NLmaps: A Natural Language Interface to Query OpenStreetMap

Carolin Lawrence
Computational Linguistics
Heidelberg University
69120 Heidelberg, Germany
lawrence@cl.uni-heidelberg.de

Stefan Riezler
Computational Linguistics & IWR
Heidelberg University
69120 Heidelberg, Germany
riezler@cl.uni-heidelberg.de

Abstract

We present a Natural Language Interface (nlmaps.cl.uni-heidelberg.de) to query OpenStreetMap. Natural language questions about geographical facts are parsed into database queries that can be executed against the OpenStreetMap (OSM) database. After parsing the question, the system provides a text-based answer as well as an interactive map with all points of interest and their relevant information marked. Additionally, we provide several options for users to give feedback after a question has been parsed.

1 Introduction

OpenStreetMap (OSM) provides a map of the world, annotated by volunteers with GPS points (*nodes*) that they consider relevant, and with as much corresponding information as is deemed of interest. For example, such nodes mark restaurants, hotels, schools or hospitals. Each node can be assigned various tags, usually from a pool of agreed upon tags by the OSM community, such as "*tourism=hotel*", "*amenity=school*" or "*name=Heidelberg University*". Alternatively, a node may be used to form a *way*, such as a road, in conjunction with other nodes. Nodes and ways in turn can be grouped together to form a *relation*, for example to mark several buildings as belonging to the same institution.

The resulting database is vast, with over 3.4 billion objects, but only offers limited searchability. For example, the search tool on the main website [http://www.openstreetmap.org] finds for the search term "*Gare du Nord*" the train station Gare du Nord in Paris, and returns it as the first search result. For "*Where are 3 star hotels in Paris*", no search result is found, even though the database contains objects marked with the tags "*tourism=hotel*" and "*stars=3*". To be able to find these objects, one would have to use the Overpass API [http://wiki.openstreetmap.org/wiki/Overpass_API] which requires extensive knowledge of not just the OSM tags used, but also of the Overpass query language. A query in this case would read "*area[name='Paris']→.a;node(area.a)[tourism= 'hotel'][stars='3'];out;*" which is not feasible for everyday use for the average user.

To be able to ask exactly such questions, we developed a semantic parser for the OSM domain that maps natural language questions to a Machine Readable Language formula (MRL) which can be executed against the OSM database. Also, we created an interface (see Figure 1 for a screenshot) that makes the parser available for online use. Furthermore, we extended our framework with various conveniences such as location detection and an interactive map that connects the text-based answer from the parser with clickable markers on an interactive map via hyperlinks. With the help of a feedback formula, the users can help us to improve the parser and to extend it to other languages in the future.

A few examples of questions that our interface can answer are:

```
What is the closest bank with ATMs from the Palace of Holyroodhouse in
Edinburgh?

Which driving school is closest to Mannheimer Straße in Heidelberg and where
is it?
```

This work is licenced under a Creative Commons Attribution 4.0 International Licence. Licence details: http://creativecommons.org/licenses/by/4.0/

