Automatic Tag Enrichment for Points-of-Interest in Open Street Map

Stefan Funke[1(✉)] and Sabine Storandt[2]

[1] University of Stuttgart, Stuttgart, Germany
funke@fmi.uni-stuttgart.de
[2] JMU Würzburg, Würzburg, Germany
storandt@informatik.uni-wuerzburg.de

Abstract. The user experience of geo-search engines and map services heavily depends on the quality of the underlying data. This is especially an issue for crowd-sourced data as e.g., collected and offered by the Open Street Map (OSM) project. In this paper we are focusing on points-of-interests (POIs), such as restaurants, shops, hotels and leisure facilities. Many of those are incompletely tagged in OSM (missing e.g., the *amenity* tag), which leads to such POIs not showing up in respective search queries or not being displayed correctly on the map. We develop methods that can automatically infer tags characterizing POIs solely based on the POI names. The idea being that many POI names already contain sufficient information for tagging. For example, 'Pizzeria Bella Italia' and 'Chau's Wok' most certainly refer to restaurants, whereas 'Cut & Color' is likely a hairdresser. We employ machine learning techniques to extrapolate such additional tag information; our approach yields an accuracy of more than 85% for the considered tags. Moreover, for restaurants, we aimed for extrapolation of the respective cuisine tag (*italian*, *sushi*, etc.). For more than 19.000 out of 28.000 restaurants in Germany lacking the *cuisine* tag, our approach assigned a cuisine. In a random sample of those assignments 98% of these appeared to be true.

1 Introduction

In the context of crowd-sourced data gathering efforts like Open Street Map (OSM), the issue of data quality and completeness is of utmost importance. Competing commercial offerings will always claim an alleged superior data quality to justify their business. Sometimes, their typically more centralized approach of data maintenance and collection indeed allows for an easier monitoring of quality issues; for crowed-based approaches this is much harder to achieve.

In the concrete case of OSM, data quality is very much dependent on the contributors' tagging discipline. While there are well thought out guidelines how to tag mapped elements, in principle every contributor can tag at his/her discretion. This freedom undisputedly has its advantages in terms of flexibility, it also creates some consistency problems, though. For example, when querying a location-based service or a geo-search engine for points-of-interest (POIs) in a certain

D. Brosset et al. (Eds.): W2GIS 2017, LNCS 10181, pp. 3–18, 2017.
DOI: 10.1007/978-3-319-55998-8_1

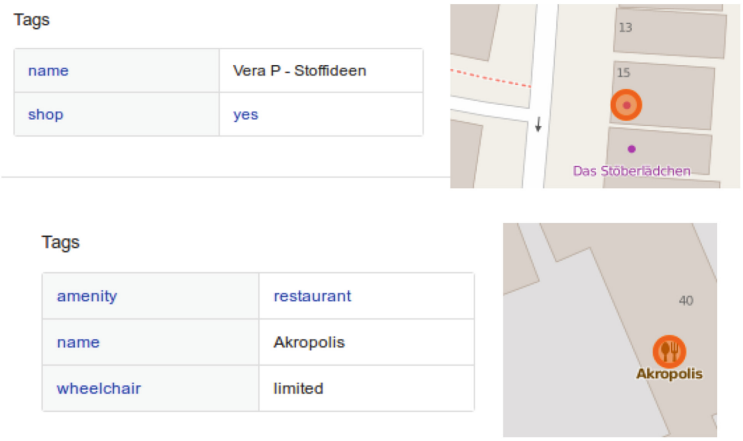

Fig. 1. Missing tag information in OSM. In the upper example, there is a *shop* tag, but it is only set to *yes*. As the POI is indeed a drapery, the *shop* tag should be set to *fabric* instead. In the lower example, most likely a Greek restaurant is depicted (considering its name). But there is no *cuisine* tag, so a search for 'Greek restaurants' will not include this POI.

region or next to the current user location, one often asks for classes ('hotels New York', 'supermarkets Berlin', 'Italian restaurants London') rather than single points ('Hotel Belvedere New York'). In OpenStreetMap (OSM), one can specify the basic class along with every POI e.g., via the *tourism* tag (*tourism=hotel*), the *shop* tag (*shop=supermarket*), the *amenity* tag (*amenity=restaurant*), or several other specialized tags as the *cuisine* tag (*cuisine=italian*) for restaurants. Not providing the appropriate tags when mapping the respective element typically leads to omission of these elements in the result list for a class-based query. These tags are also useful to categorize search results. For example, when searching for 'Venice beach' the user should be informed that there are beaches, hotels, fitness studios and clothing stores with that name. See Fig. 1 for two examples of missing or wrong tags.

In OSM, there are plenty of POIs where important tags are not provided. Many of those POIs exhibit a *name* tag (as e.g. 'Sunset Hotel', 'Walmart', 'Thai Bistro'), though, which already contains some information. In this paper, we investigate methods for automatic extrapolation of tags based on POI names. Using machine learning tools we extract typical words and phrases that occur in *name* tags and learn respective POI classifiers.

