

MTO: A Multi-turn Conversational Text-to-OverpassQL Dataset for Enhanced OpenStreetMap Query Generation

Haodi Zhang¹, Xinrui Zhu¹, Mingze Kong²,
Zhidan Liu³, Tao Fan⁴, Kaishun Wu³, Yuanfeng Song*

¹Shenzhen University ²Chinese University of Hong Kong (Shenzhen)

³The Hong Kong University of Science and Technology (Guangzhou)

⁴The Hong Kong University of Science and Technology

{zhanghd.ustc}@gmail.com {2310275007}@email.szu.edu.cn

{mingzekong}@cuhk.edu.cn {tfanac}@cse.ust.hk {songyf}@outlook.com

Abstract

We propose a comprehensive framework for constructing multi-turn Text-to-OverpassQL dialogue datasets. Under this framework, we introduce the first multi-turn Text-to-OverpassQL dataset built upon the OverpassNL corpus. Our dataset comprises over 7,800 dialogues, each containing 2 to 4 user utterances, resulting in more than 20,000 individual utterances aligned with executable Overpass queries. To generate high-quality multi-turn dialogues, we design a four-stage pipeline. First, we convert Overpass queries into syntax trees using a custom parser developed based on the official OverpassQL grammar. This enables structural manipulation while preserving syntactic and executable validity. Second, we apply a diverse set of tree-editing templates, including both simple keyword-level changes and complex structural decompositions, to produce multiple valid and diverse Overpass queries. Third, we leverage a prompt-based approach to guide large language models in generating context-aware natural language questions, ensuring increasing inter-turn dependency across the dialogue. Finally, we implement a hybrid filtering strategy that combines manual annotation with model-assisted selection to validate alignment and correctness at scale. In addition to presenting the dataset, we evaluate the performance of several mainstream large language models and demonstrate that our end-to-end baseline model achieves competitive results. This work offers a new benchmark for studying executable semantic parsing and contextual understanding in map-based query tasks.

1 Introduction

In recent years, natural language interfaces to databases (NLIDBs) have received growing attention for their potential to bridge the gap between human language and structured query sys-

tems (Yang et al., 2024; Ren et al., 2024; Qu et al., 2024). Among the various structured data sources, OpenStreetMap (OSM) stands out as one of the most comprehensive and widely used open-source geospatial databases. To interact with and extract data from OSM, users often rely on OverpassQL, a domain-specific, expressive query language designed for querying the OSM database. However, OverpassQL has a steep learning curve, requiring users to understand both the syntax and the complex schema of OSM. This presents a significant barrier for non-expert users, motivating the development of systems that translate natural language queries into OverpassQL automatically.

While prior research has made notable progress in translating single-turn questions to structured queries (e.g., SQL, SPARQL), the translation to OverpassQL remains relatively under-explored. More importantly, real-world user interactions are often multi-turn in nature, where each user utterance depends on prior context. However, to the best of our knowledge, no existing dataset supports multi-turn dialog-based Text-to-OverpassQL tasks, limiting the development and evaluation of such systems.

To facilitate research on natural language interfaces for geospatial databases, especially in realistic multi-turn interaction settings, we introduce the first multi-turn Text-to-OverpassQL dataset. Built upon the OverpassNL dataset (Staniek et al., 2024), which contains single-turn question-query pairs, our dataset expands this resource by constructing over 7,878 multi-turn dialogues, each consisting of 2 to 4 utterances, resulting in more than 21501 utterance-query pairs. These dialogues are grounded in executable OverpassQL programs and designed to reflect real-world information-seeking behaviors.

How to construct high-quality multi-turn dialogue datasets remains a fundamental and open

*Corresponding author.

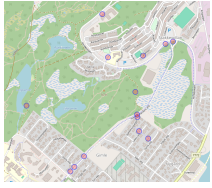
Natural Language	Bike lanes 500 meters around the top of a hill or mountain in Troms.
Overpass Query	<pre>{{geocodeArea:"Troms"}}->.searchArea; (node["natural"="peak"](area.searchArea);)->.peaks; way["highway"="cycleway"](area.searchArea)(around.peaks:500); out center;</pre>
Search Result	 <pre><way id="25200436"> <center lat="69.6879604" lon="18.9977258"/> <nd ref="5743949416"/> <nd ref="5743949424"/> <nd ref="5743949413"/> <nd ref="5743949412"/> <nd ref="5743949411"/> <tag k="bicycle" v="designated"/> <tag k="foot" v="designated"/> <tag k="highway" v="cycleway"/> <tag k="name" v="Lokes veg"/> </way></pre>

Figure 1: Example of overpass query and query result

problem. Existing approaches can generally be classified into three categories. The first category relies on extending existing datasets through manual rewriting or augmentation, where researchers transform single-turn data into multi-turn interactions (Guo et al., 2021; Yu et al., 2019a). While this approach benefits from leveraging well-established corpora, it is inherently limited in scale and diversity due to the heavy reliance on human annotation. The second category involves building expert systems and recruiting real users to engage in interactive dialogues (Budzianowski et al., 2018; Qi et al., 2025). Although this method yields more natural and contextually coherent data, it is extremely labor-intensive and costly, making it difficult to scale to large datasets. The third category, which has recently gained increasing attention, leverages large language models (LLMs) (Brown et al., 2020; Ouyang et al., 2022; Besta et al., 2024) to automatically generate multi-turn dialogues based on existing datasets (Ryu et al., 2024). This paradigm offers significant advantages in scalability and efficiency, but it raises critical challenges regarding accuracy, faithfulness, and alignment between natural language and structured representations. Addressing these challenges is essential for ensuring that automatically generated dialogues are not only diverse and coherent but also executable and valid, which underscores the importance of designing robust validation and filtering strategies.

To generate the dataset, we present a novel method for constructing such a dataset based on OverpassQL. Building upon the OverpassNL dataset, which provides high-quality single-turn question-query pairs, we propose a multi-stage pipeline to generate diverse and executable multi-turn dialogues. Our approach begins by converting

Overpass queries into syntax trees using a custom-built parser based on official grammar rules. This enables structural manipulation while preserving query validity. We then apply a set of tree-editing templates, ranging from simple keyword modifications to complex structural decompositions, to create diverse variations of Overpass queries. These are used as input candidates for a prompt-based framework that guides LLMs to construct coherent multi-turn dialogues with increasing contextual dependency. Finally, to ensure the correctness and alignment between natural language and query semantics, we design a semi-automatic validation strategy combining manual annotation with model-assisted filtering.

This method strikes a balance between annotation cost and dataset quality. By leveraging LLMs for large-scale generation while incorporating syntax-tree constraints and human-in-the-loop validation, our framework achieves both scalability and reliability. Compared with purely manual construction, our approach dramatically reduces the resource overhead, while in contrast to fully automated generation, it effectively mitigates the risks of incoherence, invalid queries, or semantic drift.

Our contributions can be summarized as follows.

- We introduce the first multi-turn Text-to-OverpassQL dataset with contextual dependencies, and compositional query structures.
- We design a data generation framework that combines human-written templates, paraphrasing techniques, and automatic OverpassQL generation.
- We evaluated a range of baseline models and demonstrated that our end-to-end baseline model achieves competitive results.

2 Related Work

2.1 NLDBs

Mapping natural language to structured queries has been extensively studied in recent years, particularly in the context of relational databases and knowledge graphs. In the SQL domain (Xie et al., 2025), early datasets such as WikiSQL (Zhong et al., 2017) introduced large-scale supervised examples for single-turn query generation, while more complex datasets like Spider (Yu et al., 2018) and SParC (Yu et al., 2019b) emphasized compositional generalization and multi-turn dialog

understanding, respectively. In parallel, semantic parsing over knowledge graphs has been explored through SPARQL-based datasets such as LC-QuAD(Trivedi et al., 2017), and WebQuestionsSP(tau Yih et al., 2016), enabling query generation over structured triples.

