Ontology-based Route Planning for OpenStreetMap

Mihai Codescu[1], Daniel Couto Vale[2], Oliver Kutz[2], and Till Mossakowski[1,2]

[1] DFKI GmbH Bremen
[2] SFB/TR 8 Spatial Cognition, University of Bremen, Germany

Abstract. We develop a web service for route finding in OpenStreetMap (OSM) following an activity-centred approach: the aim is not only to assist the user in travelling from A to B, but also to perform a series of specified activities along the way. This is particularly important e.g. for electric mobility, where activities can take place while the battery of the car is recharging. For specifying activities, our tool uses an ontology of spatially-located activities. This ontology then needs to be related to OSM which provides semantic metadata in form of tags. We organised the tags into another ontology (which is then connected to the first one via an ontology mapping), providing thus not only better reference for the OSM community, but also allowing to enrich the ontological semantics of the tags, to deal with their evolving nature and to extract implicit information using ontology-based data access. Moreover, we report first results on an ongoing query corpus experiment involving the OSM community targeted at improving the automated understanding of free text input for certain route finding tasks.

1 Introduction

We develop a web-based system DO-ROAM[3] for activity-oriented route planning. In our system, geolocations can be specified in different ways for finding a route. While traditional route planning works with identifiable locations referred to by names or coordinates, DO-ROAM allows specification of kinds of activities and facilities, whose identities and locations are only determined later on. It also includes a simple temporal planning component.

Several aspects of a massive data-intensive system such as DO-ROAM must be tackled. On the content aspect, it is unfeasible to produce the necessary amount of data in a centralised way. For this reason DO-ROAM draws on data from OpenStreetMap, which provides not only entity coordinates, but also a fair amount of metadata, like their names, opening hours, activities, URLs and the like. Since metadata obtained as volunteered geographical information (VGI) through collaborative and community-based efforts evolve in a bottom-up way contain a lot of noise (typos, redundancies, etc.) and are subject to constant change, we intend to make the access to information such as the OpenStreetMap data more structured and stable through the application of ontologies. Our goal is to extract an ontology from VGI in an automated way.

On the usability aspect of our system, offering an intuitive user interface makes it more likely that a web-based tool appeals to a broad user base. Route planning is

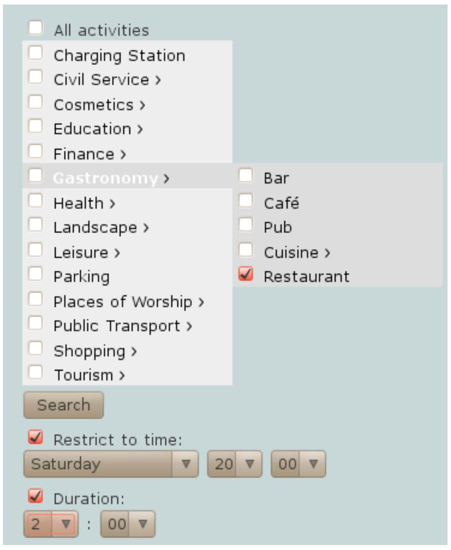
[3] A prototype is freely available at www.do-roam.org; DO-ROAM stands for *Data and Ontology driven Route-finding Of Activity-oriented Mobility*.

an intrinsically complex activity. In the interaction model of DO-ROAM, one can, for instance, browse the map laterally, zoom in and out, localise oneself, and, most importantly, specify either a location or a route composed of an origin, a destination and possibly intermediary activities. For this last task, we offer two specification methods, a guided one through drop down menus and an unguided one through a text field. Very central to the intuitiveness of this interface is the correct analysis of most direction queries frequently typed in the text field according to the linguistic conventions of each covered locale.

This work extends [6] in various ways. The motivating use cases in Sect. 2 are new. The architecture of the tool, presented in Sect. 3, has been updated from [6] to include the route planning component. The ontology of tags, OSMonto, is structured in a cleaner way (using roles, see Sect. 4), leading to a new version of the ontology mapping that links OSMonto to the ontology of activities, designed in [6] (Sect. 5). Some details regarding generation and maintenance of OSMonto can be found in [6]. Finally, we present the results of an experiment conducted with German native speakers that will support an automatic analysis of direction queries for the German-speaking regions of Europe (Sect. 6).

