data and convert it into textual form. In §3.2, we present the first synthetic spatial text datasets. We make these datasets made publicly available to facilitate model training and fine-tuning. This naturally leads to the next question: *Can spatial text be effectively leveraged by models during inference to derive spatial knowledge?* In §3.3, we propose a method for assisting models in inferring spatial information using retrieval-augmented generation. Last, in §4, we conduct a comprehensive evaluation of our approach and test the generated datasets across a range of open-source models.

2 Related Work

Related work that leverages external spatial information to assist LLMs includes GeoLLM (Manvi et al., 2024), GeoLLM-Engine (Singh et al., 2024) and GeoGPT (Zhang et al., 2023). GeoLLM focuses on regression tasks such as the prediction of population density; it uses auxiliary map data from OpenStreetMap from which the nearby locations of the given (query) location are fetched and passed to the LLM as a fine-tuned prompt. GeoLLM-Engine is an environment of tool agents for earth observation applications. It capitalizes a LLM in order to convert natural language instructions into a set of tasks over satellite images. For this, it performs function calls to geospatial APIs, dynamic maps/UIs and external multimodal knowledge bases. GeoGPT employs an LLM for interpreting the users' demands from the input and calls an external GIS tool from a pool of available ones to solve the task. Some of these tools serve processes that pertain to data collection, data loading and data analysis. GeoLLM, GeoLLM-Engine, and GeoGPT employ a distinct methodology from SpaRAGi, focusing on tasks unrelated to spatial reasoning.

Another line of research fine-tunes an LLM to enhance its understanding of spatial context. MaaSDB (Qi et al., 2023) envisions a spatial database system for enhanced user accessibility by training LLMs on data retained in a spatial database. In this way, the machine learning models can be utilized as a spatial database, enabling a new generation-based query paradigm that replaces the traditional retrieval-based one. LLM-Geo (Li and Ning, 2023) is a prototype that operates as an autonomous GIS that can produce and execute Python code for spatial data loading and visualization. By exploiting the capabilities of the LLM natural language understanding, reasoning and code generation, it manages to generate at first a step-by-step workflow that is formed as a directed acyclic graph given users' data and spatial question. The graph consists of a series of connected operations and nodes. The LLM is reused, as the graph is passed to it in order to generate code function in each operation node. Then, the generated code is collected and submitted to the LLM along with the graph and the users' input to create the final program. The program is executed producing the results that can be static maps, charts, new datasets, etc.

Concerning how well a LLM exploits information beyond of its pre-trained knowledge base, there exist several RAG benchmarks for the evaluation process. Most of them study the efficiency of the retrieval and the response generation by means of question-answering instances. Specifically, the main aspects that are studied are the context relevance (how pertinent the retrieved context to the query is), the context utilization (the extent of the context that is used by the generator to produce the response), error handling (ability to handle errors that exist in documents) and completeness (how well the response incorporates all the relevant information in the context). RGB (Chen et al., 2024) focuses on data that pertain to news, while RAG-Bench (Friel et al., 2024) cover different domains.

CRAG (Yang et al., 2024) is a comprehensive factual question-answering benchmarking that aims at defining types of questions from different domains given their diverse and dynamic nature. BERGEN (Rau et al., 2024) emphasizes on the LLM-based semantic evaluation of answers, highlighting the importance of using efficient retrievers as they can affect the RAG response generation. MIRAGE (Xiong et al., 2024) measures the accuracy of the predicted correct answer choices on multi-choice questions, but for the medical domain. Similarly, LegalBench-RAG (Pipitone and Alami, 2024) emphasizes in the legal domain measuring the effectiveness of the retrieval phase and the legal reasoning of LLMs. UDA (Hui et al., 2024) focuses on the RAG assessment on lengthy and highly unstructured external data such as those found in PDFs and HTML tables. MultiHop-RAG (Tang and Yang, 2024) assesses multi-hop queries, i.e. queries that require retrieving information from multiple documents to reason and arrive at an answer. It evaluates the quality of the retrieved set for the query and the reasoning capability of the LLM.

3 SpaRAGi

The geometry of a spatial entity is represented by a sequence of geographic coordinates (longitude, latitude). To compute spatial relations between entities from their raw representations, costly operations, such as line intersection detection, point-in-polygon tests (to detect containment of an object into another), and distance calculations (for proximity detection) must be applied (de Berg, 1997).

