OverpassNL: A Community-Generated Dataset and Real-World Semantic Parser for OpenStreetMap

Anonymous ACL submission

Abstract

We present OverpassNL, a complex dataset that pairs queries to the OpenStreetMap (OSM) database with natural language questions. It is based on nearly 10,000 queries issued by OSM users and developers in the Overpass query language. The Overpass queries were translated into suitable natural language forms by 15 trained computational linguistics students. The resulting dataset can be used as training data for real-world semantic parsing. The complexity of OverpassNL stems from both the nature of real-world queries and the expansive underlying OSM database. While existing semantic parsing datasets such as Spider (Yu et al., 2018) use formulaic synthetic queries and achieve complexity by combining multiple simple underlying databases, there is no natural split into database schemata in OSM (Yu et al., 2018) nor does Overpass provide a clear structure for slot-filling (Yao et al., 2019). The complexity of the task is shown by the mere 21% execution accuracy achieved by a generic neural semantic parser. We enhance the model by using different types of additional information and by training data augmentation, thereby increasing the performance to 36% execution accuracy.

1 Introduction

Semantic Parsing allows the mapping of natural language queries into a corresponding structural form. This form can be a structured query language like SQL (Yu et al., 2018), a programming language like Bash (Lin et al., 2018), If-Then recipes (Quirk et al., 2015), or a NoSQL language like Overpass[1] – a query language for the real-world, large-scale, and widely-used OpenStreetMap (OSM) database of geographic information.[2]

Most existing semantic parsing datasets face the constraint that they only use a single database with a small number of tables, thus limiting the number of possible queries. For example, GeoQuery (Zelle and Mooney, 1996; Iyer et al., 2017) contains eight tables, Restaurant (Tang and Mooney, 2000; Popescu et al., 2003) three tables, and IMDB (Yaghmazadeh et al., 2017) sixteen tables. Recent works such as Zhong et al. (2017) or Yu et al. (2018) try to alleviate this problem by combining multiple different databases found on the internet. The combined databases comprise different tasks, thus the meta-database consists of smaller, independent databases. Taking the respective database schemata into account during training and testing allows for a drastic reduction of the complexity of semantic parsing (Zhang et al., 2019). Such simplifications by schema information cannot be exploited for real-world semantic parsing of the OSM database. OSM consists of element types (nodes, ways, and relations), each with associated database information such as *current, history, current_tags*, and *history_tags*. It thus comprises 39 tables that are combined into one large and complex database instead of a meta-database that consists of several different and unrelated databases.

Moreover, our dataset OverpassNL is generated from complex real-world queries of users and developers using the Overpass API. In contrast, existing datasets such as Spider start from artificial formulaic questions from which queries are generated by computer science students. We created OverpassNL by letting computational linguistic students create natural language counterparts for real-world Overpass queries. We acquired those queries using Overpass Turbo[3], an online visualization tool for Overpass that exploits the full expressivity of the Overpass language. All annotators received training and went through a test to ensure high quality annotations. The resulting dataset consists of

[1] https://wiki.openstreetmap.org/wiki/Overpass_API

[2] https://www.openstreetmap.org. For statistics on usage and database, see Table 5 and Figure 4 in the Appendix.

[3] https://overpass-turbo.eu/

nearly 10,000 Overpass queries, each accompanied by a natural language question.[4]

We use OverpassNL to train a semantic parser that allows access to the OSM database via natural language questions. A state-of-the-art sequence-to-sequence model trained on our dataset achieves 21% execution accuracy, showing that semantic parsing of our dataset is indeed a challenging task. Since we cannot take advantage of additional information like the database schema, we increase the performance of our model by 1) retrieving similar examples from the training data as additional inputs, 2) clustering the data and augmenting them with the resulting cluster information, 3) extracting key-value pairs by fuzzy matching the natural language questions to an already existing knowledge source and 4) creating a synthetic dataset for further data augmentation. The best combination of these data enhancement techniques achieves an significant increase in execution accuracy to 36%.

2 Related Work

Text-to-SQL parsing has been popular since the 1990s. Many different datasets have been created, e.g., ATIS (Dahl et al., 1994; Iyer et al., 2017), GeoQuery (Zelle and Mooney, 1996) and WikiSQL (Zhong et al., 2017), each with their own shortcomings, like only using a single database each. This problem was addressed by Yu et al. (2018), who created a collection of databases pairing natural language questions to SQL queries. This dataset was later extended further into the contextual setting by Yu et al. (2019b) and into the conversational or interactive setting by Yu et al. (2019a). Another semantic parsing task turns natural language expressions into If-Then recipes (Quirk et al., 2015) which connect actions (*starting an alarm*) to triggers (*specific time is reached*). This task has been extended into an interactive setting by Yao et al. (2019). In the field of OpenStreetMap semantic parsing, preliminary work was already done by Haas and Riezler (2016) who translated natural language queries into a self-designed Machine Readable Language. We do not follow this approach since Overpass presents a more expressive language that is used by the OSM community.