As a result we can augment the existing OSM data by inferred tags and improve the data quality. This can be done either fully autonomously or with humans in the loop who verify the augmentations suggested by our algorithms – again, a community-based approval mechanism for such changes might be an interesting option. There are several tools out there for detecting data inconsistencies and missing data (e.g. Keep Right[1]) to make them visible to the

[1] http://wiki.openstreetmap.org/wiki/Keep_Right.

community and spur corrections. Such services could also feature our suggestions for tag enrichment. But ideally, mapping and tagging tools for OSM would already provide automatic suggestions as soon as a POI name was typed in (see Fig. 2 for examples). In that way the user is encouraged to enter more information, and with already listing the best options for doing so, it is just a matter of one click.

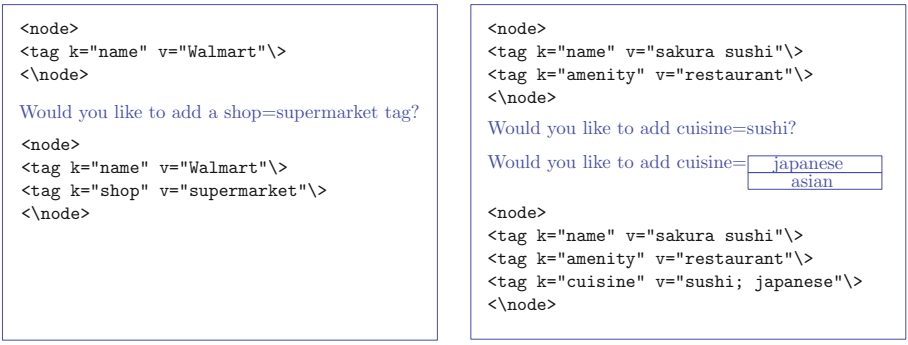

Fig. 2. Dialog system that encourages users to provide more information during tagging by making specific suggestions based on learned name classifiers.

1.1 Related Work

Numerous papers use machine learning (ML) techniques to work on/for OSM data. Basically, ML can be employed either on the application level – leaving the underlying data pool untouched – or to verify and even augment the underlying data pool. For the former, e.g., [7] propose the use of artificial neural networks and genetic algorithms to infer land-use patterns without directly feeding the results back into the OSM data pool. For the latter, e.g., inferring the structure of the road network by analyzing GPS traces using ML techniques was discussed in [4]. There are plenty of other studies on completeness and correctness of the OSM data, see e.g. [6] or [8]. The problem is that there either needs to be a ground truth one can compare to in an automated way (as investigated in [3] for building footprints in Munich), or data has to be manually compared to proprietary data. In both cases, the ground truth sample sizes are typically limited. For our application, though, there is a large pool of correctly and completely tagged POIs from which features can be extracted.

Machine learning was also applied in order to automatically assess the quality of the road network data [10]. Here, characteristics of certain street types (as e.g., motorways) were learned, including features such as total street length, number of dead-ends, number of intersection points and connectivity to other street types. In the quality analysis, feature vectors of streets are compared to the learned feature vector for the respective street type. If they do not resemble each other it is assumed that the data quality of the considered street is poor.

In our scenario of extrapolating POI tags, it is easy to verify that tags are missing. But determining what kind of tags should be extrapolated is challenging.

Preliminary results on automated quality improvement of OSM data were also reported in [9]. Here, Artificial Neural Networks (ANN) are applied to distinguish residential and pedestrian streets; features include node count within a bounding box and betweenness centrality. In [5], also missing street sections as well as street names were automatically extrapolated using ML (based on Random Forests) with an accuracy of over 90%. But in all these cases, the tags that should be learned were predefined (street type expressed as *highway* tag or *street name*), while in our application a variety of tags (e.g.*amenity, shop, tourism*, etc.) might apply, and there are typically hundreds of possible values for each (e.g. *amenity=restaurant, amenity=pub, amenity=kindergarten, amenity=bus_station* and so on). In [11], an automatic recommendation system for OSM tags was discussed, making use of SVMs. The focus there is rather on the geometry of the entities, like number of contained points and area. Names are only used if they contain precomputed indicator words as school or park. With our approach, we can also extrapolate tags for POIs where the name does not explicitly contain the amenity (e.g. *50's diner* is recognized as a burger restaurant).

There are also several approaches for other auto tagging applications, e.g. using deep convolutional neural networks as described in [2] for music classification or for tagging web services [12]. But there, the input differs significantly from the structured OSM data we consider in this paper.

1.2 Contribution

We describe a framework for automatic tag extrapolation based on POI names. We explain in detail how to process the OSM data and how to determine extrapolatable tags. Then we introduce a machine learning approach primarily based on k-grams of POI names. We apply our framework to extrapolate selected *tourism, leisure, amenity, shop* and *cuisine* tags for the dataset of Germany. Our experimental evaluation shows the ability of our framework to enrich the OSM data. For example, for *cuisine*, we can extrapolate more than 70% of missing tags with a precision of 98%.

2 POIs in OSM

We are interested in nodes in the OSM data that potentially are POIs. Nodes in OSM come with a specific ID and geo-coordinates (lat/lon). In addition, tags in form of key-value-pairs (k, v) can be specified, as shown in this example:

```
<node id=‘‘360485476’’ visible=‘‘true’’
uid=‘‘11374’’ lat=‘‘47.9955298’’ lon=‘‘7.8447728’’>
<tag k=‘‘name’’ v=‘‘Backshop’’/>
<tag k=‘‘shop’’ v=‘‘bakery’’/>
<tag k=‘‘wheelchair’’ v=‘‘yes’’/>
</node>
```

In the following, we will list tags that indicate POIs of various kinds (like *shop*) and explain their taxonomy.

