Learning to Classify Spatiotextual Entities in Maps

Giorgos Giannopoulos(✉), Nikos Karagiannakis, Dimitrios Skoutas, and Spiros Athanasiou

IMIS Institute, "Athena" Research Center, Athens, Greece
giann@imis.athena-innovation.gr

Abstract. In this paper, we present an approach for automatically recommending categories for spatiotextual entities, based on already existing annotated entities. Our goal is to facilitate the annotation process in crowdsourcing map initiatives such as OpenStreetMap, so that more accurate annotations are produced for the newly created spatial entities, while at the same time increasing the reuse of already existing tags. We define and construct a set of training features to represent the attributes of the spatiotextual entities and to capture their relation with the categories they are annotated with. These features include spatial, textual and semantic properties of the entities. We evaluate four different approaches, namely SVM, kNN, clustering+SVM and clustering+kNN, on several combinations of the defined training features and we examine which configurations of the algorithms achieve the best results. The presented work is deployed in OSMRec, a plugin for the JOSM tool that is commonly used for editing content in OpenStreetMap.

1 Introduction

The Semantic Web and Linked Data practices have been gaining increasing interest the last years and are being adopted by crowdsourcing and mapping initiatives. In conjunction with the widespread use of smartphones and GPS enabled devices, this has resulted in a large number of RDF datasets containing geospatial information, which is of high importance in several application scenarios, such as navigation, tourism, and location-based social media.

In particular, OpenStreetMap (OSM) is an initiative for crowdsourcing map information from users. It is based on a large and active community contributing both data and tools that facilitate the constant enrichment and enhancement of OSM maps. An important feature of OSM is a large hierarchy of categories[1] for annotating spatial entities on the map. Its Linked Data counterpart, LinkedGeoData, serves the whole OSM dataset as RDF data adhering to a respective ontology[2] which maps OSM categories into equivalent OWL classes.

[1] http://wiki.openstreetmap.org/wiki/Map_features.
[2] http://linkedgeodata.org/ontology.

H. Sack et al. (Eds.): ESWC 2016, LNCS 9678, pp. 539–555, 2016.
DOI: 10.1007/978-3-319-34129-3_33

One of the most prominent tools for editing OSM data is JOSM[3], a graphical tool that allows users to create and edit spatial entities in OSM. Through its graphical user interface, the user can draw the geometry of a spatial entity. Then, she can annotate the entity with categories (alt. classes, tags), which are represented in the form of key-value pairs and assign semantics to the entity. Each entity may belong to multiple categories; for example, a "building" can be further characterized as "school" or "house".

The categories used to semantically annotate spatial entities on the map can either be selected from the already existing hierarchy of categories mentioned above or be defined for the first time by the user. Although this provides a lot of flexibility, which is an important requirement when relying on crowdsourcing, it also increases the complexity of maintaining the taxonomy of classes. Ideally, already existing categories should be reused as much as possible when creating new entities, while new categories should only be introduced when a new entity appears that cannot be appropriately classified and characterized by the existing ones. However, manually browsing through the class hierarchy to determine which category(-ies) is the appropriate one for an entity is a time consuming task, especially for a non-expert user who is not already familiar with the OSM taxonomy. Thus, it may often result in choosing a category poorly (e.g., selecting a more general one, although a more specific one exists) or introducing a new category while in fact an appropriate one (e.g., a synonym) already exists.

To deal with these shortcomings, in this work we propose a process that trains recommendation models on existing, annotated spatiotextual datasets in order to subsequently recommend categories automatically for newly created entities. The main contribution of our work lies on defining and implementing specific training features in order to capture the relations between the spatial and textual properties of each spatial entity with the categories/classes that characterize it. Essentially, this way, the proposed framework takes into account the similarity of the new spatial entities to existing ones that are already annotated with categories. This similarity is specified at several levels: *spatial similarity*, e.g. the number of nodes of the feature's geometry; *textual similarity*, e.g. common important keywords in the names of the features; and *semantic similarity*, i.e. common or related categories that characterize already annotated entities.

To evaluate the proposed methodology, we perform an extensive experimental evaluation that assesses the effectiveness of several feature subsets deployed in the frame of two classification algorithms: (i) Support Vector Machines (SVM) and (ii) k-Nearest Neighbors (kNN). In addition, we assess two hybrid solutions: (iii) clustering+SVM and (iv) clustering+kNN. The experimental results show that a proper combination of classification algorithm and training features can achieve recommendation precision of more than 90 %, rendering the proposed approach suitable for deployment on real-world use cases. Indeed, to further validate it in real-world scenarios, the proposed method is implemented and made

[3] https://josm.openstreetmap.de/.

available as a JOSM plugin[4] [9], allowing the real-time and effective annotation of newly created spatial entities into OSM.

The rest of the paper is organized as follows. In the next section, we briefly review related work. Then, Sect. 3 describes our proposed method, including the defined training features and the assessed algorithms. Section 4 presents the evaluation of training features and algorithms in terms of recommendation performance, while Sect. 5 concludes the paper.

2 Related Work

During the past years, the amount of Volunteered Geographic Information is constantly increasing, while its value and importance in numerous applications and services is constantly becoming more recognized and prominent. Hence, the quality of this content is becoming a crucial factor. Nevertheless, so far few works address the problem of semantically enriching crowdsourced spatiotextual data.

