SpatialWebAgent: Leveraging Large Language Models for Automated Spatial Information Extraction and Map Grounding

Shunfeng Zheng[1], Meng Fang[2], Ling Chen[1]
[1]University of Technology Sydney, New South Wales, Australia
[2]University of Liverpool, Liverpool, the United Kingdom
Shunfeng.Zheng@student.uts.edu.au, Meng.Fang@liverpool.ac.uk, Ling.Chen@uts.edu.au

Abstract

Understanding and extracting spatial information from text is vital for a wide range of applications, including geographic information systems (GIS), smart cities, disaster prevention, and logistics planning. This capability empowers decision-makers to gain crucial insights into geographic distributions and trends. However, the inherent complexity of geographic expressions in natural language presents significant hurdles for traditional extraction methods. These challenges stem from variations in place names, vague directional cues, and implicit spatial relationships. To address these challenges, we introduce SpatialWebAgent, an automated agent system that leverages large language models (LLMs). SpatialWebAgent is designed to extract, standardize, and ground spatial information from natural language text directly onto maps. Our system excels at handling the diverse and often ambiguous nature of geographic expressions—from varying place names and vague directions to implicit spatial relationships that demand flexible combinations of localization functions—by tapping into the powerful geospatial reasoning capabilities of LLMs. SpatialWebAgent employs a series of specialized tools to convert this extracted information into precise coordinates, which are then visualized on interactive maps. A demonstration of SpatialWebAgent is available at https://sites.google.com/view/SpatialWebAgent.

1 Introduction

The ability to extract spatial information from natural language is fundamental across diverse fields such as geographic information systems (GIS), smart city planning, disaster management, and logistics. Recent advancements in natural language processing (NLP), particularly with the emergence of LLMs (Brown et al., 2020), have revolutionized how we approach tasks like text comprehension, information extraction, and automated reasoning. Trained on vast datasets, these models excel at understanding and processing natural language. The transformation of free-form text into structured geographic entities is crucial for accurate spatial analysis, real-time event monitoring, and optimized resource allocation (Li et al., 2021). By leveraging LLMs, we can automate this extraction process, empowering decision-makers to derive valuable insights from unstructured data and make timely, informed choices (Gao et al., 2022).

However, geospatial reasoning presents a formidable challenge due to the inherent diversity and ambiguity within geographic expressions. This includes variations in place names (e.g., "New York," "NYC," "The Big Apple"), vague directional phrases (e.g., "nearby," "north of"), and complex implicit spatial relationships (Zhang et al., 2020; Goodchild and Li, 2021; Gritta et al., 2018). For instance, phrases like "the café next to the school" require not only entity extraction but also an understanding of their relative positioning (Yin et al., 2021). Traditional methods, such as rule-based systems and classical NLP techniques, often struggle with this complexity (Bommasani et al., 2021), exhibiting limited generalization across varied formats and contexts (Leidner and Lieberman, 2011). While these methods primarily focus on extracting fixed spatial entities, they frequently fail to capture the relationships between them (Gelernter and Balaji, 2013). Furthermore, their reliance on explicitly structured or well-formed input renders them less robust when dealing with informal or intricate spatial descriptions common in real-world text (Karimzadeh, 2018). A critical limitation is their inability to resolve place name ambiguities; many locations share identical names, such as "Burwood" (found in both Sydney and Melbourne) or "Victoria Harbour" (present in multiple countries). Although some approaches (Syed et al., 2024) attempt to refine geographic references by extracting place

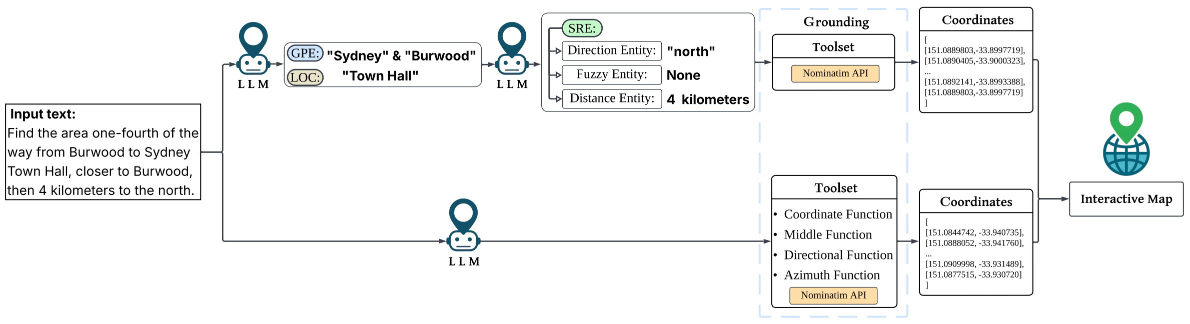


Figure 1: Overall SpatialWebAgent workflow: The top panel illustrates extraction of GPEs/LOCs and their associated SREs to compute spatial relationships, followed by coordinate generation via a specialised toolset. The bottom panel shows how the agent leverages LLM geospatial reasoning with the toolset to resolve complex spatial logic and map the resulting coordinates.

names via NLP and querying geocoding APIs, they often default to the first API result when faced with ambiguities, compromising accuracy and leading to incorrect spatial interpretations.