2 Motivating Use Cases

In the following, we describe how our system can be used in two complex use cases to specify activities and plan a route through locations where one can do them. In creating use cases, one of our concerns was electric mobility because of the rather limited reach of electric cars and relatively long charging times. The maps are provided by OpenStreetMap, but the approach is flexible enough to include data from other sources.



✓ Restaurant

✓ Restrict to time: Saturday 20 00

✓ Duration: 2 : 00

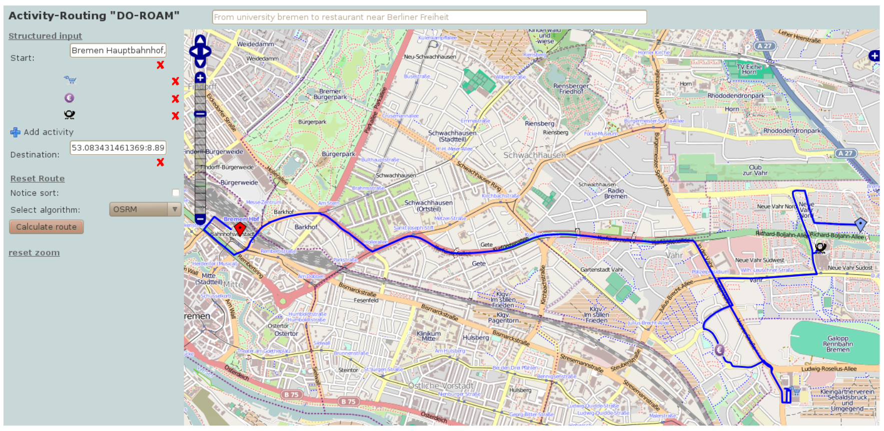
Fig. 1. Use case 1: activity selection.

[1] presents several navigation scenarios based on a GIS ontology; this work inspired the use cases presented here. We now describe how DO-ROAM supports use cases 1 and 2.

Use Case 1: Betty is a tourist and wants to find out which activities are in reach by foot from the nearest charging station. She also knows that she will be getting hungry soon and thus looks for all restaurant which will be open the next 2 hours.

Use Case 2: Maria wants to visit a friend in Bremen. On her way from the train station to her friend's flat she would like to go shopping and visit a bank and a post office. She needs a system generating a route containing all these places.

The screenshot in Fig. 1 illustrates how to use the interface to find all restaurants open during the next 2 hours. In use case 2, Maria should first select a start and end point of a route. Afterwards she should add places for shopping, a bank and a post office. A routing algorithm must then be selected (in our case OSRM) and then she clicks “Calculate route” which generates the route shown in Fig. 2. The system also supports resetting the route at any given time.


Activity-Routing "DO-ROAM"

From university bremen to restaurant near Berliner Freiheit

Structured input

Start: Bremen Hauptbahnhof,

Add activity

Destination: 53.083431461369:8.89

Reset Route

Notice sort:

Select algorithm: OSRM

Calculate route

reset zoom

Fig. 2. Use case 2: route planning

3 Architecture of DO-ROAM

DO-ROAM is a web application intended to assist spatio-temporal planning of activities and routes. At the moment, a prototype is available at http://do-roam.org. It is implemented in Ruby on Rails and its architecture is shown in the Fig. 3. A previous version of DO-ROAM, described in [6], was developed on top of the OSM rails port. We have chosen to build our system as a new, clean Rails project to gain simplicity. Our work has been inspired by the activity-oriented interactive route planning system Digital Travel Mate [9], see http://www.digitaltravelmate.net, which allows for finding locations and planning routes in a fictional map by specifying kinds of holiday activity. Routes can also be interactively corrected. Our focus has been on coping with the challenges of moving away from a *hard-coded* prototype map with a small predefined set of activities to a real-world application scale decoupled from the map.