A crucial difference of our dataset to existing semantic parsing data is that it consists of real-world queries issued by users and developers trying to satisfy a genuine information-seeking task by executing a query against a large-scale database of geographical information (see Section 3.2).

[4]Data will be downloadable under http://anon-link upon acceptance of the paper.

Variable	ONL	Spider	WikiSQL	ATIS
NL len.	9.35	12.09	12.27	10.47
Vocab. size	11,259	7,230	29,857	950
Query len.	203	116	57	1,020
String ops.	24%	2%	0%	0%
# DBs	1	200	26,521	1
# tables/DB	38	5.1	1	32

Table 1: Analysis of the complexity in OverpassNL (ONL) versus other datasets. Natural language question length (*NL len.*) and query length (*Query len.*) are averaged. *String ops.* refers to string operations like regular expressions.

3 OverpassNL Dataset

3.1 Dataset Creation

We extracted all 150,000 queries that were logged on the Overpass Turbo API with no pre-selection procedures. We filtered out duplicates, which left us with around 50,000 examples. A randomly selected 10,000 of these were manually annotated. The queries are therefore "standard" representative user queries. We hired 15 computational linguistics students for annotation of database queries with natural language questions. The annotators received a tutorial, solved some training examples and completed a test to ensure they understood the task. Then they were shown random examples of queries and results using the annotation interface shown in Figure 1. The task of the annotators was to create natural language question corresponding to the given Overpass query. This resulted in a dataset of 9,609 paired question-parse pairs. We separated those into train (7,109), dev (1,500) and test data (1,000). An example of a query-question pair is as follows:

question *Ways with "name" tag containing values "Power" or "power" edited by user with ID 2041564 in the Philippines*

query
```
[out:json];( {{geocodeArea:
Philippines}} ->.searchArea; way["
name"~"^.*[Pp]ower"] (uid
:2041564) (area.searchArea);); out
body;>;out skel;
```

This approach to database creation has two main advantages: First, teaching annotators to interpret

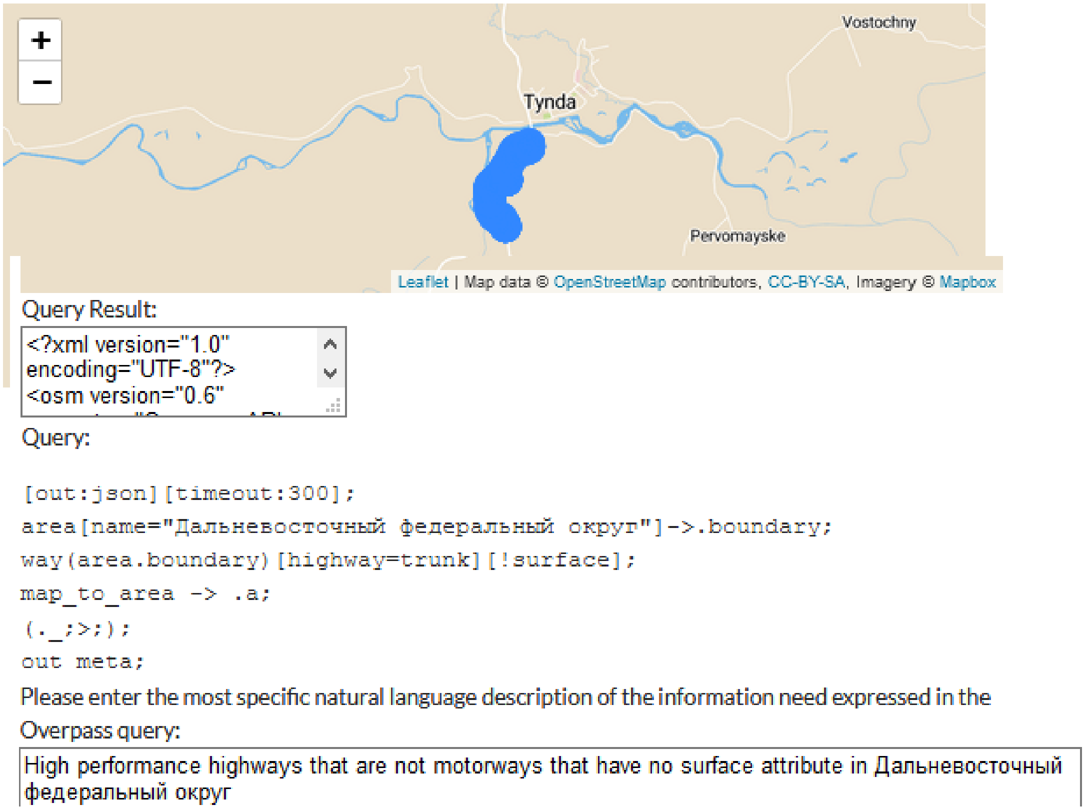


Figure 1: Annotation interface showing the query that needs to be translated is in the middle of the figure, and the output of the query on top. An example translation produced by an annotator is shown at the bottom.

existing queries into natural language is easier than training them to produce queries in the Overpass language. Using the existing Overpass queries is therefore a way to efficiently create a dataset of paired question-query tuples. Second, the original queries were entered by developers and users, thus the queries satisfy a real-world information need and exploit the full expressivity of Overpass instead of being based on the annotators' limited knowledge of the Overpass language.