2.1 Restaurants, Cafes, Pubs and Fast-Food Facilities

Let us first consider tags associated with going out to eat or drink. We differentiate restaurants, cafes, pubs and fast-food facilities in accordance with the OSM Wiki[2]. The *amenity=restaurant* tag is by far the most frequent *amenity* tag. As specified in the Wiki, *amenity=restaurant* should be used 'for a generally formal place with sit-down facilities selling full meals served by waiters and often licensed (where allowed) to sell alcoholic drinks'. The cuisine tag can be used in addition to further refine what kind of restaurant it is. A cuisine can refer to the ethnicity of the food (*cuisine=chinese*), to the way of food preparation (*cuisine=wok* or *cuisine=grill*), to the food itself (*cuisine=pasta*) or to other classifications (*cuisine=fine_dining*).

Instead of *amenity=restaurant*, one should use the tag *amenity=fast_food* for 'a place concentrating on very fast counter-only service and take-away food'. The *cuisine* tag is used in this context as well (e.g. *cuisine=burger*). Nevertheless, also *amenity=restaurant* and *cuisine=fast_food* or *cuisine=burger* are commonly used. The tag *amenity=cafe* should be used for 'a generally informal place with sit-down facilities selling beverages and light meals and/or snacks', including coffee-shops, tea shops and bistros. Again, combinations like *amenity=restaurant* and *cuisine=coffee_shop* are often used instead.

For drinking, the tags *amenity=pub, amenity=biergarten* and *amenity=bar* are intended. All are used for establishments that sell 'alcoholic drinks to be consumed on the premises'. Hereby, a pub should indicate a facility where you can sit down, food is available and the atmosphere is rather relaxed. In contrast, a bar is assumed to be more noisy, with music and no meal-like food. A biergarten is like a pub, but outdoors. Also combinations like *amenity=pub* and *biergarten=yes* are possible. Other *amenity* tags associated with eating and drinking are *bbq, drinking_water, food_court* and *ice_cream*.

Note, that there is overlap between all mentioned amenities, and tags are combined in various ways to classify places. Therefore we consider all of them together in our learning approach.

2.2 Shops, Services and Entertainment

Besides restaurants, cafes, pubs and similar facilities, there exists a large variety of other amenities that mark POIs. For example, places for entertainment as *cinema, theatre, casino* or *nightclub* fall into that category. But also *parking, post_office, post_box, fuel* (for gas stations), *public toilets, library, dentist* and other facilities that are public or provide some kind of (health) service are valid *amenity* tags. Usually, they are more easily to classify than facilities associated with eating and drinking. But there is some overlap with another important tag,

[2] https://wiki.openstreetmap.org/wiki/Main_Page.

namely the shop tag. It should be used for all kind of facilities where products are sold, as e.g. supermarkets, kiosks, bakeries, clothing stores, furniture stores, and many more. There are also nodes tagged with *amenity=shop, amenity=shopping* or variants thereof. For those the OSM Wiki encourages to check whether a shop tag can be used instead.

2.3 Hotels, Tourism Spots and Leisure Facilities

The *tourism* tag is used to describe possibilities for paid lodging as *hotel, hostel, motel, camp_site* and so on. But also in the context of sights and attractions the *tourism* tag should be used, including *zoo, museum* or *theme_park*. There is also an attraction tag for specification, e.g. *attraction=big_wheel*. Alternatively, one can tag a sight primarily according to its type, e.g. *waterway=waterfall* and then add *tourism=yes*. The *leisure* tag applies to all kind of facilities where people can spend their spare time. Most prominent representatives are *playground* and *sports_centre*. When it comes to e.g. parks and gardens there is some overlap with the *tourism* tag, though.

3 Extrapolation Framework

To be able to extrapolate missing tag information our overall plan is to identify characteristic properties (also called features) of the name tag value that are 'typical' for a certain *amenity, cuisine, shop*, etc. tag. For those OSM nodes which should bear a respective tag, (e.g. *shop=hairdresser*) but do not because either the information was not provided or added in a non-conformal way (e.g. as part of the informal description tag), we hope to infer the missing tag by examining its name tag for characteristic features typical for nodes actually bearing this tag. This section gives an overview of the necessary steps for this task.

3.1 Data Extraction and Processing

We only consider named nodes in the OSM data, i.e. the tag k = 'name' is required. Most named OSM nodes refer to streets or parts of the public transport system (as e.g. *amenity=bus_station, name=Norris_Street*). Such nodes are not POIs according to our definition. They are excluded by checking for presence of *highway* and *public_transport* tags. Moreover, we pruned nodes tagged with *cemetery, power, fire_hydrant, historic, natural* and *man_made*. In order to learn the correlation between names and certain tags, we need to have POIs with complete information, that is, a name and the tags we are interested in. These POIs will serve as training data in our machine learning approach. For extrapolation, we consider the POIs that potentially miss tags of a certain kind.