Spatial, domain-specific knowledge is missing from foundation models, giving room for improvement via RAG. LLMs are tailored to handle natural language, so the model relies on external specialized tools to process the spatial data. This is usually expensive on time and resources, leading to added response delays during inference. Additionally, to the best of our knowledge, no spatial datasets in text format currently exist, despite their potential to be more interpretable and accessible to LLMs compared to raw spatial data.

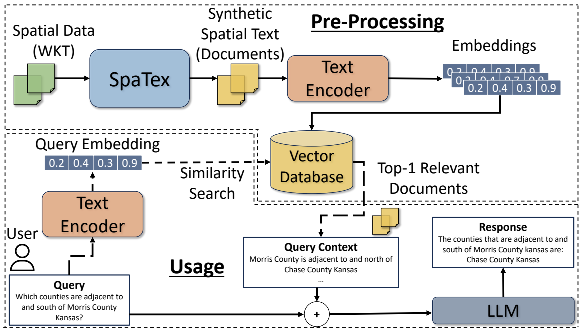


Figure 2: SpaRAGi's overview, including $SpaTex$'s synthetic spatial text generation stage and the embedding and indexing of the generated texts.

We hypothesise that if spatial knowledge is expressed comprehensively (natural language) and concisely (lack of noise, redundancy) in textual form, then the LLM may be able to infer spatial relationships between objects. Also, it may allow the user to query spatial data in natural language, as illustrated in Figure 1. We use RAG to enhance a spatial query with related context, in order to guide models to infer the correct response. Specifically, synthetic spatial texts are first generated from raw spatial data. All texts are then embedded and indexed in a vector database for fast retrieval (approximate k nearest neighbour similarity search). Then, upon a spatial query, all related texts are first retrieved from the index based on their vector similarity with the query's embedding. The retrieved texts are added as context to the query and then the contextualized query is given as a prompt to an LLM for the response generation. An overview of this framework is illustrated in Figure 2.

3.1 $SpaTex$: Synthetic Spatial Text Generator

Spatial knowledge may contain various different aspects and metrics, such as the distance between entities, their topological relationships (e.g. adjacent, intersect) and the cardinal direction of an entity in relation to another one (e.g. north, southwest). We refer to any type of relation between two geographical entities as a *spatial relation*. To extract these spatial relations and generate meaningful synthetic spatial text that describes them comprehensively and concisely, we introduce $SpaTex$, a rule-based spatial-to-text data generator that takes as input spatial data collections in raw format (WKT, CSV etc.). The output text encapsulates in natural language the relations between (nearby) pairs of objects.

3.1.1 Spatial Relation Detection

For the detection of topological relations, we use the standard Dimensionally Extended 9-Intersection model (DE-9IM) (Clementini et al., 1993). DE-9IM defines a 3×3 matrix where the rows and columns represent two objects' interior, boundary and exterior areas. The combination of values in the matrix defines the exact topological relationship for two objects. Moreover, $SpaTex$ calculates the cardinal direction between nearby objects in relation to one another, as well as their in-between distance and their common area (if any) in square kilometres.

For two input spatial datasets R and S, $SpaTex$ performs a *spatial join* $R \bowtie S$ between them, an operation that identifies all pairs of objects $\langle (r, s) | r \in R, s \in S \rangle$ that intersect with each other. For each dataset, a *self-join* is performed ($R \bowtie R$ and $S \bowtie S$), to identify relations between objects in the same dataset as well.

The grand majority of object pairs in real-world spatial datasets are *disjoint* (Georgiadis and Mamoulis, 2023), so we only detect and generate non-disjoint topological relations, as disjointness can be implied. This saves us both the effort and the overhead of encoding and retrieving disjoint relations. In general, spatial relations between objects that are disjoint and far from each other can be inferred by LLMs and do not need to be explicitly defined in the context. For example, describing two entities as adjacent implies that their borders touch and thus, LLMs can infer that since they touch, they are not disjoint with each other.

