Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings

Volume 15 *Seoul, South Korea* Article 56

2015

Automatic Improvement of Point-of-Interest Tags For OpenStreetMap Data

Sabine Storandt
Insitut für Informatik, Albert-Ludwigs-Universität Freiburg 79110 Freiburg, Germany

Stefan Funke
FMI, Universität Stuttgart 70569 Stuttgart, Germany

Recommended Citation
Storandt, Sabine and Funke, Stefan (2015) "Automatic Improvement of Point-of-Interest Tags For OpenStreetMap Data," *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*: Vol. 15 , Article 56.
DOI: https://doi.org/10.7275/R5D21VS0
Available at: https://scholarworks.umass.edu/foss4g/vol15/iss1/56

AT1-01

AUTOMATIC IMPROVEMENT OF POINT-OF-INTEREST TAGS FOR OPENSTREETMAP DATA

Sabine Storandt[1] and Stefan Funke[2]

[1]Insitut für Informatik, Albert-Ludwigs-Universität Freiburg
79110 Freiburg, Germany
Email: storandt@informatik.uni-freiburg.de
[2]FMI, Universität Stuttgart
70569 Stuttgart, Germany
Email: funke@fmi.uni-stuttgart.de

ABSTRACT

The user experience of any OpenStreetMap (OSM) based service heavily depends on the quality of the underlying data. If the service deals with points-of-interest (POIs), consistent and comprehensive tagging of the respective map elements is a necessary condition for a satisfying service.

In this paper, 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' most certainly indicates an Italian restaurant. As the OSM data contains hundred of thousands POIs for Germany alone, we aim for a tool that can accomplish tag extrapolation in an automated way. In a first step, we automatically extract typical words and phrases that occur in names associated with a certain tag. For example, learning indicators for 'shop=hairdresser' on German OSM tags led to high scores for 'fris', 'cut', hair' and 'haar'. Having available such indicator phrases, we use standard machine learning techniques to derive the probability for a POI to exhibit a certain tag. If this probability exceeds a certain threshold, we assign the tag to the POI in an automated fashion. We used our extrapolation framework to create new amenity, shop, tourism, and leisure tags. The accuracy of our approach was over 85% for all considered tags. Moreover, for POIs tagged with amenity=restaurant, we aimed for extrapolating the respective cuisine tag. For more than 19 thousand out of 28 thousand 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 particular in the context of community-based or crowd-sourced data gathering efforts like OpenStreetMap (OSM), the issue of data quality is of utmost importance. Irrespectively, how sophisticated and advanced the employed algorithms are, the user experience when using services based on the gathered data is first and foremost affected by the quality of the underlying data. Competing commercial offerings will always claim an alleged superior data quality to justify their business. Sometimes, their typically much more centralized approach of data maintenance and collection indeed allows for an easier monitoring of quality issues, for crowed-based approaches like OSM 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 geo-search engine or a location-based service for points-of-interest (POIs) in a certain 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. Moreover, class tags are 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.

Unfortunately, in OSM, there are still plenty of POIs where the class is not provided via the appropriate tags. Many of those POIs exhibit a name tag (as e.g. 'Sunset Hotel', 'Wal Mart', 'Pizzaria Bella Italia'), though, which already contains some information about the respective class. In this paper, we investigate methods for automatic extrapolation of classes based on POI names. Using machine learning tools we extract typical words and phrases that occur in name tags associated with a certain class 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.

1.1 Related Work

Numerous papers use machine learning (ML) techniques to work on/for OSM data. In the following we pick a few examples without claiming completeness. 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., (Hagenaur et al., 2012) 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, (Jiliani et al., 2014) make use of machine learning techniques to assess correctness of highway tags. (Fathi et al., 2010) aim at inferring the structure of the road network by analyzing GPS traces using ML techniques.

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. POINTS-OF-INTEREST 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="2360485476" visible="true" version="5" changeset="21956784"
timestamp="2014-04-26T15:28:19Z" user="q_un_go" 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. 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 *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. *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, *amenity=pub, amenity=bar* and *amenity=biergarten* 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, 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 a *amenity, cuisine, shop* ... 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 machinery is then applied to the derived feature vectors of the names.

3.4 Machine Learning

We use the *Random Forest* (Breiman, 2001) 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 - let's say 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.

Furthermore, we consider the accuracy of the classification, that is, the percentage of correctly classified POIs among all considered POIs.

4. EXPERIMENTS

We implemented the described framework using C++ and Python. For the machine learning part, we relied on the scikit-learn package (Pedregosa, 2011). Our experiments were conducted on a single core of an Intel i5-4300U CPU with 1.90GHz and 12GB 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

4.1.1 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*, 18701 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* and 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-run 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 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.