A recent work for addressing the semantic heterogeneity of OSM data proposing a tag recommendation plugin for JOSM is presented in [1]. The main idea of this approach is to recommend additional similar categories for a spatial entity in OSM, when the user has already inserted a category. The recommendation process is held by constructing a semantic network for OSM. This network holds the scores for semantic similarities between pairs of OSM tags and the recommendation process is based upon these scores. The authors also focus on evaluating the effectiveness of the tool and user satisfaction by performing a thorough user study. Thus, they solve a very similar problem to ours, but from another perspective. Given that, the two approaches could complement each other in order to further increase the recommendation effectiveness of the two individual systems.

The work presented in [2] proposes a machine learning based solution for assessing the semantic quality of OSM data. This work focuses on classifying road segments in OSM, thus it specializes only on geometrical and topological features of the specific entities and reduces the space of recommendation categories from more than 1000 to only 21.

In a broader context, relating to recommendations of geospatial data, a Semantic Web Knowledge System is described in [3] that is able to recommend Points of Interest (POIs) to drivers using a content-based recommendation approach. Specifically, they utilize a kNN classifier that takes into account historical POIs the driver has visited to recommend new POIs. In [4], a method is presented for recommending tags for photos, exploiting geospatial proximity, image similarity and several other estimators between photos. The authors feed these estimators into several machine learning algorithms, training this way classifiers that recommend tags for new photos. In [5], the authors utilize Gaussian mixture models to represent the geospatial profile of a user, extracted by microblog posts on music listening events. Considering either user geographic positions or geographic neighbourhoods of the user, they exploit these models into a collaborative filtering approach for identifying similar users and recommending music.

[4] http://wiki.openstreetmap.org/wiki/JOSM/Plugins/OSMRec.

The works in [6,7] perform analyses on OSM annotation statistics and on semantic relations between point geometries in OSM respectively. For the latter, the results of their analysis can be exploited from systems that perform category recommendations or quality assessment of the annotation of the spatial entities. Finally, [8] presents a thorough overview of recommendation concepts and methods on location based social networks.

3 Recommendation Models

3.1 Problem Definition and Method Outline

We cast the problem as a multilabel classification task. Given a set $\mathcal{E}$ of spatial entities $e_i \in \mathcal{E}$ and a set $\mathcal{T}$ of categories $t_n \in \mathcal{T}$: (i) properly represent each entity as a feature vector $v_{e_i} = <g_i, w_i, c_i>$, where g_i, w_i and c_i are sets of geospatial, textual and semantic attributes of the entity; (ii) learn a function $\mathcal{F}(v_{e_i}, t_n) => \{true, false\}$ that maps entities to categories.

We further break down this task into three distinct sub-tasks: (a) analyse spatial entities into meaningful attributes (training features) that properly describe the entities and capture their latent relation to the categories that annotate them; (b) train a machine learning algorithm to "learn" relations between attributes of the spatiotextual entities and the respective categories and (c) input the new (test) entities into the trained algorithm to produce category recommendations for them.

In Sect. 3.2 we describe the defined features that correspond to geometric, textual and semantic properties of the entities. The next step is to feed training entities, expressed through their features, into a classification algorithm that utilizes them to classify new entities to categories. We applied both model based (Support Vector Machines) and memory based (k Nearest Neighbour) state-of-the-art classification methods[5]. Further, we tested two hybrid solutions that first create clusters of similar training spatial entities and then apply either SVM or kNN respectively. Each of the algorithms is described in Sect. 3.3. During the third step, depending on the algorithm, either the trained model is applied on the new entity, or the new entity is matched to the training ones (using the cosine similarity of the respective feature vectors of the entities), in order to produce category recommendations. In the case where the similarity between a cluster and an (external) entity needs to be computed, this is done by considering the average feature vector of all entities contained in the cluster and, similarly, applying the cosine similarity with the vector of the (external) entity. The general rule used for producing the average vector of each cluster is by applying an OR operation on the boolean training features (since we wanted the cluster to be characterized with the specific training feature value if at least one of the clusters

[5] More sophisticated or recent algorithms could have been tested, however, the focus of this work lies mainly on defining meaningful and effective training features. Thus, these two algorithms were selected as intuitive representatives of two categories of classification algorithms that have shown to be effective in several scenarios.

entities was characterized by it) and computing the average value of the training features represented by double values.

3.2 Feature Selection

In our scenario, the entities to be classified are the spatiotextual entities that exist on the map (e.g. buildings, roads, parks) and the variable to be predicted is the category(ies) of the entities. Note that an entity may belong to multiple categories. To learn a classifier for assigning entities to categories, we represent each spatiotextual entity by a feature vector. We consider features that capture spatial, textual and semantic properties of the entities, as listed next.