DO-ROAM consists of a graphical user interface, a data integration component and a route planning engine, see Fig. 3. The tool can be used in two complementary manners. Firstly, it can display locations where a certain activity takes place, possibly at a specified time. The locations are found using the OSM tags of the map elements, which we organised in an ontology, described in Sec. 4. We provide two alternative interfaces for the activity search. The first is a simple text-based search, similar to those existing in tools like OpenStreetMap or Google Maps. We describe the functionality of the text search in detail in Sec. 6. The second interface provides an overview of the activities

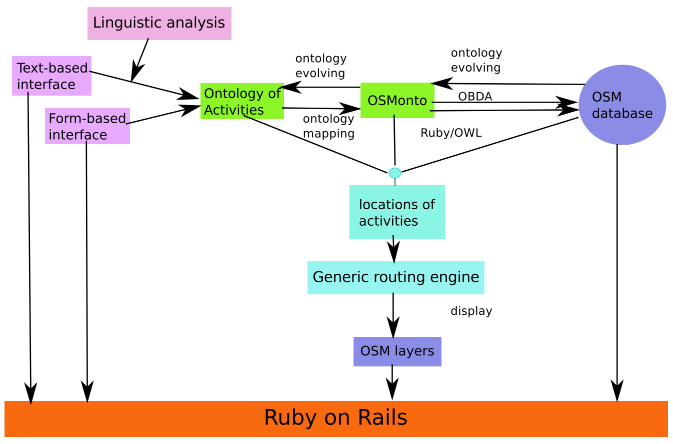


Fig. 3. Architecture of DO-ROAM

by displaying them in a tree-like structured taxonomy. Since the OSM tags representing the locations for activities are too cryptic to be presented as such, we designed an ontology of spatially-located activities to ease interaction with the user. This ontology and the way it relates to the ontology of tags are described in Sec. 5. After an activity has been specified and a restriction on opening hours selected, DO-ROAM displays markers on the map for each location where the desired activity takes place, as in Fig. 2. Moreover, the list of the names of the search results is displayed on the left side, and pop-ups providing more information on the corresponding location are available for each marker.

Secondly, the tool generates routes that include a number of locations where certain activities take place. This is done in a two stage process. At a first stage, the user must specify the starting and ending point, by either selecting a location on the map via a context menu, searching for an address or a name or directly giving the latitude and longitude. After that, the user selects a number of activities that she wants to perform along the route, in the same way as above. After each selection, the user can continue specifying activities or can proceed to the second stage to generate a route.

To find a route, we use routing engines developed by the OpenStreetMap community and available as web services (OSRM, YOURS). In the case of route planning for electric cars, we have integrated the routing engine available at http://greennav.org [2, 8], which generates energy-efficient routes (at the moment only in Bavaria). The user chooses a car type and the state of charge of the battery and the route is optimised regarding status of charge at the end of the route. Accounted for is the fact that the battery can be recharged by braking and driving downhill. The system sends the list of coordinates of the starting point, intermediate locations for activities and ending point to the routing engine, which then computes the route. The user can also specify whether the order in which the activities have been specified should be noticed or not; in the second case, the generated route will include the activities in any possible order.

As discussed in [6], there is an easy association between the tags occurring in the ontology and the database. This allows us to apply ontology-based data access (OBDA)

[7, 5] for data integration. This relies on the representation of ontologies within the Ruby on Rails framework using the library Rails-OWL also developed as part of this project, see again [6].

4 OSMonto: An Ontology of OSM Tags

OpenStreetMap's database consists of nodes, ways and relations, which can be tagged with information about the respective map element. The convention is that any user is free to introduce his own tags, but it is recommended to use existing tags and only have new ones if they are not already covered by the existing ones. The tags of the map elements are represented as (key, value) pairs. An element of the map may have multiple tags (see below for an example of an OSM node with its tags in an XML representation. This format has been developed by the OSM community. The listed tags vary from node to node).

```
<node id="834034642"
   lat="53.0871310" lon="8.8091071"
   version="7" changeset="6027662"
   user="Kerridge" uid="324245"
   timestamp="2010-10-13T09:51:39Z">
    <tag k="addr:city" v="Bremen" />
    <tag k="addr:country" v="DE" />
    <tag k="addr:housenumber" v="20" />
    <tag k="addr:postcode" v="28215" />
    <tag k="addr:street" v="Theodor-Heuss-Allee" />
    <tag k="amenity" v="charging_station" />
    <tag k="name" v="Elektrotankstelle swb" />
    <tag k="note" v="telephone reservation necessary" />
    <tag k="opening_hours" v="Mo-Fr  6:00-18:00" />
    <tag k="operator" v="swb" />
    <tag k="phone" v="+49 421 3593186" />
  </node>
```