To overcome these challenges, we introduce SpatialWebAgent, an agent system that leverages Large Language Models (LLMs) to automatically identify, extract, standardize, and ground spatial information from text, as shown in Figure 1. SpatialWebAgent combines the advanced geospatial reasoning capabilities of LLMs with a suite of specialized tools. This creates a robust, fully automated system capable of processing complex geographic information, including the implicit spatial relationships often found in natural language. Specifically, our agent system first extracts spatial entities from the text. It then autonomously applies various tools to convert these entities into precise coordinates, and finally visualizes the results on interactive maps. This seamless transformation from unstructured geographic text to visualized spatial data significantly simplifies tasks such as location interpretation, geographic query resolution, and the development of downstream applications.

Our contributions are as follows:

- **Automated Spatial Agent System:** We introduce SpatialWebAgent, a novel pipeline that seamlessly transforms natural language into structured spatial data and interactively visualizes it, streamlining the entire process from extraction to map-based representation.

- **Extensive Empirical Evaluation:** We conducted a rigorous empirical evaluation, performing experiments on both entity extraction datasets and a specialized tool dataset. We assessed several LLMs to evaluate their ability to infer coordinates from diverse natural language descriptions, providing insights into their performance on complex spatial understanding tasks.

- **Interactive Web Prototype:** We developed a web prototype of SpatialWebAgent, demonstrating its practical utility in processing natural language queries and visualizing spatial information.

2 SpatialWebAgent

SpatialWebAgent leverages Large Language Models (LLMs) to accurately extract spatial information from text and visualize it on maps. The system's workflow involves two primary modules: the Spatial Entity Extraction Module and the Spatial Grounding Module.

2.1 Spatial Entity Extraction Module

This module is responsible for identifying and categorizing location-based entities within natural language text. SpatialWebAgent extracts three types of entities:

- Geopolitical Entities (GPEs): Identifiable locations with geopolitical relevance, such such as countries, cities, or administrative regions (e.g., "France," "Sydney").

- Location Entities (LOCs): Physical landmarks or geographic features anchored to fixed reference points (e.g., "Eiffel Tower," "Sydney Opera House").

- Spatial Relation Entities (SREs): Descriptions of spatial relationships based on GPEs and LOCs, including directional cues (e.g., "north," "second street on the left"), distance-based terms (e.g., "10 kilometers away"), and vague expressions (e.g., "nearby," "border").

To accurately extract these entities, we designed an LLM-based pipeline using carefully crafted prompts (detailed in Appendix B). The process begins by extracting GPEs and LOCs. These are then used to decompose the query into smaller clauses, which are further processed to extract SREs through three additional steps: identifying directional expressions, distance terms, and fuzzy spatial references.

For an SRE to be uniquely localizable, both orientation and distance must be present. Cases involving incomplete spatial information, such as directional-only descriptions, are handled by the Spatial Grounding Module as discussed in Section 2.2. This multi-stage extraction process enables our system to robustly handle both explicit and implicit spatial references.

2.2 Spatial Grounding Module

The Spatial Grounding Module is designed to interpret and resolve intricate spatial descriptions, converting extracted spatial entities into precise geographic coordinates and ultimately visualizing them. This module comprises two key sub-sections: Entity Grounding and Advanced Grounding.

2.2.1 Entity Grounding

To compute and visualize spatial information, SpatialWebAgent encodes GPEs, LOCs, and SREs into geographic coordinates.

For GPEs and LOCs, we use the Nominatim API (OpenStreetMap contributors, 2024) to retrieve their coordinates. If the API returns multiple candidate locations, the system prompts the LLM to disambiguate based on the administrative region associated with each location. Additionally, all candidate locations are listed on a dedicated localization page, allowing for secondary human confirmation if needed.

For SREs, once the coordinates of their corresponding GPEs or LOCs are determined, we compute new spatial coordinates using the extracted relative relationships (e.g., cardinal/ordinal directions, distance-related expressions). In scenarios where information is incomplete or ambiguous, the system employs specific fallback grounding strategies:

- If only an orientation is provided (e.g., "south-west of X"), we represent the location as a directional arrow originating from the reference point.

- If only a distance is mentioned (e.g., "10 km from X"), we render a circular region centered at the GPE/LOC with the specified radius.

- When both orientation and distance are present, we combine them to infer a more precise region.

Following the previous work (Syed et al., 2024), we adopt a graphical slicing method to present a comprehensive and interpretable visual profile of the spatial data. We retain the design choice of visualizing directional areas based on projected contours, as it aligns well with human intuition and preserves semantic coherence. The final results, including both the extracted GPE/LOC coordinates and the inferred SRE locations, are visualized on interactive maps using OpenStreetMap, providing users with an intuitive exploration of spatial relationships (an example is shown in Appendix D).

2.2.2 Advanced Grounding

To address complex and multi-layered geographic expressions in natural language, we enhance the Spatial Grounding Module with a specialized toolset featuring four core localization functions that support compositional spatial reasoning in LLMs.

This toolset offers two key advantages: First, it enables the system to autonomously identify and handle a wide variety of spatial relation patterns described in natural language, including directional, distance-based, and complex compositional relationships. Second, it allows the LLM to translate its semantic understanding of location into precise geographic coordinates, which can then be visualized and interpreted by subsequent processes. The LLM performs Chain-of-Thought (CoT) reasoning, then selects and composes appropriate functions from this toolset based on the input text, ultimately outputting a set of target geographic coordinates representing the region of interest.

We define the following four localization functions within the toolset:

1. **Coordinate Function**: Takes a specified location entity (GPE or LOC) as input and outputs its corresponding geographic coordinates.

2. **Directional Localization Function**: Given a location's coordinates, a specified direction, and a distance, this function calculates the coordinates of the area located in the given direction at the specified distance from the initial location.

3. **Middle Localization Function**: Computes the coordinates of the region situated between two given locations based on their respective coordinates.