Beyond these mainstream efforts, less attention has been given to natural language interfaces for geospatial databases such as OSM. OverpassQL, the dominant query language for OSM, poses unique challenges due to its nested syntax, spatial filters, and domain-specific semantics. While some recent works (Staniek et al., 2024; Zhang et al., 2025; Wan et al., 2026) have attempted to generate Overpass queries from user questions, these are often limited to single-turn settings or handcrafted templates, lacking broader linguistic and contextual diversity.

Compared to prior work, our focus is on multi-turn semantic parsing in the geospatial domain, where queries evolve through contextual interactions. To our knowledge, no existing dataset addresses this task. Our work aims to fill this gap by constructing the first benchmark for multi-turn Text-to-OverpassQL, supporting research on dialog-aware spatial query generation.

2.2 Multi-turn Dialog Dataset Construction

Constructing high-quality multi-turn dialog datasets for semantic parsing remains a significant challenge, as it requires modeling both linguistic variation and contextual dependencies. Existing multi-turn datasets(Guo et al., 2021; Henderson et al., 2014; Hemphill et al., 1990) such as SPaC(Yu et al., 2019b) and CoSQL(Yu et al., 2019a) extend the single-turn Spider dataset to dialog settings by collecting crowd-sourced question sequences grounded in real SQL schemas. These datasets emphasize context tracking, coreference resolution, and query refinement. Similarly, SMCaFlow(Semantic Machines et al., 2020) introduces complex task-oriented dialogs using a compositional semantic formalism, with utterances either written manually or generated via paraphrasing.

Other works(Dong et al., 2025; Li et al., 2024) employ semiautomatic approaches to reduce annotation cost. For example, the EHR-SeqSQL(Ryu et al., 2024) dataset constructs multi-turn Text-to-SQL dialogues by decomposing each SQL query into a sequence of subqueries, reorganizing them into stepwise sequences, and then generating cor-

responding natural language descriptions through predefined templates. These descriptions are subsequently refined into dialogue utterances using LLMs. Similarly, the PRACTIQ(Dong et al., 2025) dataset generates contextual dialogue data by modifying SQL statements and database schemas to create new query scenarios, and then leveraging language models to expand the original data into multi-turn dialogues. Both approaches highlight the effectiveness of combining symbolic query decomposition with LLM-based generation for constructing realistic and coherent dialogue datasets.

Inverse-annotation approaches provide another line of work for generating question corpora from structured representations. Deriu et al. (Deriu et al., 2020) propose inverse data annotation for building question answering corpora by generating natural-language questions from complete meaning representations. Our setting differs in that we reconstruct multi-turn conversations from incremental AST transformations, which requires state tracking, clause dependencies, and contextual resolution across turns; these multi-step constraints are not present in single-step inverse-annotation settings.

Inspired by these strategies, our dataset adopts a hybrid construction pipeline combining template-based query generation, dialog simulation, and paraphrase augmentation, tailored for the geospatial domain. Unlike prior datasets focused on relational or task-oriented domains, our data reflects realistic user intents for querying OSM using OverpassQL, requiring compositional reasoning and context-sensitive query generation.

3 The MTO Dataset

To construct a multi-turn dialog dataset grounded in realistic map-based information-seeking scenarios, we build upon the existing OverpassNL dataset (Staniek et al., 2024), which contains 8.5k high-quality pairs of natural language queries and executable OverpassQL programs. The original dataset was carefully curated via crowdsourcing and rigorous validation, making it a reliable foundation for extension.

We note that some utterances in our examples may refer to regions via placeholders (e.g., “within this area”). Such references are not underdetermined in our benchmark: they are grounded to the concrete latitude–longitude bounds from the source OverpassNL instances, and the placeholder is replaced with the corresponding bounds during

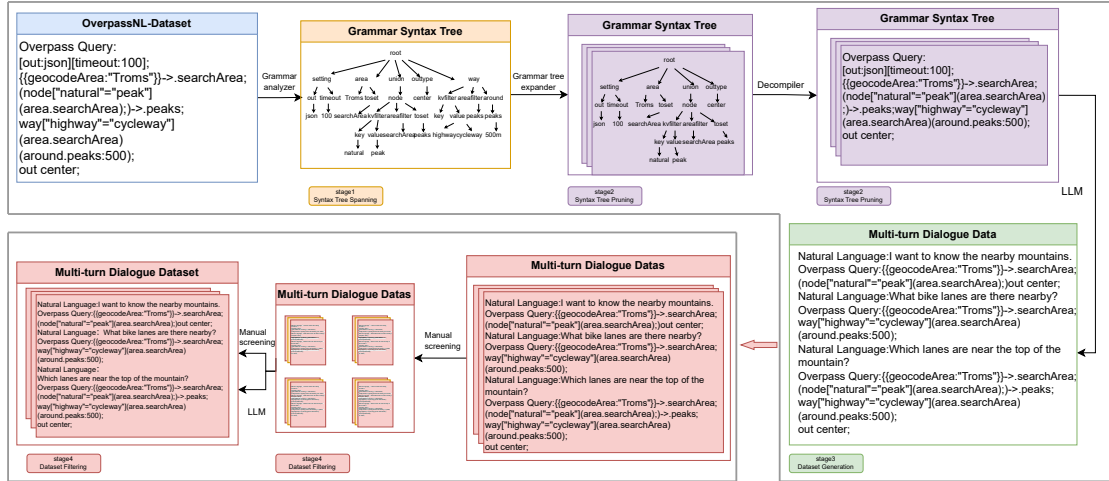


Figure 2: Overview of our dataset construction pipeline, consisting of four main modules: (1) SYNTAX TREE SPANNING This module parses Overpass queries into ASTs using a custom parser built from official grammar specifications. The resulting trees serve as the basis for executable and structurally sound query transformations. (2) SYNTAX TREE PRUNING A set of rule-based transformations is applied to the ASTs to produce diverse and coherent Overpass queries. These transformations ensure variability while maintaining semantic integrity. (3) DATASET GENERATION Leveraging a prompt-driven large language model framework, this module selects logically connected OverpassQL sequences and generates corresponding multi-turn dialog utterances that reflect natural conversational flow. (4) DATASET FILTERING A small portion of the generated data is manually annotated and designated as ground truth. These samples are used to validate the model’s outputs; incorrectly labeled batches are further filtered or corrected via human review, ensuring the overall quality and accuracy of the dataset. Note: Please zoom in to view the details

execution-based evaluation.

To ensure the validity and executability of extended data, we first develop a custom OverpassQL syntax tree parser that transforms Overpass queries into abstract syntax trees (ASTs). We then apply controlled tree-level pruning operations to produce modified but still executable query structures. To further enhance query diversity, we design a set of syntax tree transformation templates that guide the generation of varied and semantically meaningful query variants.

Given the pool of extended Overpass queries, we construct a prompt-based framework that leverages LLMs to select coherent query sequences and generate corresponding multi-turn dialog utterances. The model is guided to preserve logical flow and contextual consistency across dialog turns.

To ensure quality, we apply a targeted human-in-the-loop verification process. Specifically, for every 10 generated dialog samples, one is designated as a ground truth instance with gold annotations. If the LLM correctly reconstructs this reference, the remaining samples in the batch are retained. Otherwise, the batch is manually reviewed and filtered to ensure correctness. This selective human supervision strategy balances scalability and quality

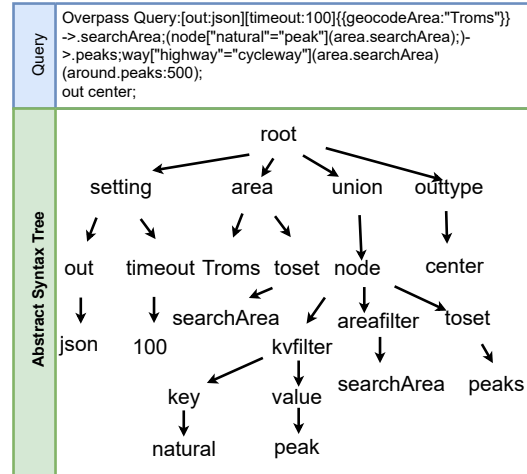


Figure 3: Example of abstract syntax trees

control.

