

Extracting Wikipedia Data to Enrich Spatial Information

Jörg Roth

Faculty of Computer Science
Nuremberg Institute of Technology
Nuremberg, Germany
Joerg.Roth@th-nuernberg.de

Abstract. Freely available geo data allow a developer to create new types of remarkable services related to the user's location. Even though current geo data sources have a high coverage and quality, they do not contain all information required by new services. This is because geo data sources usually focus on object geometries and object types. Important information is often missing. As an example: city entries mainly contain the city name and border, but not the name of mayor, amount of taxes, year of foundation, number of districts etc. These data are available in online encyclopediae such as Wikipedia, but there is no obvious approach to relate both sources. Our objective was thus to create an automatic import from Wikipedia articles that describe geo objects and extract all relevant data. To extract processible values we are able to identify property types such dates, money values, powers, heights, sizes etc. This makes it possible to use these data for further computation, e.g. to search for maxima, build averages and sums or to create comparative conditions in queries.

Keywords. Geo Data, Encyclopedic Data, Data Fusion

1 Introduction

Geo data form the foundation for different kinds of new services. Many innovative services have a relation to locations and are able to identify a user's position, display a map of friends or compute driving distances. Often, location-based services are sub-components of social services or community services.

Open Street Map (OSM) is a great source for geo data and enables such services. In contrast to closed geo services such as Google Maps, Open Street Map applications cannot only use pre-defined services to display maps – developers are able to create arbitrary new services as they are able to access the underlying structured geo data. All geometric information (in particular their vector representation) and non-geometric information (e.g. names, types, properties and relations to other objects) are available per object. A service developer can load the freely available geo data and import it into own formats appropriate for the respective service.

The geo data itself, however, still is incomplete for some services. This is because OSM data historically was mainly intended for map rendering. Even though con-

tributors can assign properties to every object, many objects mainly contain a name, a type, and geometry.

Other sources may fill this gap. As the counterpart of OSM for non-geographic data we may consider Wikipedia: an online encyclopedia and similar to OSM organized as a community project. Many objects appear in both databases, especially entries about cities, regions and countries, but also about touristic sights. Our main approach was to identify these data to enrich our geo database. We pursued the following goals:

- We want to integrate the short description of geo objects that appear in Wikipedia into our geo object database.
- We want to take over object properties from Wikipedia to our database. If these properties have numerical characteristics, we want to transfer the value and unit in a processible representation. In particular, all numerical values are mapped to a comparable format that, e.g. abstracts from the physical unit stated the Wikipedia article (e.g. miles vs. km).
- We want to index all Wikipedia texts that are related to geo objects by a text search index. Thus, we can use these words in search queries to find objects.

It was in particular not a goal to take vector geometries from Wikipedia. Even though for some objects such information is available in Wikipedia articles, we rely on the original OSM source. As a major objective, all these goals must be achieved by autonomous mechanisms that work without any user intervention. All potential decisions must be formulated as a-priori rules. Once started, the processing of many million geo objects must run in a batch manner.

In this paper, we present an import mechanism of encyclopedic data to a geo database inside the HomeRun project.

2 Related Work

We first distinguish *knowledge* from *data*. This distinction is not obvious. We here consider *knowledge* as collection of higher-level structures about an object, e.g. triples of *subject–predicate–object*. In contrast, *data* are plain values in e.g. columns of a database table. This means, extracting data follows a more traditional approach of storing objects in a table-oriented manner. Many projects that deal with Wikipedia as source try to extract *knowledge*. YAGO2 [6], e.g., is an ontology-driven project that extracts many million triples from Wikipedia, also spatial information. In contrast [1] takes OSM as source and transfers geo object information to a knowledge database. Even though a knowledge-based approach is more general and flexible, it also has to deal with the problem to understand value/unit pairs in a processible manner. In this paper, we only discuss the import of *data* that, however, can be stored in knowledge database in later stages.

Our approach deals with two data sources – geo data on the one hand and encyclopaedic facts on the other hand. Combining these data can be executed in two directions: starting from a geo object we look up the corresponding encyclopedia entry or

starting from an encyclopedia entry we may look up the corresponding geo object. As our goal was to enrich an existing geo database we pursued the first direction. The geo data source Open Street Map already has a formalized structure [3]. The most important issue is: how we make use of encyclopedic data from Wikipedia to enrich existing geo data. This especially means: is there an additional usage scenario for Wikipedia articles besides presenting articles to people.