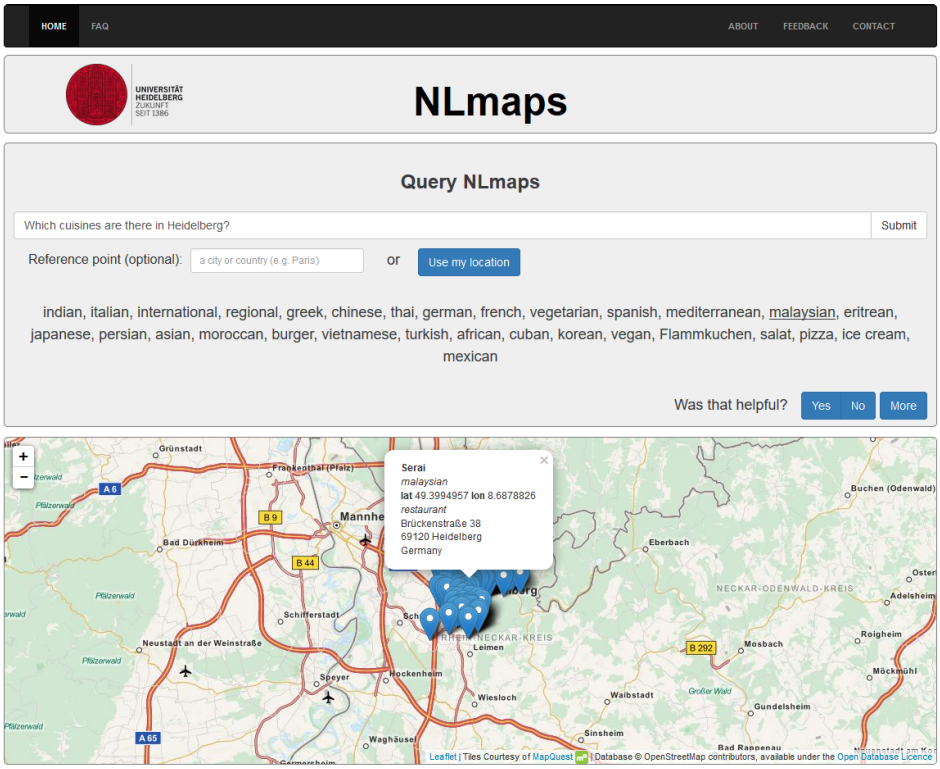

Figure 1: A screenshot of the user interface after a question has been processed, and the underlined link in the answer box was clicked to open up the corresponding marker's popup.

```
Where are the closest bank and the closest pharmacy from the Rue Lauriston
in Paris?
```

In the following, we present a semantic parsing approach that uses hard constraints to search a geographic information system that can cope with complex spatial expressions, such as "*nearby*" or "*in walking distance*". The type of questions that can be answered, may be found in Table 2.

The system can be freely accessed by anyone and all components of the semantic parser are publicly available. For detailed license notes of the individual components see `nlmaps.cl.uni-heidelberg.de`.

To the best of our knowledge, this is the first approach of a natural language interface to OSM using semantic parsing. Overpass-turbo[1] supplies a wizard that uses a simple set of rules to match a few basic natural language words to Overpass, e.g. "*tourism=museum in Vienna*". Other purely string matching approaches are Nominatim[2] and GeoNames[3].

2 System Architecture

Area Recognition. The interface offers the users a text field to type in their question and to submit it to the system. Once the system receives such a submitted question, the processing pipeline is started. First, possible area references (locations that are areas) mentioned in the question need to be identified. For example, the system will recognize "*Heidelberg*" as a city name in "*Which cuisines are there in Heidelberg?*". Words following "*in*" and "*around*" are marked as area references. Additionally, we mark a word as an area reference if it is preceded by "*of*" and if the previous word was "*vicinity*" or a cardinal direction ("*Where are restaurants in the east of Heidelberg?*").

In a next step, the identified area references are mapped to an OSM object. This is accomplished using the OSM tool Nominatim [`http://wiki.openstreetmap.org/wiki/Nominatim`] which,

[1] `http://overpass-turbo.eu/`
[2] `http://wiki.openstreetmap.org/wiki/Nominatim`
[3] `http://www.geonames.org/`

given a search term, searches the name and address fields of all OSM objects and returns a ranked list. The object has to be either a way or a relation, as a single node (i.e. a GPS point) does not span an area. The ranked list is searched until either a way or a relation is found. The resulting object will then be used to constrain the search for objects of interest to lie within the area spanned by said object. One main drawback of this strategy is that a name which applies to several cities will only ever match the first city found. Thus a person living in Heidelberg, Pennsylvania, will always be disappointed with answers that concern Heidelberg, Germany. Because of this, we offer two alternative input methods to the user. The first is a text field where the user can specifically enter the area that shlould be searched. If "*Heidelberg, Pennsylvania*" is entered, then Nominatim will return the correct area. To alleviate the effort required on the user's part, we offer another option which can be selected by clicking the button "*Use my location*". This will ask the Geolocation API [`http://www.w3schools.com/html/html5_geolocation.asp`] to locate the user's device. Of course, the user is first asked for permission. Once the user's GPS location has been determined, Nominatim's reverse geocoding feature can provide the name of the city in which the GPS coordinates lie.