An overview of the data generation pipeline is shown in Figure 2, and further implementation details are provided in the following sections.

3.1 Data Construction

3.1.1 Stage 1. Syntax Tree Spanning

To ensure the correctness, executability, and structural diversity of our dataset, we represent Overpass queries in the form of syntactic trees that conform

to the official OverpassQL grammar. Based on the formal syntax specification provided in the OverpassQL documentation¹, we implement a custom grammar and develop a dedicated parser that converts OverpassQL strings into structured ASTs, as illustrated in Figure 3.

The parser is designed to handle the full expressiveness of OverpassQL, including recursive constructs, nested filters, and compound statements. Once parsed, the AST provides a robust and semantically faithful representation of the original query, serving as the foundation for controlled query transformations. These transformations include operations such as subtree pruning, substitution, or reordering, which allow us to generate syntactically valid and executable variants of the original queries.

By working at the syntax tree level, we can precisely manipulate query components without violating OverpassQL's execution constraints, thereby ensuring that all generated queries remain both valid and semantically interpretable. This syntactic representation also facilitates downstream modules such as diversity-oriented generation and dialog grounding, which rely on structurally coherent OverpassQL samples.

3.1.2 Stage 2. Syntax Tree Pruning

In this module, we design a collection of transformation templates and rule-based operations to edit the OverpassQL syntax trees, with the aim of generating diverse and coherent query variants. Figure 4 shows the content of some extension methods. These generated queries serve as candidate inputs for the subsequent prompt-based dialog construction, where the language model selects and arranges them into contextually grounded multi-turn conversations.

We categorize the transformation templates into simple and complex types. The simple templates involve localized edits to specific components of a query, such as modifying constraints on dates and locations, altering output formats, or adjusting result limits. They also include changes to independent query expressions, such as editing tag-value pairs or modifying the output type. These modifications preserve the structure of the original query while introducing sufficient variation to support downstream data generation.

¹https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL

In contrast, the complex templates involve structural transformations at a higher level of abstraction. These include decomposing queries based on the use of the "map_to_area" keyword, extracting subqueries from compound BLOCK structures, and splitting queries that rely on intermediate variable assignments and references. These transformations produce semantically meaningful fragments that can function as standalone queries, enabling finer-grained control over dialog turn construction.

Among these modification types, we found that the most challenging aspect is maintaining correct clause-to-clause referencing and linkage while preserving both executability and semantic consistency. In practice, a single query is often decomposed into multiple subclauses whose intermediate results are stored in a set structure; different subclauses may include or reference the results of others. When decomposing a query, incorrect handling of these inclusion and reference relationships can yield queries that remain syntactically valid but reference a non-existent set, leading to execution errors and making the data unusable; an incorrect extraction order can also cause the subclause semantics to become inconsistent. Therefore, compared with simpler modifications such as adjusting the number of control statements, changing output settings, or adding/deleting/editing key-value pairs, the extraction of subclauses in complex queries is significantly more challenging.

Although Table 1 lists both SQL- and OverpassQL-based datasets (e.g., EHR-SeqSQL and MTO), their executability requirements are fundamentally different. SQL is schema-driven and table-centric, where executability largely depends on selecting the correct tables, fields, and joins under a fixed relational schema. In contrast, OverpassQL is geospatial and schemaless, and queries are constructed procedurally through heterogeneous object retrieval, spatial operators (e.g., `around`, `within`), and set-based subclauses whose intermediate results must remain consistent across turns. As a result, a syntactically valid OverpassQL query may still fail due to missing set references or inconsistent geographic context, making contextual tracking and procedural reasoning particularly critical.

3.1.3 Stage 3. Dataset Generation

In this module, we design a prompt framework aimed at guiding the language model to select semantically related Overpass queries from a given

Category	Definition	Example
Extension of settings	Modifications to query statements, including adding, altering, or removing temporal, spatial, and output constraints.	Original query: [out:json]{{geocodeArea:"Troms"}}->.searchArea;(node["natural"]="peak")(area.searchArea);->.peaks;way["highway"]="cycleway"(area.searchArea)(around.peaks:500);out center; Extended query: [out:json][date:"2020-6-26T00:00:01Z"]{{geocodeArea:"Singapore"}}->.searchArea;(node["natural"]="peak")(area.searchArea);->.peaks;way["highway"]="cycleway"(area.searchArea)(around.peaks:500);out center;
Extension of filter	Modifications to query filters, including replacing, adding, or removing semantically similar key-value pairs.	Original query: [out:json][timeout:25];(way["natural"]="wood";relation["natural"]="wood");out;>;out skel qt; Extended query: [out:json][timeout:25];(way["landuse"]="wood"["admin_level"]="6";relation["landuse"]="wood");out;>;out skel qt;
Extension based on block	Block-based decomposition, where internal content is extracted as one query and the remaining structure forms another.	Original query: [out:json]{{geocodeArea:"Troms"}}->.searchArea;(node["natural"]="peak")(area.searchArea);-node["natural"]="wood"(area.searchArea);out center; Extended query: 1.[out:json]{{geocodeArea:"Troms"}}->.searchArea;node["natural"]="peak"(area.searchArea);out center; 2.[out:json]{{geocodeArea:"Troms"}}->.searchArea;node["natural"]="wood"(area.searchArea);out center;
Extension based on set	Statements from set definitions are extracted as independent queries, with set references treated separately. For complex dependencies, topological sorting organizes query components.	Original query: [out:json]{{geocodeArea:"Troms"}}->.searchArea;(node["natural"]="peak")(area.searchArea);->.peaks;way["highway"]="cycleway"(area.searchArea)(around.peaks:500);out center; Extended query: 1.[out:json]{{geocodeArea:"Troms"}}->.searchArea;node["natural"]="peak"(area.searchArea);out center; 2.[out:json]{{geocodeArea:"Troms"}}->.searchArea;way["highway"]="cycleway"(area.searchArea);out center;

Figure 4: Definition and Example of Extended template, Note that the query are simplified for illustration purposes.

pool and to generate corresponding natural language questions. The generated questions are required to be contextually coherent and to exhibit a logical progression across dialogue turns. Furthermore, the model is instructed to assess the complexity of each query when generating questions, ensuring a balanced mixture of simple and complex information needs. The details of the prompts used in our dataset construction are provided in Appendix A.

To facilitate the model’s understanding of the task, we manually construct three reference examples representing different levels of query complexity (simple, medium, and complex). These in-context examples demonstrate how to generate high-quality natural language questions, how to establish contextual connections between turns, and how to assess the complexity of each OverpassQL input.

During our initial experimentation, we observed that the model tended to generate questions that, although relevant to the input query, were too independent in nature—that is, each question could

be resolved using a single query without referring to the dialogue history. This contradicts the core objective of building a multi-turn dialogue dataset, where inter-turn dependency is essential. To address this issue, we revised our prompt design to explicitly emphasize the importance of contextual linkage. Specifically, the model is now instructed to ensure that each question depends on prior turns, making it infeasible to resolve in isolation. To further illustrate this phenomenon and the effectiveness of our revised prompt strategy, we provide detailed case studies in the Appendix C.1.