Currently, OpenStreetMap's tags are organised and maintained through a collection of Wiki pages[4] that list the popular tags and specify their intended use. We here propose to complement this by organising the tags into an ontology, which we call OSMonto[5]. The OSMonto tag ontology is written in the OWL profile EL [3], which is a lightweight subset of OWL. The OSMonto ontology will bring the following advantages:

- an ontology provides an easier overview of the tags and their hierarchical structure than a Wiki does. Browsing the tag ontology can be done using ontology editors like Protégé;
- ontology mappings can provide different views on the tag ontology:
    - tags can be enriched with an ontological semantics by mapping existing ontologies to the tag ontology;
    - different tags that are used for the same concept (due to local differences or the evolving nature of tags) can be united to one ontological concept through a mapping;

[4] See http://wiki.openstreetmap.org/wiki/Tags and http://wiki.openstreetmap.org/wiki/Features

[5] See the project's homepage at http://osmonto.do-roam.org/ and the ontology at http://osmonto.do-roam.org/tags.owl

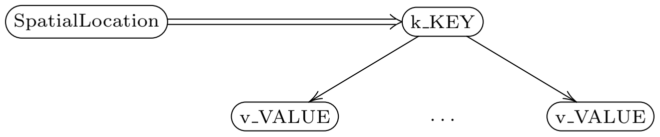
- search and navigation tools can use their own, purpose-driven ontologies (e.g. an ontology of activities that is shown to the user) and map them to the tag ontology.

The ontological perspective opens the door to *ontology-based data access* that can provide an enriched query language for the OpenStreetMap database. The ontology mappings that are necessary for obtaining the views can be generated semi-automatically or even automatically with the help of ontology matching tools. This approach provides a relatively simple, yet effective solution to the generally rather hard problem of how to relate data to ontologies.

OSMonto offers an easy and compressed overview of the keys and their values which are used in OSM. It resembles the page "Map Features" in the OSM wiki, but does not include descriptions of the tags and thus delivers a quicker overview of keys and especially their values. Also, other than the wiki page, it orientates more on the tags which are really in use at the moment (through the Taginfo website) as we only include the ontology the tags that are used with a certain frequency, see [6] for a larger discussion on this. So it is more an interesting device for users who want to make use of the existing database rather than for users interested in information how to tag something. Moreover, since all tags are in English, the ontology provides a high-level, natural-language-agnostic way of browsing the information while staying close to the original structuring of tags.

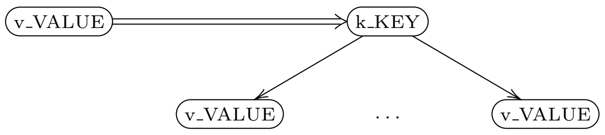
Since OSMonto is an ontology for spatial locations, we decided to introduce a class called $SpatialLocation$, around which the other classes are centred. The tags are then decomposed hierarchically according to the keys: the key becomes a superconcept of its values and we introduce an object property with domain $SpatialLocation$ having as range the class introduced for the key. Since it is possible that a key and a value have the same name whilst the names of the concepts are required to be unique in OWL (OSM has for example `station` as value of the key `railway` but also a key named `station`), we decided to prefix all keys with "k_" and all values with "v_". Notice however that some values can appear for more than one key: for example `tower` is a value both of `man_made` and `power`. In such cases we prefix the name of the value with the name of the tag (e.g. $k\_power\_v\_tower$).

E.g., a tag `<k = "amenity" v = "charging_station">` introduces a concept $k\_amenity$ with a subconcept $v\_charging\_station$ and a role $has\_k\_amenity$ with the domain $SpatialLocation$ and range $k\_amenity$. Locations of activities are then identified using the existential restriction, in our case $\exists\, k\_amenity \bullet v\_charging\_station$. In general, the graph introduced by a tag `<k = KEY v = VALUE>` is represented below:

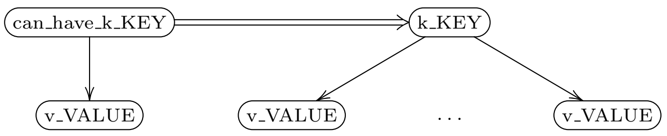


We have followed this approach whenever the value of the tag is an OSM constant rather than a string/numeral (for example, `amenity` admits as values `bank`, `cinema`, `hospital` and so on).