SpaRAGi takes advantage of spatial inference as much as possible to reduce the volume of the generated text by $SpaTex$. To this end, we partition the data space using a uniform grid and assign each spatial entity to the partitions (i.e., tiles) that it spatially overlaps. $SpaTex$ then performs a partition-to-partition spatial join (Patel and DeWitt, 1996) for each cell; hence, we only compute and generate the spatial relations between objects of the same tile. For any pair of objects in a partition, we first compare their Minimum Bounding Rectangle (MBR(r)). If the MBRs do not intersect, then we only compute the relative cardinal direction between them (e.g., north of) and their distance; otherwise, we compute the DE-9IM matrix. For overlapping objects, we only generate the topological relation (e.g., overlaps, inside, covers); if the relation of the objects is adjacent, we also compute their cardinal direction relation.

The partitioning approach employed by $SpaTex$ has two advantages. First, we avoid computing an excessive (and redundant) number of spatial relations, which can be inferred; for two entities (e.g., counties) in different partitions, their relation should be disjoint and the cardinal direction relation can be inferred by the cardinal directions of entities that enclose them (e.g., states). Second, each partition is processed independently and in parallel, scaling up the relation generation process.

3.1.2 Text generation

The spatial-to-text translation rules are of great importance to our framework, as the output must be readable, properly formatted synthetic spatial text that is comprehensible by any LLM.

We explore various formats for $SpaTex$'s output, such as generating a single text per unique entity in the data or a separate snippet for each $\{subject, relation, object\}$ sentence. Another thing to consider is how much "flavour" text is necessary or preferred in the output. In this paper, we analyse and compare 3 approaches for the synthetic spatial text format:

- *Entity*: Grouping all spatial relations in a single text for each unique entity in the datasets.
- *Triplet*: We output all distinct spatial relations between two entities separately, phrased as plainly and simple as possible.
- *Rich-Triplet*: The relations are kept separately again, but each one is expressed using variant phrasing and richer vocabulary than the Triplet approach.

Each approach has its pros and cons, for example entity-based grouping generates fewer but larger texts than the other two approaches. If such a text is retrieved to be used as context for a query, it might contain irrelevant information, adding noise to the model during inference. On the other hand, both triplet-based approaches contain more but smaller texts, which increases the threshold for how many texts should be retrieved regarding a query, as the spatial knowledge for a specific entity is spread around in multiple texts. $SpaTex$'s text generation process for each format is illustrated in Figure 4. The previous stage of detecting the spatial relations of Figure 3 is common to all approaches.

3.2 Generated Spatial Datasets

We use the TIGER (SpatialHadoop, 2015) datasets for the States (50 entities), Counties (3225 entities) and Zip-codes (33144 entities) in the USA. We performed one self-join for each dataset as well as their cross-join, to capture all possible relations between any related Counties, States and Zip-codes. The overall time for the $SpaTex$ generation and the encoding of the produced texts in a commodity machine was less than 3 minutes. We generated the following synthetic spatial text datasets:

1. CSZe (36.4K entities): each text corresponds to a unique entity in the datasets exclusively. All of the entity's relations with other entities are contained in this text.

2. CSZt (487K entities): instead of being grouped by entity, relations are stored as separate texts in the triplet form $\{subject, relation, object\}$. Sentences are kept plain and simple.

3. CSZt-r (487K entities): this is a modified version of CSZt, but all of the texts have richer text, describing the same relations using more words and different phrasing.

CSZe has the fewest number of texts compared to the other two datasets; this results to lengthy texts that contain more words as shown in Table 1. CSZt and CSZt-r have the same number of texts, differentiating in the counted words and the length of the texts. Since CSZt-r is a phrase-enriched version of CSZt, each of its texts have greater length and are composed by more words on average. Notice that even though that CSZt-r incorporates more phrases, it still has in average fewer words and lower length per text compared to the CSZe dataset.

Dataset	Word count				Length			
	avg	min	max	std	avg	min	max	std
CSZe	235	28	60755	835	1459	166	326K	4762
CSZt	11.6	8	29	3.2	63.3	40	177	17.9
CSZt-r	65.2	40	118	8.2	385	233	777	45.4

Table 1: Statistics of the three generated datasets.

3.3 Retrieval for Spatial Inference

All generated texts are encoded to vectors through a pre-trained encoder. The embeddings are then added to a vector DB and indexed for fast retrieval. This way, the texts generated by $SpaTex$ are used in the model as context relevant to a given query. A spatial query q (e.g., Does Stanton County Nebraska contain Zip-code 68779?) passed to the model, is first embedded using the same text encoder we used to embed the texts. Then, through approximate k nearest neighbour (AkNN) similarity search, the k most relevant texts to the query are retrieved and added as context to it (Figure 1 shows query q with $k = 1$). The formatted prompt then is given to the model and it has a *Question* part and a *Question Context* part, so that the model can respond to the contextualized query in a single pass. A few examples of SpaRAGi's prompts are shown in Table 4 of the Appendix.