- ***Spatial properties***
  - ***Type***. Six distinct geometry types are considered: Point, LineString, Polygon, LinearRing, Circle and Rectangle. Each type is treated as a boolean feature, so six positions in the feature vector are used.
  - **Geometry**
    - *Points.* This denotes the number of points that the geometry of the entity comprises when represented as a point set. It provides an indication of how complex the geometry is.
    - *Area.* This represents the size of the entity's geometry.
    - *Mean.* Mean edge length is a feature used to capture properties of the shape of the entity.
    - *Variance.* Variance of edge lengths is also related to the entity's shape.
- ***Textual properties***
  - *Text.* For each entity of the training set, we extract the textual description of its name. We consider each word separately and count their frequency within the dataset. Then, we sort the list of words by their frequency, filtering out words with frequency below a certain threshold (set to 20 in our experiments). Finally, we apply a stopword list[6], removing this way words without any particular meaning. What remains are special meaning identifiers, such as "Road", "Avenue", "Park", "Court", etc. Each of these keywords is used as a separate boolean feature, realizing, thus, a bag-of-words model for representing the text of the entities.
- ***Semantic properties***
  - *Categories.* This is a set of 1,421 boolean features, corresponding to each of the OSM categories. This feature is used in the scenario where an entity has been previously annotated with one or more categories, and we wish to recommend additional relevant categories for this entity[7].

[6] https://code.google.com/archive/p/stop-words/.

[7] These features are only mentioned here for completeness, since they are utilized in the implemented OSMRec prototype - JOSM plugin. However, they are not included in the experimental evaluation that is described next.

We should note here that we select different representations for the numeric features (double precision number or set of boolean features) to comply with the functionality of the different models and similarity functions we apply. Namely, we consider boolean features in the case of SVM, since it is well known that such models perform better when the input feature vectors are vertically normalized within (or at least close) to the interval [0,1]. Thus, for example, for the feature "Area of Geometry", defining several area ranges corresponding to separate feature positions, we allow the model to relate these different areas (feature positions) with different training classes. On the other hand, in the case of clustering or applying kNN, where similarity measures such as the cosine similarity are applied, using the exact value of a feature (e.g. area of geometry) is preferable in order to better quantify the difference between two entities.

3.3 Algorithms

SVM. The first algorithm applies multilabel SVM classification, using the $SVM^{multiclass}$ implementation[8], considering as training items the spatiotextual entities themselves and as labels the categories that characterize them. The method maps the training entities into a multidimensional feature space and aims at finding the optimal hyperplanes that discriminate the entities belonging to different categories. The optimality of the hyperplane depends on the selected parameter C, which adjusts the trade-off between misclassified training entities and optimal discrimination of correctly classified entities. The output of the training process is a model (essentially a weight vector) that is able to map the feature vector of a new, unannotated entity to a set of categories, providing, also, a matching score for each category. This way, one can obtain the top-n most fitting categories for a new spatial entity.

kNN. The second algorithm searches for k-Nearest Neighbours in the set of training entities. The algorithm compares the new entity with each one of the training entities and recommends the categories that characterize the most similar training entities. As similarity measures we consider *cosine similarity* and *Euclidean distance*. The two similarity functions were initially applied to the feature vectors of the respective entities. However, we empirically observed that, in the specific setting, boosting the vector-calculated similarity score with the area-based and point-based similarity scores between two entities improves the precision of the algorithms. Specifically, we use the following formula in our experiments to calculate the similarity score S between two entities u, v:

$$\begin{aligned} S_{cos} &= cosSim + 2 * (1 - areaDist) + 2 * (1 - pointDist) \\ \text{where } areaDist &= \frac{|area_u - area_v|}{\max area_u, area_v} \\ \text{and } pointDist &= \frac{|points_u - points_v|}{\max points_u, points_v} \end{aligned} \tag{1}$$

[8] https://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html.

where *cosSim* is the cosine similarity on the whole feature vector of the two entities and is interchanged in our experiments with the similarity that is based on the euclidean distance

$$euSim = 1 - euDist/\max euDist \tag{2}$$

The similarity of a trained entity with the new entity is propagated to the categories that annotate the training entity. So, for each candidate category, we get an aggregate score from all matching training entities that are annotated by it. This way, we can get the top-n most fitting categories for a new spatial entity.

clustering+SVM. The third algorithm, SVM on clustered entities, first clusters the training entities, to identify groups of similar entities, based on the training features that represent them. Then, the entities assigned to each cluster go through the SVM training process in separate groups. This procedure produces a number of SVM models equal to the number of clusters. The rationale of this approach is that the clustering step is expected to produce subsets of the training entities containing more homogeneous entities, w.r.t. to the defined training features. Thus, the SVM training step is performed on more coherent groups of entities, with the expectation to produce more specialized SVM models. Then, a test entity is matched only with some of these models (corresponding to the most similar training entities) and the eventually used category recommendation models are trained only by entities similar to the test entity.

We applied an Expectation Maximization clustering algorithm (using the implementation provided by Weka[9]). The algorithm starts by assigning initial probabilities to items belonging to clusters and then it iteratively re-defines clusters and re-assigns items to clusters, until it converges. The specific implementation provides the option either to define a fixed number of clusters to be formed, or to allow the system to result to the most fitting number of clusters. Thus, we performed our experiments using both options.