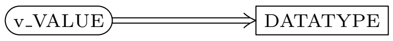
Another design decision is to take into account tag dependencies. For example, when a node is tagged with `<k = "amenity" v = "restaurant">` it is possible (but not mandatory) that the cuisine of the restaurant is also tagged as `<k = "cuisine" v = "seafood">`. Here we distinguish two cases: some tags can be used only in the presence of a certain tag (for example, a node can be tagged with `<k = "theatre:genre" v = "comedy">` only if it is tagged with `<k = "amenity" v = "theatre">`) but some tags can be used in the presence of more than one tag (for example, not only restaurants have cuisine, but also pubs or fast-foods). In the first case, we simply introduce an object property from the value that allows the dependency (in our case, $v\_theatre$) to the dependent tag (in our case $k\_theatre : genre$) like in the figure below:



while in the second case, we introduce a superclass of all tags that enable presence of the dependent tag (in our example $can\_have\_k\_cuisine$ is a superclass of v_bar, $v\_restaurant$ and so on) and use this superclass as domain of the object property:

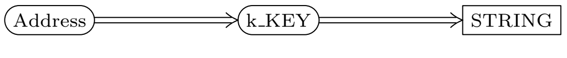


Notice that it is possible that the value of a tag is not only an OSM constant (like in the case of `cuisine`, which can be for example `chinese`, `italian`, and also `sushi` or `vegetarian`) but a string/numeral (for example, number of rooms in a hotel). In this case, we introduce data properties, with the same two subcases as above.



A special case to consider is the tags that admit as value `yes` or `no`. For such tags we do not introduce a class with the name of the tag having a subclass v_no (or rather $k\_TAG\_v\_no$ to ensure unique names), but rather an object property with the name of the tag having as range a concept yes/no, with subconcepts v_yes and v_no. Finally, since `smoking` can not only be tagged as permitted or allowed, but also as allowed in dedicated, separated or isolated areas, we grouped these concepts as subclasses of a concept we called $Decision$.

In OpenStreetMap, addresses are not represented compactly, but using a number of tags (e.g. for city, country, house number, street, postal code) prefixed with `addr:`. The values of these tags are simply strings. There are a number of tags that enable the presence of address tags. Following the same principle as above, we grouped them as subclasses of a new concept which is then the domain of an object property $has\_address$. The range of this object property is $Address$. We then introduce for each address tag a concept $addr : TAG$ and an object property $has\_addr : TAG$ with domain $Address$ and range $addr : TAG$, and a data property relating $addr : TAG$ to the datatype $String$:



5 Enriching the Semantics of Tags

In our envisioned scenarios, activities play a central role. We aim to present them to the user as a more structured interface element for guiding their selection. Moreover, we want to allow a certain degree of flexibility by performing a lexical analysis on the free text queries of the user and trying to match synonyms of the used words with concepts of the ontology of tags. Notice that the structure of OSM tags does not always provide a clean ontological classification of related concepts, and duplications sometimes occur.

Towards these goals, we have therefore designed an ontology of activities. The concepts of the ontology refer to locations where a certain activity takes place. This provides an abstraction level from the representation of the data in the databases and thus the user can express queries using a vocabulary closer to natural language. Notice that from an ontological perspective, the ontology developed is a task ontology: here the main motivation is not to create and specify a model of a domain, but to solve a well-defined task, namely, searching locations. We refer the interested reader to see [6] for a larger discussion on the design of this ontology.

We then connect the ontology of activities to concepts in the ontology of tags via an ontology mapping.[6] This also provides a way to semantically connect related tags.

For example, when searching for a place to swim, OSM offers a wide range of tags: `<k="amenity" v="swimming_pool">`, `<k="leisure" v="swimming_pool">` as well as `<k="sport" v="swimming">`. Sometimes this occurs because of changes in the tagging system, which are not immediately taken over by the users in the data. In the ontology of activities, we can have a single concept $Swimming$ which is mapped to the derived concept $\exists\, has\_k\_amenity \bullet v\_swimming\_pool \sqcup \exists\, has\_k\_leisure \bullet v\_swimming\_pool \sqcup \exists\, has\_k\_sport \bullet v\_swimming$.