4. **Azimuth Localization Function**: Similar to the Directional Localization Function, but utilizes a precise bearing angle as input to determine the target area's coordinates.

Except for the Coordinate Function, which outputs the actual boundary of a specific region, the other three functions project a circular area from the centroid of the result location. The area of this projected circle is dynamically adjusted to equal that of the original basic location—or the average of multiple input locations—so as to better reflect human common-sense reasoning about spatial extent.For instance, if users mention "the western part of a specific district," the resulting circular highlight will have the same area as that district. If the query refers to a location between two cities, the system computes the highlight area as the average of the two.

By prompting the LLM to integrate these four functions along with the extracted entities, it can effectively select and compose the appropriate operations to localize regions described in complex scenarios. For example, to determine the coordinates of the phrase `'An area located 4 kilometers west between Loc_A and Loc_B'`, the following steps are taken:

1. The *Coordinate Function (Loc_A)* and *Coordinate Function (Loc_B)* are used to retrieve the coordinates of "Loc_A" and "Loc_B".

2. The *Middle Localization Function* (step 1) is applied to compute the midpoint between the two locations.

3. The *Directional Localization Function* (step 2, west, 4 km) is then used with the midpoint as a reference, incorporating the direction westward and a distance of 4 km to infer the target region.

LLMs autonomously determine the selection, order, and input parameters for these functions, facilitating accurate geospatial localization. Once the target region's coordinates are identified, they can be visualized using platforms such as OpenStreetMap. An example of the final output is provided in Appendix D.

3 Experiments

3.1 Entity Extraction Evaluation

This subsection is for extracting geographic information from informal text, we utilize a series of datasets that cover different aspects of geographic entity recognition. To ensure a comprehensive evaluation, we select four diverse datasets.

CoNLL-03 and OntoNotes 5.0: These two datasets are foundational for recognizing GPEs and LOCs, including countries, cities, and regions globally. However, both datasets are limited to only recognizing GPEs and LOCs types of geographic entities, leaving more complex spatial relationships unaddressed. They provide a strong baseline for recognizing geographic names but do not capture the full diversity of spatial expressions present in informal text (Tjong Kim Sang and De Meulder, 2003; Schweter and Akbik, 2020).

HarveyNER: The HarveyNER dataset, a newer and more comprehensive geographic NER resource, expands on this by incorporating both standard and relative entity, thus capturing more complex geographic expressions, including long and intricate location mentions often found in informal text (Chen et al., 2022).

		Standard Entity		Relative Entity	Total
LLM Model	**Dataset**	**GPE (%)**	**LOC (%)**	**SRE (%)**	**Total (%)**
Llama-3-8B	CoNLL-03	12.3	17.8	–	11.9
	OntoNotes 5.0	15.5	18.0	–	14.9
	HarveyNER	15.5	18.0	11.3	10.9
	PADI-web	16.0	17.5	13.8	10.9
Mistral-7B-0.3	CoNLL-03	33.3	37.3	–	32.9
	OntoNotes 5.0	34.8	38.0	–	24.1
	HarveyNER	34.8	38.0	28.8	21.9
	PADI-web	33.5	36.3	30.3	21.5
Gemma-2-10B	CoNLL-03	35.0	33.3	–	32.6
	OntoNotes 5.0	36.0	34.5	–	31.9
	HarveyNER	36.0	34.5	27.5	20.0
	PADI-web	35.3	33.8	29.0	22.9
GPT-4o	CoNLL-03	92.3	91.5	–	79.9
	OntoNotes 5.0	90.3	93.8	–	84.6
	HarveyNER	88.0	89.8	80.8	77.3
	PADI-web	89.5	84.0	82.3	77.4
Gemini Pro	CoNLL-03	80.5	80.0	–	79.5
	OntoNotes 5.0	82.0	80.3	–	80.0
	HarveyNER	83.8	80.3	66.5	66.4
	PADI-web	84.0	80.0	67.0	66.3
DeepSeek-R1	CoNLL-03	74.5	80.0	–	73.4
	OntoNotes 5.0	75.0	82.3	–	81.1
	HarveyNER	73.8	83.0	62.0	61.9
	PADI-web	76.0	83.3	63.8	60.4