Semantic Parsing. After these preprocessing steps have been accomplished, the question is sent to a semantic parser. The parser employed here is a SMT system that translates from natural language to a machine readable language (MRL), following an approach introduced by Andreas et al. (2013). A MRL for the OSM domain as well as a corpus, NLMAPS, containing 2,380 question-MRL pairs was introduced by Haas and Riezler (2016) [`http://www.cl.uni-heidelberg.de/statnlpgroup/nlmaps/`]. Using this corpus, split into 1,500 training examples and 880 test examples, a SMT system can be trained, using GIZA++ (Och and Ney, 2003) and the SMT framework CDEC (Dyer et al., 2010), including a MERT run. Finally, the MRL returned by the SMT system is executed against the OSM database using an extension of the Overpass API, OVERPASS NLMAPS [`https://github.com/carhaas/overpass-nlmaps`]. All relevant information is collected from the returned database objects and is then compiled into 2 different output formats.

Answer Presentation. Answers are returned in the interface's answer box in text format. Additionally, the output, formatted in GeoJSON, is used to place markers in the appropriate GPS locations on an interactive map. If a marker is clicked, this output supplies further information that may be of interest to the user in a popup. It first provides the object's name. Second it lists information directly related to the text-based answer. In the example of Figure 1, this would be the cuisine served at the restaurant the marker points to. Further it provides the exact latitude and longitude of the objects but also, again using Nominatim's reverse geocoding feature, a human readable address (due to resource reasons this is currently only supported for selected countries). Lastly, it lists all values of an OSM tag's key-value pair ("*amenity=restaurant*") which were used in the query that returned the object.

To browse the results, the users can now move around the map and click on markers for more information. Alternatively, they can, if applicable, click on an element in the text-based answer which will then open all relevant markers' popups. In the example in Figure 1, the user clicked "*malaysian*" and all restaurant markers that serve malaysian food where opened.

Backoff. Should the semantic parser not find an answer, then the system backs off and queries Nominatim. Should Nominatim now find an answer, that information is then presented to the user.

3 Semantic Parser Training

For the semantic parser to work in this setting we extended the parser described in Haas and Riezler (2016) in several ways[4].

Area IDs. One of the issues of the original model is its difficulty to generalize to area references not seen during training. It purely relies on the word to be passed through. Our first parsing model alleviates

[4]Their model labelled "*+intersect +stem +cdec +pass +cfg*" is called "*original*" in Table 1 and the extensions are added consecutively onto that model.

NLMAPS		Precision	Recall	F1
1	original	89.90	64.02	74.78
2	+IDs	88.67	66.17	75.78
3	+SE	89.56	65.67	75.71
SE		Precision	Recall	F1
1	original	71.56	39.27	50.71
2	+IDs	63.32	40.4	49.29
3	+SE	90.86	71.64	80.11

Table 1: Semantic Parsing results on the NLMAPS and SE test sets for the models introduced. Results are an average over 3 runs because the tuning algorithm MERT introduces randomization.

Type	Example
aggregation	How many...?
GPS location	Where...?
existence	Is there...?
specific key search	What is the name/website/...?
distance	How far apart...?
cardinal direction	...in the north of...?
closest	closest hotel
radius search	hotels in walking distance
exclusive or	a bar or restaurant
union	a butcher and a bakery

Table 2: An overview of the questions types the system has seen during training.

this issue by the area reference recognizer and lookup components described above. For example, the question "*Where are restaurants in Heidelberg?*" is changed to "*Where are restaurants in 3600285864*" by the area recognizer and lookup components. We thus modified the MRLs in the training corpus to reflect this change accordingly.