For high values of k the context may grow out of control. For example, dataset CSZe has an average word count of 235 in its texts. This may lead to large amounts of noise (i.e. information unrelated to the query) to be added as context for a query. Various mechanisms can be employed at this stage to filter out unnecessary information from the retrieved texts. For this preliminary analysis, we follow the naive approach of appending the retrieved texts as context in their entirety.

4 Experimental Analysis

Queries To assess the performance of SpaRAGi, we generated a query set with 1000 random spatial relation queries. To do so, we sampled random texts from our datasets and generated yes/no questions from them with 50-50 chance for each. For example, sampling the text "Stanton County Nebraska contains Zip-code 68779." can generate either the "Does Stanton County Nebraska contain Zip-code 68779?" query (yes) or the "Is Stanton County Nebraska inside of Zip-code 68779?" query (no). This query set was used in all of our experiments, regardless of which dataset was loaded for retrieval. Each generated query is accompanied by a 'yes' or 'no' answer that is used to measure the correctness of the responses, as well as the text ID from which the query originated which we call *ground truth*. We opted to run each query three times and the response with the highest occurrence

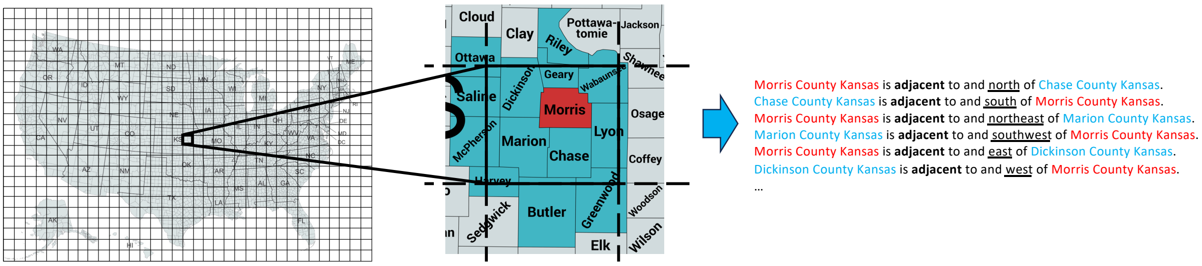


Figure 3: The Spatial relation identification process by $SpaTex$ that uses a global grid to group nearby entities and compute their spatial relations.

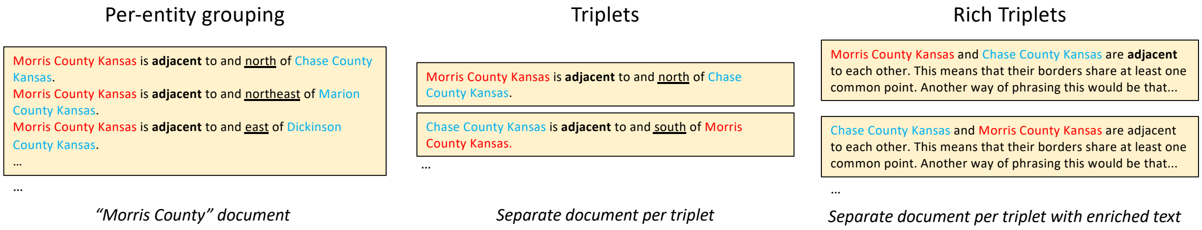


Figure 4: Synthetic spatial text generation process by $SpaTex$, generating from left to right: i) a separate text for each unique entity, grouping all of its relations together ii) a text per triplet $\{subject, relation, object\}$ in plain sentences and iii) a text per triplet but with each text enriched and the relation expressed using multiple phrasings.

frequency was selected as the final result.[5]

Embeddings & Indexing In our implementation, all dataset and query embeddings were created using the mixedbread-ai/mxbai-embed-large-v1 sentence embedder (Li and Li, 2023). We use FAISS (Johnson et al., 2019) to index the embeddings, which achieves a very good throughput in AkNN queries while preserving good retrieval accuracy.[6]

Models In all of our experiments, we use meta-llama/Llama3.1-8B-Instruct quantized to 4 bits and without any fine-tuning. We use an NVIDIA GeForce RTX 3060 with 12GB of memory for all of our experiments.