3.2 Complexity of Semantic Parsing Data

As the example in Section 3.1 shows, queries in OverpassNL often make use of regular expressions. In contrast, queries in the Spider dataset (Yu et al., 2018) consist only of simple string matching operations, such as strings starting, containing or ending with a specific (sub)string. ATIS (Dahl et al., 1994; Iyer et al., 2017) and WikiSQL (Zhong et al., 2017) queries do not even contain string operations, but only exact matches. Statistics comparing dataset complexity of OverpassNL to Spider, WikiSQL, and ATIS are given in Table 1. All of these properties show that OverpassNL offers a setting that has been lacking in research so far. We work with a new query language with its own challenges, such as regular expressions and the NoSQL-style that allows concise queries against a complicated database. Moreover, the underlying database consists of only one highly connected database, making it possible to issue many different queries, resulting in a high vocabulary size.

4 (Neural) Semantic Parsing

In addition to the dataset, we also present a first cut on semantic parsing, showcasing the complexity of the talk. We first employ a generic sequence-to-sequence neural network (Sutskever et al., 2014) the encoder-decoder variant from (Luong et al., 2015). We use Joey NMT (Kreutzer et al., 2019) as framework to build the baseline parser.

Given a dataset $D = \{(X_n, y_n)\}_{n=1}^{N}$ of natural language questions X and corresponding queries y, standard supervised training is performed by minimizing the average cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\log p(y_{n,t} \mid y_{n,<t}, X_n), \quad (1)$$

where the sum is over all timesteps $t = 1$ to $t = T_n$

for sample n.

The natural language question is fed into a bi-directional RNN (GRU) to generate the hidden states $h \in \mathbb{R}^{|X| \times m}$, where $|X|$ is the number of source inputs and m is the hidden state size. The decoder takes its previous hidden state s_{t-1} and calculates a context vector c_t with an attention mechanism (Bahdanau et al., 2015) such that $c_t = \text{att}(s_{t-1}, h)$. This context vector is then used for prediction by passing it through another feed-forward and softmax layer to generate the output distribution. Meta-parameter settings used in our experiments can be found in Table 6 in the Appendix.

5 Evaluation Measures

We use a parseval-style (Black et al., 1991) evaluation metric that matches a generated query q_{pred} against a gold standard parse q_{gold} and counts how often the predicted key-value pairs $\text{kv}(q_{\text{pred}})$ match their counterparts $\text{kv}(q_{\text{gold}})$ in the gold standard parse. This is similar to the component matching done in Yu et al. (2018). Our `parse_match` metric is based on the Dice Coefficient (Dice, 1945) where the key-value pairs in predicted and gold parse is measured:

$$\text{parse\_match} = \frac{1}{|\text{Q}|} \sum_{q \in \text{Q}} \frac{|\text{kv}(q_{\text{pred}}) \cap \text{kv}(q_{\text{gold}})|}{\max(|\text{kv}(q_{\text{pred}})|, |\text{kv}(q_{\text{gold}})|)}. \tag{2}$$

Furthermore, we use a grounded evaluation metric that executes the queries against the OpenStreetMap database and computes an execution accuracy by matching the predicted results against the correct result. It is computed as follows:

$$\text{exec\_acc} = \frac{\sum_{q \in \text{Q}} \delta(\text{res}(q_{\text{gold}}), \text{res}(q_{\text{pred}}))}{|\text{Q}|}, \tag{3}$$

where $\delta(i, j)$ is the Kronecker delta and $\text{res}(q)$ is a function that executes the query q and returns the results.

However, sometimes a hypothesis query executes, but produces only a part of the correct output. Therefore we use an additional metric that computes a `part_exec` average over partially correct query results:

$$\text{part\_exec} = \frac{1}{|\text{Q}|} \sum_{q \in \text{Q}} \frac{|\text{res}(q_{\text{gold}}) \cap \text{res}(q_{\text{pred}})|}{|\text{res}(q_{\text{gold}})|}. \tag{4}$$

6 Experiments

6.1 Experimental Setup

A state-of-the-art sequence-to-sequence model trained on the dataset achieves an execution accuracy of 21% when executing the predicted queries against the OSM database, showing that semantic parsing of the OverpassNL dataset is indeed a challenging task. We find that the difficulty stems from three sources: 1) The correct use of database keys and values, since a database schema cannot be provided; 2) The complex syntax of Overpass queries; 3) The limited size of the dataset. An example of a predicted and gold parse for a natural language question can be found in Table 7 in the Appendix. Our goal is to solve these problems by the following three approaches:

1. **db_info**: Adding additional information such as possible database keys and values to the model input (countering difficulty 1).