3.2 Selection of Extrapolatable Tags

Not all tags are suitable for extrapolation. First, there need to be sufficiently many POIs which exhibit a certain tag to allow the machine learning approach to work. There are plenty of tags in the OSM data which occur only once or very few times, either because they are over-specified (e.g. *cuisine=asian;curry;noodle*), too specific (e.g. *cuisine=self made cake*), home-brewed (e.g. *cuisine=german-bohemian*), exhibit spelling errors (e.g. *cuisine=chineese*), are not in English (e.g. *cuisine=bürgerliche_küche*), simply used wrong (e.g. *cuisine=music*) or indeed rare (e.g. *cuisine=israelian*). Therefore, we count how often a certain tag or a combination of tags occurs and only further consider tags whose respective count exceeds 200. Second, there are tags which subsume each other or overlap in terms of their semantics. For example, *cuisine=asian* is used but also *cuisine=japanese, chinese, vietnamese, thai* amongst others. To accommodate for such dependencies, we first group tags and specify their relations manually. If we consider two tags to be interchangeable like *cuisine=steakhouse* and *cuisine=steak*, we merge them into one. For a class subsuming several others, we check whether the subclasses are large on their own. If that is the case, we try to learn the more specific group. Otherwise, we cumulate the names of all subgroups and try to learn the more general group.

3.3 Feature Extraction

Once we fixed the set of classes/tags, we need to specify suitable features (characteristic properties of the name tag) that allow to learn the correlation between names and tags. We want to identify words and phrases that are typical for certain classes. Consider for example this list of names of hairdressers of a city in Germany:

Claudia's Frisierstube, Cut & Color, Der Goldene Schnitt, emporio, Freiseur Ryf, Frerich, Friseur Ganter, Friseur Roth, Friseur Ryf, Friseur Salon H.Jonas, Frisör Charisma, Frisörsalon Annette, Frisuren-Atelier, Gutjahr Hairlounge, Haar-WG, HaarBalance, HaarBar, Haarstudio Burger, Haarstudio Marina Lindle, Haarstudio Marita, Hair Body Soul, Hair Saloon, hairkiller, HairSpeed, Helbling, Horst Fischer Friseursalon, Nölle, Power Hair Styling, Salon Carmen, Salon Haargenau, Toni & Guy, Via Style

We observe that e.g. 'fris', 'seur', 'haar' (the German word for hair), 'hair', 'styl', 'salo' and 'studio' appear multiple times and therefore might be good indicators for *shop=hairdresser*. Determining indicator phrases manually for thousands of POIs in hundreds of classes is impractical, though. To automatize the process, we proceed as follows. Let N be the list of names associated with a certain tag (e.g., *shop=hairdresser*). For each name in N we construct all k-grams for k between 3 and 10. A k-gram of a string/word is a consecutive substring of length k. For example, all 4-grams for 'Hair Styling' would be 'Hair', 'air', ir S', 'r St', ' Sty', 'Styl', 'tyli', 'ylin', 'ling'. We count for each k-gram how often it occurs in N. We consider a k-gram to be significant when at least two percent of names in N share this k-gram. If a significant k-gram is a substring of

another significant k-gram with a similar count (e.g. considering *cuisine=burger*, 'onald' and 'McDonald's' both appear 753 times), we prune the smaller k-gram as we assume it has no significance on its own. As a counter-example, 'burger' appears more often in the list than 'Burger King', therefore both k-grams are kept. After this pruning step, we have for each class a final list of indicator phrases (k-grams) at hand, each with a percentage specifying the fraction of nodes of this class exhibiting the respective k-gram. Then we construct for each name a so-called feature vector. A feature vector of a name is a vector with as many real-valued entries as there are class/significant k-gram combinations. The entry corresponding to a certain indicator phrase and class is set to the length of the phrase multiplied by the percentage of nodes in the class containing this k-gram. Here the intuition is that long shared sequences between the name and the names in N, as well as a shared sequence with many names in N indicate a high correlation with the class. Standard machine learning can then be applied to the derived feature vectors of the names.

3.4 Machine Learning

We use the Random Forest [1] approach for learning the classifier, as it allows to take care of dependencies between the feature vector entries. We expect to learn a classifier for POI names that can decide which tag (from a given set) should be assigned. As it might very well be the case that no tag is suitable, we have to accommodate for that. Therefore, we not only aim for the classification itself but rather for a probability distribution over the classes. So for each name to classify, we derive a probability for every class denoting how likely it is that the name belongs to this class. The sum over all class probabilities for a name always equals 1. If no class has a significantly higher probability than the others, it can be assumed that none of the classes fit.