4.1 SpaRAGi Retrieval Evaluation

To measure SpaRAGi's retrieval accuracy, we test whether the ground truth of each query was retrieved for that query and if yes, with what rank among the top-k retrieved texts (i.e. rank of similarity). Note that during inference, even if the ground truth is not retrieved, a correct response to the query may be inferred from the rest of the retrieved texts. However, to benchmark the retrieval, we only take into account the ground truth and do not measure the rest of the retrieved texts' relativity to the query.

We perform each experiment for varying retrieval size k to thoroughly analyse its effect. To evaluate retrieval, we use the following measures:

- Mean Reciprocal Rank (MRR) evaluates the rank of the ground truth within the list of retrieved texts, calculated as the reciprocal of its rank and averaged across all queries.
- Precision-at-One (P@1) measures the proportion of queries for which the ground truth is retrieved as the top-ranked text, irrespective of the value of k.
- Success Rate (SR) indicates whether the ground truth was retrieved at all, without considering its rank in the results.
- Mean Rank (MR) computes the average rank of the ground truth text across all queries where it was successfully retrieved, focusing only on successful retrievals.

Figure 5 analyses SpaRAGi's retrieval accuracy for each dataset. Specifically, Figures 5a and 5b showcase that MRR and P@1 remain unaffected by the increasing value of k. This is the default when measuring P@1, whilst a steady MRR indicates that the rank of the ground truth does not necessarily change much in the list of the retrieved texts as k increases. This indicates that the correct text is either retrieved at the highest rank or not retrieved at all (for $k = 10$). In both metrics, CSZt-r performs the best, exceeding CSZt by approximately 0.2 and CSZe by even more.

[5] Running the query set three times takes 1 hour on an NVIDIA GeForce RTX 3060, 30 minutes on an A100, and 10 minutes on an H200 on average for each model.

[6] The retrieval cost of FAISS for k up to 10 was 10-15ms.

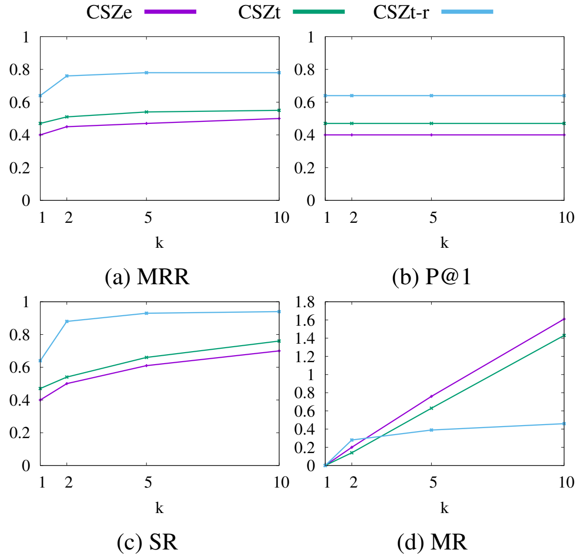


Figure 5: MRR (a), P@1 (b), Success Rate (c), MR (d) of SpaRAGi's retrieved texts per dataset and k.

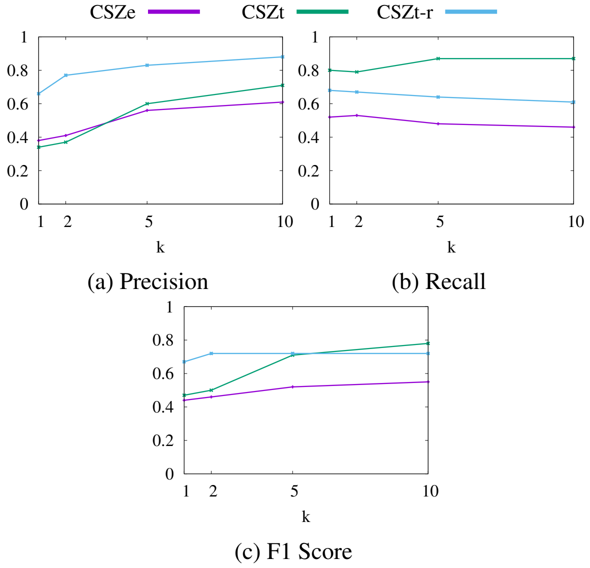


Figure 6: Classification performance of SpaRAGi's generated responses (yes/no) per dataset and k.