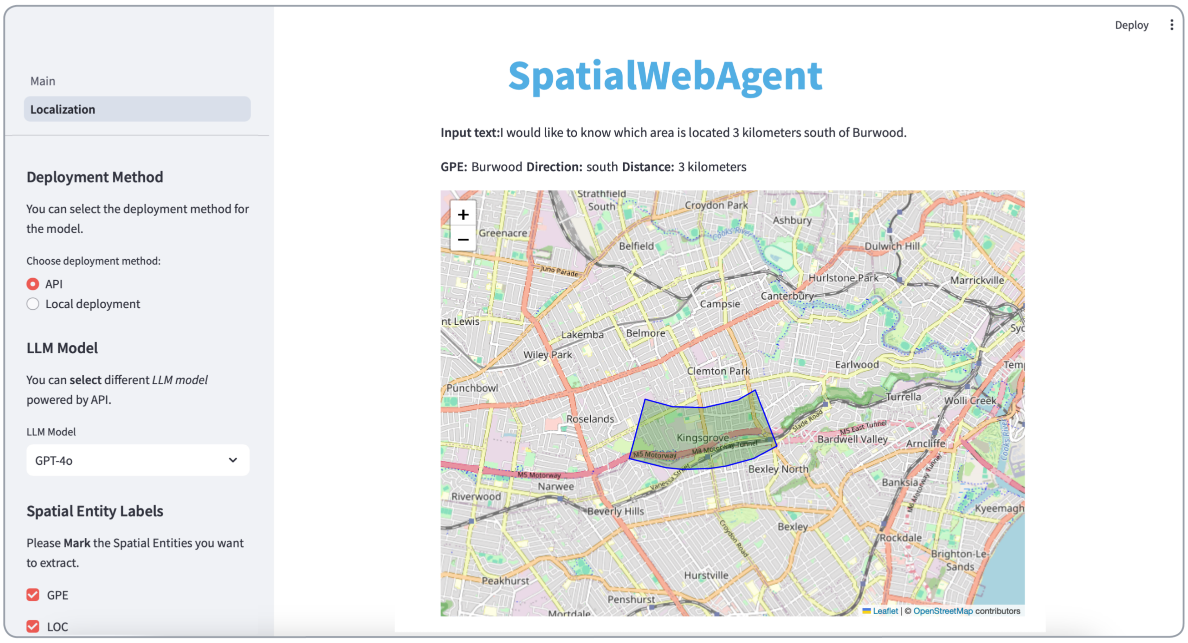
Table 1: Entity Extraction Evaluation: Performance of Different LLMs on Entity Recognition Across Various Datasets. The evaluation of **GPE** and **LOC** entities requires the LLM to accurately extract all entities for each data point to be considered correct. For **SRE**, the LLM only needs to correctly extract the SRE-related entities without the need to verify their corresponding standard entity. The Total score reflects the ability of the model to correctly extract all relevant entity types from the entire query.

PADI-web: A collection of natural language related to animal diseases . This dataset serves as a benchmark for the previous work (Syed et al., 2024), providing a comprehensive resource for evaluating models focused on animal health information. The PADI-web dataset contains a wide range of textual data, including descriptions of various animal diseases, symptoms, treatment methods, and geographical distributions. (PADI-web, 2023).

For each dataset, we sample 350 positive examples per entity type plus 50 negative examples, yielding 400 test instances per category.

We compare six LLMs—Llama-3-8B (Meta, 2024), Mistral-7B-0.3 (Jiang et al., 2023), Gemma-2-10B (Team, 2024b), GPT-4o (OpenAI, 2024), Gemini Pro (Team, 2024a), and DeepSeek-R1 (DeepSeek-AI, 2025).

The results are shown in Table 1. The closed-source models: GPT-4o, DeepSeek-R1, and Gemini Pro, demonstrated strong accuracy, with GPT-4o performing the best, which shows that the closed-source models were generally able to accurately recognize the corresponding entity types in most normally expressed language inputs. However, certain special text formats still led to recognition errors. For more details, please refer to Appendix C.


Deploy

Main

Localization

Deployment Method

You can select the deployment method for the model.

Choose deployment method:

[✓ API]

Local deployment

LLM Model

You can **select** different *LLM model* powered by API.

LLM Model

GPT-4o

Spatial Entity Labels

Please **Mark** the Spatial Entities you want to extract.

[✓ GPE]

[✓ LOC]

SpatialWebAgent

Input text:I would like to know which area is located 3 kilometers south of Burwood.

GPE: Burwood **Direction:** south **Distance:** 3 kilometers

Figure 2: The screenshot displays SpatialWebAgent in action. Given the input *"I would like to know which area is located* ***3 kilometers south*** *of* ***Burwood****."*, the system identifies the GPE *Burwood* and the SRE *3 kilometers south*, and computes the final location using a geographic function.

3.2 Spatial Grounding Evaluation

We evaluate the ability of LLMs to compose spatial functions from our toolset and resolve complex, multi-step geospatial instructions. As no existing benchmark addresses this task, we constructed a dataset of 100 challenging queries:

- Five trained annotators created the 100 queries and drafted matching natural-language instructions along with their ground-truth function sequences.
- Each example was reviewed by at least three different annotators, with any discrepancies resolved by consensus (see Appendix E).

Listing 1: An example of function composition in our benchmark dataset.

```
{
"index": 12,
"instruction": "Find the area one-fourth
     of the way from Burwood to Sydney
    Town Hall, closer to Burwood.",
"steps": [
{"id": 1, "function": "Between", "inputs
    ": ["Burwood", "Sydney Town Hall"]},
{"id": 2, "function": "Between", "inputs
    ": [1, "Burwood"]}
]
```

An example of one data point is illustrated in Listing 1. To ensure the quality of the dataset, we followed a set of annotation guidelines, which are detailed in Appendix E.

Model	Total Accuracy (%)
GPT-4o	87
DeepSeek-R1	83
Gemini-Pro	80
Llama-3-8B	5
Mistral-7B-0.3	7

Table 2: Evaluation of the proposed Hierarchical Geometric Function Localization method on various LLMs. This is the first attempt to integrate fine-grained geospatial reasoning capabilities into LLMs via function composition. We adopt a one-shot in-context learning setting to guide function selection and reasoning.

We evaluated different LLMs using this dataset to assess the models' capabilities in handling complex geospatial reasoning, the results are presented in Table 2. Given the challenging nature of the task, which involves multi-step spatial function interpretation, reliable performance was observed only in strong open-source models known for their reasoning abilities. Among them, GPT-4o achieved the best performance with an accuracy of 87%. Overall, all three models—GPT-4o, DeepSeek-R1, and Gemini-Pro—demonstrated strong capabilities and are well-suited for handling hierarchical geospatial function interpretation tasks.

In contrast, small-parameter open-source models showed unsatisfactory performance. While they can mimic the output format based on in-context examples, they are only capable of handling very simple logic. When faced with multi-layered reasoning or ambiguous questions, they often fail. Advanced closed-source models, however, are able to recognize subtle reasoning detours to generate the final answers. For example, in Listing 1, these models can infer that the final location lies between 'Burwood' and the output of step 1. In comparison, small open-source models tend to produce rigid outputs such as "id": 2, "function": "Relative", "inputs": [1, "east", "d/4"], without successfully reasoning through the correct logic.

3.3 Web Prototype

Our SpatialWebAgent web prototype—built with Streamlit—offers an intuitive, interactive environment for extracting and visualizing geographic information directly from user queries, shown in Figure 2. In the central panel, users enter free-form text containing spatial descriptions. The left-hand sidebar provides filter options for selecting categories of interest such as geopolitical entities (GPE), generic locations (LOC) or specific spatial relations (e.g., SRE, RSE). Once the user confirms these selections, the system invokes the corresponding extraction modules, automatically identifies the requested entities and plots them on an embedded map. For example, when a query specifies "the area located 3 km south of the Burwood district," SpatialWebAgent computes the target coordinates and highlights that zone in real time.

4 Conclusion

In this paper, we introduced SpatialWebAgent, an automated agent system designed to extract geographic named entities from natural language queries and pinpoint their precise spatial locations. Our system accomplishes this by skillfully combining the geospatial reasoning capabilities of Large Language Models (LLMs) with specialized tools, ultimately grounding and visualizing these results on interactive maps. SpatialWebAgent effectively tackles the challenge posed by ambiguous and complex spatial expressions within text. It first accurately identifies diverse spatial entities from natural language inputs. Then, by intelligently integrating a suite of spatial localization tools, the system precisely infers geographic coordinates, even for intricate or abstract spatial scenarios. These inferred coordinates are then seamlessly grounded and displayed on interactive maps, effectively bridging the gap between unstructured text and structured geographic data. This work highlights a promising direction for automated geospatial analysis systems and suggests significant potential applications across GIS, location-based services, and advanced spatial data processing.

Limitations

The model's accuracy in extracting SRE entities requires improvement, as the three subcategories of SRE are prone to confusion by the model. This issue leads to errors in recognizing and distinguishing between various spatial relationships, which hinders overall performance. Additionally, the model's performance is suboptimal in low-parameter open-source models. While SpatialWebAgent can operate effectively, achieving high accuracy still necessitates the use of closed-source model APIs.

Ethical Considerations

This work involves automated extraction and geocoding of location references from unstructured text using LLMs. The system may process data that includes names of geographic locations, organizations, or individuals. To mitigate privacy concerns, we only use publicly available and non-sensitive textual inputs. No personally identifiable information (PII) is collected, stored, or used in this work.

The mapping component relies on third-party geocoding APIs (e.g., Nominatim), which may return ambiguous or multiple candidate locations. While our system includes mechanisms to prompt LLMs for disambiguation and allow user confirmation, geocoding errors could still lead to misrepresentation of spatial intent. We encourage cautious interpretation when using this system for high-stakes applications.

Additionally, the use of pretrained language models may inadvertently reproduce geographic, cultural, or geopolitical biases present in the training data. Future work will focus on bias mitigation, fairness-aware prompting, and increased transparency in location reasoning.

References

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, and 1 others. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Pei Chen, Haotian Xu, Cheng Zhang, and Ruihong Huang. 2022. Crossroads, buildings and neighborhoods: A dataset for fine-grained location recognition. In *NAACL*. Association for Computational Linguistics.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jing Gao, Tao Zhang, and Yan Liu. 2022. Deep learning for spatial entity recognition in text. *Computers, Environment and Urban Systems*, 92:101745.

Joel Gelernter and Saket Balaji. 2013. An algorithm for local geoparsing of microtext. *GeoInformatica*, 17(4):635–667.

Michael F. Goodchild and Wenwen Li. 2021. Gis and natural language processing: A new perspective. *Annals of GIS*, 27(1):1–13.

Milo Gritta, Mohammad Taher Pilehvar, Nut Limsopatham, and Nigel Collier. 2018. What's missing in geographical parsing? *Proceedings of ACL*, pages 1235–1246.

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. *Preprint*, arXiv:2310.06825.

Manouchehr Karimzadeh. 2018. Geoai: Geospatial artificial intelligence for geographic knowledge discovery. *Advances in Geographic Information Science*, pages 1–18.

Jochen L. Leidner and Michael D. Lieberman. 2011. Detecting geographical references in the form of place names and spatial expressions. *ACM Computing Surveys (CSUR)*, 39(3):1–35.

Xiao Li, Yu Liu, and Qingyun Du. 2021. Extracting spatial information from text using deep learning models: A survey. *International Journal of Geographical Information Science*, 35(2):365–392.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenStreetMap contributors. 2024. Nominatim: OpenStreetMap Geocoding API. Accessed: 2024-02-19.

PADI-web. 2023. Padi-web1 dataset. Accessed: 2023-12-01.

Stefan Schweter and Alan Akbik. 2020. Flert: Document-level features for named entity recognition. *Preprint*, arXiv:2011.06993.

Mehtab Alam Syed, Elena Arsevka, Mathieu Roche, and Maguelonne Teisseire. 2024. Geospacy: A tool for extraction and geographical referencing of spatial expressions in textual data. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (EACL 2024)*, pages 115–126, Montpellier, France. Association for Computational Linguistics.

Gemini Team. 2024a. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

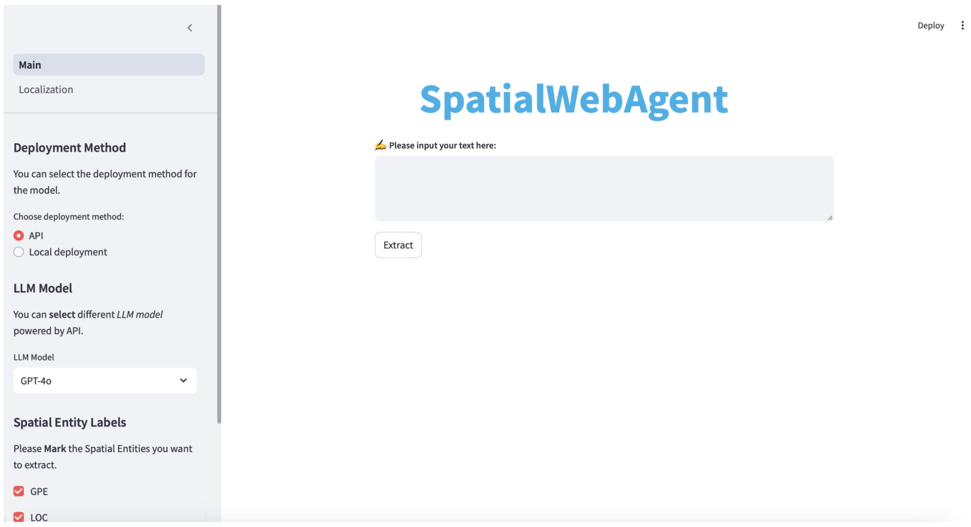
Gemma Team. 2024b. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Peng Yin, Jie Zhang, and Hongyuan Zha. 2021. Spatial relationship extraction from text using neural networks. *Geoinformatica*, 25:231–254.

Wei Zhang, Pengfei Liu, and Qiang Shen. 2020. Handling ambiguities in geographic texts using nlp techniques. *Transactions in GIS*, 24(3):519–538.

A System Screen Shot



SpatialWebAgent

Please input your text here:

Deployment Method

You can select the deployment method for the model.

Choose deployment method:

[✓ API]
Local deployment

LLM Model

You can **select** different *LLM model* powered by API.

LLM Model: GPT-4o

Spatial Entity Labels

Please **Mark** the Spatial Entities you want to extract.

[✓ GPE]
[✓ LOC]

Figure 3: The screen shot of SpatialWebAgent, users can specify the type of entities they want to extract and specific language models. For open source models, users can specify the model address.

B The Prompts of Different types of Named Entity Extraction

Here we list the different prompts provided to LLM for extracting different types of spatial entities.

Entity: GPE extraction prompt

System Prompt: You are a professional geographer. Your task is to extract all fuzzy spatial entities (keywords) from a given text. Fuzzy spatial keywords can include terms like *nearby, near, vicinity, close, beside, next, adjacent, immediate, border, surrounding, neighbourhood, proximity, territory, locality*, and similar terms.

For each fuzzy spatial keyword, wrap the name in a unique character sequence, such as `[###ENTITY###]`. If there are multiple entities, output them in the following format:
`[###ENTITY1###, ###ENTITY2###, ###ENTITY3###]`

Here is an example:
Text:
```
"The park is located nearby the lake, with several cafes close to the walking paths, and a small
garden adjacent to the main entrance."
```
Expected Output:
```
[###nearby###, ###close###, ###adjacent###]
```

Figure 4: An example prompt for extracting GPEs.

Entity: LOC extraction prompt

System Prompt: You are a professional geographer. Your task is to extract all location entities (LOC) from a given text. Location entities can include physical locations such as landmarks, geographical features, mountains, rivers, oceans, and places, but do not include political or administrative divisions such as countries or cities (these are considered geopolitical entities).

For each location entity, wrap the name in a unique character sequence, such as `[###ENTITY###]`. If there are multiple entities, output them in the following format:
`[###ENTITY1###, ###ENTITY2###, ###ENTITY3###]`

Here is an example:
Text:
```
"The Grand Canyon is one of the most spectacular natural wonders in the world, located in the state
of Arizona. Nearby, the Colorado River flows through the canyon, carving its way through the rugged
terrain. In the north, the Rocky Mountains stretch across several states, including Colorado and
Wyoming."
```
Expected Output:
```
[###Grand Canyon###, ###Arizona###, ###Colorado River###, ###Rocky Mountains###, ###Colorado###,
###Wyoming###]
```

Figure 5: An example prompt for extracting LOCs.

Entity: SRE (direction) extraction prompt

System Prompt: You are a professional geographer. Your task is to extract all spatial entities (directional keywords) from a given text. Spatial entities can include directional keywords such as *north, south, east, west*, and more specific terms like *northeast, northwest, southeast, southwest*, as well as terms indicating locations like *center, central, downtown*, and *midtown*.

For each spatial entity, wrap the name in a unique character sequence, such as `[###ENTITY###]`. If there are multiple entities, output them in the following format:
`[###ENTITY1###, ###ENTITY2###, ###ENTITY3###]`

Here is an example:
Text:
`"The hotel is located in the downtown area of New York, just south of Central Park, with a beautiful view of the southeast corner."`

Expected Output:
`[###downtown###, ###south###, ###southeast###]`

Figure 6: An example prompt for extracting SREs (direction).

Entity: SRE (fuzzy) extraction prompt

System Prompt: You are a professional geographer. Your task is to extract all fuzzy spatial entities (keywords) from a given text. Fuzzy spatial keywords can include terms like *nearby, near, vicinity, close, beside, next, adjacent, immediate, border, surrounding, neighbourhood, proximity, territory, locality*, and similar terms.

For each fuzzy spatial keyword, wrap the name in a unique character sequence, such as `[###ENTITY###]`. If there are multiple entities, output them in the following format:
`[###ENTITY1###, ###ENTITY2###, ###ENTITY3###]`

Here is an example:
Text:
`"The park is located nearby the lake, with several cafes close to the walking paths, and a small garden adjacent to the main entrance."`

Expected Output:
`[###nearby###, ###close###, ###adjacent###]`

Figure 7: An example prompt for extracting SREs (fuzzy).

Entity: RSE (distance) extraction prompt

System Prompt: You are a professional geographer. Your task is to extract all concrete distance keywords from a given text. Concrete distance keywords must include both a numeric value and a specific distance unit. These units can be in various formats, such as *kilometer, mile, meter, foot, inch, centimeter*, or their abbreviations (e.g., *km, mi, m, ft, cm, mm, yd*, etc.).

For each extracted distance keyword, wrap the entire expression (number + unit) in a unique character sequence, such as `[###ENTITY###]`. If there are multiple entities, output them in the following format:
`[###ENTITY1###, ###ENTITY2###, ###ENTITY3###]`

Here is an example:
Text:
`"The park is located 3 km away from the city center, while the nearest supermarket is only 500 meters from here, and the lake is about 1 mile further down the road."`

Expected Output:
`[###3 km###, ###500 meters###, ###1 mile###]`

Figure 8: An example prompt for extracting SREs (distance).

C Case Study

C.1

In this case study, we demonstrate the ability of our system to extract geographic entities from unstructured text. Below are the input, model output, and target output for a given example.

Legend:

- 🟨: **Location (LOC)**.
- 🟦: **Geopolitical Entity (GPE)**.
- 🟩: **Relative Spatial Entity (SRE)**

Input and Target Output:

ORE - IMC TBN - 70,000 tonnes Dampier / Kaohsiung 20-30/12 $ 5.25 fio 35,000 shinc / 30,000 shinc yellow China Steel.

Model Output:

{‘Dampier’: ‘GPE’, ‘Kaohsiung’: ‘GPE’, ‘China’: ‘GPE’}

In this case, we found that the model sometimes struggles to differentiate between GPE (Geopolitical Entity) and LOC (Location). However, this does not pose a significant issue for locating coordinates using the Nominatim API, as both are considered absolute spatial entities.

At times, the model incorrectly identifies certain terms containing place names, such as "company" or region/country names in products, as geographic entities.

C.2

Input and Target Output:

The company’s new headquarters is located roughly 5 miles south of the Sydney city center. 5 miles south is a directional spatial entity.

Model Output:

{‘south’: ‘RES_1’, ‘center’: ‘RES_1’}

{None}

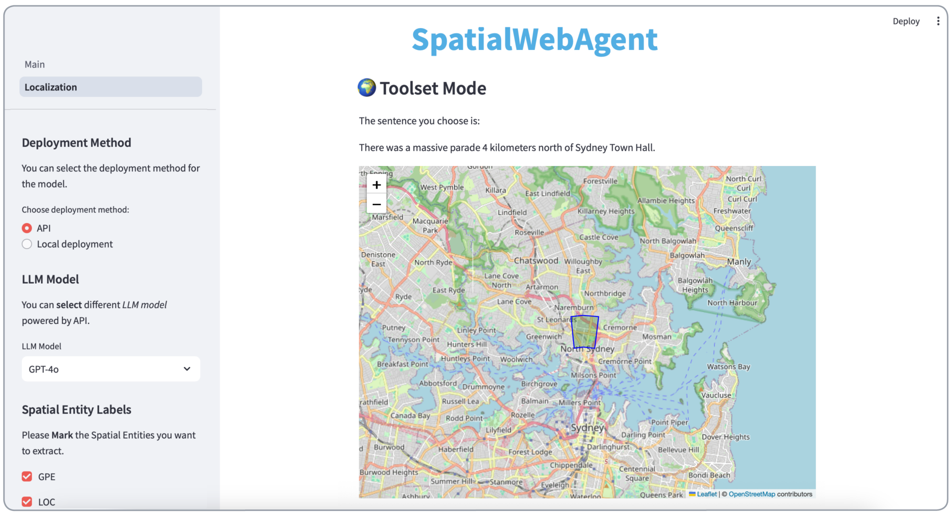
{‘5 miles’: ‘RES_3’}

Here, the model erroneously classifies the word "center" in "city center" as a directional spatial entity.

D Visualization Examples

Query 1:

There was a massive parade 4 kilometers north of Sydney Town Hall .


Deploy

Main

Localization

Deployment Method

You can select the deployment method for the model.

Choose deployment method:

[✓ API]

Local deployment

LLM Model

You can **select** different *LLM model* powered by API.

LLM Model

GPT-4o

Spatial Entity Labels

Please **Mark** the Spatial Entities you want to extract.

[✓ GPE]

[✓ LOC]

SpatialWebAgent

Toolset Mode

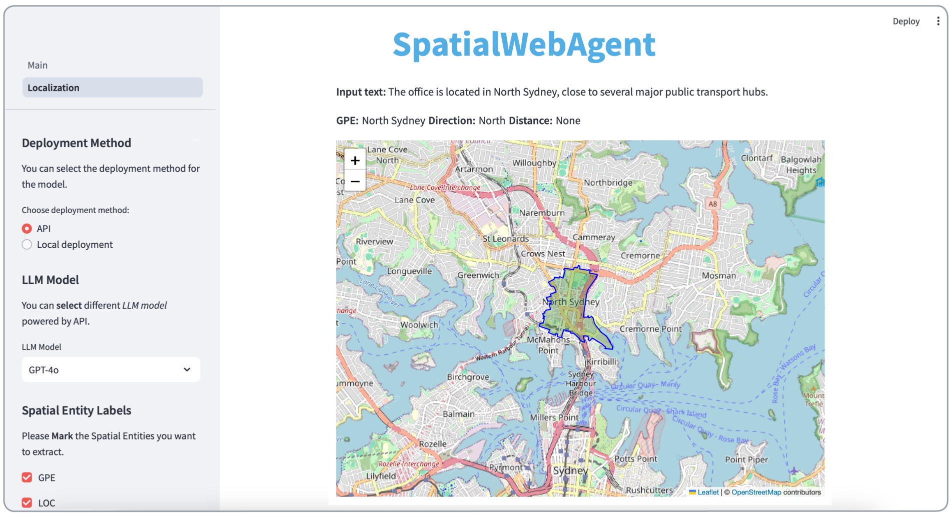
The sentence you choose is:

There was a massive parade 4 kilometers north of Sydney Town Hall.

Figure 9: When using 'Toolset Method', the LLM automatically extracts the necessary parameters, selects the appropriate function, and inputs the corresponding arguments. In this example, the chosen function is *directional*, and the input parameters are 'Sydney Town Hall', '4 kilometers', and 'north'.

Query 2:

The office is located in North Sydney , close to several major public transport hubs.


Deploy

Main

Localization

Deployment Method

You can select the deployment method for the model.

Choose deployment method:

[✓ API]

Local deployment

LLM Model

You can **select** different *LLM model* powered by API.

LLM Model

GPT-4o

Spatial Entity Labels

Please **Mark** the Spatial Entities you want to extract.

[✓ GPE]

[✓ LOC]

SpatialWebAgent

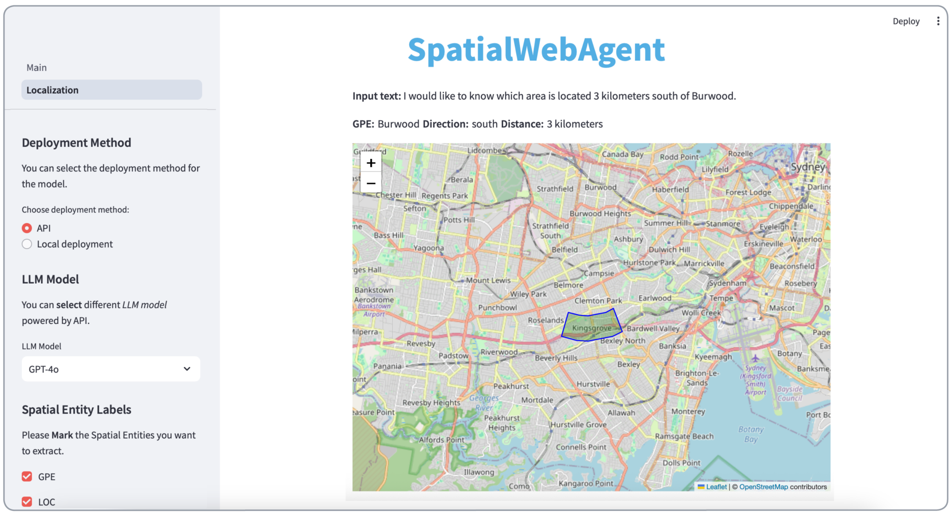
Input text: The office is located in North Sydney, close to several major public transport hubs.

GPE: North Sydney **Direction:** North **Distance:** None

Figure 10: In this query, we only one GPE which is 'North Sydney'.

Query 3:

I would like to know which area is located 3 kilometers south of Burwood .


Deploy

Main

Localization

Deployment Method

You can select the deployment method for the model.

Choose deployment method:

[✓ API]

Local deployment

LLM Model

You can **select** different *LLM model* powered by API.

LLM Model

GPT-4o

Spatial Entity Labels

Please **Mark** the Spatial Entities you want to extract.

[✓ GPE]

[✓ LOC]

SpatialWebAgent

Input text: I would like to know which area is located 3 kilometers south of Burwood.

GPE: Burwood **Direction:** south **Distance:** 3 kilometers

Figure 11: In this query, ‘Burwood’ is the GPE and ‘3 kilometers south’ is the SRE.

E Annotation Guidelines

Instruction abstraction and reasoning complexity: The instructions are intentionally designed to require multi-level spatial reasoning. For example, given the randomly sampled functions—`"steps"` field in Listing 1—a literal interpretation might be: ‘Find the point between location A and location B, then find the point between that and location A.’ However, we abstract this into a more concise and cognitively demanding instruction, such as: ‘Find the area one-fourth of the way from location A to location B, closer to A.’

Topical diversity of scenarios: To ensure broad coverage of real-world geographic expressions, the dataset includes instructions spanning diverse contexts. These range from urban navigation and public infrastructure to natural landmarks and administrative zones. This diversity exposes models to various spatial reference patterns and linguistic formulations, encouraging generalization beyond narrow domains.

Variation in geographic scale: The spatial entities referenced in the dataset vary significantly in scale, from fine-grained local features (e.g., buildings or suburbs) to coarse-grained global references (e.g., countries or regions). This variation ensures that the task reflects the hierarchical nature of spatial reasoning in natural language, requiring models to adapt their localization strategies based on the level of geographic granularity.