Since the number of concepts and roles is quite large, providing such a mapping manually would be a very tedious process. We can, however, use an ontology matching tool to obtain a list of pairs of concepts that are in correspondence. This approach is known to be very effective - e.g. with the ontology matcher Falcon, the degree of automation reaches 80%. However, since in our case locations are identified by existential restrictions, we do not only have to verify and confirm the matches produced by the tool, but also to add ourselves the appropriate role restriction. While we have done this manually so far, automating this process is relatively straightforward and we are in the process of developing a specialised matching tool to cope with this issue.

Example 1. In the DO-ROAM project, we created a unified concept for charging stations for electric cars. It combines the tags `fuel:electricity=yes` (fuel stations with a possibility to charge electric cars) and `amenity=charging_station` (a device solely for charging electric vehicles). The mapping then is:

$$\begin{aligned}&ChargingStation \mapsto\\ &\quad \exists\, k\_amenity \bullet v\_charging\_station \sqcup\\ &\quad (\exists\, k\_amenity \bullet v\_fuel \sqcap \exists\, k\_fuel\_electricity \bullet v\_yes)\end{aligned}$$

[6] This procedure makes it also less dependent of the specific data base (in this case of Open Street Map). A parallel connection to Google Maps e.g. is planned. This would be realised with another ontology mapping.

Example 2. The user may search not only for single activities, but classes of activities, for example, not just restaurants or fast-food, but gastronomy in general The corresponding concept in the ontology of tags is obtained by taking first the union of all subclasses of $Gastronomy$:

$$Gastronomy \mapsto Bar \sqcup Restaurant \sqcup \ldots$$

and then mapping them individually to the ontology of tags:

$$Gastronomy \mapsto \exists\, k\_amenity \bullet v\_bar \sqcup \exists\, k\_amenity \bullet v\_restaurant \sqcup \ldots$$

Example 3. We can also do more specific searches, like looking for an italian restaurant. All restaurants with an italian cuisine are italian:

$$ItalianRestaurant \mapsto Restaurant \sqcap \exists\, hasCuisineOfNationality \bullet Italian$$

and again this termed is mapped structurally to the ontology of tags:

$$ItalianRestaurant \mapsto \exists\, k\_amenity \bullet v\_restaurant \sqcap \exists\, k\_cuisine \bullet v\_italian$$

6 Query Corpus Experiment

Building an intuitive query field interface for specifying routes demands a less prescriptive and more descriptive approach to human-machine interaction. Query fields present a design challenge because they do not constrain human behaviour (guide users) as much as drop down menus, check boxes and radio boxes, thus compelling users to type what they assume the system can answer without making the available search space explicit. As we shall see in the examples, user queries are not similar to any text in books or webpages nor to any corpus of other genre. For this reason, we could not use a general purpose corpus of representative German texts and were forced to observe what users would really write most frequently in our website to create a suitable text analyser for their queries.

To collect a corpus of route queries, we conducted a preliminary controlled experiment in Bremen, Germany with 12 participants: 7 males and 5 females; all spoke German with their parents, partners and outside the family. Participants had to perform 10 tasks of planning routes with the map. For each task there was the description of a situation in which a route plan was needed (as in Example 4) and the user was required to use the query field in our website to find the desired route. The intended results were precomputed before the experiment and presented to the user on query submission independent of what the user had written in the query field. Our goal with this experiment was to verify how much variation there is in user queries when they search the same routes and what query patterns the system is expected to understand.

Example 4. Task: Du sitzt in der Bremer Kneipe “Zum Feldschlößchen” und möchtest wissen, wie Du zum Restaurant “Das kleine Lokal” in Bremen kommst.

Translation: You are at the bar “To The Little Field Castle” in Bremen and want to know how you get to the restaurant “The Little Restaurant” in Bremen.