In order to match a new entity to appropriate cluster(s), we calculate the cosine similarity of this entity's feature vector against *all* the entities contained in the training set. Based on the K[10] most similar training entities e_j, $j \in [1, K]$ that emerged from this process and, using their cosine similarity scores $cos(e_j, e_{new})$ with the new entity's feature vector e_{new}, we assign similarity weights between each cluster and the new entity. Specifically, the value of a cluster weight w_{c_i} is calculated by the frequency of the cluster's appearance in the set of K most similar training entities and additionally, the similarity of each training entity (belonging to the cluster) with the new entity:

$$w_{c_i} = \sum_{j=1}^{K} cos(e_j, e_{new}) * \mathbb{1}_{(e_j \epsilon c_i)} \tag{3}$$

[9] http://www.cs.waikato.ac.nz/ml/weka/.

[10] Experimentally tuned to $K = 50$ by testing the recommendation results while changing its value in the interval $[10, 100]$ with step 10.

with $\mathbb{1}$ being the indicator function.

Then, we apply each of the M matched SVM models (corresponding to the matched clusters above) to produce a ranking of categories for the new entity. Let $r_{c_i,j}$ denote the ranking of the j-th category by the ranking function of cluster c_i; the final ranking score between a category j and the new entity is given by linearly weighting the cluster rankings with the cluster importance w_{c_i}:

$$S(e_{new}, j) = \sum_{i=1}^{M} w_{c_i} * r_{c_i,j}. \tag{4}$$

Finally, we consider the top-n categories ranked according to the process above.

clustering+kNN. The fourth algorithm, kNN on clustered entities, first clusters the training entities to identify groups of similar entities (using the same aforementioned Expectation Maximization clustering algorithm), based on the training features that represent them. Then, it performs a kNN algorithm that, for each new spatiotextual entity, finds the most similar cluster. To do so, each cluster is represented by its average feature vector, as described in Sect. 3.1. The same rationale described in the clustering+SVM case applies here. Through clustering, we expect to specialize the groups of training entities on which the kNN algorithm is applied, only to those training entity groups that are found more similar to the test entity.

Upon clustering, the process is identical to the one followed in kNN algorithm with the difference that the training items are now clusters of entities and their labels are the total of the categories that characterize the entities of the cluster. The similarity function compares a new spatial entity with the average vector of each cluster, and assigns it a ranked list of categories, based on the similarity score of the entity with each cluster. Defining the set of all categories N that belong to a ranked list of matching clusters c_i, $i\epsilon[1, \mathcal{C}]$, that is all the categories that characterize entities belonging to the respective list of clusters and $w_{c_i,n}$ the matching score of the cluster c_i's centroid (where category n belongs to cluster c, with normalized frequency $f_{c_i,n}$) to the new entity, the aggregate matching score of each category with the new entity is calculated as follows:

$$S_n = \sum_{i=1}^{K} w_{c_i,n} * f_{c_i,n} \tag{5}$$

In our experiment, we tried out $K = 1$ and $K = 5$, i.e. we considered the 1 and the 5 Nearest Neighbours-clusters of the new entity for detecting matching categories[11].

[11] In the case of clustering+kNN, our experiments consistently showed that the best results were achieved with a low number of clusters, specifically, 14 and 28 for Athens and London respectively. Thus, given this upper limit for k, we indicatively evaluated the 1-NN and 5-NN cases.

4 Experimental Evaluation

Next, we present the evaluation of the proposed methods w.r.t. the recommendation precision they achieve. First, we describe the dataset used and the evaluation methodology. Then, we compare the four algorithms and discuss the results.

4.1 Dataset and Evaluation Methodology

We performed our evaluation on two distinct subsets of OSM data covering parts of: (i) Athens, Greece and (ii) London, UK, which we exported through the Overpass API[12] from the OSM website. Each of the two datasets contains a total of about 20,000 spatiotextual entities which were properly divided into training, validation and test sets, as will be described next. Table 1 presents some statistics on both datasets.

Each dataset is partitioned into five subsets of similar sizes. Then, combining each time different subsets to make (a) the training (3 subsets), (b) the validation (1 subset) and (c) the test set (1 subset), we create five different arrangements for five-fold cross-validation. In every fold, the validation set was used to tune the parameters of the classification model and the test set was the one where the actual evaluation of the method was performed. In this evaluation, we considered the setting where test entities were clear of annotations, that is, they were to be annotated for the first time. Thus, we do not consider the Semantic properties mentioned in Sect. 3.2.

We experimented varying the following parameters. Parameter C of the SVM algorithm was set with the following values: {0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0}. For Cl, the number of created clusters for the two hybrid approaches, we set the values: {10, 50, 100, 200} and we also considered the automatic cluster creation option of the algorithm that decided the optimal number of clusters to be created. Finally, for the number k of Nearest Neighbours, we set 10 for the kNN algorithm and experimented with 1 and 5 for the clustering-kNN approach.

Table 1. Dataset statistics.

Statistics	Athens	London
Distinct classes	186	306
Classes per entity	1.0	1.1
Majority class #1 (Building/Building) frequency	7133	8036
Majority class #2 (ResidentialHighway/Footway) frequency	3850	1942
Majority class #3 (Footway/UnclassifiedHighway) frequency	1122	1083
Categories with frequency = 1	20 %	25 %
Categories with frequency ≤ 2	33 %	38 %

[12] http://overpass-api.de/.

As evaluation measure, we consider the precision of category recommendations, i.e. the ratio of correct recommendations to the total recommendations:

$$P = \frac{\#correct\_category\_recommendations}{\#total\_category\_recommendations} \tag{6}$$

We consider three variations of the measure, depending on how strictly we define the recommendation correctness:

P^1: In this case, a recommendation is considered correct if the recommended category with the highest rank from the recommendation model is indeed a category that characterizes the test entity.

P^5: In this case, a recommendation is considered correct if one of the five recommended categories with the highest rank from the recommendation model is indeed a category that characterizes the test entity.

P^{10}: Similarly, a recommendation is considered correct if one of the ten recommended categories with the highest rank from the recommendation model is indeed a category that characterizes the test entity.

The rationale behind the definition of these three measures is that, in practice, it is common to provide the top-N recommendations. So, using these measures, we wanted to evaluate some representative cases, setting $N = 1, 5, 10$. Note that the above measures are slight variations of the commonly used measure of *Precision@N* which measures the number of correct results in the top-N rank positions. Our measures are defined in a boolean manner, indicating whether *at least one* correct category is found within the recommended ones to annotate the entity. Given that, whether the recommendation set consists of 1, 5 or 10 categories, the recommendation is considered successful if at least one category from it indeed characterizes the respective entity. Another reason for this adaptation is the very low average number of categories that annotate each entity in our datasets (and in OSM in general), that lies a little above 1 category per entity. This means that measuring *Precision@N* with $N > 1$ would be meaningless.

Finally, we note that when we used a validation step, the best performing configuration was chosen according to the highest P^1 **validation** value (since this is the strictest of the applied measures), while when no validation was applicable, then we considered the P^1 **test** value. Thus, the overall precision Table 8 presents the best sets of P^1, P^5 and P^{10} values considering the best P^1 as described above.

4.2 Algorithm Comparison and Discussion

We first present individual analysis results for each algorithm separately.

SVM. Table 2 presents the test values on precision of the SVM algorithm, for several training feature configurations. A general observation is that Spatial properties (either geometry types or geometry values) highly contribute to the precision of the model. On the other hand, Text features result to very low quality recommendations, at least when used on their own. The best precision in terms of P^1 comes by combining Spatial properties and Text. Type features used on their own provide the best values for P^{10} in both datasets.

Table 2. SVM feature combinations for Athens and London.

Features	Athens test set				London test set			
	C	P^1	P^5	P^{10}	C	P^1	P^5	P^{10}
Type	0.1	59.25	**82.85**	**91.06**	0.001	52.15	73.54	**83.25**
Points and area	100	47.76	72.65	79.10	100	47.30	59.29	62.59
Mean and variance	0.01	53.73	77.66	87.39	0.1	44.71	69.32	78.19
Points, area & type	0.01	59.15	76.55	89.95	100	47.26	58.55	67.44
Text	1	6.78	9.85	9.94	1	14.05	23.27	24.93
Points, area, type & text	10	59.49	70.55	75.67	1000	**59.59**	73.20	80.02
Spatial properties & text	1000	**59.99**	81.61	89.56	1000	55.44	**75.92**	81.43
Spatial properties	1000	59.50	79.69	88.42	1000	50.34	66.09	75.90

Table 3. kNN feature combinations for Athens and London.

Features	Athens test set			London test set		
	P^1	P^5	P^{10}	P^1	P^5	P^{10}
Type	42.57	53.62	54.96	44.67	54.50	56.40
Points and area	40.06	51.00	52.25	43.2	52.74	54.55
Mean and variance	57.42	69.32	71.56	51.88	63.06	65.40
Points, area & type	42.66	53.66	54.93	44.60	54.42	56.35
Points, area, type & text	46.91	58.06	59.46	54.13	64.04	66.29
Spatial properties & text	**59.98**	**71.57**	**73.45**	**58.81**	**70.15**	**72.39**
Spatial properties	56.58	68.13	69.92	52.01	62.77	64.94
Points, area, type - double values	42.57	53.62	54.96	44.67	54.51	56.40

kNN. Table 3 presents the recommendation precision values for the kNN algorithm. Spatial properties and Text combined give the highest precision in both datasets. However, these values are lower than the respective ones produced by SVM. Further, the difference in precision increases while moving from P^1 (negligible) to P^{10} (high). The overall better performance of SVM compared to kNN can be probably attributed to the generalization properties of the SVM model: kNN compares straightforwardly training and test entities and utilizes only categories from matching training entities. On the other hand, SVM, produces a model that assigns a score to *every available category* for the respective test entity, based on the importance of the entity's features for the trained model. Essentially, by finding optimal hyperplanes that discriminate different classes-categories in the training feature space, SVM can effectively handle both the potential sparseness of the training data and outlier data (i.e. incorrect entity annotation cases) that would otherwise negatively affect the model's training.

Table 4. Clustering+SVM feature combinations for Athens and London for the automatic cluster selection mode.

Features	Athens test set					London test set				
	C	Clust.	P^1	P^5	P^{10}	C	Clust.	P^1	P^5	P^{10}
Type	0.01	14	**31.21**	**68.25**	**75.71**	0.01	21	33.57	**57.10**	**66.16**
Points and area	0.01	14	28.02	46.66	59.54	0.001	21	26.37	36.60	50.93
Mean and variance	0.1	14	28.98	47.23	59.41	0.001	28	26.74	37.57	52.75
Points, area & type	0.001	14	27.86	46.69	60.01	0.01	28	26.61	37.50	51.97
Points, area, type & text	100	14	28.41	48.80	57.94	1000	28	35.18	45.87	51.12
Spatial properties & text	1000	14	29.37	45.28	57.94	1000	28	**38.02**	45.14	54.62
Spatial properties	100	14	28.65	41.76	55.65	1000	28	30.02	37.35	48.33

Table 5. Clustering+SVM feature combinations for Athens and London with manual selection of clusters.

Features	Athens test set					London test set				
	C	Clust.	P^1	P^5	P^{10}	C	Clust.	P^1	P^5	P^{10}
Points, area & type	1000	10	44.07	56.99	**68.91**	0.001	10	25.26	44.06	55.20
Spatial properties & text	1000	10	40.98	53.97	59.53	0.001	10	23.38	37.36	49.16
Spatial properties	1000	10	44.46	57.22	64.78	0.01	10	22.76	36.29	48.21
Points, area, type & text	1000	10	43.45	**63.48**	64.75	1000	10	25.30	40.37	48.21
Type	0.1	10	44.70	60.33	65.20	0.1	10	29.60	44.37	57.20
Points, area & type	0.01	50	43.37	45.18	59.03	0.01	50	**43.37**	50.18	**60.71**
Points, area, type & text	0.01	50	43.37	45.18	59.03	0.001	50	23.56	33.60	48.08
Spatial properties	1000	50	42.39	50.33	61.31	0.01	50	23.28	32.80	46.81
Spatial properties & text	1000	50	43.06	51.43	62.02	0.01	50	23.98	32.82	47.34
Type	0.01	50	38.02	39.37	41.67	0.01	50	25.01	34.40	38.56
Points, area & type	0.01	100	40.15	45.10	59.70	0.001	100	19.76	35.44	56.01
Points, area, type & text	0.001	100	**45.26**	58.34	68.32	1000	100	25.51	42.79	48.68
Spatial properties	0.01	100	40.18	51.36	61.44	0.01	100	26.14	37.77	57.51
Spatial properties & text	0.001	100	40.18	51.35	61.46	1000	100	32.71	**53.68**	59.58
Type	-	100	-	-	-	-	100	-	-	-
Points, area & type	0.001	200	30.39	39.04	50.60	0.001	200	-	-	-
Points, area, type & text	0.001	200	**45.26**	58.34	68.32	0.001	200	-	-	-
Spatial properties	0.1	200	26.91	39.97	48.01	0.001	200	23.68	32.08	43.12
Spatial properties & text	100	200	38.75	44.76	53.40	0.01	200	23.97	23.60	43.80
Type	-	200	-	-	-	-	100	-	-	-

clustering+SVM. Tables 4 and 5 present recommendation precision values for the hybrid clustering+SVM approach we propose. The first table presents results using the automatic number of cluster selection of the EM algorithm for Athens and London, while the second one the respective results when we manually provide several values for the number of clusters parameter. The best values are produced by manual selection of the number of clusters and, in most cases, for low values of this parameter (10 or 50 clusters). However, the results are rather poor, compared to the SVM and kNN algorithms. A possible explanation is that the clustering process is not performed with the proper objective in order to facilitate and enhance the upcoming SVM training process. Namely, the clusters are created using an improper clustering criterion, w.r.t. the specific

task. Another issue might lie on the selection of the matching process, where we match test entities with training entities straightforwardly and not with a representation of the clusters they belong to, as we do in the clustering+kNN case. Nevertheless, since this general approach of creating ensembles of classifiers and properly combining them to obtain better results has worked in several other classification settings, we intend to further investigate it.

clustering+kNN. Tables 6 and 7 present recommendation precision values for the hybrid clustering+kNN approach we propose for applying 1-NN and 5-NN on clusters respectively. In this case, we report only results produced by the automatic number of clustering selection option of EM algorithm. The manual setting of the parameter produced consistently worse results and these results are, thus, omitted, due to lack of space. This hybrid solution produces very good recommendation precision, that even surpasses the simple SVM model for P^5 and P^{10}. Here, in contrast with the clustering+SVM case, the clustering step facilitates the upcoming kNN classification process. This can be intuitively explained as follows: by considering clusters of similar entities, instead of individual entities, we basically enhance the "Nearest Neighbours" of test entities; the test entities are still matched with similar training entities, but now, through the matched clusters, a more rich/diverse set of categories can be mapped to the test entities. So, when the precision regards the top-5 or top-10 recommended entities, finding correctly recommended categories becomes more probable.

Another observation is that the 1-NN configuration provides much better results than the 5-NN. This can be explained by the fact that each cluster already contains several categories to be recommended. So, if we consider many clusters, the recommendation set of categories gets too diverse, hurting, thus, the recommendation precision. Also, we can observe that relatively small numbers of clusters achieve the best precision values (14 for Athens - 28 for London). Finally, again, the geometry related features are the ones providing the highest precision; specifically, the Type for Athens and the Spatial properties for London.