Former research deals with similar questions. We can identify two major directions: access and semantic analyses. Different former work addresses the problem of *accessing* articles [7, 21]. For a longer time, the only programmatic way was to import so called Wikipedia *dumps*. They contain all articles in the *Wikitext* format [2] that forms the basis for rendering articles in HTML. The articles themselves are structured in XML, whereas also information about the articles are stored, e.g., a classification and information about creation and modifications. The goal of access platforms was to get an API to load the deep structure of articles, sections, paragraphs and links to other articles.

Once access to articles is established, the next goal was to conduct a semantic analysis. Thus, some platforms perform basic pre-computing to simplify such analyses. E.g. the platform presented in [20] derives a list of triples (*noun-phrase relation noun-phrase*) for every article. The main goal: classify articles and detect relations between articles and the respective concepts they describe. Some approaches focus on *relatedness* between concepts and try to answer questions such as 'How related are *Cat* and *Dog*?'. This is answered using text mining or analysing links between articles [7]. In [21] further natural language processing (*NLP*) techniques are incorporated. Further work use Wikipedia meta data, in particular the article classification and links between articles to detect semantic relatedness [4, 9, 10]. In [18] the authors suggest new types of meta data (typed links and attributes) that should be contributed by article authors to support a semantic analysis.

Even though several existing work deals with automatic analyses of Wikipedia articles, they mainly either focus on the technical access or to create relations between concepts. Incorporating content into other data repositories is not intended. To try to access certain properties of entities described in Wikipedia articles and to assign these values to (geo) objects is a new approach.

3 Incorporating Wikipedia Entries to Geo Objects

The *HomeRun* project [11] has a long tradition to deal with geo data. HomeRun is a platform for low-cost development of location-based services (also small scale services). It provides a set of basic services, e.g. import of geo data from public sources, map rendering and route planning. HomeRun also supports mobile devices to execute applications, even running 'offline', i.e. with all geo data stored on the device.

The main data source currently is Open Street Map, even though the HomeRun import chain also was able to take geo data from other geo data sources, e.g. from land survey offices. Merging data with different sources already is an important topic, as some information is missing or does not have the desired quality. E.g.:

- The OSM positions are only stored in 2D without height information, i.e. only represent a projection on the Earth's surface. Our import thus adds elevation information from NASA [8].
- For redundancy reasons, borders of cities, regions, states etc. are in OSM represented as an unordered collection of lines (so-called *relations*). If a border is incomplete, it is not possible to reconstruct a ring. Thus, we additionally load borders from another source [5], if borders cannot properly be generated from OSM.

As a next step, we want to add a further source: Wikipedia. This, however, differs in many aspects from existing work: first, information is not geometric as our other sources. Second and more important: the content is primarily intended to be read by users and it is not prepared for further automatic processing.

However, some existing features are encouraging:

- There is a special tag in the OSM object structure that indicates the URL to the corresponding Wikipedia article, if available. Thus, it is not required to 'guess' a link to an article, e.g. using the object's name.
- Even though an article is like a book or book chapter structured for human readers, there are some parts that in principle can automatically be read (Fig. 1). Foremost: for many types of articles there is an *infobox* that indicates important properties as pairs of keys and values.

The image shows a screenshot of the Wikipedia article for Nuremberg. Several elements are highlighted with red boxes and labeled with arrows:

- short description:** A box around the first paragraph of the article text.
- typical image:** A box around the main image of Nuremberg's skyline.
- map images or icons:** A box around the map and flag icons.
- infobox entries:** A box around the structured data table on the right side of the article.

The infobox entries include:

Country	Germany
State	Bavaria
Admin. region	Middle Franconia
District	Urban district
Government • Mayor	Ulrich Maly (SPD)
Area • City	196.46 km ² (71.99 sq mi)
Population (2015-12-31) ^[1] • City	488,976
 • Density	2,700/km ² (6,900/sq mi)
 • Urban	763,654 (includes Erlangen, Fürth and Schwabach)
 • Metro	3,500,000
Time zone	CET/CEST (UTC+1/+2)
Postal codes	90000-90491
Dialling codes	0911, 09122, 09129
Vehicle codes	N
Registration	
Website	nuremberg.de/g

Fig. 1. Example Wikipedia article and important entries to import

- Moreover: there is a common understanding in the community, which properties are expected for a special type of article – e.g. for cities we have properties such as *Country*, *Area*, *Population* and *Population Density*.
- A typical article contains a short description of the specified object. Also, this short description can be found in the article structure.

We thus have a good chance to automatically read certain content. Our goal was to pre-process content as much as possible. E.g. if a certain property indicates a numerical value, we want to be able to use this value for computations and statistical analyses later. Thus, our mechanism must be able to understand different number formats including physical or other units. Only if numerical interpretation fails, we take the original text, but then further computation is disabled.

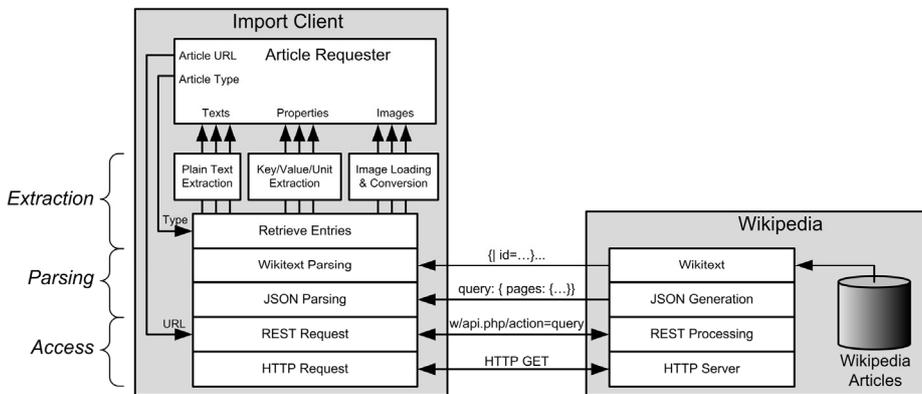


Fig. 2. Stack to access Wikipedia articles

For text mining among our geo data, we additionally add the article's plain text to our symbio-spatial search engine [16]. This engine allows to formulate complex queries that contain texts, addresses, object types but also spatial relations (e.g. *nearby a lake*). Textual queries currently only use object names and addresses, but we now are able to extend it to index entire articles. As some of these words may not represent the actual object, search words from articles get lower priorities compared to object names – however they significantly improve the search experience.

3.1 Retrieving Articles and Entries

Fig. 2 shows the execution stack to get articles and article entries. The approach is based on the Wikipedia REST API [19] to get the raw articles' format in Wikitext. This API also allows writing back edited articles. Thus, it is possible to develop an own authoring environment to read, edit and write Wikipedia articles without the need to use the Web-based editing facilities offered by Wikipedia. However, for our approach, we only need to read articles from Wikipedia.

It would also be possible to load Wikipedia articles directly using HTTP with the respective URL that is, e.g., a result of a browser search. As all URLs are built in a

straight-forward manner, it is very easy to generate the respective HTTP request. E.g. to get the HTML-formed article for the city of Nuremberg, the URL is

```
https://en.wikipedia.org/wiki/Nuremberg
```

However, the result is formed in HTML, thus underwent an additional step. As HTML pages both contain the page structure as well as layout definitions, it is more difficult to get the original article structure. Thus, we decided to use the REST API that provides the basic article definition. Another benefit: the API allows reading only a certain section of an article. Thus, entire processing can focus on interesting parts of the article rather than the entire text. We divide the steps to get desired article entries as follows (Fig. 2):

Access: These layers are responsible to transfer the actual data from the Wikipedia server to the requesting client:

- *HTTP:* The basis forms an HTTP GET request to the Wikipedia server. This request is always answered, even if a requested article is not available.
- *REST:* The request is structured according to the REST API. This means, the URL encodes the requested article, but also some parameters, e.g.

```
https://en.wikipedia.org/w/api.php
?action=query&titles=Nuremberg
&prop=revisions&rvprop=content
&rvsection=0&format=json
```

Parsing: These layers read the actual content and structure:

- *JSON:* The lower layer provides JSON decoding of results. It contains information about the success of the request. If the request was not successful, JSON variables contain a description of the problem. In case of success the variable `query: pages` contains the requested article section.
- *Wikitext:* The article structure is then parsed using a Wikitext parser. This is because of two reasons: first, we need to know the structure to find interesting parts of the article, e.g. the infobox or the short description. As these parts are not explicitly indicated in the article document, we need to detect them using structural information. Second: we need the structure to produce plain texts or to extract property values in a later stage.

Extraction: The upper layers access entries inside the article and convert them to the required format:

- *Retrieve Entries:* Once we know a list of requested entries, we try to detect them in the article structure. The type of article passed from the Article Requester helps this component to identify significant entries.
- The loaded entries now undergo a final step that depends on the entry types. Currently we support plain text, images and pairs of key values with units.

The result entries are now ready to be stored in the geo object's data. In our case of the HomeRun database, geo objects refer to entries in a properties table that holds all non-geometric characteristics. Until now, this table only contains properties expressed by object tags in Open Street Map. Now additional properties of Wikipedia are stored there.

3.2 Retrieving and Processing Entries

Once the content and structure is parsed, we try to identify interesting entries in the article. As Wikipedia stores a plenty of different articles with different structures, this is not trivial. The problem is, even if the layout is obvious for human readers, it is not obvious for a program to identify, e.g. the short description or infobox. Some examples of misleading document structures:

- The short description is not necessarily the first text in the article. Sometimes, e.g., there is a text declaring an article is a redirect from another article.
- Sometimes the first descriptive text section is about which other articles have a similar topic. Such a text contains a lot of links to other articles that are meaningless in plain texts stored in our database.
- An infobox is usually stored as Wikitext table. However, also the table of content or some images may appear as table.

As a result, we applied a heuristic approach to identify certain entries in the article. In this approach we formulate a set of rules that have to be fulfilled in order to get the right entry in the structure tree. Some examples:

- The short description is a text (not table) that does not contain any file download link, image or table. In addition some texts must not appear in the description, e.g. '*For other uses...see...*'.
- The infobox is either the first, second or last table of the first section with two columns and a set of expected property keywords in the first column (e.g. *State*, *Population*, *Postal codes*).

Wikipedia authors make use of so-called *templates*. Templates are building blocks of Wikitext fragments that provide a basic structuring and layout. To ensure a similar look of articles, Wikipedia makes heavily use of templates for, e.g. tables of contents, disambiguation references, maps etc. The Wikipedia API reflects the usage of templates with two modes to get articles: the requester can either load an article still with templates or can load an article where all templates are replaced by their respective Wikitext fragments. The latter case is called *expanded mode*. Even though, templates would simplify to find important entries in the article, we decided to use the expanded mode, because we do not have access to the underlying template definitions and they may change without prior notice.

Once appropriate entries are detected, a type-dependent conversion is applied. For plain text entries, all Wikitext tags are removed. As it is also possible for an article author to embed HTML into Wikitext, we also have to consider HTML tags. If tags

enclose references, they have to be removed. Formatting tags are removed at all. Finally, a conversion of character sets is applied to get the plain text in the desired encoding.

If the desired entry is an image, the image is loaded – usually the embedded images are represented by an URL and not embedded in Wikitext. In addition, images undergo a technical transformation regarding size, resolution and image format, to meet the requirements for the later usage (e.g. for a smart phone app or for map rendering).

3.3 Properties of Key/Value/Unit

Original geo objects imported from Open Street Map already contain properties. Even though Open Street Map allows assigning arbitrary property tags to characterize geo objects, they typically are used to

- define an object type (e.g. lake, highway, tree);
- define object names (for different purposes);
- control rendering of maps (e.g. tell if an object should not be painted at all);
- provide information for route planning (e.g. speed limits);
- provide additional information to the object's geometry;
- offer additional information about the object, e.g. opening hours, type of restaurant, parking prices.

Very often, information of the latter case is missing. This is because Open Street Map mainly is used to draw maps and maybe to support route planning – the origin of Open Street Map was to collect information about *streets*. This is also one reason why classification of geo objects from Open Street Map is very difficult [14].

Wikipedia on the other hand provides a lot of additional information, not stored in Open Street Map, e.g. population, important people, object classification, organisation, history, usage or costs. Even though the actual object geometry is exactly defined by OSM, additional geometric information can be read from Wikipedia, e.g. volumes or surface areas of barrier lakes.

The respective information may be spread over an article, thus not within reach for automatic processing. Fortunately, infoboxes as shown in Fig. 3 indicate the most important properties in tables of keys and values.

A major goal is not only to copy the pairs of key/value from the table, but to process values in such a way to enable further processing. This means:

- Keys must be mapped to a representation that allows to check for equality.
- Numbers must be transferred from their textual representation to native numbers.
- Units must be recognized and all properties of the same kind have to be mapped to the same unit in order to be comparable and to create sums of property values. E.g., units are converted between Imperial and metric units (e.g. miles to km), but also scaled to the same basic unit, e.g. m^2 to km^2 .

We want to fulfil these goals, because we want to support queries such as: 'What is the average capacity of barrier lakes in Bavaria?', or 'What were the average building

costs of Universities in Germany built between 1950 and 1960?', or 'What is the sum of power in Megawatts of all power plants in the North of Germany?'

These goals are surprisingly hard to achieve, because also infoboxes are primarily intended to be read by people, not programs. Some problems, we have to face:

- Equal keys have different representations regarding upper/lower case, abbreviations or the usage of hyphens.
- Different meanings of keys may have the same texts. E.g. the German '*Land*' may mean '*Country*' or '*Federal State*' in different articles.
- Some infoboxes are nested or have section headings. Here, a certain key is meaningless without the knowledge of the section heading. E.g. for universities, we may not only have '*employees*', but also '*academic*' and '*non-academic*' employees. Thus, the key '*academic*' solely is misleading if we do not take into account the section heading.

Gottleuba Dam	
	
Country	Germany
Location	Sächsische Schweiz-Osterzgebirge, Saxony
Coordinates	 50°50′04″N 13°55′51″E﻿ / ﻿50.83444°N 13.93083°E﻿ / 50.83444; 13.93083
Construction began	1965
Opening date	1976
Dam and spillways	
Impounds	Gottleuba
Height	53.2 m (174.5 ft)
Length	327 m (1,073 ft)
Width (crest)	7 m (23 ft)
Dam volume	270,000 m ³ (9,500,000 cu ft)
Spillway capacity	176 m ³ /s (6,215 cu ft/s)
Reservoir	
Total capacity	14.02 hm ³ (11,370 acre-ft)
Catchment area	35.3 km ² (13.6 sq mi)
Surface area	660,000 m ² (7,104,181 sq ft)
Power station	
Type	Conventional
Installed capacity	53 KW

Fig. 3. Example of a Wikipedia infobox

- Even for numeric values, there exist several ways for textual representations. E.g. we can use blanks or ',' to separate thousands, or we can use scientific representation such as $3.5 \cdot 10^6$. Small numbers can be written as words, e.g. '*none*', '*zero*' or '*one*'.
- There exist words or letters that modify the value, e.g. 'million', or 'mio.'. Some letters, e.g. k or M are considered to be part of the physical unit, but sometimes lead to confusion. E.g. km² obviously does not mean thousand m² but (km)². The

problem is even worse, as authors faulty write, e.g. 'K' instead of 'k', or m2 instead of m².

- Even for a certain unit, there exist multiple spellings. E.g. for monetary costs, we have '€', 'Euro', 'Eur.', and 'EUR' only for €-values.

There may be textual supplements behind the value, e.g. '*measured 2005*' or '*see below*'. We have to detect and remove these additions to get the raw unit. However, this is not trivial, as most physical units also contain letters, similar to the additional words.

Table 1. Wikipedia properties (selection)

Property	Restrictions, Variations	Numb/Unit
Height, Depth, Elevation	min, max, height above valley bottom, height difference, depth of reservoirs	m
Area	...of city, countries, regions	km ²
Area	...of barrier lakes, estates, parks, places	m ²
Length, Width	...of barrier lakes, buildings	m
Volume, Capacity	...of barrier lakes, reservoirs	m ³
Volume per Time	Spillway capacity, rate of flow	m ³ /s
Slice Plane, Radius, Circumference	...of pipes, tunnels	m ² , m resp.
Date	start, stop of commencement, extended, start of operation, idle since, closed, demolition	date
Costs	budget, sales, building costs	€
Building Type	...of castles, walls, ruins, new buildings	-
Power	...of power plants, transformers, power storages	MW
People	architect, builder, planer	-
Government, Administration	mayor, vice, district administrator, chair, director	-
Name	official name, local name	-
Affiliation	state, country, region, district, city, quarter, municipality	-
Number of parts	number of quarters, number of regions	integer
Population	...of city, region, country	integer
Density of population	...of city, region, country	1/km ²
People counts	number of e.g. visitors, employees, members	integer
Address	town hall, administration, head office	-
Keys	Official keys such as NUTS, LOCODE, AGS, IBNR, BIC	resp. format
Vehicle Registration	First letters of licence plates	resp. format
Dialing Codes	First digits of telephone numbers	resp. format