To prevent strong priming, we avoided offering textual formulations in the situation descriptions that could be reused in a direction query by making sure that referent names such as *Zum Feldschlößchen*, *Das Kleine Lokal*, and *Peter-Weiss-Straße* were used exclusively as tokens of identifying phrases or clauses such as *in der Kneipe "Zum Feldschößchen"* (in the bar "To The Little Field Castle") and in *zum Restaurant "Das Kleine Lokal"* (to the restaurant "The Small Restaurant"), or as location relata in locating clauses such as *der in der "Peter-Weiss-Straße" wohnt* (who lives in the "Peter Weiss Street"). Referents were chosen to cover a broad set of name and entity kinds in OSM data. Name kinds includes names with or without parts such as case markers, class markers, and separators while place kinds include facilities, shops, streets, districts, and cities. A careful analysis of the route query corpus showed participants have taken one of two strategies to specify routes. The most frequent strategy consisted of writing a frame with slots for the entity names or entity kind names in the same order as they are intended to be visited. Separating the slots, we found punctuation symbols such as hyphens and colons, the English keywords "route", "from" and "to", and the equivalent German keywords "weg", "von" and "nach"/"bis". This query pattern shows most users expect websites to spot entity names or entity kind names in the query and to use the order of frame slots as the order of locations to be visited. This expectation resulted in queries such as *Route: St.-Johannis-Schule - Café - Buchladen - Hohentor* (Route: St. Johannis School - café - book store - Hohentor).

The second strategy consisted of writing a sequence of phrases in fluent German, each one carrying a case marker identifying its function in the route query (origin, destination or intermediary paths). As case markers were used, the order of query constituents was not always the same as the order of locations to be visited as one can see in *von der St.-Johannis-Schule zum Hohentor via Buchladen und Café* (from the St. Johannis School to Hohentor through bookstore and cafe).

In the task of going from The Little Field Castle to The Small Restaurant, the frame with slots was favoured by most participants as illustrated below.

Frame with Slots (5 examples out of 10)

(1) route Feldschlößchen "das kleine lokal"
(route Little Field Castle "the small restaurant")

(2) route feldschlößchen das kleine lokal
(route little field castle the small restaurant)

(3) route Zum Feldschlößchen bis Das Kleine Lokal
(route To The Little Field Castle until The Small Restaurant)

(4) Feldschlößchen to Das Kleine Lokal
(Little Field Castle to The Small Restaurant)

(5) Zum Feldschlößchen Bremen - Das kleine Lokal Bremen
(To The Little Field Castle Bremen - The Small Restaurant Bremen)

Phrase Sequence with Case Markers (2 examples out of 2)

(6) vom Feldschlößchen zum Kleinen Lokal
(from the Little Field Castle to the Small Restaurant)

(7) vom feldschößchen zum kleinen lokal
(from the little field castle to the small restaurant)

In the frame slots, on the one hand, participants have used either the nominative form of an entity name such as *Zum Feldschlößchen* and *Das Kleine Lokal* (To The Little Field Castle, The Small Restaurant) or they have used an inflexible form composed of the parts of an entity name that do not carry case markers such as *Feldschlößchen* for *Zum Feldschlößchen*. Entity names with a discontinuous case marker such as *Das Kleine Lokal* (the word *Das* and the last *e* in *Kleine*) had no inflexible form. Entity kind names such as *Café* (cafe) and *Buchladen* (bookstore) were used without articles.

When participants used sequences of phrases with case markers, on the other hand, queries were very similar to one another. Case markers were used to indicate whether a location was an origin such as *vom Feldschlößchen* for *Zum Feldschlößchen* (from The Little Field Castle), a destination such as *zum Kleinen Lokal* for *Das Kleine Lokal* (to The Small Restaurant) or a path such as *via Buchladen und Café* (through bookstore and cafe). These two strategies demand two different approaches for handling queries. For the strategy of frame with slots, – detectable by the presence of special punctuation and keywords, – names can be spotted either in their nominative or invariable form, both of which can be precomputed from the OSM database with rules. For the strategy of phrases with case markers, spotting names is not possible. For instance, the nominative form *Das Kleine Lokal* (The Small Restaurant) is not spottable in the phrase *zum Kleinen Lokal* (to the Small Restaurant). Even if spotting such names were possible, their order would not necessarily correspond to the order of locations in the route. For this reason, we intend to implement a combinatory categorial grammar to produce further processable logical forms out of such queries.

7 Conclusions and Future Work

The route planning component is further developed at the moment. Future improvements concern, for instance, better mechanisms for the reduction of the search results. At the moment, the first 35 results are shown to improve both readability for the user and the speed of DO-ROAM. Desirable would be a ranking algorithm for the results. This could be achieved by creating a user profile recording favourite search results or by using a webpage ranking like in Google Maps. We also plan to make routes modifiable in a way going beyond from route modifications in Google maps, because the place of an activity might also be changed. An interesting open research question is to remove the separation between searching locations for activities and route generation, that is, the system will find the best places for activities and the best route simultaneously.