3.1.4 Stage 4. Dataset Filtering

Although the use of syntax trees in the earlier stages ensures the correctness and executability of Overpass queries, additional verification is still necessary to address potential issues introduced during the generation process by LLMs. These issues include inconsistencies between natural language utterances and their corresponding Overpass queries, or logical flaws in the generated dialogue turns. Given the scale and complexity of the generated data, we adopt a group-based sampling and valida-

Metric	Query Language	Number of Instances	Conversational	of Interactions	of Turns
NLmaps	MRL	2,380	✗	-	-
SParC	SQL	12,726	✓	4,298	12,726
CoSQL	SQL	15,598	✓	3,007	15,598
EHR-SeqSQL	SQL	31,669	✓	9,195	31,669
OverpassNL	OverpassQL	8352	✗	-	-
MTO	OverpassQL	21501	✓	7,878	21,501

Table 1: Comparison of Text-to-Query datasets.

tion strategy to efficiently ensure dataset quality.

Specifically, we randomly sample 10% of the generated data and conduct manual annotation to determine whether each instance meets the inclusion criteria. These annotated samples are then grouped into batches of 10, with one instance in each group designated as the ground truth. The model is tasked with evaluating all 10 samples in a group, including identifying the one that matches the ground truth annotation. If the model correctly identifies the ground truth and provides accurate judgments on the remaining samples, the outputs marked as acceptable by the model are accepted into the final dataset. If the model misjudges the ground truth or shows uncertainty, the entire group is passed to human annotators for further inspection and correction.

While this batch-based strategy is efficient at scale, it may allow some low-quality dialogues to pass due to uneven error distribution within a batch; we discuss this limitation and potential improvements in the Limitations section.

This group-based semi-automatic validation approach balances efficiency and accuracy, ensuring that only high-quality, executable, and contextually consistent data is retained in the final corpus.

3.2 Data Statistics

In total, our framework generates 7,878 multi-turn dialogues, comprising 21,501 individual utterances, with an average of 2.73 utterances per dialogue. In terms of dialogue length distribution, the dataset contains 2,197 two-turn dialogues, 5,594 three-turn dialogues, 74 four-turn dialogues, and 13 dialogues with more than four turns. To better capture the range of query complexity, we further categorize dialogues into three levels: 2,446 low-complexity dialogues, 5,343 medium-complexity dialogues, and 89 high-complexity dialogues.

Our dataset is built upon the OverpassNL dataset,

which covers approximately 76% of the core syntactic features of OverpassQL and includes keys representing 91% of the frequently used tags in OSM. In our extension, we further incorporate additional keys collected directly from the official OSM documentation, thereby expanding the linguistic and structural coverage beyond OverpassNL. Overall, our dataset encompasses 95% of the content in OverpassNL while introducing a substantial set of newly integrated keys and query structures.

This distribution and coverage analysis highlights two important characteristics of our dataset. First, the majority of dialogues fall within the two-to three-turn range, aligning with practical user interaction patterns in information-seeking scenarios. Second, the inclusion of dialogues across multiple complexity levels, combined with expanded query coverage, ensures both linguistic and structural diversity. Together, these properties demonstrate that the dataset achieves a balance between realism, diversity, and scalability, making it a robust benchmark for advancing research on multi-turn semantic parsing in map-based query tasks.

We include several representative examples from the dataset in the Appendix E to illustrate the characteristics of the constructed dialogues.

3.3 Comparison to Other Datasets

In the domain of Text-to-OverpassQL, no prior work has addressed the construction of multi-turn dialogue datasets. Existing OSM-related corpora, such as NLMap (Lawrence and Riezler, 2016) with 2,380 examples and OverpassNL (Staniek et al., 2024) with 8,352 text-to-query pairs, are limited to single-turn interactions. In contrast, the SQL domain offers several multi-turn benchmarks, including SParC (Yu et al., 2019b) (over 4,000 dialogues, 3–4 turns on average), CoSQL (Yu et al., 2019a) (3,000 dialogues, 3 turns per dialogue), and the more recent EHR-SeqSQL (Ryu et al., 2024)

Model	Overpass Query Similarity				Execution Accuracy			Validity (%)
	KVS	TreeS	chrF	OQS	EM	EX	EX _{soft}	
GPT-4.1mini	44.3	41.7	54.9	46.9	4.4	15.1	23.3	85.9
qwen3-235b	43.7	35.0	43.6	40.7	3.7	14.5	20.3	83.6
deepseekV3.1	49.7	41.9	50.8	47.4	4.9	16.0	23.8	93.93
gemini2.5-flash	47.3	44.3	45.8	45.8	4.4	13.8	21.0	87.7
Byt5-Large (finetuned)	59.0	65.4	67.4	64.0	7.1	15.9	25.0	97.68

Table 2: Results on the Test set of MTO.

(9,000 dialogues, 3 turns), highlighting the importance of multi-turn semantic parsing in complex query settings.

Motivated by this gap, we introduce the first multi-turn Text-to-OverpassQL dataset, constructed by extending OverpassNL with additional query keys from OSM documentation. The dataset consists of 7,878 dialogues and 21,501 turns (averaging 3 turns per dialogue), each paired with executable OverpassQL queries. This resource establishes a new benchmark for multi-turn semantic parsing in geospatial data querying.

4 Experiments

We evaluate the performance of several mainstream models on our dataset and provide a fine-grained analysis of their generated outputs. To further illustrate the strengths and limitations of these models, we include case studies with representative examples in the Appendix C.2.

4.1 Experiment Setup

4.1.1 Prompting Protocol

Unless otherwise specified, we evaluate models using standard prompting without advanced reasoning or tool-use prompting (e.g., Chain-of-Thought, explicit thinking prompts, or external tool calls). We leave a systematic study of such advanced prompting strategies for future work.

4.1.2 Models

In our evaluation, we tested the dataset across multiple mainstream LLMs, including GPT-4.1-mini, Qwen3-235B, DeepSeek V3.1, and Gemini 2.5-Flash, which cover a diverse range of strong general-purpose LLMs. In addition, we fine-tuned a local model, ByteT5-Large, to further validate the dataset in a controlled setting. The task of mapping natural language to Overpass queries can be formulated as a sequence-to-sequence (seq2seq) problem, which naturally aligns with models based on the encoder-decoder architecture. We chose

the T5 family due to its strong performance across various seq2seq tasks (Raffel et al., 2020), particularly in Text-to-SQL tasks (Scholak et al., 2021). Within this family, ByteT5 was selected because it has demonstrated superior performance compared to CodeT5 in related structured prediction tasks (Staniek et al., 2024). Furthermore, we opted for the Large variant of ByteT5, since multi-turn dialogue requires stronger contextual understanding and more robust in-context learning capabilities. Further details of the models and training procedures are provided in the Appendix B.

4.1.3 Metrics

We adopt widely used evaluation metrics in the field (Staniek et al., 2024), covering both query similarity and execution accuracy perspectives. For query similarity, we report Character F-score (chrF), which captures lexical overlap at the character n-gram level and is more robust to minor surface variations than word-level measures; Key-Value Similarity (KVS), which explicitly evaluates whether the predicted queries preserve the correct key-value pairs critical for OverpassQL semantics; and the Tree Similarity metric (TreeS), which compares the syntactic tree structures of predicted and gold queries to assess structural fidelity. To further quantify semantic alignment, we include Exact Match (EM), which measures the percentage of predictions identical to gold queries, Execution Accuracy (EX), which evaluates correctness by executing both predicted and reference queries on the OSM database and comparing the retrieved results, and Soft Execution Accuracy (EX-SOFT), which relaxes the strictness of EX by allowing partial overlaps in the retrieved sets. The detailed computation procedures of all evaluation metrics are provided in the Appendix D for reproducibility and clarity.

Together, these metrics provide a comprehensive evaluation framework, jointly reflecting lexical correctness, structural validity, and execution-level utility, which are all essential in assessing the

performance of models on multi-turn OverpassQL generation.

4.2 Main Results

Table 2 reports the performance of different models on the test set of MTO across both query similarity and execution-based metrics. Overall, we observe that ByT5-Large (fine-tuned) substantially outperforms all baseline models. In terms of query similarity, it achieves the highest scores across all metrics, with 59.0 KVS, 65.4 TreeS, 67.4 chrF, and 64.0 OQS, showing clear improvements over strong closed-source systems such as GPT-4.1mini and Gemini 2.5-flash. Notably, the gains in TreeS and chrF indicate that the fine-tuned ByT5-Large model is particularly effective in capturing the structural and surface-level fidelity of Overpass queries.

For execution-level evaluation, ByT5-Large also obtains the best Soft Execution Accuracy (25.0), while achieving a competitive Execution Accuracy (15.9) that is on par with GPT-4.1mini (15.1) and slightly lower than DeepSeek V3.1 (16.0). These results suggest that although ByT5-Large excels at generating queries that are structurally and semantically aligned with gold references, there remains a gap in strict execution accuracy compared to models such as DeepSeek.

In addition to similarity and execution accuracy, we quantify OverpassQL validity as the percentage of syntactically executable model outputs, reported in Table 2.

In summary, the results highlight that fine-tuning domain-adapted encoder-decoder models like ByT5 can provide consistent improvements in multi-turn Overpass query generation, particularly in terms of query similarity and robust execution.

5 Conclusion

In this work, we introduced MTO, the first large-scale multi-turn semantic parsing dataset for geospatial queries over OSM. Unlike existing single-turn corpora such as NLMap and OverpassNL, our dataset explicitly models contextual dependencies and query compositionality across multiple interaction turns, thereby bridging a critical gap in Text-to-OverpassQL research. To construct MTO, we designed a syntax-tree-driven generation framework that systematically extends single-turn queries into coherent multi-turn dialogues, while ensuring both executability and linguistic diversity.

We conducted extensive evaluations across several widely used LLMs (e.g., GPT-4.1mini, Qwen3-235b, DeepSeek V3.1, Gemini 2.5-Flash) and a finetuned ByT5-Large model. Results demonstrate that MTO poses significant challenges for existing models, particularly in handling contextual reasoning and query structure alignment. At the same time, ByT5-Large achieves notable improvements under all metrics, highlighting the potential of byte-level sequence-to-sequence models for this task.

We hope that MTO will serve as a new benchmark for advancing research in multi-turn semantic parsing for geospatial databases, and inspire future work on more robust dialogue systems that can seamlessly support interactive, real-world map exploration.

Limitations

We propose a novel framework for constructing multi-turn dialogue datasets that ensures syntactic correctness while significantly reducing manual involvement. However, the method relies on domain-specific grammar parsers and tree-editing templates, which must be carefully designed for each target domain. This constraint reduces the generalizability of the framework and poses challenges for direct transfer to new or low-resource domains.

Our evaluation focuses on standard prompting and does not explore advanced prompting methods such as Chain-of-Thought or tool-use prompts, which may further improve LLM performance on multi-turn OverpassQL generation.

In addition, we do not evaluate finetuned decoder-only architectures (e.g., open-source instruction-tuned LLMs adapted to our task). A systematic investigation would require additional model engineering and computational resources, which are beyond the scope of the present work.

Our validation strategy is batch-based, which may allow some low-quality dialogues to pass due to uneven error distribution. A more rigorous evaluation would involve independently sampling and validating each dialogue, ideally with stratified sampling across modification types. In future work, we plan to design a new validation pipeline that incorporates independent sampling, finer-grained error tracking, and automated consistency checks.

Our experiments primarily evaluate baseline LLMs under unconstrained decoding. While this setting reveals raw model limitations, struc-

tured decoding (e.g., grammar-based constraints or function-calling formats) and few-shot demonstrations could significantly increase OverpassQL validity. A comprehensive exploration of these techniques is beyond the scope of this paper.

We also evaluated the MTO-fine-tuned ByT5 model on OverpassNL and observed moderate transfer (chrF 55.34, KVS 46.64, TreeS 53.18, OQS 51.71, EM 4.9, EX 14.1, EX_{soft} 21.2), but it underperforms a ByT5 model finetuned directly on OverpassNL. We attribute this gap to distributional differences in natural-language phrasing (OverpassNL is more diverse and human-written, while MTO follows AST-guided transformations) and structural differences between single-turn and multi-turn query styles. Future work will explore joint multi-dataset training, style-aware data augmentation, and domain-adaptive finetuning to improve robustness across heterogeneous OverpassQL datasets.

Finally, our work focuses on conversational text-to-OverpassQL generation and does not address the inverse direction (OverpassQL-to-text) or the generation of natural-language system responses. Developing OverpassQL-to-text resources and models is a promising direction for future work.

References

- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. *Language models are few-shot learners*. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. *MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling*. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.
- Jan Deriu, Katsiaryna Mlynchyk, Philippe Schlapfer, Álvaro Rodrigo, Dirk von Grunigen, Nicolas Kaiser, Kurt Stockinger, Eneko Agirre, and Mark Cieliebak. 2020. *A methodology for creating question answering corpora using inverse data annotation*. In *Annual Meeting of the Association for Computational Linguistics*.
- Mingwen Dong, Nischal Ashok Kumar, Yiqun Hu, Anuj Chauhan, Chung-Wei Hang, Shuaichen Chang, Lin Pan, Wuwei Lan, Henghui Zhu, Jiarong Jiang, Patrick Ng, and Zhiguo Wang. 2025. *PRACTIQ: A practical conversational text-to-SQL dataset with ambiguous and unanswerable queries*. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 255–273, Albuquerque, New Mexico. Association for Computational Linguistics.
- Jiaqi Guo, Ziliang Si, Yu Wang, Qian Liu, Ming Fan, Jian-Guang Lou, Ziji Yang, and Ting Liu. 2021. *Chase: A large-scale and pragmatic Chinese dataset for cross-database context-dependent text-to-SQL*. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2316–2331, Online. Association for Computational Linguistics.
- Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. *The ATIS spoken language systems pilot corpus*. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.
- Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. *The second dialog state tracking challenge*. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A. Association for Computational Linguistics.
- Carolin Lawrence and Stefan Riezler. 2016. *NLmaps: A natural language interface to query OpenStreetMap*. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 6–10, Osaka, Japan. The COLING 2016 Organizing Committee.
- Lei Li, Yuqi Wang, Runxin Xu, Peiyi Wang, Xiachong Feng, Lingpeng Kong, and Qi Liu. 2024. *Multimodal ArXiv: A dataset for improving scientific comprehension of large vision-language models*. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14369–14387, Bangkok, Thailand. Association for Computational Linguistics.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1

- others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Zhiyang Qi, Takumasa Kaneko, Keiko Takamizo, Mariko Ukiyo, and Michimasa Inaba. 2025. [KokoroChat: A Japanese psychological counseling dialogue dataset collected via role-playing by trained counselors](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 12424–12443. Association for Computational Linguistics.
- Ge Qu, Jinyang Li, Bowen Li, Bowen Qin, Nan Huo, Chenhao Ma, and Reynold Cheng. 2024. [Before generation, align it! a novel and effective strategy for mitigating hallucinations in text-to-SQL generation](#). In *Findings of the Association for Computational Linguistics ACL 2024*. Association for Computational Linguistics.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).
- Tonghui Ren, Yuankai Fan, Zhenying He, Ren Huang, Jiaqi Dai, Can Huang, Yinan Jing, Kai Zhang, Yifan Yang, and Xiaoyang Sean Wang. 2024. [Purple: Making a large language model a better sql writer](#). *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 15–28.
- Jaehye Ryu, Seonhee Cho, Gyubok Lee, and Edward Choi. 2024. [EHR-SeqSQL : A sequential text-to-SQL dataset for interactively exploring electronic health records](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16388–16407, Bangkok, Thailand. Association for Computational Linguistics.
- Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. [PICARD: Parsing incrementally for constrained auto-regressive decoding from language models](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Semantic Machines, Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, and 27 others. 2020. [Task-oriented dialogue as dataflow synthesis](#). *Transactions of the Association for Computational Linguistics*, 8:556–571.
- Michael Staniek, Raphael Schumann, Maike Züfle, and Stefan Riezler. 2024. [Text-to-overpassql: A natural language interface for complex geodata querying of openstreetmap](#). *Transactions of the Association for Computational Linguistics*, 12:562–575.
- Wen tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. [The value of semantic parse labeling for knowledge base question answering](#). In *Annual Meeting of the Association for Computational Linguistics*.
- Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. [Lc-quad: A corpus for complex question answering over knowledge graphs](#). In *International Semantic Web Conference*, pages 210–218. Springer.
- Zhuoyue Wan, Wentao Hu, Chen Jason Zhang, Yuanfeng Song, Shuaimin Li, Ruiqiang Xiao, Xiao-Yong Wei, and Raymond Chi-Wing Wong. 2026. [Osmt: Bridging openstreetmap queries and natural language with open-source tag-aware language models](#). In *2026 IEEE 42nd International Conference on Data Engineering (ICDE)*.
- Xiangjin Xie, Guangwei Xu, Lingyan Zhao, and Ruijie Guo. 2025. [Opensearch-sql: Enhancing text-to-sql with dynamic few-shot and consistency alignment](#). *Proc. ACM Manag. Data*, 3(3).
- Jiayi Yang, Binyuan Hui, Min Yang, Jian Yang, Junyang Lin, and Chang Zhou. 2024. [Synthesizing text-to-SQL data from weak and strong LLMs](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.
- Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, and 5 others. 2019a. [CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. [Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.
- Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. [SPaRC: Cross-domain semantic parsing in context](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

Haodi Zhang, Xinrui Zhu, Xuan Wang, Yuanfeng Song, Hanlin Gu, and Lin Ma. 2025. [Natural language to overpass query: A multi-step approach using task decomposition and key-value correction](#). In *2025 26th IEEE International Conference on Mobile Data Management (MDM)*, pages 1–11.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

A Prompts

A.1 Dataset Construction

The primary objective of the prompts for dataset construction is to guide the model in effectively utilizing the extended data, selecting appropriate samples for dialogue construction. Additionally, the prompts are designed to enable the model to better simulate realistic human conversational needs, thereby generating more natural dialogues. This approach also mitigates issues such as weak contextual coherence, lack of relevance, and excessive independence between utterances, which are discussed in detail in Appendix C.1. The specific prompts are illustrated in Figure 12.

A.2 Model Evaluation Generation

The prompts used during model evaluation are designed to ensure that the model’s performance can be assessed in a controlled and consistent manner. Specifically, they aim to elicit responses that reflect the model’s ability to handle realistic conversational scenarios, maintain contextual coherence, and generate relevant, human-like outputs. By carefully structuring these prompts, we reduce biases caused by ambiguous instructions and enable a more reliable comparison across different models. The detailed design of these evaluation prompts is presented in Figure 11.

B Model Details

For LLMs evaluation, we employed several widely adopted proprietary APIs. Specifically, we utilized Qwen3-235B-a22b on the Alibaba Cloud platform in its default non-reasoning mode, DeepSeek-V3.1 with standard inference parameters, the OpenAI API to access GPT-4.1-mini (snapshot released on *April 14, 2025*), and Gemini-2.5-Flash through its official API. These models were selected to represent different families of LLMs, covering diverse architectural designs and training paradigms. This setup enables a comprehensive assessment of model performance on multi-turn Text-to-OverpassQL tasks, providing insights into their ability to capture contextual dependencies and generate executable queries.

B.1 Finetune

For model adaptation, we fine-tuned ByT5-Large² on our constructed dataset. Each input sequence

²<https://huggingface.co/google/byt5-large>

consisted of the natural language query enriched with dialogue context, while the target sequence was the corresponding OverpassQL query. Both inputs and outputs were truncated or padded to a maximum length of 512 tokens. Training was performed using the HuggingFace Transformers framework with the Adam optimizer, a learning rate of 3×10^{-5} , and a batch size of 4 per device. The model was trained for 4 epochs with a 90/10 train-validation split, and the best checkpoint was selected based on validation performance. Gradient accumulation was used for stability, bfloat16 precision was enabled, and checkpointing was conducted at the end of each epoch.

All experiments were run on a single NVIDIA L40 GPU with 46,068 MiB memory, requiring approximately 5 hours to complete fine-tuning.

C Case study

C.1 Data generation

In the third stage, where large language models were tasked with selecting and organizing samples from the extended data to construct multi-turn dialogues, we identified several issues. These included incoherent context transitions, overly independent turns, and weak inter-turn dependencies, which undermine the objective of modeling natural conversational flows. Representative examples of such shortcomings are illustrated in Figure 7. To mitigate these problems, we iteratively refined the prompt design and provided high-quality exemplars, thereby guiding the models to generate dialogues that are more fluent, coherent, and strongly contextually linked.

C.1.1 Illustrative case: subclause extraction with set linkage

As discussed in Section 3, complex modifications often decompose a single OverpassQL program into multiple turns. A common failure mode is breaking clause-to-clause linkage via set variables: e.g., an early turn defines an intermediate set, while a later turn filters, unions, or aggregates over that set. If the extraction order is incorrect or the set name is not propagated consistently across turns, the resulting query can remain syntactically well-formed but become non-executable due to an undefined set reference; even when all sets are defined, an incorrect extraction order can lead to subtle semantic inconsistencies across subclauses. This is a key reason why subclause extraction and reordering

are notably harder than local edits such as changing output settings or editing key–value filters.

C.2 Model evaluation

During our experiments, we observed several limitations of current large language models (LLMs) in generating OverpassQL queries within multi-turn dialogues. First, the models sometimes produce outputs that do not conform to OverpassQL syntax, resulting in invalid or unusable data. This issue is compounded by a low degree of relevance to the preceding dialogue context. Second, in multi-turn settings, the models frequently fail to generate complete query results, producing outputs that only supplement previous data or contain partial information, thereby yielding incomplete representations of the intended requests. Examples illustrating these issues are shown in Figures 5 and 6, highlighting both syntactic errors and the incompleteness of dialogue-driven query generation.

From our experiment logs, we found several recurring causes of invalid or non-executable OverpassQL outputs: (1) missing or inconsistent set references across subclauses (e.g., using an undefined intermediate set in later statements), (2) incomplete or malformed clauses/blocks (e.g., unbalanced brackets or missing keywords), and (3) inconsistent tag filters or spatial scopes across turns that lead to contradictory constraints. While we do not provide a fully systematic categorization due to scope constraints, these patterns are representative of the main challenges that LLMs face in multi-turn OverpassQL generation.

D Metrics

- **Character F-score (chrF)** chrF is an evaluation metric that measures text similarity based on character n-gram precision and recall, producing an F-score that reflects overall quality. Because chrF operates on the character-level, it is well suited for Overpass queries which consist of words and special characters alike.
- **Key-Value Similarity (KVS)** Key-Value Similarity (KVS) is an evaluation metric that measures the alignment between predicted and reference queries by comparing their key-value tag pairs, thereby assessing semantic consistency. The definition of KVS is as follows:

$$KVS(q_G, q_R) = \frac{|KV(q_G) \cap KV(q_R)|}{\max(|KV(q_G)|, |KV(q_R)|)} \quad (1)$$

where the operator $KV(\cdot)$ returns all key-value pairs as well as all individual keys and all individual values. This metric captures the semantic relatedness of two queries.

- **XML-tree Similarity (TreeS)** The Tree Similarity metric (TreeS) is computed by comparing the XML tree representation of the queries. All key-value pairs and variable names are removed, and the number of matching subtrees between the generated and reference query is recursively calculated.
- **Overpass Query Similarity (OQS)** OQS is designed to quantify the overall compatibility between generated and reference queries. It integrates three complementary metrics—chrF, KVS, and TreeS, by combining them into a single weighted score. In this work, equal weights (1:1:1) are assigned to the three components, ensuring a balanced evaluation of lexical similarity, key-value accuracy, and structural alignment.
- **Exact match (EM)** Exact String Match (EM) evaluates whether the generated query exactly matches the reference query at the string level, providing a strict measure of correctness.
- **Execution accuracy (EX)** Execution Accuracy (EX) measures whether the generated query, when executed, yields the same results as the reference query, thus assessing functional correctness beyond surface-level similarity.
- **Soft Execution Accuracy (EXSOFT)** It is computed as the overlap of returned elements, normalized by the maximum number of elements returned by either the generated or reference query.

E Dataset Examples

Figures 8- 10 present representative dataset examples, including three of simple complexity, two of medium complexity, and two of complex complexity.

F OverpassQL Composition

A brief introduction to OverpassQL Statement composition.

- **Settings:** Settings must be declared in the first non-commented statement of the Overpass QL source code. These typically include output type settings, timeout settings, maximum number of returned elements, and more.
- **Block Statement:** A block statement is a special control structure used to define a set of commands that will be executed within a specific context. Block statements help organize complex queries, particularly when dealing with recursive queries, subqueries, or the need for temporary variables. Common types include union blocks, recursive blocks, loop blocks, difference blocks, and others.
- **Standalone Statement:** A standalone statement refers to statements that can exist and execute independently, without needing to run within the context of other statements. The primary example is the query statement, which filters and selects elements that meet specified conditions using tags.
- **Filters:** Filter statements are typically used within query statements to filter the selection of elements. Filtering conditions often include tags, bounding boxes, element IDs, and others.
- **Evaluators:** Evaluators are tools used to calculate or evaluate expressions within a query. These expressions can be used for filtering, sorting, or performing other operations based on attributes and conditions. Evaluators determine whether objects meet specific conditions by assessing their attribute values or other expressions during the query process, guiding further operations or output.

```

{
  "conversation": [
    {
      "nl": "Can you show me all ways where motor vehicles are not allowed?",
      "overpassql":
      "[out:json][timeout:25];way[\"access\"=\\"no\"][\"motor_vehicle\"=\\"yes\"];way[\"access\"
      =\"no\"][\"motor_car\"=\\"yes\"];way[\"access\"=\\"no\"][\"vehicle\"=\\"yes\"];out;",
      "response": "ways with motor_vehicle = no;"
    },
    {
      "nl": "Can you also include their related nodes and details?",
      "overpassql":
      "[out:json][timeout:25];(way[\"access\"=\\"no\"][\"motor_vehicle\"=\\"yes\"];way[\"access\"
      =\"no\"][\"motor_car\"=\\"yes\"];way[\"access\"=\\"no\"][\"vehicle\"=\\"yes\"]);out;>;out skel
      qt;",
      "response": "ways with motor_vehicle = no { out skel qt; }"
    }
  ],
}

```

Figure 5: Example of LLMs failing to generate data correctly.

```

{
  "conversation": [
    {
      "nl": "Can you show me bicycle-designated paths?",
      "overpassql": "[out:json][timeout:25];way[\"highway\"=\\"path\"][\"bicycle\"=\\"designated\"
      ](bbox);out;" ,
      "response":
      "way[\"highway\"=\\"path\"][\"bicycle\"=\\"designated\"](bbox)->.bike;(way.bike;- (way.bik
      e[\"foot\"=\\"yes\"];way.bike[\"foot\"=\\"designated\"]););out meta geom;
      {bbox=area:3606195356}"
    },
    {
      "nl": "Do any of them allow foot traffic?",
      "overpassql":
      "[out:json][timeout:25];(way[\"highway\"=\\"path\"][\"bicycle\"=\\"designated\"](bbox));w
      ay[\"highway\"=\\"path\"][\"bicycle\"=\\"designated\"](bbox););out;>;out skel qt;",
      "response": "way.bike[\"foot\"=\\"yes\"];way.bike[\"foot\"=\\"designated\"];out;"
    },
    {
      "nl": "I want to see all the designated bicycle paths from before as well.",
      "overpassql":
      "[out:json][timeout:25];(way[\"bicycle\"=\\"designated\"](bbox));way[\"bicycle\"=\\"design
      ated\"](bbox);way[\"bicycle\"=\\"designated\"](bbox);way[\"bicycle\"=\\"designated\"](bb
      ox)););out;>;out skel qt;",
      "response": ".bike;out;"
    }
  ],
}

```

Figure 6: Example of LLMs generating only supplementary code.

```

{
  "conversation": [
    {
      "nl": "Can you show me all hospitals with intensive care units in Germany?",
      "overpassql":
        "[out:json][timeout:60];{{geocodeArea:"Germany"}}->.searchArea;(nwr["amenity"="hospital"]["healthcare:speciality"~"intensive"](area.searchArea));out center;>;out skel qt;"
    },
    {
      "nl": "Now show me all hospitals with intensive care units in Maxéville",
      "overpassql":
        "[out:json][timeout:60];{{geocodeArea:"Maxéville"}}->.searchArea;(nwr["amenity"="hospital"]["healthcare:speciality"~"intensive"](area.searchArea));out center;>;out skel qt;"
    }
  ],
}

      ↓                ↓

{
  "conversation": [
    {
      "nl": "Can you show me all hospitals with intensive care units in Germany?",
      "overpassql":
        "[out:json][timeout:60];{{geocodeArea:"Germany"}}->.searchArea;(nwr["amenity"="hospital"]["healthcare:speciality"~"intensive"](area.searchArea));out center;>;out skel qt;"
    },
    {
      "nl": "What about those specifically in Maxéville?",
      "overpassql":
        "[out:json][timeout:60];{{geocodeArea:"Maxéville"}}->.searchArea;(nwr["amenity"="hospital"]["healthcare:speciality"~"intensive"](area.searchArea));out center;>;out skel qt;"
    }
  ],
}

```

Figure 7: Example of LLMs generating only supplementary code.

```

{
  "conversation": [{
    "nl": "Can you find restaurants within this area?",
    "overpassql":
      "(node[\"amenity\"=\\"restaurant\"](bbox));way[\"amenity\"=\\"restaurant\"](bbox);rel
ation[\"amenity\"=\\"restaurant\"](bbox););out;>;out skel qt;"
  },
  {
    "nl": "Show me only the nodes and ways from those restaurants.",
    "overpassql":
      "node[\"amenity\"=\\"restaurant\"](bbox);way[\"amenity\"=\\"restaurant\"](bbox);out;"
  }
  ],
  "complexity": "simple"
},
-----
{
  "conversation": [{
    "nl": "Can you show me all universities and colleges currently mapped?",
    "overpassql":
      "[out:json];(relation[\"amenity\"=\\"university\"];relation[\"amenity\"=\\"college\"];node[\"a
menity\"=\\"university\"];node[\"amenity\"=\\"college\"];way[\"amenity\"=\\"university\"];way[\"
amenity\"=\\"college\"];);out tags qt bb;"
  },
  {
    "nl": "Can you also include their metadata as it was on February 1, 2013?",
    "overpassql":
      "[out:json][date:\\"2013-02-01T05:18:54Z\"];(relation[\"amenity\"=\\"university\"];relation[
\"amenity\"=\\"college\"];node[\"amenity\"=\\"university\"];node[\"amenity\"=\\"college\"];way[\"
amenity\"=\\"university\"];way[\"amenity\"=\\"college\"];);out tags qt bb;"
  }
  ],
  "complexity": "simple"
},
-----
{
  "conversation": [{
    "nl": "Can you find all supermarkets in Bordeaux Métropole?",
    "overpassql":
      "[out:json];area[\"name\"=\\"Bordeaux
Métropole\"]->.searchArea;(node[\"shop\"=\\"supermarket\"](area.searchArea);way[\"shop\"=\\"
supermarket\"](area.searchArea));out center;>;out skel qt;"
  },
  {
    "nl": "Show only the nodes and ways without the area context.",
    "overpassql":
      "[out:json];node[\"shop\"=\\"supermarket\"](.searchArea);way[\"shop\"=\\"supermarket\"](.
searchArea);out;"
  }
  ],
  "complexity": "simple"
}

```

Figure 8: Example of simple complexity data.

```

{
"conversation": [
  {
    "nl": "Can you show me the recent changes to way 10074543 in the last day?",
    "overpassql":
      "[out:xml][timeout:600];way(uid:10074543)(newer:\\"{{date:1day}}\");node(uid:10074543)(newer:\\"{{date:1day}}\");out;"
  },
  {
    "nl": "Can you include detailed metadata for those changes?",
    "overpassql":
      "[out:xml][timeout:600];(way(uid:10074543)(newer:\\"{{date:1day}}\");node(uid:10074543)(newer:\\"{{date:1day}}\"));out meta;>;out meta qt;"
  },
  {
    "nl": "Can you give me the same with the exact timestamp of April 28, 2021, 14:47:01 UTC?",
    "overpassql":
      "[out:xml][timeout:600][date:\\"2021-04-28T14:47:01Z\"];(way(uid:10074543)(newer:\\"{{date:1day}}\");node(uid:10074543)(newer:\\"{{date:1day}}\"));out meta;>;out meta qt;"
  }
],
"complexity": "medium"
},
-----
{
"conversation": [
  {
    "nl": "Can you show me all the natural bedrock features within this bounding box?",
    "overpassql":
      "[out:json][timeout:25];node[\\"natural\\=\"bedrock\\"]({{bbox}});way[\\"natural\\=\"bedrock\\"]({{bbox}});relation[\\"natural\\=\"bedrock\\"]({{bbox}});out;"
  },
  {
    "nl": "Could you include the detailed geometry of these bedrock features as well?",
    "overpassql":
      "[out:json][timeout:25];(node[\\"natural\\=\"bedrock\\"]({{bbox}});way[\\"natural\\=\"bedrock\\"]({{bbox}});relation[\\"natural\\=\"bedrock\\"]({{bbox}}));out;>;out skel qt;"
  },
  {
    "nl": "I want to see only those features that were added or updated since May 15, 2007.",
    "overpassql":
      "[out:json][timeout:25][date:\\"2007-05-15T16:11:54Z\"];(node[\\"natural\\=\"bedrock\\"]({{bbox}});way[\\"natural\\=\"bedrock\\"]({{bbox}});relation[\\"natural\\=\"bedrock\\"]({{bbox}}));out;>;out skel qt;"
  }
],
"complexity": "medium"
}

```

Figure 9: Example of medium complexity data.

```

{
"conversation": [{
  "nl": "How many swimming pools are there in France?",
  "overpassql":
    "[out:xml][timeout:125];{{geocodeArea:"France"}}->.searchArea;nwr["amenity"="swimming_pool"](area.searchArea)->.b;out count;"
}],
{
  "nl": "How many of them do not have a leisure tag or a name tag?",
  "overpassql":
    "[out:xml][timeout:125];{{geocodeArea:"France"}}->.searchArea;nwr["amenity"="swimming_pool"](area.searchArea)->.b;nwr["amenity"="swimming_pool"]["leisure"!~\.*\"]["name"!~\.*\"](area.searchArea)->.a;(nwr.b;-nwr.a);out count;"
}],
{
  "nl": "Of those, how many are not actually separate buildings?",
  "overpassql":
    "[out:xml][timeout:125];{{geocodeArea:"France"}}->.searchArea;nwr["amenity"="swimming_pool"](area.searchArea)->.b;nwr["amenity"="swimming_pool"]["leisure"!~\.*\"]["name"!~\.*\"](area.searchArea)->.a;(nwr.b;-nwr.a["building"!~\.*\"]);out count;"
}],
"complexity": "complex"
},
-----
{
"conversation": [{
  "nl": "Can you show me all power-related infrastructure in Seattle?",
  "overpassql":
    "[out:xml][timeout:100];{{geocodeArea:"Seattle"}}->.searchArea;(node["power"](area.searchArea);way["power"](area.searchArea);relation["power"](area.searchArea));out meta;>;out meta qt;"
}],
{
  "nl": "Can you give me only the nodes among them?",
  "overpassql":
    "[out:xml][timeout:100];{{geocodeArea:"Seattle"}}->.searchArea;node["power"](area.searchArea);out;"
}],
{
  "nl": "Now show me all of these in India instead, including ways and relations.",
  "overpassql":
    "[out:xml][timeout:100];{{geocodeArea:"India"}}->.searchArea;(node["power"](area.searchArea);way["power"](area.searchArea);relation["power"](area.searchArea));out meta;>;out meta qt;"
}],
{
  "nl": "Can you repeat that but only including data that was added after April 5, 2006?",
  "overpassql":
    "[out:xml][timeout:100][date:"2006-04-05T14:57:23Z"];{{geocodeArea:"India"}}->.searchArea;(node["power"](area.searchArea);way["power"](area.searchArea);relation["power"](area.searchArea));out meta;>;out meta qt;"
}],
"complexity": "complex"
}

```

Figure 10: Examples of complexity data.

Task Description:

You are an OverpassQL map search expert. Based on my requests and questions, generate the corresponding OverpassQL query statements. Please follow the steps below:

1. Understand the natural language query:
 - Extract key locations, query targets (such as castles, tram lines, etc.), and any additional conditions (such as time, service type, specific attributes, etc.).
2. Generate the OverpassQL query statement:
 - Construct the appropriate OverpassQL query based on the extracted information.
 - Ensure the query is efficient and accurately retrieves the required geographic data.
 - If needed, include output optimizations (like `out skel qt`) to ensure query efficiency.
 - You only need to generate the corresponding OverpassQL query without any explanation!
 - Don't have any comments in the OverpassQL query!
 - Output only plain text!

Task Objective:

Generate OverpassQL queries that are in accordance with the specifications, ensuring they can execute correctly and return the desired data.

Conversation Begins:

Now, please answer the following queries with the correct OverpassQL syntax:

Figure 11: Prompt used to generate answers for LLMs.

You are an expert in constructing high-quality conversational datasets from OverpassQL queries. You will be given a list of OverpassQL queries. Your task is to extract **one coherent multi-turn conversation** based on these queries.

Each turn in the conversation should consist of:

- A realistic **natural language question** ("`\n`") a user might ask
- The corresponding **OverpassQL query** ("`overpassql`")

Later turns should build on earlier ones — for example, by narrowing the query, changing filters, or adding spatial constraints.

At the end, assign a **complexity label** to the entire conversation: `"simple"`, `"medium"`, or `"complex"`.

Output Format

Output a **JSON array**, where each object represents a complete multi-turn conversation with the following fields:

- `"conversation"`: a list of turns, each turn containing:
 - `"nl"`: user natural language question
 - `"overpassql"`: corresponding OverpassQL query
- `"complexity"`: one of `"simple"`, `"medium"`, or `"complex"`

Restrictions

- You don't need to use all the queries.
- Each conversation should have at least 2 turns.
- Need concise questions and reflect relevance, to be precise, except the first round, only pronouns are needed to refer to repeated words in each round, each round of questioning needs to be combined with the previous round of questioning to get the result, and it is impossible to get the result alone!

Example Output

Json

...

```
{example_output}
```

...

Below is the list of OverpassQL queries to choose from:

...

```
{queries}
```

...

Generate a JSON array without any explanation.

Figure 12: Prompt used to generate datasets for LLMs.