To control and structure the recognition of values and units, we provide a table of keys and their values/units. Table 1 shows some of them, but the table is by far not

complete. We detected 348 different keywords and assigned rules to understand the values. Besides the physical units for keys, we have lists of all spellings (and typical misspellings) of units, a list of words for value multipliers and patterns for additional texts. Moreover we have a list of typical textual expressions that actually indicate a number, e.g. *'uninhabited'* for *'Population: 0'*.

4 Evaluation And Sample Scenarios

4.1 Evaluation of Results

We fully implemented and integrated the approach in our HomeRun tool chain. During the import of OSM data a lot of pre-processing is performed [12] and the original data is enriched. This is required, as original OSM data has several drawbacks regarding geometry representation, route planning and classification of objects. This is an ideal point to query Wikipedia articles for geo objects.

OSM offers its data in compressed XML files – so called *planet* files. There exist sub files for continents, countries and regions. The following analysis is based on the file *Germany* from two time stamps: Nov. 2015 (1) and Jan. 2017 (2). Table 2 shows basic numbers.

Table 2. Wikipedia import statistics (OSM file Germany)

Category	Count (1)	Count (2)
Geo object statistics		
OSM objects in the file	43407661	50864433
OSM objects with Wikipedia link	71040	83169
Ratio	0.164%	0.164%
Success/Failures		
Successfully processed	56041	64781
Success ratio	78.9%	77.9%
Failed GET/JSON indicated error	3804	4178
Failed parsing Wikitext	11195	14210
Infobox properties		
Infobox properties (total count)	383266	430917
Infobox properties with value or unit error	1203	1447
Error ratio	0.31%	0.36%
Avg. infobox properties per geo object	6.84	6.65

A first observation: even though the amount of total geo objects increased, the ratio of geo objects with Wikipedia entry remains nearly the same. It is considerably low with 0.164%. One explanation: most of the geo objects with a reference are areas with

an administrative border such as cities, regions or counties. But these types of objects only represent a very small amount of geo objects in the OSM database (only 0.05%).

From the amount of geo objects with a Wikipedia link, some references could not successfully be processed, due to technical errors:

- Some accesses produce low-level errors related to HTTP GET or the JSON result does not contain a Wikipedia article. Usually the reason for this was an outdated URL in the OSM entry. Sometimes, the original page was replaced by a hub page to represent the different meanings of a term.
- A high amount of errors is a result of parsing the Wikitext content. This was because either the sources were malformed, or the Wikitext parser was not able to successfully parse the structure. Note that Wikitext may contain embedded HTML, thus can be very complex – actually the Wikitext parser must also contain an HTML parser to get the entire structure tree. The amount of parsing can be reduced using another (esp. more tolerant) parser.

The last section in Table 2 shows statistics about infobox properties. The amount of read failures is very low (0.31%). Most of them are a result of unrecognized additional texts in the value descriptions. Most of them cannot easily be solved by automatic mechanisms, as they significantly affect the interpretation of values. Some examples:

- Texts such as '*in summertime*' or '*only department...*' limit the value to certain times, locations or impose other limitations. As a result, the value cannot be used *as is*.
- Some texts indicate ranges or open intervals, e.g. '*more than*' or '*value₁ – value₂*'. As the distribution of values inside these intervals is not given, we cannot express this property by a single value.

The second reason for failures were typos in units (e.g. m² instead of m³ for volumes). In principle, most of the problems with properties and values cannot easily be solved in the current workflow and format for Wikipedia articles, as they conflict the major goal of Wikipedia: to provide a human-readable article that may contain additional values and properties that again are interpreted by people.

Table 3 shows the distribution of Wikipedia articles to geo object types. Our classification of geo objects is a so-called *strong classification* [14] in contrast to the *weak classification* of the original OSM source.

As stated above, most geo objects with a corresponding Wikipedia article are regions with an administrative border e.g., cities. The second type of objects are those with a touristic or historic meaning such as castles, museums, monuments or touristic sights. The third type are objects related to traffic and transportation, e.g. railway stations, canals, routes or roads.

If we look at the degree of objects of a certain type that are covered by Wikipedia articles, only cities have a sufficient coverage (95.8%). For other object types the coverage is too low, for e.g., a detailed statistical analysis. However, for certain objects, an application can benefit from the additional Wikipedia entries.

Table 3. Distribution of Wikipedia Articles to Geo Object Types (file *Germany*, Jan. 2017)

Geo object type	Objects with Wikipedia article	Total number of objects	Ratio (%)
City	11053	11543	95.8
County	400	557	71.8
Parish	980	1413	69.4
Provinc.			
Town	1492	2410	61.9
Castle	1483	4356	34.0
District	3100	9934	31.2
Suburb	1567	8913	17.6
Museum	793	6928	11.4
Railway Station	731	6986	10.5
Village	7587	80879	9.4
Theater	211	2358	8.9
Church	1404	18733	7.5
Protected Landscape	283	3910	7.2
Archeol. Site	486	7171	6.8
Touristic Site	3041	52217	5.8
Historic. Site	1874	45999	4.1
Monument	413	10267	4.0
Route	500	13520	3.7
Mountain	772	22081	3.5

Geo object type	Objects with Wikipedia article	Total number of objects	Ratio (%)
Canal	402	12268	3.3
Chur. Instit.	1179	37640	3.1
Chapel	208	7708	2.698
Tower	224	8560	2.617
Bike Route	223	10181	2.190
Hike Route	318	20351	1.563
River	4425	383797	1.153
School (basic & sec.)	435	38121	1.141
Rail Track (demount.)	303	26593	1.139
Park	371	32873	1.129
Graveyard	350	33290	1.051
Wayside Cross	311	34990	0.889
Rail Track	768	104773	0.733
Highway	326	46007	0.709
Pedestr. Area	265	38398	0.690
Fed. Highway	883	153621	0.575
Industr. Area	235	43508	0.540
Lake	1083	217195	0.499
Bridge	948	278986	0.340

4.2 Sample Scenarios

The imported entries are incorporated into the HomeRun database format and reside side-by-side with entries originated by Open Street Map. All HomeRun services take the required data from the HomeRun database in their respective data representation. Whereas the map rendering service *dorenda* [13] operates on the HomeRun SQL database, the route planning service *donavio* [15] first transfers the road network in an own format, specialized for high-performance graph algorithms.

The original SQL database now allows executing queries on entries from Wikipedia. E.g. if we want to get the power facility with the highest output power, we simply enter:

```

select d_id from domain_properties
where p_id=10411 and double_value=
(select max(double_value)
from domain_properties where p_id=10411)

```

In this query, the property ID 10411 represents the Wikipedia infobox entry '*power plant capacity in Megawatts*'. The result `d_id` of this query is the geo object's ID. With this, it is possible to query everything known from this object e.g., its name, geometry, or further properties. As another example, we query universities with more than 40000 students:

```

select d_id from domain_properties
where p_id=11519 and int_value>40000

```

Here, the property ID 11519 represents '*number of enrolled students*'. Note that such queries are only possible, since the respective entries are taken from Wikipedia – these properties currently are not available in OSM.



Fig. 4. Smart phone widget that presents a description of the current location

As another example, we extended our *HomeRun Reverse Geocoding* framework [17]. It provides a purely textual description of the current location that may, e.g. be read aloud by text-to-speech services of a smart phone for blind people. In the older version, it only provides a small text that summarizes city, address, nearby places or important sights. With our new approach, we are able also to print the short description from Wikipedia of the most important geo object in the nearer area (Fig. 4).

5 Conclusions

In this paper we presented an approach to import data from Wikipedia to enrich our geo data inside the HomeRun project. The import mechanism was fully established and integrated to HomeRun's import tool chain. The general results are encouraging: we now get a lot of additional properties as processible value with unit, currently not available in the geo data source. In addition, we get a short description and typical images of the corresponding object. We can use these entries for different types of applications and services.

It causes considerable efforts to get processible properties from texts. This is because Wikipedia texts are not intended for this type of usage. The problem is very similar to semantic Web approaches: if sources are primarily prepared to be rendered for users, it is difficult to convert them into machine readable content afterwards. We solved this problem with a rules-based approach that relies on the *strong classification* of the HomeRun geo data representation. Another solution would be to extend article structures to store such properties. This, however, would change the overall goal and article authors have to be convinced, to administrate such structures.

References

1. Auer S., Lehmann J., Hellmann S. 2009: LinkedGeoData: Adding a spatial dimension to the web of data, The Semantic Web - ISWC 2009, LNCS 5823, 731-746
2. Barrett, D. J., 2008: MediaWiki (Wikipedia and Beyond), O'Reilly
3. Bennett, J., 2010: OpenStreetMap, Packt Publishing
4. Gabrilovich, E., Markovitch, S., 2007, Computing semantic relatedness using Wikipedia-based explicit semantic analysis, Proceedings of the 20th international joint conference on Artificial intelligence (IJCAI'07), Hyderabad, India, Jan. 6-12, 2007, 1606-1611
5. Global Administrative Areas, <http://gadm.org/>
6. Hoffart, J., Suchanek, F. M., Berberich, K., Weikum, G., YAGO2, 2013: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia, Artificial Intelligence, 194, 28-61
7. Milne, D., Witten. I. H. 2013: An open-source toolkit for mining Wikipedia, Elsevier, Artificial Intelligence 194 (2013) 222-239
8. NASA, Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER), <http://asterweb.jpl.nasa.gov/>
9. Ponzetto, S. P., Strube, M., 2007, Deriving a large scale taxonomy from Wikipedia, AAAI'07 Proceedings of the 22nd national conference on Artificial intelligence - Volume 2, July 22-26, 2007, Vancouver, British Columbia, 1440-1445

10. Prato, A., Ronchetti, M., 2009, Using Wikipedia as a Reference for Extracting Semantic Information from a Text, Third International Conference on Advances in Semantic Processing, 2009. SEMAPRO '09. Oct. 11-16, 2009, Sliema, Malta, 56 - 61
11. Roth, J., 2010: Die HomeRun-Plattform für ortsbezogene Dienste außerhalb des Massenmarktes, in A. Zipf, S. Lanig, M. Bauer (eds.) 6. GI/ITG KuVS Workshop Location based services and applications, Heidelberger Geographische Bausteine Heft 18, 2010 (in German)
12. Roth, J., 2010: Übernahme von Geodatenbeständen aus Open Street Map und Bereitstellung einer effizienten Zugriffsmöglichkeit für ortsbezogene Dienste, Praxis der Informationsverarbeitung und Kommunikation (PIK), Vol. 13, No. 4, 2010 (in German)
13. Roth, J., 2013: Combining Symbolic and Spatial Exploratory Search – the Homerun Explorer, Innovative Internet Computing Systems (I2CS), Hagen, June 19-21 2013, Fortschritt-Berichte VDI, Reihe 10, Nr. 826, 94-108
14. Roth, J., 2014: From Weak to Strong Geo Object Classification, in Schau V., Eichler G., Roth J. (eds): Proc of the 10th Workshop Location-based application and Services (LBAS) Sept. 16-17 2013, University of Jena, Germany, Logos Verlag Berlin, 3-12
15. Roth, J., 2015: Predicting Route Targets Based on Optimality Considerations, International Conference on Innovations for Community Services (I4CS), Reims (France) June 4-6, 2014, IEEE xplore, 61-68
16. Roth, J., 2014: Fast Spatio-Symbolic Searching in Huge Geo Databases, Proc. of the 11th Workshop Location-based application and Services (LBAS), Sept. 18-19, 2014, Telekom Innovation Laboratories, Darmstadt, Germany, Logos Verlag, 2015
17. Roth, J., 2015: Generating Meaningful Location Descriptions, International Conference on Innovations for Community Services (I4CS), July 8-10, 2015, Nuremberg (Germany), IEEE xplore, 30-37
18. Völkel, M., Krötzsch, M., Vrandečić, D., Haller, H., Studer, R., 2006, Semantic Wikipedia, Proceedings of the 15th international conference on World Wide Web (WWW '06), May 23–26, 2006, Edinburgh, Scotland, 585-594
19. Wikipedia 2017, MediaWiki action API, https://www.mediawiki.org/wiki/API:Main_page/en
20. Wu, F., Weld, D. S., 2010, Open Information Extraction using Wikipedia, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, 11-16 July 2010, 118–127
21. Zesch, T., Müller, C., Gurevych, I., 2008: Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary, Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), May 28-30, 2008, Marrakech, Morocco