Search Engine Queries. A first test of the system showed that people tend to enter, search engine style, short queries, rather than fully grammatically correct sentences. The discrepancy between the training data and these queries causes a significant drop in performance. We thus changed the complex sentences of NLMAPS into these shortened forms, resulting in a second data set, SEARCHENGINE (SE). The current parser model performs very badly on this new test set (see Table 1), confirming the reports of the first testers. Training a new model on both the original and the new training data shows a big increase in performance (model "+*SE*"). The model is now better on the SE test set than on the NLMAPS test set. We attribute this to the fact that all words that are not strictly necessary are removed in a search engine style query, allowing the model to exclusively focus on the important words.

4 User Feedback Mechanisms

To be able to improve our system in the future, we implemented mechanisms that allow us to gather user feedback after a question has been answered. The most direct feedback option is directly integrated into the answer box. It asks for single point feedback to the question "*Was that helpful?*" where the user can select the "*Yes*" or the "*No*" button.

Alternatively, the user can click the "*More*" button which then opens another window that overlays the previous. Here the user can provide more detailed feedback. The questions become progressively more complex, in the sense that the further the user progresses, the more detailed knowledge is required about OSM, the Overpass query language, and the MRL. Each question asks the user if an intermediate result from various steps in the pipeline is correct or not. The intermediate result is printed in a text box which the user can edit if it is wrong. Alternatively, the user can also merely indicate if it is correct or not, using buttons next to the text box. The user can stop at any time, thus submitting an incomplete form, or close the feedback window without answering any questions at all.

If a user can correct the MRL, then this highest level feedback equals a training example with a gold answer. Any supervised learning can be used to further improve the parser. However, this type of feedback is by far the hardest to give. A more likely scenario is one, where a user who regularly used the Overpass query language, visits the site and corrects the Overpass query in the feedback form. This would still provide a high quality supervision signal, though only partial. The same holds true for the other feedback questions where the correct output or the 0/1 feedback can be seen as a partial supervision signal of varying degrees of detail.

We will use this feedback to test various algorithms for response-based learning (Kwiatowski et al. (2013), Berant et al. (2013), Goldwasser and Roth (2013), Szepesvári (2009), Bubeck and Cesa-Bianchi

(2012), *inter alia*) to improve the parser. A new challenge will be to incorporate the different levels of feedback into one algorithm.

5 Conclusion

We presented an online interface with which the OSM database can be queried using natural language. While the parser is not yet close to answering every question posed, it already shows promising results. Considering that previously a simple question like "*Where are 3 star hotels in Paris*" needed detailed knowledge of OSM and the Overpass query language, we think that the interface takes a large step towards making the vast and interesting knowledge of the OSM database available to the everyday user.

In the future, we will use feedback gained from the users to improve the system further, using algorithms for learning from partial feedback. Additionally, we want to extend our system to work for multiple languages. To this end, we will train SMT systems that translate the question into English, which can then be parsed with the current system. To improve the translations into English by the SMT system, we can again make use of the feedback users provide by employing response based learning algorithms, particularly the algorithms introduced by Riezler et al. (2014). The user feedback as well as the question logs will also be used to further improve the system in future work.

Acknowledgments

We would like to thank the OSM developers Roland Olbricht and Martin Raifer for their support. The research reported in this paper was supported in part by DFG grant RI-2221/2-1 "Grounding Statistical Machine Translation in Perception and Action".

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sofia, Bulgaria.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Seattle, WA.

Sébastian Bubeck and Nicolò Cesa-Bianchi. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden.

Dan Goldwasser and Dan Roth. 2013. Learning from natural instructions. *Machine Learning*, 94(2):205–232.

Carolin Haas and Stefan Riezler. 2016. A corpus and semantic parser for multilingual natural language querying of OpenStreetMap. In *In Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, San Diego, California, June.

Tom Kwiatowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Seattle, WA.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Stefan Riezler, Patrick Simianer, and Carolin Haas. 2014. Response-based learning for grounded machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Baltimore, MD.

Csaba Szepesvári. 2009. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool.