Automatic recommendations of categories for geospatial entities

Giorgos Giannopoulos
IMIS Institute, R.C. ATHENA
giann@imis.athena-innovation.gr

Nikos Karagiannakis
IMIS Institute, R.C. ATHENA
nkaragiannakis@imis.athena-innovation.gr

Dimitrios Skoutas
IMIS Institute, R.C. ATHENA
dskoutas@imis.athena-innovation.gr

Spiros Athanasiou
IMIS Institute, R.C. ATHENA
spathan@imis.athena-innovation.gr

ABSTRACT

Over the last years, thanks to Open Data initiative and the Semantic Web, there has been a vast increase on user contributed data. In several cases (e.g. OpenStreetMap, Geonames), the respective data include geospatial information, that is the coordinates and/or the precise geometries of buildings, roads, areas, etc. In such cases, proper schemas are defined to allow users to annotate the entities they contribute. However, browsing through a large and unstructured list of categories in order to select the most fitting one might be time consuming for the end users. In this paper, we present an approach for recommending categories for geospatial entities, based on previously annotated entities. Specifically, we define and implement a series of training features in order to represent the geospatial entities and capture their relation with the categories they are annotated with. These features involve *spatial* and *textual* properties of the entities. We evaluate two different approaches (SVM and kNN) on several combinations of the defined training features and we demonstrate that the best algorithm (SVM) can provide recommendations with high precision, utilizing the defined features. The aforementioned work is deployed in OSMRec, a plugin for JOSM tool for editing OpenStreetMap.

Categories and Subject Descriptors

[**Information systems**]: [Recommender systems]

Keywords

Category recommendation, spatial entities, OSM

1. INTRODUCTION

OpenStreetMap (OSM)[1] is an initiative for crowdsourcing map information from users. It is based on a large and active community that contributes both data and tools that facilitate the constant enrichment and enhancement of OSM maps. One of the most prominent tools of OSM is JOSM[2], a graphical tool that allows users to (a) download all the spatial features (i.e. roads, buildings, stations, areas, etc.) of a geographic area from OSM, (b) visualize these features on a map overlay, (c) add, change or delete spatial features and (d) reload the changed dataset into the publicly accessible OSM map dataset. Specifically, through its interface, the user can draw the geometry of a spatial feature. Then, she can annotate the feature with categories, in the form of key-value pairs. These spatial entities can also be annotated by categories (classes, tags) that assign semantics to them. Each entity may belong to multiple classes; for example, a building can be further characterized as school or bar.

The system allows the selection of such categories from an existing list[3] or the definition of new categories by the user. At this point lies the problem we handle, which is expected in such crowdsourcing approaches: the user can define an arbitrary category which (a) might already exist in a slightly different form or (b) might have no actual semantic meaning in the context of OSM and, thus, degrade the quality of the inserted information. Besides, the user might be discouraged by the large amount of available categories and quit the task of annotating the inserted spatial entity. Trying to avoid (a) and (b) by restricting the category selection only to categories that already exist in the OSM dataset has important disadvantages. First, it would cancel a large part of the crowdsourcing character of OSM and, second, it is not guaranteed that the OSM categories, in their current form, cover every aspect of characterizing spatial features. Instead, the most suitable course of action is to guide the users during their annotation process, recommending to them already existing categories to annotate the new spatial entities.

In this work, we propose such a process, that trains recommendation models on existing, annotated geospatial datasets and is able to recommend categories for newly created enti-

[1]https://www.openstreetmap.org/
[2]https://josm.openstreetmap.de/
[3]http://wiki.openstreetmap.org/wiki/Map_features

LocalRec'15, September 19, 2015, Vienna, Austria.

ties. The main contribution of our work lies on the problem specific training features we define 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 already existing (and annotated with categories) ones in several levels: spatial similarity, e.g. the number of nodes of the feature's geometry and textual similarity, e.g. common important keywords in the names of the features.

We perform an extensive 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 Neighbour (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. To this end, the proposed framework is implemented as a JOSM plugin[4], allowing the real-time and effective annotation of newly created spatial entities into OSM.

To the best of our knowledge, this is the first approach on recommending annotation categories for spatial entities in OSM. Another existing work on a similar problem [1] 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.

The rest of the paper is organized as follows. Section 2 describes our proposed method, including the defined training features and the assessed algorithms. Section 3 presents the evaluation of training features and algorithms in terms of recommendation performance and Section 4 concludes.

2. RECOMMENDATION MODELS

Next, we describe the model training and recommendation process we follow. In general, our goal is, given a training set, that is a set of spatial entities that are already annotated with categories, to be able to produce category recommendations for each new, unannotated entity. Thus, we aim at (a) learning correlations between attributes of the spatial entities and the respective categories and (b) "matching", through these correlations, new entities with already annotated ones, in order to produce recommendations.

The first step to this end is to define proper, problem specific training features, that describe each entity and capture its latent relation with the category that annotates it. In Section 2.1 we describe the defined features that correspond to geometric and textual 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 art classification methods. Each of the algorithms is described in Section 2.2.

2.1 Training features

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

In our scenario, the training items are the geospatial entities that have already been annotated with categories. However, since the same entity might be annotated with more than one category, we consider, for each entity, as many training items as its categories. The items are respesented in exactly the same way, w.r.t. the values of their training features, with the exception of the different label that corresponds to the different class. Each item is represented as a feature vector, with each feature corresponding to a property of the item. Next, we present the implemented features:

- **Geometry type**. Six distinct geometry types are identified: Point, LineString, Polygon, LinearRing, Circle and Rectangle. This is a boolean feature, so six positions in the feature vector are assigned to it.
- **Number of geometry points**. Depending on the algorithm this might be a double precision number, or a set of boolean positions in the feature vector that are used to represent it. For the latter case, each position represents a different range. In total, we define (based on a statistical analysis of the frequencies of entities having certain numbers of points) 13 ranges, thus mapping this integer characteristic into 13 boolean features: [1-10], (10-20], ..., (200-300], (300-500], (500-1000], (1000-...). So, according to the number of points of an entity's geometry, the proper position is set to 1, while the rest positions are set to 0.
- **Area of geometry**. Depending on the algorithm this might be a double precision number, or a set of boolean positions in the feature vector that are used to represent the various ranges of areas we consider. In the latter case, we define intuitively (and considering that in this problem setting we are mainly interested in entities that are buildings) 24 ranges with increasing length in square meters, until the area of 4000m2, where we consider the 25th range of (4000-...).
- **Mean edge length**. Depending on the algorithm this might be a double precision number, or a set of boolean positions in the feature vector representing different ranges of the mean length of the geometry's edges. In our case, we define 23 ranges, starting from length less than 2meters and ending with length more than 200meters.
- **Variance of edge lengths**. Depending on the algorithm this might be a double precision number, or a set of boolean positions in the feature vector representing different variations of a geometry's edges from the mean value. Likewise, we define 36 ranges, based on statistics on our training dataset.
- **Textual features**. 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 in descending frequency and filter out words with frequency less than 20. Finally, we apply a stopword list and remove words without any particular meaning. What remains are special meaning identifiers, such as avenue, church, school, park, etc. Each of these special keywords is used as a separate boolean feature.

We should note here that we select different representations (double precision number or set of boolean features) in order to favour the functionality of the different models and similarity functions we apply. Namely, we consider boolean features in the case of SVM to facilitate the effectiveness of the training model, 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]. That is, defining several area ranges corresponding to separate feature positions, we allow the model to correlate these different areas (feature positions) with different training classes. On the other hand, in the case of k-NN, where similarity measures such as the Euclidean distance or 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.

2.2 Algorithms

SVM. The first algorithm applies multiclass SVM classification considering as training items the geospatial 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 discriminated 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 is the well known approach of performing kNN on the initial set of training entities. That is, 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 are 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:

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

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

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

.

3. EXPERIMENTAL EVALUATION

Next, we present the evaluation of the proposed methods, w.r.t. the recommendation precision they achieve. First, we describe the dataset and the evaluation methodology. Then, we compare the two algorithms and, finally, we discuss the contribution of individual groups of training features.

3.1 Dataset and Evaluation methodology

We performed our evaluation on a subset of OSM data covering Athens, Greece, which we exported through the Overpass API[5] from OSM's site. The dataset contains overall $111,491$ geospatial entities which were properly divided into training, validation and test sets, as will be demonstrated next. The total number of distinct OSM categories/classes where $1,421$. The dataset is partitioned into five subsets of similar sizes. Then, combining each time different subsets to make (a) the training, (b) the validation and (c) the test set, 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. Of course, the validation part was applied only with SVM, since kNN does not train any model to be tuned.

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

$$P = \frac{\#correct\_category\_recommendations}{\#total\_category\_recommendations} \tag{3}$$

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 geospatial 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 geospatial 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 geospatial entity.