In Figure 5c, SR increases with k for all three datasets, which is expected as more texts are retrieved and by extension, it is more possible that the ground truth is retrieved among them. CSZt-r achieves the highest success overall, with its SR reaching 94% for $k = 10$ while CSZt peaks at 76% and CSZe at 70%. Note that CSZt-r reaches a high success rate (88%) very fast for $k = 2$, while for the other two datasets SR gradually improves with k. This suggests that a low value of k is sufficient for dataset CSZt-r to retrieve the ground truth, which can help minimize context noise by avoiding the retrieval of less relevant or unrelated texts.

Similarly, Figure 5d shows the average rank of the ground truth in the retrieved texts (if it exists in the list) growing with k. A low MR indicates that the ground truth can be successfully retrieved with a small value of k. However, as k increases, the ground truth is retrieved in more cases, leading to an increase in the MR. The MR of the ground truth converges quickly for CSZt-r, reinforcing the assertion that a relatively low k is sufficient to achieve high retrieval accuracy in CSZt-r.

In summary, all metrics confirm that CSZt-r has the best retrieval accuracy among the datasets we used in our experiments. On the other hand, CSZe's per-entity compression of all related relations performs the worst in terms of retrieval, indicating that its texts' embeddings are distorted by noise and affect the ground truth similarity search negatively.

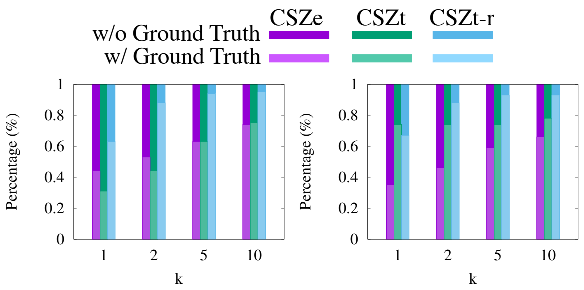
4.2 SpaRAGi Generation Evaluation

To assess SpaRAGi's performance in successfully responding to spatial queries, we perform a binary classification task on the generated responses. A response to a query is considered *correct* if it matches the query's *correct answer* (yes or no), otherwise it is considered *incorrect*. Example prompts, along with their responses and their evaluation, are shown in Table 4 of the Appendix.

We measured the classification performance using Precision, Recall and F1 score for increasing k, shown in Figures 6a, 6b and 6c, respectively. Note that for all datasets, the Precision is gradually increases k, which means reduction of false positives (i.e. queries whose correct answer is 'no' and are answered as 'yes') as more texts are retrieved. This increase of Precision, combined with the relatively steady MRR of Figure 5a, shows that the additional, seemingly unrelated, texts that are being retrieved as k increases, actually contribute positively when added as context, assisting the model into responding correctly to more queries.

As concluded in §4.1, CSZe performed the worst in terms of retrieval accuracy among our datasets. This is mirrored in CSZe's generation evaluation as well, performing worse than the rest of the datasets in terms of Precision, Recall and F1 score.

Even though CSZt-r has the best retrieval accuracy, it is eventually outperformed in response generation by CSZt for high k. This is correlated with CSZt's high Recall (i.e., fewer cases of respond-



(a) Correct response cases (b) Incorrect response cases

Figure 7: Proportional breakdown (%) of correct and incorrect response cases, based on whether the ground truth was *retrieved* (light-coloured stacked bars) or *not retrieved* (dark-coloured stacked bars).

ing 'no' to queries whose correct answer is 'yes'), combined with its relatively good Precision. The high Recall can be attributed to CSZt's plain and simple sentences, with little noise that might mislead the generation. On the other hand, CSZt-r's Recall drops as k increases; the noise of the richer text, sometimes negatively affects generation. Although CSZt-r quickly reaches a high F1 score, CSZt eventually outperforms CSZt-r for $k = 10$.

To assess *how much* the retrieved texts assisted the model in responding correctly to the queries, in Figure 7, we study the correlation between the ground truth retrieval and the correctness of the response. Specifically, for queries where the model responded *correctly* (Figure 7a), we separate cases where the ground truth was successfully retrieved (light-coloured bars) to those where the ground truth is missing from the context (dark-coloured bars). Respectively, we perform the same for the queries to which the model responded *incorrectly*, shown in Figure 7b. We observe in the correct response cases that the phrase-enriched per-triplet dataset (CSZt-r) benefits to a greater extent than the other two datasets, as in every case the proportional percentage of the contained ground truth is higher and increasing with k. The same is observed for the incorrect response cases, but for k higher than 1, indicating that even with the ground truth as context, the model can still infer an incorrect response to certain queries. Furthermore, both the SR of retrieval (Figure 5c) and the F1 score of the generation (Figure 6c) are increased with k, verifying that in general, the added context to the query helps to improve its generation for the queries.

4.3 Model Comparison

Table 2 performs a baseline comparison between various models on our spatial queries, to identify which model has the best out-of-the-box performance, measured by their F1 scores. All models ran without SpaRAGi, on a A100 GPU, with the exception of meta-llama/Llama3.1-70B-Instruct which we ran on a H200 due to its large memory requirement. Llama3.1-70B-Instruct, the largest model we evaluated, achieved the best performance among all models, with Llama3.1-8B-Instruct following closely in second place.

Model	# of Parameters	F1 score
mistralai/Mistral-7B-Instruct-v0.1	7B	0.45
ibm-granite/granite3.2-8b-instruct-preview	8B	0.19
meta-llama/Llama3.1-8B-Instruct	8B	0.58
mistralai/Ministral-8B-Instruct-2410	8B	0.35
mistralai/Mistral-Nemo-Instruct-2407	12.2B	0.18
microsoft/phi-4	14.7B	0.44
meta-llama/Llama3.1-70B-Instruct	70B	0.61

Table 2: The models that we tested on our query set and how they performed in our response generation benchmark based on their F1 scores.

Framework	F1 score
Llama-8B	0.58
Llama-8B + SpaRAGi (CSZt)	0.78
Llama-70B	0.61
Llama-70B + SpaRAGi (CSZt)	0.91

Table 3: SpaRAGi's response generation improvement (in terms of F1 score) on small (Llama3.1-8B-Instruct) and relatively large (Llama3.1-70B-Instruct) models for our query set. SpaRAGi was deployed using the CSZt dataset and $k = 10$.

As seen in Table 3, SpaRAGi employed on a small model like Llama3.1-8B-Instruct and with the CSZt dataset as context, outperforms the significantly bigger Llama3.1-70B-Instruct in terms of response generation accuracy by 0.17. When combined with Llama3.1-70B-Instruct, SpaRAGi improved its performance by 0.3, increasing its F1 score to 0.91 for our spatial queries.

5 Conclusions

This study presented SpaRAGi, a novel approach for generating synthetic spatial text and assisting large language models in answering spatial queries through retrieval-augmented generation. Our experimental analysis shows that employing SpaRAGi on models (small or large), leads to improving their response generation for spatial queries by 35% to almost 50%. The ultimate goal of this work is to study the spatial inference capabilities of LLMs on open-ended spatial questions rather than yes/no queries. In the future, we will explore how can RAG facilitate better spatially-informed discussion between the user and the model in natural language.

Limitations

This preliminary version of SpaRAGi has the following limitations: 1) due to resource limitations, we were unable to perform most of our experiments on large models that require high-end GPUs to run. However, we included one large model (meta-llama/Llama3.1-70B-Instruct) to support our claim that SpaRAGi helps smaller models match or surpass large models in terms of spatial inference. 2) the synthetic spatial text generation is not automated in terms of spatial data retrieval. This means that spatial datasets need to be manually collected and then pre-processed by our $SpaTex$ generator to generate the synthetic spatial text datasets that are actually used in the RAG mechanism. Additionally, many publicly available real-world spatial datasets lack metadata (name, description etc.), which creates the need for some data curation before being able to be used.

References

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17754–17762. AAAI Press.

Eliseo Clementini, Paolino Di Felice, and Peter van Oosterom. 1993. A small set of formal topological relationships suitable for end-user interaction. In *SSD*, Lecture Notes in Computer Science. Springer.

Mark de Berg. 1997. *Computational geometry: algorithms and applications*. Springer.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on RAG meeting llms: Towards retrieval-augmented large language models. In *ACM SIGKDD*.