3.5 Evaluation

In order to check whether the selected features allow for an accurate classification, we first perform a 5-fold cross validation. Here the training data (the set of POIs with known tags) is split into five equal sized parts $P1, P2, P3, P4, P5$. Then for each part Pi, we train on the other four parts and classify the feature vectors in Pi on that basis. So we train on $P2, P3, P4, P5$ and check whether the resulting classifier works as intended for set P1 (for which we know the correct classification), and repeat for P2 vs $P1, P3, P4, P5$, as well as P3 vs $P1, P2, P4, P5$, etc. As a quality measure we compute the following statistical standard machine learning quantities (averaged over the 5 experiments):

Recall: for a specific category as e.g. *amenity=hair_dresser*, we consider the ratio (#items correctly classified by our algorithm)/(#items that really have *amenity=hair_dresser* items)

Precision: for a specific category we consider the ratio (#items correctly classified by our algorithm)/(#total number of items that are classified as *amenity=hair_dresser* by our algorithm)

A perfect precision score of 1.0 (or 100%) means that every item classified as having *amenity=hair_dresser* by our algorithm is indeed a hairdresser (but does not imply that every hairdresser was found). On the other hand, a perfect recall score of 1.0 means that all hairdressers were actually classified as having *amenity=hair_dresser* by our algorithm. Note that for tag enrichment, the quality measures precision and accuracy are more important than recall – because adding a wrong tag to a POI is problematic and should be avoided by all means, while not being able to add extra tags to all POIs is not as severe and partially unavoidable (e.g. if the POI name does not contain any useful information at all).

4 Experimental Results

We implemented the described framework using C++ for the feature extraction and Python for machin learning. In particular, we relied on the scikit-learn package [13] there, also using the predefined standard parameters. Our experiments were conducted on a single core of an Intel i5-4300U CPU with 1.90 GHz and 12 GB RAM. The Germany data set extracted from OSM as basis for all our experiments contains 771,325 named nodes. Among those, we identified 84,618 with insufficient tagging (about 12,000 contained only the name tag, the others only non-classifying additional tags as e.g. *wheelchair=yes/no*, *opening_hours*, website or Wikipedia references and address information).

4.1 Amenity and Cuisine Tags for Eating and Drinking

Restaurants, Fast Food Facilities, Cafes, Pubs, Bars and Biergartens. Filtering our data set for eating and drinking related amenities, the following distribution was observed: 60,819 POIs with *amenity=restaurant*, 18,823 with *amenity=cafe*, 18,701 with *amenity=fast_food*, 14,484 with *amenity=pub*, 3,862 with *amenity=bar*, 2,078 with *amenity=biergarten*, 786 with *amenity=ice_cream*, 746 with *amenity=drinking_water*, 391 with *amenity=bbq*.

Conducting a cross-validation on this data, we observed that pub, bar and biergarten are not sufficiently separable with our basic approach as many bars and biergartens are indeed tagged with *amenity=pub*. Also *pub* and *restaurant* were confused frequently. Therefore, we inserted an additional step: We first learned a classifier for *bar* and *biergarten* and applied it to all POIs with *amenity=pub*. Then we excluded those classified as *biergarten* or *bar* from the training data for *pub* and re-ran the experiment. The precision increases from 68% to 82%. Moreover, we used the classifiers for *pub, bar* and *biergarten* to prune the training data for *restaurant* and the *ice_cream classifier* to prune *cafe* names. Based on the remaining training data, we learned the final classifier. In the cross-validation, the overall accuracy was 76%.

Next, we applied the learned classifier to data with missing tags. We only assigned a tag automatically when the classification probability was 100%. In that way, we created 461 new tags. For 100 of these, we manually checked the

correctness by using the OSM search engine and Google on the name (and possibly further associated tags). In 85% of the instances, the assigned tag was valid. Examples for misclassification are e.g. 'kaffeemaschinenservice kafas' classified as *cafe* (but should be a *shop*), 'uh80 ga weingarten' classified as *biergarten* but really is a *fire_hydrant*, and 'lind haustechnik' classified as *restaurant* (because 'haus' as part of 'gasthaus' occurs quite frequent in German restaurant names) but is a building service.

Cuisine Tags. In total, about 1500 different *cuisine* tags among POIs with *amenity=restaurant* were contained in the data set. Many of those occurred only once or very few times. The most frequent ones are listed in Table 1. They all either indicate ethnicity or type of food. If a *cuisine* tag contained multiple entries (as *cuisine=pizza;kebab*), we counted the POI in both categories.

Table 1. Overview of cuisines.

Ethnicty	Frequency	Type of food	Frequency
Italian	7,365	Pizza	3,275
German	6,753	Kebab	2,926
Regional	6,673	Ice_cream	1,921
Greek	3,002	Burger	1,491
Chinese	1,847	Coffee_shop	723
Asian	1,808	Sandwich	568
Turkish	1,299	Steak_house	310
International	787	Sushi	305
Indian	661	Fish	244
Thai	582	Chicken	125
Bavarian	532	Seafood	116
Mexican	394	Vegetarian	108
Spanish	369		
Japanese	281		
Vietnamese	281		
French	240		
American	163		
Croatian	108		

We performed the following modifications manually to increase the performance and meaningfulness of our approach. We merged *regional* and *german* into one group as they are both too diverse to easily tell them apart. Also *bavarian* was integrated into this group. In contrast, *japanese, chinese, thai* and *vietnamese* were considered each on their own and not accumulated into the *asian* group.