2. **query_templates**: Providing templates to help with the difficult syntax by retrieving similar examples from the training data or by clustering the data and providing the cluster ID (countering difficulty 2).

3. **data_augmentation**: Creating a synthetic silver training dataset by templating and substituting tokens in questions and queries (countering difficulty 3).

6.1.1 Database Information

The OSM database is accessible through Overpass using *keys* and *values*. However, it is hard for the model to find the correct key for a value that appears in a natural question because the keys are often very general and cannot simply be inferred from the value. To avoid this difficulty, we aim to find the corresponding keys through string matching in order to provide the keys and values along with the input question. As shown in the example below, the keys and values are simply appended to the input string with a `[SEP]` token separating the real natural language question and the additional information. Keys and values are marked with `[K]` and `[V]`, respectively:

Charging stations *around* ***motorway*** *A 8 in Germany.* `[SEP]` `[K]` *amenity* `[V]` *charging_station* `[K]` *highway* `[V]` *motorway* `[SEP]`

Data	#Examples with add. info	Percentage
train	4117	58.76 %
dev	576	38.40 %
test	572	57.20 %

Table 2: Statistics about Additional Information (*keys and values* that was added to the data

The approach `db_info` makes use of a nominatim table[5]. This table maps OSM entries to categories to be used as keys and values in Overpass. Similar to Lin et al. (2020), we apply a fuzzy string matching algorithm to obtain the additional information from the nominatim table. The exact algorithm is explained in Appendix A.6. Naturally, matches can only be found if the word in the natural language question (± two characters) appears in the nominatim table. This is not the case for all examples in the OverpassNL dataset: Overall, for the train and test set, keys and values could only be added in around 60 % of the cases. Exact numbers can be found in Table 2.

6.1.2 Query Templates

In approach `retrieve`, we follow Hashimoto et al. (2018) to retrieve for every natural language question x the most similar question-query pair (x', y') from the training data, using BERTScore (Zhang et al., 2020) as similarity metric. These additional inputs are fed into different encoders with their own attention mechanisms. The output of the encoders are then concatenated in the order x, x', y' and fed into the decoder, turning this into a multi-source setup (Zoph and Knight, 2016). This gives the model access to a similar question-query pair through the additional encoded input.

In approach `cluster`, we provide the model with additional information about the type of query. This approach is inspired by previous approaches to use control tags as additional inputs (Sennrich et al., 2016a). We first embed the natural language questions with BERT (Devlin et al., 2019), cluster the data with the k-Means clustering algorithm (k=10), and then augment the data with a special tag indicating the corresponding cluster. For example, similar natural language questions like *planetarium in current view* and *places of worship in current view* will be assigned to the same cluster. Examples for clusters are given in Fig. 6 in the Appendix.

[5] `https://wiki.openstreetmap.org/wiki/Nominatim/Special_Phrases/EN`

gold question *Recycling in admin level 10 areas with the name Kupferdreh*

silver question *Restaurants in admin level 5 areas with the name La Vida*

gold query
```
area["name"="Kupferdreh"]
  [admin_level=10]->.a;
  (node(area.a)
  ["amenity"="recycling"];);
```

silver query
```
area["name"="La Vida"]
  [admin_level=5]->.a;
  (node(area.a)
  ["amenity"="restaurant"];);
```

Figure 2: Example for a silver example creation. The underlined part in the gold example is replaced by random values from the training data to create the silver data.

6.1.3 Data Augmentation

Lastly, we conduct two further data augmentation strategies. In approach `substitution`, we generate silver data by jointly templating both natural language questions and queries, replacing tokens occurring in both question and query. Afterwards we insert random values from the training data into the template slots. This resulting data was then filtered by removing nonsense natural language questions according to their sentence probability predicted by GPT-2 (Radford et al., 2018). The sentence probability was normalized by sentence length and thresholded with a value of 0.0001. A full example can be seen in Figure 2. The final silver dataset contains 14,000 examples and is used on its own or combined with the other approaches.

In approach `back-trans`, we made use an approach inspired by backtranslation (Sennrich et al., 2016b). We use the existing question-query pairs to train a query-to-question model that was then used to generate natural language questions for queries that were not given manual annotations with questions. This process resulted in additional 19,000 data points that were added to the training data.