4.1.2 Cuisine Tags

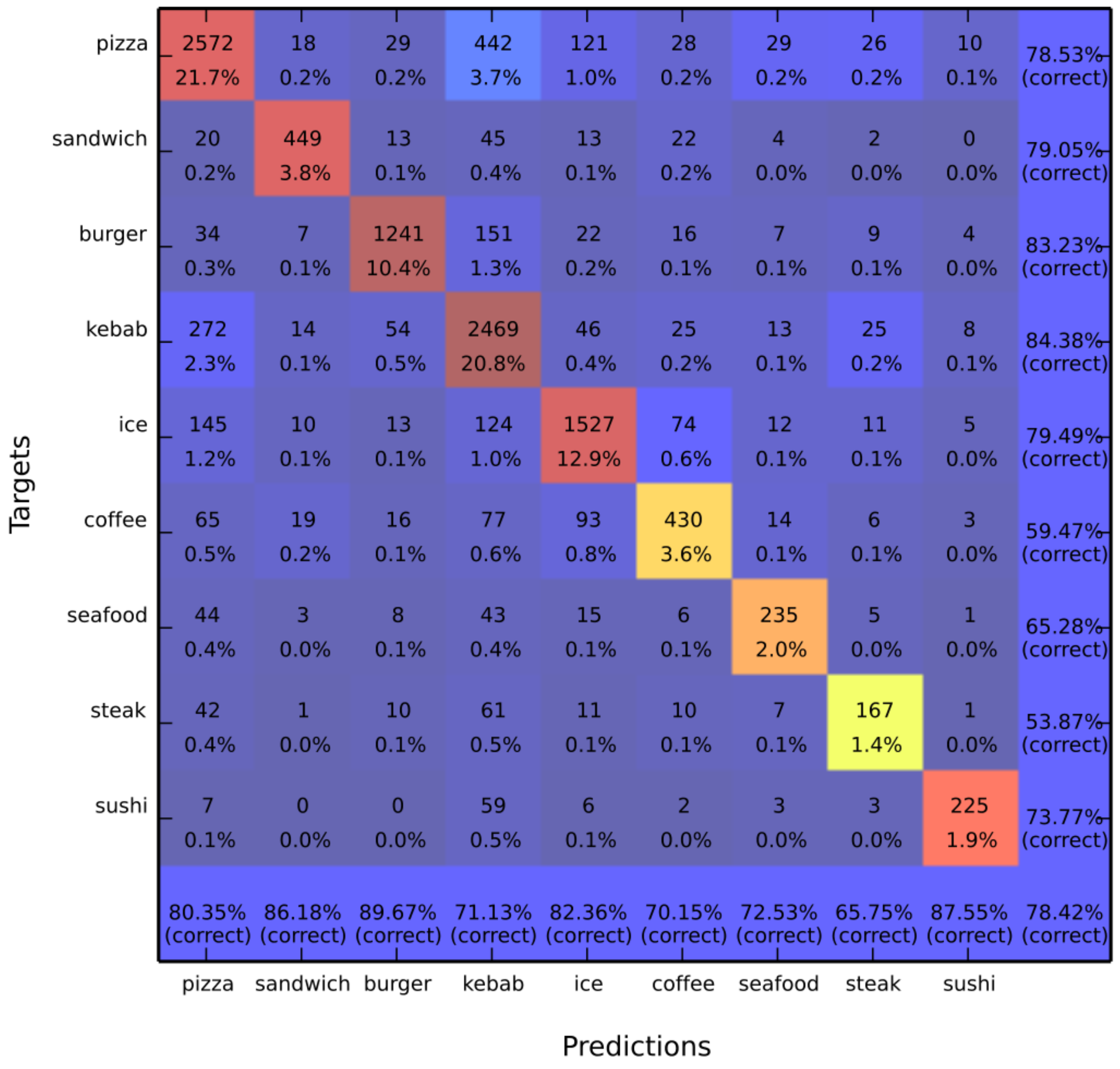
In total, about 1500 different *cuisine* tags among POIs with *amenity=restaurant* were contained in the data set. Many of those occurred only once or very few times. The most frequent ones are listed in Table 1. They all either indicate ethnicity or type of food. If a *cuisine* tag contained multiple 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
asian	1,808	sandwich	568
bavarian	532	burger	1,491
german	6,753	kebab	2,926
chinese	1,847	chicken	125
greek	3,002	ice_cream	1,921
japanese	281	coffee_shop	723
mexican	394	fish	244
regional	6,673	seafood	116
indian	661	vegetarian	108
french	240	steak_house	310
thai	582	sushi	305
international	787		
turkish	1,299		
spanish	369		
croatian	108		
american	163		
vietnamese	281		

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. 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 Figure 1.



Targets \ Predictions	pizza	sandwich	burger	kebab	ice	coffee	seafood	steak	sushi	
pizza	2572 21.7%	18 0.2%	29 0.2%	442 3.7%	121 1.0%	28 0.2%	29 0.2%	26 0.2%	10 0.1%	78.53% (correct)
sandwich	20 0.2%	449 3.8%	13 0.1%	45 0.4%	13 0.1%	22 0.2%	4 0.0%	2 0.0%	0 0.0%	79.05% (correct)
burger	34 0.3%	7 0.1%	1241 10.4%	151 1.3%	22 0.2%	16 0.1%	7 0.1%	9 0.1%	4 0.0%	83.23% (correct)
kebab	272 2.3%	14 0.1%	54 0.5%	2469 20.8%	46 0.4%	25 0.2%	13 0.1%	25 0.2%	8 0.1%	84.38% (correct)
ice	145 1.2%	10 0.1%	13 0.1%	124 1.0%	1527 12.9%	74 0.6%	12 0.1%	11 0.1%	5 0.0%	79.49% (correct)
coffee	65 0.5%	19 0.2%	16 0.1%	77 0.6%	93 0.8%	430 3.6%	14 0.1%	6 0.1%	3 0.0%	59.47% (correct)
seafood	44 0.4%	3 0.0%	8 0.1%	43 0.4%	15 0.1%	6 0.1%	235 2.0%	5 0.0%	1 0.0%	65.28% (correct)
steak	42 0.4%	1 0.0%	10 0.1%	61 0.5%	11 0.1%	10 0.1%	7 0.1%	167 1.4%	1 0.0%	53.87% (correct)
sushi	7 0.1%	0 0.0%	0 0.0%	59 0.5%	6 0.1%	2 0.0%	3 0.0%	3 0.0%	225 1.9%	73.77% (correct)
	80.35% (correct)	86.18% (correct)	89.67% (correct)	71.13% (correct)	82.36% (correct)	70.15% (correct)	72.53% (correct)	65.75% (correct)	87.55% (correct)	78.42% (correct)

Figure 1. Accuracy of the learned food type classifier in a cross-validation.

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. Main number of mix-ups occurred for *german/italian, mexican/spanish, thai/chinese* and *greek/turkish*.

For our evaluation on unclassified data, we extracted 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 Table 2, 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 'café 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.

Table 2. Result excerpts for cuisine classification. Red entries indicate misclassification.

food type		**ethnicity**	
pizzahaus	c = pizza	schwaben-bräu	c = german
fischerklause	c = seafood	ginnheimer wirtshaus	c = german
la stella	c = pizza	china imbiss drache	c = chinese
pizzeria capriccio	c = pizza	pizzeria capriccio	c = italian
eiscafé rialto	c = ice_cream	bauernstübchen	c = german
pizzeria italia	c = pizza	gameiro pizza-express	c = italian
pizzeria venezia	c = pizza	taverna ilios griechisches restaurant	c = greek
50's diner	c = burger	zur feurigen bratwurst	c = german
block house	c = steak_house	pizzeria italia	c = italian
fischhaus	c = seafood	kartoffelhaus	c = german
calimero	c = ice_cream	pizzeria venezia	c = italian
ristorante pizzeria isola d'ischia	c = pizza	einkehr	c = german
nordsee	c = seafood	gasthof pension drexler	c = german
pizzeria marino	c = pizza	brauhaus am schlössle	c = german
rosenburger hof	c = burger	winzerhof weinstuben	c = german
nazar kebap stube	c = kebab	schusterstübchen	c = german

chilli peppers rock cafe	c = coffee_shop	sushi for friends	c = japanese
eis-cafe da vinci	c = ice_cream	il capriccio	c = italian
steakhouse cheyenne	c = steak_house	deutscher hof	c = german
eiscafé dolce vita	c = ice_cream	delphi	c = greek
fischkombüse	c = seafood	zum bembelsche	c = german
baguetterie filou	c = sandwich	mykonos	c = greek
classic western steakhouse	c = steak_house	rhodos	c = greek
shaki sushi	c = sushi	zum neuen schwanen	c = german
cafe kamps	c = coffee_shop	sausalitos	c = mexican
trattoria la grappa	c = pizza	my thai	c = thai
sakura sushi & grill	c = sushi	mr. kebab	c = turkish
speisekammer	c = ice_cream	dschingis khan	c = chinese
piccola italia	c = pizza	el paso	c = mexican
döner haus	c = kebab	café mallorca	c = spanish

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 *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 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 equal 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.

Table 3. K-grams and their for selected shops.

bakery	**supermarket**	**clothes**	**hairdresser**	**butcher**
38.75 bäcker	12.01edeka	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

4.3 Tourism and Leisure Tags

We identified 168 different *tourism* tags of which 16 occurred more than 200 times. *Information* (45,879) and *hotel* (12,228) and *attraction* (9,404) had the highest counts. The others 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 *information* tagged entities 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).

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 especially for *cuisine*, a significant fraction of missing tags was identified with our approach.

In future work, other tags beside 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. Maybe it is possible to learn also the set of indicator tags for each class in an automated way. Furthermore, other countries besides 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.

6. REFERENCES

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., and Duchesnay, E., 2011, Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830

Jilani, M., Corcoran, P., and Bertolotto, M., 2014, Automated Highway Tag Assessment of OpenStreetMap Road Networks. *Proc. 22nd ACM SIGSPATIAL Int. Conference on Advances in Geographic Information Systems*

Hagenaur, J., Helbich, M.. 2012. 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 (June 2012), 963-982.

Fathi, A., Krumm, J., 2010. Detecting Road Intersections from GPS Traces. *Proc. 6th Int. Conf. On Geographic Information Science, LNCS, Vol. 6292*

Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32.