An ontology for OpenStreetMap tags similar to OSMonto has been developed in the EU FP-7 project LOD2 [10]. This ontology does not employ our rule to leave out tags that are only rarely used; this produces a lot of noise in particular what the data properties concerns. OSMonto not only avoids this noise, but also employs roles (specified with their domains and ranges) more systematically than the LOD2 ontology does. Altogether, we think that OSMonto provides a cleaner approach, while its extraction from the OSM tags still is an algorithmic process (see the description in Sect. 4).

The potential uses of OSMonto, our ontology of OpenStreetMap tags, that we have pointed out can be realised only if the ontology is kept up to date with the current development of OpenStreetMap tags. To ensure this, further research about manual and automatic update facilities is needed, including incorporating related work done for instance in the Web 2.0 context (see e.g. [4]). Probably an automatic update via TagInfo and the tagging wiki pages can do most of the work, keeping the number of necessary centralised manual corrections at a minimum. On the other hand, links to existing ontologies might suggest useful ontological structuring principles that need to be implemented manually. Eventually, the ontology may also give some fruitful insights into how to extend and structure the realm of OpenStreetMap tags.

As for the further improvement of the text field interface, we have invited members of the OpenStreetMap community to participate in the query corpus experiment and we shall have a larger corpus to support our automatic analysis in the near future.

Acknowledgements. We would like to thank Christian Clausen for doing implementation work and Gregor Horsinka for help with OpenStreetMap. This work has been supported by the German Research Foundation (DFG), Project I1-[OntoSpace] of the SFB/TR 8 “Spatial Cognition”.

References

1. Neeharika Adabala and Kentaro Toyama. Purpose-driven navigation. In M. Andrea Rodríguez, Isabel F. Cruz, Max J. Egenhofer, and Sergei Levashkin, editors, *GeoS*, volume 3799 of *Lecture Notes in Computer Science*, pages 227–233. Springer, 2005.
2. Andreas Artmeier, Julian Haselmayr, Martin Leucker, and Martin Sachenbacher. The shortest path problem revisited: Optimal routing for electric vehicles. In Rüdiger Dillmann, Jürgen Beyerer, Uwe D. Hanebeck, and Tanja Schultz, editors, *KI*, volume 6359 of *Lecture Notes in Computer Science*, pages 309–316. Springer, 2010.
3. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL Envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 364–369, 2005.
4. Silvia Bindelli, Claudio Criscione, Carlo A. Curino, Mauro L. Drago, Davide Eynard, and Giorgio Orsi. Improving search and navigation by combining ontologies and social tags. In *OTM ’08: Proceedings of the OTM Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems*, pages 76–85. Springer, 2008.
5. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The MASTRO system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011.
6. Mihai Codescu, Gregor Horsinka, Oliver Kutz, Till Mossakowski, and Rafaela Rau. DO-ROAM: Activity-oriented search and navigation with OpenStreetMap. In C. Claramunt, S. Levashkin, and M. Bertolotto, editors, *Fourth International Conference on GeoSpatial Semantics*, volume 6631 of *LNCS*, pages 88–107. Springer, 2011.
7. Antonella Poggi, Mariano Rodriguez-Muro, and Marco Ruzzi. Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In Patel-Schneider, editor, *Proc. of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 DC)*, volume 496. CEUR-WS, 2008.
8. Martin Sachenbacher, Martin Leucker, Andreas Artmeier, and Julian Haselmayr. Efficient energy-optimal routing for electric vehicles. In Wolfram Burgard and Dan Roth, editors, *AAAI 2011*. AAAI Press, 2011.
9. Inessa Seifert. Region-based model of tour planning applied to interactive tour generation. In Julie A. Jacko, editor, *HCI (3)*, volume 4552 of *LNCS*, pages 499–507. Springer, 2007.
10. Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. Linkedgeodata: A core for a web of spatial open data. *Semantic Web Interoperability, Usability, Applicability*, submitted. http://www.semantic-web-journal.net/content/linkedgeodata-core-web-spatial-open-data.