6.2 Experimental Results

As shown in the top part of Table 3, the baseline performance of our model achieves only 21% execution accuracy. Using approach `retrieve` to retrieve similar question-query pairs through the additional encoded input increases the model per-

Approach	Model		exec_acc	part_exec	parse_match
-	baseline		0.21	0.43	0.22
Database Information	db_info		0.33	**0.62**	0.35
Query Templates	cluster		0.35	0.6	0.35
	retrieve		0.33	0.57	0.3
Data Augmentation	back_trans		0.33	**0.62**	0.35
	substitution		0.35	0.6	0.31
Combined	cluster	db_info	**0.36**	**0.62**	**0.37**
Combined	cluster	retrieve	0.35	0.61	0.33
Combined	cluster	substitution	0.34	0.61	0.32

Table 3: Accuracy in percent of different semantic parsing models: A baseline, enhanced by retrieving similar question-query pairs (+ `retrieve`), augmenting the data with special cluster tags (+ `cluster`) and adding more training data using automatic generated data (+ `substitution` or + `back_trans`). All results are significantly better than the baseline ($p < 0.001$).

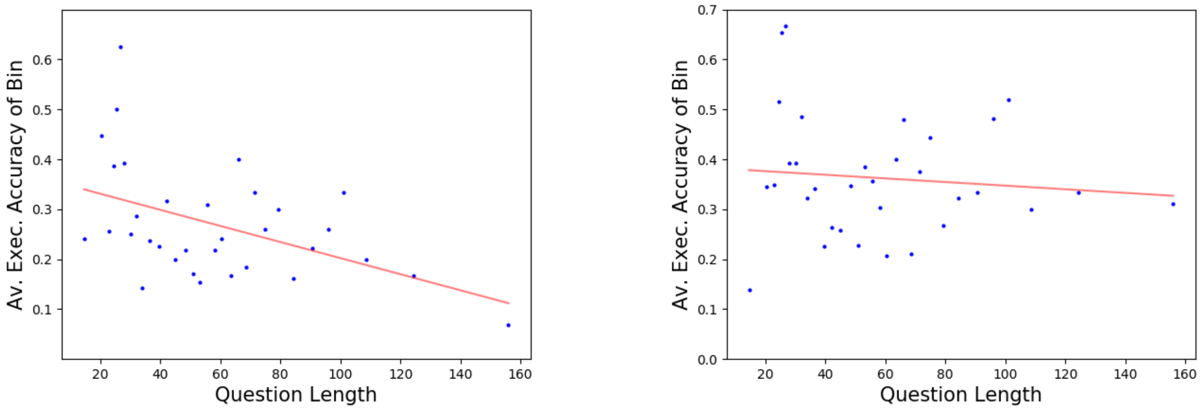
formance by 12 points to 33% execution accuracy. The `db_info` approach reaches the same performance. Allowing the model to easily generate similar queries by using approach `cluster` also leads to a better performance with 35% execution accuracy. Finally, using approach `substitution` to add the silver data to our training data, the model also achieves 35% execution accuracy.

The bottom part of Table 3 shows the results for the best combinations of approaches. Combining `cluster` with either `retrieve` or `substitute` achieves a score of 35% execution accuracy. Combining `cluster` and `db_info` yields the highest improvement, reaching 36% execution accuracy. Combining all three methods does not lead to further improvements. We conjecture that this result can be explained by a certain amount of redundancy in the information provided by retrieving similar instances or adding silver data, with the most accurate addition to the `cluster` information being provided by the explicit keys and values in the `db_info` approach. This combination also reaches the highest values according to the partial execution and parse match metrics.

In order to investigate the interaction of data properties and parsing performance, we took a closer look at the data characteristics of question length. Our hypothesis was that the dataset poses increased difficulties due to increased question length: The longer the question, the harder to find the correct query. In order to test this hypothesis, we use an LMEM-based significance test (Riezler and Hagmann, 2022) to investigate the interaction between the question length and the execution accuracy. With a p-value of < 0.01, question length makes a significant difference. This can be confirmed by fitting a line to the results split by question length, as can be seen in Figure 3. The negative gradient confirms our observation. An advantage of the best model (`db_info + cluster`) is that it seems to close the gap between the performance difference of long and short examples. As Figure 3 shows, the base model (left) performs worse the longer the natural language question gets. However, the best model seems to perform equally well independent of the question length. As the p-value shows, the line of best fit is not significantly different from a horizontal line, which would indicate no performance loss due to the question length. Further information on the distribution of the test data due to question length can be found in Figure 5 in the Appendix.

model	parse	static	nominatim
baseline	0.05	0.02	0.11
+cluster	0.05	0.03	0.05
+retrieve	0.05	0.02	0.04
+db_info	0.07	0.02	0.05
+cluster+db_info	0.06	0.02	0.03

Table 4: Analysis of error types that make queries not executable against the database. Parse errors are errors like missing closing brackets, static errors are wrong keywords and nominatim errors are errors that hinder nominatim to return area ids for locations.



(a) Base Model: The line of best fit is significantly different from a horizontal line (p: 0.006).

(b) Base Model enhanced with `cluster` tags and `db_info`: The line of best fit is not significantly different from a horizontal line (p: 0.58).

Figure 3: Interaction of execution accuracy and question length in the base model and the best models. The questions were binned based on their sentence length. The average execution accuracy of each bin (blue dots) is measured on the y-axis. The line of best fit is illustrated in red.

7 Error Analysis

An error analysis (Table 4) shows that for the baseline parser, 11% of the queries do not yield a correct result due to nominatim errors. In these cases, the geolocation service provided by nominatim[6] cannot find an id for a query string like ´Nermany' instead of 'Germany'. For the best model that uses `cluster` and `db_info`, the nominatim error rate for the dataset is significantly lower at 3%. The nominatim error rate is the lowest even compared to models that use only one enhancement, the lowest being `retrieve` having an error rate of 4%.

An inspection of selected examples shows that the baseline model seems to have a problem with hallucination by inserting values in the hypotheses that appear often in the query but are different from the values given in the questions. Giving the model access to query templates via `cluster` or `retrieve` appears to make the model hallucinate less. In the following example, the baseline model inserts the correct uid only in one of the two places, whereas the improved model correctly predicts the correct uid in both places.

Question: Ways and nodes with the uid 9847941 newer than yesterday

Baseline:
```
(way(uid:9847994)
(newer:"{{date:1day}}");
node(uid:9847941)
(newer:"{{date:1day}}"););out;
```

cluster:
```
(way(uid:9847941)
(newer:"{{date:1day}}");
node(uid:9847941)
(newer:"{{date:1day}}"););out;
```

`Cluster` and `retrieve` also seems to reduce the generation of typos, as can be seen in the following example, where the baseline model produces the typo "miltary" instead of "military".

Question: Way with the attribute usage having a value military in Colorado

Baseline:
```
geocodeArea:Colorado->
.searchArea; (
way["usage"="miltary"]
(area.searchArea);); out;
```

cluster:
```
geocodeArea:Colorado->
.searchArea; (
way["usage"="military"]
(area.searchArea);); out;
```

Interestingly, even if `cluster` or `retrieve` approaches have never seen a certain value in the training data (like "furnace" in the following example), they seem to be able to copy better from the source than the baseline model.

Question: furnace shops in current view

[6] https://nominatim.openstreetmap.org/ui/search.html?q=Germany

Baseline:
```
( node["shop"="furniture"]
    ({{bbox}});
    way["shop"="furniture"]
    ({{bbox}});
    relation["shop"="furniture"]
    ({{bbox}}););out;
```

cluster:
```
( node["shop"="furnace"]
    ({{bbox}});
    way["shop"="furnace"]
    ({{bbox}});
    relation["shop"="furnace"]
    ({{bbox}}););out;
```

Looking at model outputs trained with the `db_info` approach, it can be seen how the performance is increased by using nominatim information (described in section 6.1.1). The nominatim table contains an entry for "florist", returning "shop-florist" as a key-value pair. The model learns that the augmentation is often of high quality, thus it only needs to focus more on the key-value information, even if the specific key-value pair has not occurred very often ("florist" only appears in one training example).

Question: florist in current view

Augmented Question: florist in current view [K] shop [V] florist

Baseline:
```
( node["historic"= "fort"]
    ({{bbox}}); way["historic"=
    "fort "]({{bbox}});
    relation["historic"=
    "fort"]({{bbox}}););out;
```

db_info:
```
( node["shop"="florist"]
    ({{bbox}});way["shop"="florist"]
    ({{bbox}});
    relation["shop"="florist"]
    ({{bbox}}););out;
```

The `db_info` approach also seems to be able to reduce certain types of halluctionations, which can be seen in the following query, where a typo in the question ("is" instead of "in") confuses the baseline model, but not the augmented model.

Question: Cinemas is current view

Baseline:
```
(node["landuse"="cemetery"]
    ({{bbox}}); way["landuse"=
    "cemetery"]({{bbox}});
    relation["landuse"="cemetery"]
    ({{bbox}}););out;
```

db_info:
```
( node["amenity"="cinema"]
    ({{bbox}}); way["amenity"=
    "cinema"]({{bbox}});
    relation["amenity"="cinema"]
    ({{bbox}}););out;
```

8 Conclusion

We introduced OverpassNL, a new dataset for semantic parsing and interpretation of Overpass queries to the OpenStreetMap database. OverpassNL is a semantic parsing dataset that builds upon complex real-world user queries issued to a large-scale complex database. We illustrate the complexity of the dataset and the difficulty of the semantic parsing task, with the baseline model only reaching around 21% of execution accuracy. We then improved the model by incorporating more information, either by feeding similar examples into the model, by exploiting similarities in the natural language questions, and by enhancing our train data with silver data. Our best model then reaches an execution accuracy of 36%.

9 Future Work

An avenue of research we aim to pursuit in the future is to use the PICARD (Scholak et al., 2021) algorithm which led to improvements on the Spider dataset by constraining the beam search to valid outputs. A reimplementation for the Overpass syntax could also yield improvements in our experiments.

Additionally we want to research the possibility of augmenting our models with even more knowledge sources, for example the contents of the OpenStreetMap wiki [7]. Lastly, we are planning to establish an interactive setup where OSM users and developers can use a semantic parser trained on OverpassNL and provide feedback for interactive machine learning.

10 Limitations

A possible limitation of the presented work could be an inherent bias in the developer-generated data, for example, a gender bias, or simply a bias towards queries that appear complex on the surface, but ask for trivial contents. We hope that a future interactive scenario will encourage users and developers to take advantage of the natural language interface to query for interesting contents.

[7] https://wiki.openstreetmap.org/wiki/

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriber. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. Plainsboro, NJ.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Lee R. Dice. 1945. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

Carolin Haas and Stefan Riezler. 2016. A Corpus and Semantic Parser for Multilingual Natural Language Querying of OpenStreetMap. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, San Diego, California.

Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 10073–10083, Red Hook, NY, USA. Curran Associates Inc.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

Julia Kreutzer, Jasmijn Bastings, and Stefan Riezler. 2019. Joey NMT: A Minimalist NMT Toolkit for Novices. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing: System Demonstrations (EMNLP-IJCNLP)*, Hong Kong, China.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. *CoRR*, abs/2012.12627.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. 2018. NL2Bash: A corpus and semantic parser for natural language interface to the linux operating system. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI)*, New York, USA.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, Beijing, China.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Stefan Riezler and Michael Hagmann. 2022. *Validity, Reliability, and Significance: Empirical Methods for NLP and Data Science*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

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.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*. Montreal, Canada.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, Hong Kong, China.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: Query synthesis from natural language. In *International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, Vancouver, USA.

Ziyu Yao, Xiujun Li, Jianfeng Gao, Brian M. Sadler, and Huan Sun. 2019. Interactive Semantic Parsing for If-Then Recipes via Hierarchical Reinforcement Learning. In *Proceedings of the 33rd Conference on Artificial Intelligence (AAAI)*, Honolulu, Hawaii, USA.

Tao Yu, Rui Zhang, He Yang 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, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 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 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China.

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 (EMNLP)*, Brussels, Belgium.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Tan, 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 (ACL)*, Florence, Italy.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI)*, AAAI'96.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. 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)*, Hong Kong, China.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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

Barret Zoph and Kevin Knight. 2016. Multi-Source Neural Translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34.

A Appendix

A.1 Overpass Statistics

number of OSM users	8.3 million
number of nodes in OSM	7.4 billion
map changes per day in OSM	4.5 million

Table 5: Database statistics of OpenStreetMap as of 2022-01-10 (`https://wiki.openstreetmap.org/wiki/Stats`).

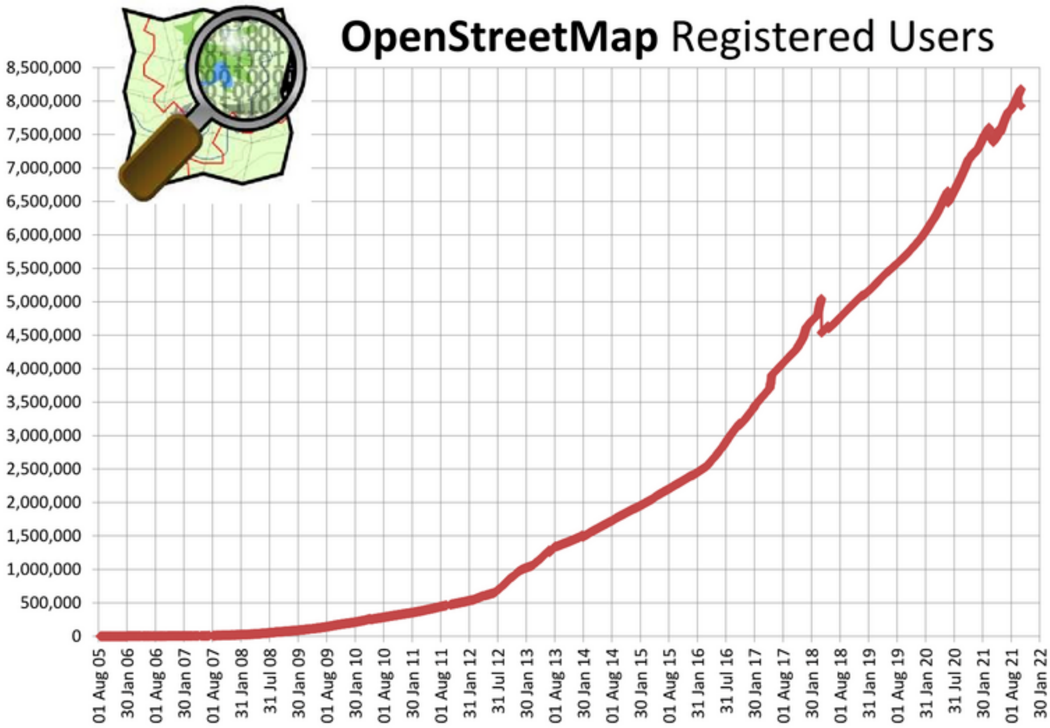


Figure 4: Accumulated registerd users (linear scale) of OpenStreetMap (`https://wiki.openstreetmap.org/wiki/Stats`)

A.2 Dataset Properties

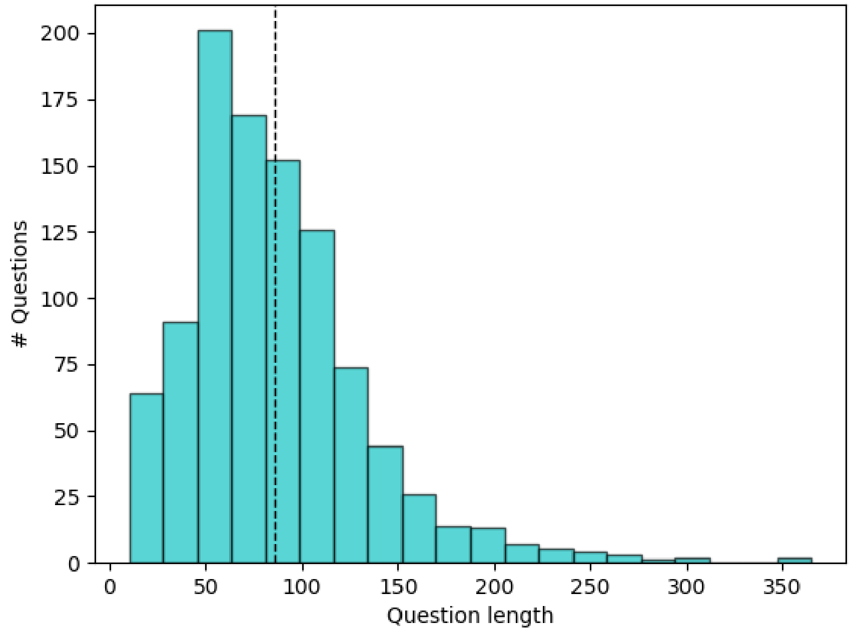


Figure 5: Distribution of the test data due to question length (in characters). The dotted line indicates the arithmetic mean.

A.3 Hyperparameter Settings

Parameter	Value
optimizer	adam
learning rate	0.0002
batch size	4
encoder rnn type	bidirectional GRU
attention	bahdanau
encoder embedding dim	620
encoder hidden dim	400
encoder layers	1
decoder rnn type	GRU
decoder embedding dim	620
decoder hidden dim	800
deccoder layers	1

Table 6: Hyperparameter settings of JoeyNMT sequence-to-sequence model used in our experiments.

A.4 Semantic Parsing Example

SRC	*Highways or routes with official name* *Rodovia Vespertino de Medeiros Bonorino in Brasil*

PRED.	`{{geocodearea:rs,brasil}}->.searcharea;`
	`(way["highway"~".*"]["official_name"~`
	`"^rodovia estadual joão cândido$"](area.searcharea);`

GOLD	`{{geocodearea:rs,brasil}}->.searcharea;`
	`(way["highway"~".*"]["name"~`
	`"^rodovia vespertino de medeiros bonorino$"](area.searcharea);`

Table 7: Semantic parsing example. SRC is the natural language question, PRED. the predicted query and GOLD the correct query.

A.5 Cluster Examples

- Cluster 0
  - `Admin level 3 in Russia`
  - `Admin level 3 in Tanzania`
  - `Admin level 4 in Angola`
- Cluster 2
  - `places I can grill outside in current view`
  - `places of worship in current view`
  - `planetarium in current view`
- Cluster 6
  - `Boundary relations in Rio Grande do Sul, Brazil with IBGE order numbers matching the regular expression "4̂3[0-9]{8}$"`
  - `Milestones in mesorregião do oeste catarinense that have a description or a reference matching "ŜC"`
- Cluster 9
  - `Nodes and ways that were changed between 2018-07-02T00:00:00Z and 2018-07-02T19:39:59Z by the user with the ID 8076784`
  - `Nodes and ways that were changed between 2019-07-10T00:00:00Z and 2019-07-10T23:59:59Z by the user with the ID 8710004`
  - ` Nodes and ways that were edited between 2019-02-11T00:00:00Z and 2019-02-11T23:55:59Z by the user with the ID 7725447`

Figure 6: Cluster examples that were used to improve the performance of our encoder-decoder model.

A.6 Fuzzy String Matching

For this algorithm, the natural language question and the whole word/phrase column from the nominatim table are converted into lower-cased character sequences and the longest subsequence match between the question and the column values is computed. The subsequence match is only considered valid if the word boundaries can be detected within ± 2 characters of the match, thereby matches that are substrings of the words in the natural language question such as “way” in “motorway” are excluded. Additionally, if there is a preposition right after the word in the natural language question, it is checked whether the preposition appears in the nominatim table in the column “operator”.