3.2 Algorithm Comparison

We performed two rounds of experiments for the SVM method. The first one regarded optimizing the classification model by training it with different parameterizations on C (trade-off parameter between the margin size and training error of the algorithm). After obtaining, through the 5-fold validation, the optimal parameters on the validation set, we run the actual evaluation of the recommendation model, using the three precision measure variations defined above. Note that the plain k-NN algorithm is a purely memory based algorithms, so it requires no training. The results for the best performing configuration, for each algorithm are given in Table 1.

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

Features	**C**	**Valid. Set**	**Test Set**		
		P^1	P^1	P^5	P^{10}
Geometry Type	5	66.95	67.86	69.45	71.03
Points and Area	200000	64.44	64.66	69.50	74.60
Mean and Variance	10000	65.087	65.17	78.69	85.63
Simple Geometry	170000	63.05	61.43	67.19	71.39
Only Textual	100000	6.71	6.01	7.83	8.261
Simple Geom. & Text.	**200000**	**79.54**	**79.04**	**87.90**	**92.07**
All	650	63.56	63.92	83.11	89.82

Table 2: SVM feature combinations.

Algorithms	**Valid. Set**	**Test Set**		
	P^1	P^1	P^5	P^{10}
SVM	79.54	79.04	87.90	92.07
kNN	-	65.53	73.03	78.05

Table 1: Recommendation precision.

It is obvious that SVM outperforms kNN by far. Regarding the strictest measure (P^1), the recommendation model achieves precision of 79%, which is a very good result, considering that it is measured on real-world OSM data without any restrictions that might favour the method. Further, when we consider the other two measures, the precision becomes even higher, reaching 88% and 92% for P^5 and P^10 respectively. Given that recommending 5 or 10 categories to the user is a realistic option, P^5 and P^{10} are suitable for evaluating in a real-world application, and thus the effectiveness of the system proves to be very high.

3.3 Training Features Analysis

The evaluation presented in Section 3.2 indicated that the SVM algorithm produces by far more precise recommendations in our setting. In this section, we analyze which training feature combinations achieve the highest precision results, when applied with SVM. To this end, we ran several experiments, following the same five-fold, training-validation-testing process, selecting each time, different groups of traing features. Next we report on these results (Table 2).

Geometry Type. This configuration consists in using vectors that contained only the features regarding the geometry type of the OSM entities. The geometry types are: polygon, linestring, linear ring, point, circle and rectangle

Points and Area. This configuration consists in using vectors that contain boolean features that correspond to the number of points and the area of the entities.

Mean and Variance. This configuration consists in using only the mean and variance features.

Simple Geometry. This configuration consists in using vectors containing only the geometry information about the OSM entities, without any class, relation or textual features in the vectors. In this experiment we also excluded the mean and variance geometry features from the vector.

Only Textual. This configuration consists in using vectors with features regarding only textual information. The precision came out very low with this configuration because only a little fraction of the OSM entities contain information about their names.

Simple Geometry and Textual. This configuration consists in using the geometry features (without the features regarding mean and variance values), plus the textual information extracted from the names of the entities. This is the best achieving configuration in the whole set of experiments, on all evaluated methods.

All. This configuration consists in using all available features. As mentioned above the class and relation features exist only in the train sets. The results are slightly worst that the best performing configuration above.

In brief, the above analysis indicated that, in the specific scenario, the most highly contributing training features are the ones considering simple geometric properties of the entities. The textual features, although they perform very poorly on their own (probably due to their sparseness) they can boost the performance of the simple geometric features.

4. 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, combined with SVM classification, result to recommendations of high precision. Although we base our work on the OSM use case, the proposed framework is general enough to be used in any other scenario that involves spatial entities and an initial set of annotations on a few of them. Further, our method is already implemented as a plugin for JOSM OSM editing tool. Our next steps involve defining further meaningful training features and performing more extended evaluation on the effectiveness of the algorithms, e.g. by excluding very frequent categories from the experimental data.

5. ACKNOWLEDGEMENTS.

This research is conducted as part of the EU project GeoKnow[6] FP7-ICT-2011-8, 318159.

6. REFERENCES

[1] Jilani, M. and Corcoran, P. and Bertolotto, M. Automated Highway Tag Assessment of OpenStreetMap Road Networks. In *Proceedings of SIGSPATIAL'14*, 2014.

[6]http://geoknow.eu/