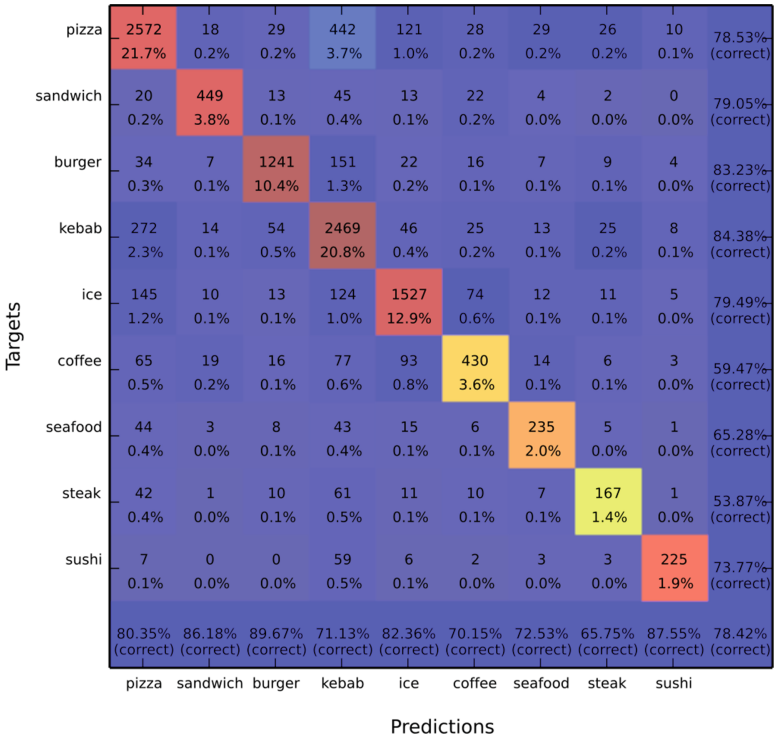

Fig. 3. Precision, recall and accuracy of the learned food type classifier in a cross-validation.

We excluded *international* as we do not expect to identify consistent phrases and words that indicate this cuisine. Regarding type of food, we merged *fish* and *sea_food* into *sea_food* as they were used synonymously and the OSM Wiki recommends to use *sea_food* for both. Furthermore, we excluded *vegetarian*, as it should not be a *cuisine* tag but a *diet* tag instead. The *croatian, american* and *chicken* groups do not contain enough POIs for consideration. So in total, we distinguish 12 ethnicity cuisines and 9 cuisines related to food type.

We first performed a cross-validation, subdivided by ethnicity and food type. For food type the results are presented in Fig. 3.

The overall accuracy is about 78%. We observe, that the accuracy is worse for groups with a small number of representatives as *coffee, seafood, steak* and *sushi*. In addition, there are some natural mix-ups as *pizza* and *kebab*, or *ice* and *coffee* which often occurred together in *cuisine* tags of our input data. For ethnicity, we achieved an overall accuracy of about 81%. For *cuisine=german*, the precision was even above 91% and the recall about 94%. Again, for smaller groups the results were worse. The largest number of mix-ups occurred for *german/italian, mexican/spanish, thai/chinese* and *greek/turkish*.

For our evaluation on unclassified data, we considered 28,218 POIs tagged with *amenity=restaurant* but without a *cuisine tag*. We tried to classify those POIs by food type and ethnicity. We only assigned an ethnicity tag when the probability for a certain class exceeded 75%, and a food type when the classifier was 100% sure. The reason for the different percentages being that we expect most POIs to belong to none of the food types in question. But the classifier creates a probability distribution over the classes with the probabilities summing up to 1. With only nine classes to consider, the chance of a false classification would be too high otherwise. In contrast, for ethnicity, we expect most POIs indeed to belong to one of the classes we consider. For 19,671 out of the 28,128 restaurants, our approach assigned an ethnicity cuisine with a sufficient probability, and for 1,460 a food type was matched. Some POIs received both an ethnicity and a food type cuisine, with the most popular combinations being *pizza;italian*, *kebab;turkish*, *ice_cream;italian* and *sushi;japanese*.

We manually checked 250 extrapolated cuisines for ethnicity and 250 for food type (by having a look at the restaurant's website). We first selected 10 examples for each considered class randomly (if possible). The remaining samples were selected completely randomly among all classified POIs. Table 2 shows an excerpt of 30 samples for ethnicity and food type cuisines assigned by our framework. For food type, two examples for misclassification can be seen: 'rosenburger hof' and 'speisekammer' both serve German food. But as those names contain 'burger' and 'eis' (the German word for ice) respectively, they get assigned *cuisine = burger* and *cuisine=ice_cream* with high confidence. Nevertheless, for the 500 samples in total, the classification accuracy was 98%. As observable in the tables, even spelling errors as 'kebap' could be taken care of with our k-gram based approach, as well as names borrowed from places or persons as 'delphi', 'dschingis khan' and 'cafe mallorca'. The reason for the better precision on real data than in the cross-validation is due to only assigning a class to a POI with unknown cuisine when the probability for that class is high enough. In the cross-validation, every POI gets assigned the class with the highest probability automatically.