Robert Friel, Masha Belyi, and Atindriyo Sanyal. 2024. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *CoRR*, abs/2407.11005.

Thanasis Georgiadis and Nikos Mamoulis. 2023. Raster intervals: An approximation technique for polygon intersection joins. In *ACM SIGMOD*.

Yulong Hui, Yao Lu, and Huanchen Zhang. 2024. UDA: A benchmark suite for retrieval augmented generation in real-world document analysis. *CoRR*, abs/2406.15187.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.

Zhenlong Li and Huan Ning. 2023. Autonomous GIS: the next-generation ai-powered GIS. *Int. J. Digit. Earth*.

Rohin Manvi, Samar Khanna, Gengchen Mai, Marshall Burke, David B. Lobell, and Stefano Ermon. 2024. Geollm: Extracting geospatial knowledge from large language models. In *ICLR*. OpenReview.net.

Jignesh M. Patel and David J. DeWitt. 1996. Partition based spatial-merge join. In *ACM SIGMOD*.

Nicholas Pipitone and Ghita Houir Alami. 2024. Legalbench-rag: A benchmark for retrieval-augmented generation in the legal domain. *CoRR*, abs/2408.10343.

Jianzhong Qi, Zuqing Li, and Egemen Tanin. 2023. Maasdb: Spatial databases in the era of large language models (vision paper). In *ACM SIGSPATIAL*.

David Rau, Hervé Déjean, Nadezhda Chirkova, Thibault Formal, Shuai Wang, Vassilina Nikoulina, and Stéphane Clinchant. 2024. BERGEN: A benchmarking library for retrieval-augmented generation. *CoRR*, abs/2407.01102.

Simranjit Singh, Michael Fore, and Dimitrios Stamoulis. 2024. Geollm-engine: A realistic environment for building geospatial copilots. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024 - Workshops, Seattle, WA, USA, June 17-18, 2024*, pages 585–594. IEEE.

SpatialHadoop. 2015. *TIGER datasets*.

Yixuan Tang and Yi Yang. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *CoRR*, abs/2401.15391.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal

Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking retrieval-augmented generation for medicine. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 6233–6251. Association for Computational Linguistics.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen-tau Yih, and Xin Luna Dong. 2024. CRAG - comprehensive RAG benchmark. *CoRR*, abs/2406.04744.

Yifan Zhang, Cheng Wei, Shangyou Wu, Zhengting He, and Wenhao Yu. 2023. Geogpt: Understanding and processing geospatial tasks through an autonomous GPT. *CoRR*, abs/2307.07930.

Appendix

Query	Prompt	Response	Evaluation
Is Carroll County Maryland southwest of Zipcode 17349? Respond with yes or no. Do NOT provide an explanation.	Question: Is Carroll County Maryland southwest of Zipcode 17349? Respond with yes or no. Do NOT provide an explanation. Question Context: Carroll County Maryland is southwest of Zipcode 17349. Additionally Carroll County Maryland is approximately 36.259572 kilometers away from Zipcode 17349 to the southwest. This means that Carroll County Maryland and Zipcode 17349 do not share a border or have common area. Topologically it is the same to say that Zipcode 17349 is to the northeast of Carroll County Maryland.	yes	TP
Is Zipcode 08042 adjacent to and south of Zipcode 08068? Respond with yes or no. Do NOT provide an explanation.	Question: Is Zipcode 08042 adjacent to and south of Zipcode 08068? Respond with yes or no. Do NOT provide an explanation. Question Context: Zipcode 08068 and Zipcode 08042 are adjacent to each other. This means that their borders share at least one common point. Another way of phrasing this would be that Zipcode 08068 and Zipcode 08042 spatially meet with each other touch or that they are neighbors. Additionally Zipcode 08068 is south of Zipcode 08042. It is the same to say that Zipcode 08042 is to the north of Zipcode 08068.	no	TN

Table 4: Prompting and response examples for various queries on SpaRAGi, on Llama3.1-8B-Instruct using the CSZt dataset and $k = 1$. The Query is what is asked by the user. The Prompt is what SpaRAGi generates as the contextualized prompt, after the retrieval is finished. The response is the model's response for the Query. In the Evaluation column we show whether the Response is correct (True Positive or True Negative) or incorrect (False Positive or False Negative).