All Methods. Table 8 presents the best achieving configuration for each tested algorithm, w.r.t. P^1. SVM achieves the highest accuracy for P^1 and clustering+kNN

Table 6. Clustering+kNN (1-NN) feature combinations for Athens and London.

Features	Athens test set				London test set			
	Clusters	P^1	P^5	P^{10}	Clusters	P^1	P^5	P^{10}
Spatial properties & text	14	51.22	71.14	79.14	21	47.54	61.43	69.71
Points, area, type & text	14	58.69	80.51	89.13	21	44.35	55.12	62.03
Mean and variance	14	44.43	55.52	60.00	28	45.92	55.81	61.83
Points, area & type	14	31.61	53.41	61.37	28	50.38	71.23	81.46
Points, area & type - double val.	14	39.62	51.04	56.51	28	45.24	56.64	65.65
Type	14	**59.15**	**82.53**	**91.08**	28	**51.67**	71.44	82.85
Spatial properties	14	55.95	79.36	87.89	28	51.24	**72.93**	**84.03**
Points and area	14	55.94	79.18	87.68	28	31.37	47.90	56.04

Table 7. Clustering+kNN (5-NN) feature combinations for Athens and London.

Features	Athens test set				London test set			
	Clusters	P^1	P^5	P^{10}	Clusters	P^1	P^5	P^{10}
Spatial properties & text	14	39.13	58.74	62.56	21	41.62	52.37	58.79
Points, area, type & text	14	38.69	58.16	61.64	21	41.62	48.77	56.05
Mean and variance	14	38.52	49.37	54.39	28	41.62	49.06	54.55
Points, area & type	14	50.53	**65.82**	72.05	28	**43.96**	56.27	63.90
Points, area & type - double val.	14	38.35	44.02	46.90	28	41.62	46.70	49.92
Type	14	**50.73**	63.86	**76.12**	28	41.62	52.89	61.29
Spatial properties	14	50.24	61.54	70.20	28	40.72	**60.17**	**67.00**
Points and area	14	38.46	60.21	65.15	28	41.62	45.12	50.89

for P^{10} in both datasets, while P^5 is a "tie" for the two methods considering both datasets. kNN comes third, with its performance degrading from P^1 to P^{10}. clustering+SVM provides the worst results of all. These results indicate that SVM is preferable when we want to provide very few recommendations, but as accurate as possible. On the other hand, when the setting allows us to provide more recommendations, with the hope that a few of them will be correct, then clustering+kNN works better.

Further, as a naive baseline, we consider the case where the majority class, i.e. the category with the highest frequency in each dataset is constantly (i.e. for each test entity) recommended, w.r.t. the P^1 measure. Similarly, we consider the cases where the 5 and 10 most frequent classes are constantly recommended for P^5 and P^{10}. The precision values are provided in the last line of Table 8 and demonstrate that the precision of the best solutions are consistently much higher than the naive baseline, with a difference ranging from 10 % to 24 %.

Table 8. Overall recommendation precision for Athens and London.

Algorithms	Athens test set			London test set		
	P^1	P^5	P^{10}	P^1	P^5	P^{10}
SVM	**59.99**	81.61	89.56	**59.59**	**73.20**	80.02
kNN	59.98	71.57	73.45	58.81	70.15	72.39
Clustering+SVM	43.45	63.48	64.75	43.37	50.18	60.71
Clustering+kNN	59.15	**82.53**	**91.08**	51.67	71.44	**82.85**
Majority class recommendation	35.65	68.33	75.76	40.17	63.07	72.97

Further, in general, it seems that applications of Geometry features or Spatial properties features give the best results, enhanced in occasions by being combined with Text features. However, in some cases (clustering+kNN), Type features seem to work very well by their own. Text features, seem to have the less effect on precision, probably due to their sparseness.

Another observation is the difference in recommendation precision between Athens and London, with Athens having slightly better P^1 values and considerably better P^5 and P^{10} values. This can be explained by examining some simple statistics from the two datasets. First, Athens is based on a more compact annotation set of 186 distinct classes, while the same number of spatial entities (20000 entities) in London are annotated by a set of 306 classes, resulting to higher heterogeneity in annotations. Second, the London dataset contains relatively more classes that do essentially hurt the evaluation results, due to their very low frequency. For example, categories with only one appearance in the dataset comprise 20 % of Athens but 25 % of London dataset, while the respective numbers for categories with at most two appearances are 33 % and 38 % respectively. Conclusively, the compactness of the annotation set and the lack of outlier categories (which is the case of the specific Athens dataset, relatively compared to the London one) seem to favour the recommendation precision.

With respect to the importance of the achieved values for the three evaluation measure variations, reaching a precision of around 60 % in a multilabel classification task, and for recommending *just one* category, is a rather important achievement. However, the other two measures are also important, considering a real world deployment of the recommendation algorithms: recommending 5 or 10 categories

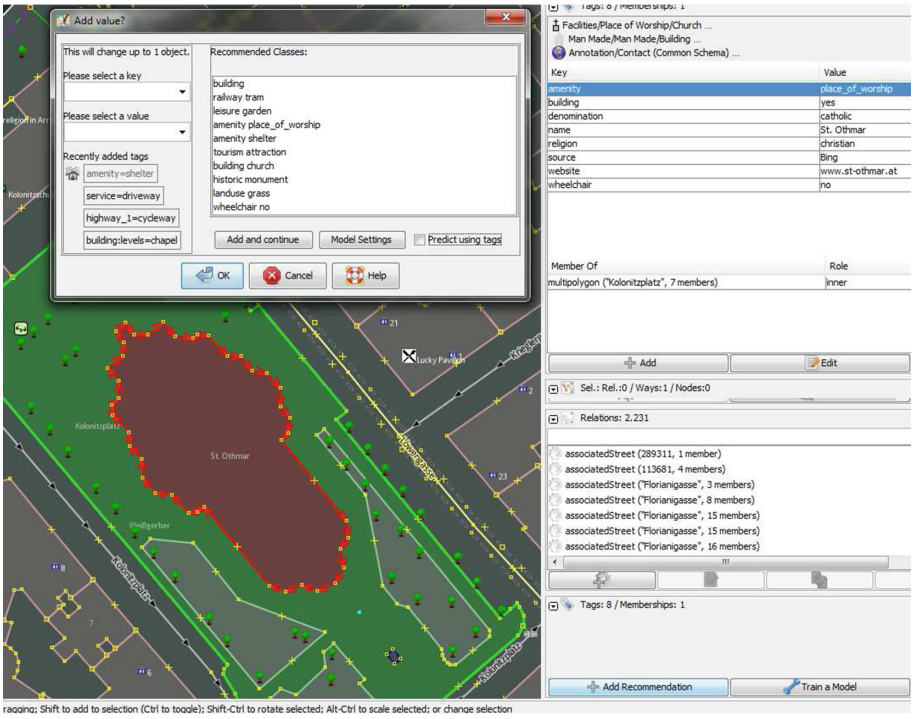

Fig. 1. Recommendations example

to the user to choose from is a realistic option, and the proposed algorithms can achieve precision up to 91 % for the 10 categories recommendation setting. This means that most times, the system will be able to recommend at least one useful category to the user.

To intuitively demonstrate the above conclusion, we present an exemplary recommendation of the JOSM plugin we have implemented using our approach, OSMRec [9]. In Fig. 1, we can see an example of recommendations for a church in Vienna. The already existing, manual annotations set (right top of the screen), contains the specific annotation pairs "*amenity => place_of_worship*", "*denomination => catholic*" and "*religion => christian*". Out of these three annotations, the most *geospatially* meaningful is the first one, "*amenity => place_of_worship*", which is identified by our recommendation model (left top panel). Further, our model recommends an even more accurate annotation, "*building => church*", that does not exist in the initial, manual annotation set. Apart from that, our model recommends a more diverse annotation, "*historic => monument*", which is very probable to also characterize the specific entity.

5 Conclusions

In this paper, we presented a framework for producing category recommendations on spatial entities, based on previously annotated entities. We defined a set of problem specific training features that where applied with four classifiers, two state of the art and two hybrid solutions we proposed, and we reported on their recommendation precision for two real OpenStreetMap datasets from Athens and London. Specifically, we showed that the recommendation precision is high enough to be used in real world applications, such as JOSM OSM editing tool, where our method is already implemented as a plugin (OSMRec). Our future work includes further generalizing and testing the implemented framework with more geospatial semantic datasets and evaluating the effectiveness of Category features by using existing categories of an entity to predict new, correct annotation categories for the entity.

Acknowledgments. This work was partially supported by EU projects GeoKnow (GA no. 318159) and City.Risks (H2020-FCT-2014-653747).

References

1. Arnaud, V., Rodolphe, D.: Improving volunteered geographic information quality using a tag recommender system: the case of OpenStreetMap. In: Arsanjani, J., Zipf, A., Mooney, P., Helbich, M. (eds.) OpenStreetMap in GIScience, pp. 59–80. Springer, Cham (2015)
2. Jilani, M., Corcoran, P., Bertolotto, M.: Automated highway tag assessment of OpenStreetMap road networks. In: Proceedings of the SIGSPATIAL 2014 (2014)

3. Parundekar, R., Oguchi, K.: Learning driver preferences of POIs using a semantic web knowledge system. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 703–717. Springer, Heidelberg (2012)
4. Silva, A., Martins, B.: Tag recommendation for georeferenced photos. In: LBSN (2011)
5. Schedl, M., Vall, A., Farrahi, K.: User geospatial context for music recommendation in microblogs. In: SIGIR (2014)
6. Mooney, P., Corcoran, P.: Annotating spatial features in OpenStreetMap. In: GIS-RUK (2011)
7. Mülligann, C., Janowicz, K., Ye, M., Lee, W.-C.: Analyzing the spatial-semantic interaction of points of interest in volunteered geographic information. In: Egenhofer, M., Giudice, N., Moratz, R., Worboys, M. (eds.) COSIT 2011. LNCS, vol. 6899, pp. 350–370. Springer, Heidelberg (2011)
8. Bao, J., Zheng, Y., Wilkie, D., Mokbel, M.: Recommendations in location-based social networks: a survey. Geoinformatica **19**, 525–565 (2015)
9. Karagiannakis, N., Giannopoulos, G., Skoutas, D., Athanasiou, S.: OSMRec tool for automatic recommendation of categories on spatial entities in OpenStreetMap. In: RecSys (2015)