4.2 Other Amenity and Shop Tags

In total, the data set contained 938 different *amenity* and 1,853 *shop* tags. The five most frequent *amenity* tags not related to eating and drinking are *bank* (18,765 times), *pharmacy* (16,256), *place_of_worship* (14,309), *parking* (10,853) and *kindergarten* (10,174). The most prominent *shop* tags are in order of frequency *bakery* (22,634 times), *supermarket* (17,655), *clothes* (14,440), *hairdresser* (13,310) and *butcher* (6,862). Overall, we identified 67 reasonable *amenity* and 73 reasonable *shop* classes. The cross-validation revealed a classification accuracy of 84%. Applied to real data, we got 4,212 new tags for previously unclassified POIs. We manually checked for each of the 140 considered classes two extrapolated POIs with that class for correctness. The accuracy was about 76%. Considering only the ten most frequent classes listed above, and 10 examples each, the accuracy was 88%, though.

Table 2. Result excerpts for cuisine classification according to food type and ethnicity. Red entries indicate misclassification.

food type	
pizzahaus	pizza
fischerklause	seafood
la stella	pizza
pizzeria capriccio	pizza
eiscafe rialto	ice_cream
pizzeria italia	pizza
pizzeria venezia	pizza
50's diner	burger
block house	steak_house
fischhaus	seafood
calimero	ice_cream
ristorante pizzeria isola d'ischia	pizza
nordsee	seafood
pizzeria marino	pizza
rosenburger hof	burger
nazar kebap stube	kebab
chilli peppers rock cafe	coffee_shop
eis-cafe da vinci	ice_cream
steakhouse cheyenne	steak_house
eiscafe dolce vita	ice_cream
fischkombüse	seafood
baguetterie filou	sandwich
classic western steakhouse	steak_house
shaki sushi	sushi
cafe kamps	coffee_shop
trattoria la grappa	pizza
sakura sushi & grill	sushi
speisekammer	ice_cream
piccola italia	pizza
döner haus	kebab

ethnicity	
zum neuen schwanen	german
sausalitos	mexican
my thai	thai
mr. kebab	turkish
dschingis khan	chinese
el paso	mexican
cafe mallorca	spanish
mykonos	greek
zum bembelsche	german
delphi	greek
deutscher hof	german
rhodos	greek
il capriccio	italian
sushi for friends	japanese
schusterstübchen	german
winzerhof weinstuben	german
brauhaus am schlössle	german
gasthof pension drexler	german
einkehr	german
pizzeria venezia	italian
kartoffelhaus	german
zur feurigen bratwurst	german
pizzeria italia	italian
taverna ilios	greek
gameiro pizza-express	italian
bauernstübchen	german
pizzeria capriccio	italian
china imbiss drache	chinese
ginnheimer wirtshaus	german
schwaben-bräu	german

Table 3. K-grams and their percentage of occurrence for selected shops.

Bakery	Supermarket	Clothes	Hairdresser	Butcher
38.75 bäcker	12.01 edeka	10.87 mode	25.45 fris	54.84 erei
38.71 rei	11.72 netto	7.94 haus	19.91 friseur	51.62 erei
33.27 bäckerei	11.42 markt	7.13 kik	15.91 haar	42.67 ger
11.36 back	10.67 rewe	5.61 textil	15.10 salon	35.08 metzger
11.20 sch	10.03 aldi	4.41 family	13.84 hair	34.99 metzgerei
5.62 ste	6.69 lidl	2.91 s.oliver	8.96 studio	24.50 fleisch
4.60 mann	6.51 penny	2.85 jeans	8.20 friseur	16.13 fleischere
2.80 konditorei	5.72 kauf	2.31 peek	5.00 haarstudio	16.13 leischerei
2.13 backstube	3.81 norma	2.20 kleid	4.62 cut	4.83 land

Table 3 lists the most frequent k-grams for the main *shop* tags. Reconsidering our example *shop=haidresser*, the main k-grams extracted by our program are close to what one would select manually. Interestingly, for *supermarket*, the k-grams almost exclusively are names of supermarket chains. We observed a similar result for gas station chains. Nevertheless, for almost all classes we identified k-grams that occurred in over ten percent of the respective class names. This fact, and the overall good classification accuracy, shows that indeed many names contain classification information.

4.3 Tourism and Leisure Tags

We identified 168 different *tourism* tags of which 16 occurred more than 200 times. *information* (45,879), *hotel* (12,228) and *attraction* (9,404) had the highest counts. The other popular ones are *viewpoint*, *artwork*, *hostel*, *museum*, *alpine_hut*, *picnic_site*, *camp_site*, *guest_house*, *caravan_site*, *chalet*, *theme_park*, *apartment*, and *zoo*. For *leisure*, 153 different tags were contained in the data. Only 9 of them exhibit a high frequency: *sports_centre* (4,622), *playground* (3,108), *marina* (1,734), as well as *park*, *water_park*, *pitch*, *stadium*, *slipway* and *nature_reserve*. We excluded *artwork*, as due to its nature where is little hope for consistent indicator phrases. Furthermore, we excluded *attraction* as this class is too diverse and the extracted k-grams were too general. The remaining 23 classes were fed in our classifier. The first cross-validation indicated too much mix-up between *information* and *hotel*. Therefore, we first created a *hotel* classifier in order to prune the *information* data. After this step, the overall accuracy improved from 62% to 73%.

For the real data, we newly augmented 3,452 POIs with a *tourism* or *leisure* tag. Computing the precision by manually looking up samples was not so easy in this case, as entities tagged with *information* are often simply signs next to hiking trails. Moreover, most classes were not assigned at all. Therefore, we restricted ourselves in the precision calculation to *hotel*, *playground*, *marina*, and *sports_centre*. We checked 50 examples for each class. The overall accuracy was 92%. For *sports_centre*, we even achieved 98% (e.g. 'tennishalle görner', 'willy-lemkens-sportpark', 'eissporthalle', 'the strike bowlingcenter', 'tanzsportzentrum', 'turnhalle herringhausen' are correct examples).

4.4 Discussion

We also tried other machine learning approaches as logistic regression but got comparable (or slightly worse) results. Analyzing the not correctly extrapolated tags in our cross-validation, we observed three main sources of error: (1) POI names without any amenity, shop or cuisine information, (2) same/very similar POI names but different amenity or cuisine etc., (3) mixed or misleading POI names, as e.g. the example in Fig. 4 shows. In all three cases, also a human reading the POI name is unlikely to come up with a correct extrapolated tag. Therefore, we conclude that only tuning the machine learning part cannot lead to drastic quality improvements.

Fig. 4. Extrapolating from the POI name *Chico's mexican restaurant* that pizza is served there is very unlikely for any reasonable classifier.

5 Conclusions and Future Work

We showed the potential of OSM name tags to serve as basis for extrapolating tags that indicate the class of a POI. Our machine learning approach for automatic tag extrapolation was proven to work well on real data. The accuracy was significantly over 80% for most considered tags. And in particular for *cuisine*, a significant fraction of missing tags was correctly inserted with our approach.

In future work, other tags beside the *name* tags could be considered to improve the results further. For example, the *opening_hour* tag could help to distinguish between restaurants and pubs. The *brand* tag could be helpful when it comes to supermarkets, gas stations, dealerships, clothing stores and so on. Also the free text tags *note* and *description* could be parsed for that purpose. Furthermore, other countries apart from Germany should be investigated. Some tags only occur in certain parts of the world, and the indicator phrases as well as their frequencies for certain tags are expected to change significantly for other countries.

As indicated in the introduction, there are different approaches for integrating the outcome of automatic tag-inference tools into the Open Street Map data pool. In spite of the high precision of our approach, the outcomes should possibly not be automatically fed into OSM without human verification and possible intervention. Mapping and tagging tools like OSMtracker[3] might incorporate the classifiers developed using our approach. By suggesting suitable tags once the user has specified the name of the new POI, the tagging process, which is often experienced as tedious and annoying (manifested in many nonsense tags), could be greatly improved, both in terms of usability for the mapper as well as the resulting data quality. As our classifiers, once learned, can make new reasonable tag suggestion for a POI in milliseconds, it would be feasible to use it a real-time dialog system.

[3] http://wiki.openstreetmap.org/wiki/OSMtracker.

References

1. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
2. Choi, K., Fazekas, G., Sandler, M.B.: Automatic tagging using deep convolutional neural networks. In: Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR, New York City, United States, 7–11 August, pp. 805–811 (2016)
3. Fan, H., Zipf, A., Fu, Q., Neis, P.: Quality assessment for building footprints data on openstreetmap. Int. J. Geogr. Inf. Sci. **28**(4), 700–719 (2014)
4. Fathi, A., Krumm, J.: Detecting road intersections from GPS traces. In: Fabrikant, S.I., Reichenbacher, T., Kreveld, M., Schlieder, C. (eds.) GIScience 2010. LNCS, vol. 6292, pp. 56–69. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15300-6_5
5. Funke, S., Schirrmeister, R., Storandt, S.: Automatic extrapolation of missing road network data in OpenStreetMap. In: Proceedings of the 2nd International Workshop on Mining Urban Data Co-located with 32nd International Conference on Machine Learning (ICML), Lille, France, 11th July, pp. 27–35 (2015)
6. Girres, J.-F., Touya, G.: Quality assessment of the French OpenStreetMap dataset. Trans. GIS **14**(4), 435–459 (2010)
7. Hagenauer, J., Helbich, M.: Mining Urban land-use patterns from volunteered geographic information by means of genetic algorithms and artificial neural networks. Int. J. Geogr. Inf. Sci. **26**(6), 963–982 (2012)
8. Haklay, M.: How good is volunteered geographical information? A comparative study of OpenStreetMap and ordnance survey datasets. Environ. Plan. B Plan. Des. **37**, 682–703 (2010)
9. Jilani, M., Corcoran, P., Bertolotto, M.: Automated quality improvement of road network in OpenStreetMap. In: Agile Workshop (Action and Interaction in Volunteered Geographic Information) (2013)
10. Jilani, M., Corcoran, P., Bertolotto, M.: Multi-granular street network representation towards quality assessment of OpenStreetMap data. In: Proceedings of the Sixth ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS 2013, pp. 19:19–19:24. ACM (2013)
11. Karagiannakis, N., Giannopoulos, G., Skoutas, D., Athanasiou, S.: OSMRec tool for automatic recommendation of categories on spatial entities in OpenStreetMap. In: Proceedings of the 9th ACM Conference on Recommender Systems, pp. 337–338. ACM (2015)
12. Lin, M., Cheung, D.W.: An automatic approach for tagging web services using machine learning techniques. Web Intel. **14**(2), 99–118 (2016)
13. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)