ABSTRACT
With the increasing availability of geopositioning information, due to the massification of devices combining GPS receivers with access to location-based services and social networks, we are also witnessing a growing interest in the analysis of human location histories. This paper compares different approaches for classifying the visited locations, within human trajectories, according to semantic categories, one of them based on heuristics and two others based on Hidden Markov Models (HMMs), relying on either supervised or unsupervised learning. HMMs can take into account the location characteristics as unobservable context, relating this information beneath a time process, in our case corresponding to trajectories. I report on a series of experiments made with a dataset that combines the previously available GeoLife trajectory dataset with information collected from the Foursquare location-based social network. The results show that a classification accuracy of 56.5% can be achieved with the supervised HMM, when considering regions with an area ranging from 740 \( m^2 \) to 1534 \( m^2 \), in the modeling of the trajectories.

Keywords
Hidden Markov Models, Semantic Location Classification, Human Mobility Analysis, Trajectory Mining, Machine Learning

1. INTRODUCTION
Interest on the analysis of human movement data has recently increased, perhaps as a consequence of the larger availability of such data. This interest is also due to the increased adoption of the nowadays currently ubiquitous geopositioning technologies, and also due to the widespread usage of location-based services and location-based social networks. In parallel, studies over movement data have shifted from raw movement data analysis to more application-oriented ways of analyzing segments of movement.

The state-of-the-art on trajectory data management includes sophisticated techniques for data modeling, storage, indexing, querying, and mining of human mobility patterns [11]. However, most of the existing methods mainly focus on the spatio-temporal features of trajectories, while the high-level semantic features of trajectory data are still less-explored.

In this paper, I address the task of classifying the individual locations, visited in the context of a trajectory, according to semantic categories such as Shopping or Nightlife. Different methods have been studied, in which a first corresponds to a set of heuristics which associate to each location, according to a set of rules, the semantic class that best describes its context. The other proposed methods are based on a well-known technique for modeling sequential data, namely Hidden Markov Modeling. I represent each visited location within a trajectory, originally associated to geospatial coordinates (i.e., latitude and longitude) and to a timestamp (i.e., composed by the day of the week and the hour of the day), according to a set of codes. These spatio-temporal location codes constitute the symbols in the hidden Markov model, whereas the semantic classes, to be assigned to each location, constitute the hidden states in the model.

I report on a set of experiments and on a detailed characterization study made with a well-known dataset in the area of trajectory analysis, namely the GeoLife dataset [22, 24, 23]. I enriched the GeoLife dataset with semantic classes for each of the visited locations, through a semi-automatic approach that corresponds to the first of our methods, leveraging on data from the Foursquare location based social network. The enriched dataset was then used as the basis for the experiments with the HMM approaches. I specifically considered both supervised and unsupervised settings, for the training of HMMs.

The rest of this paper is organized as follows. Section 2 summarizes the most important related work. Section 3 presents the proposed methods, discusses the representation of the coordinates for the individual locations through the application of a hierarchical triangular mesh, and details the training and the inference of semantic trajectory patterns with Hidden Markov Models. Section 4 presents the evaluation results for the proposed methods, starting with a description of the dataset and then presenting a detailed discussion over the obtained results. Finally, Section 5 presents our conclusions and provides suggestions for future work.
2. RELATED WORK

Previous works have addressed distinct issues involving trajectory data. The book by Zheng and Zhou has for instance presented an extensive and detailed survey [25].

Parent et al. explained the basic concepts involved in trajectory analysis, and they also discussed several techniques for the enrichment of trajectories with semantic meaning, and for the extraction of behaviour knowledge from trajectories [11]. Trajectory enrichment concerns of the addition of contextual knowledge to the raw data, from a contextual data repository (e.g., from geo databases). The contextual links happen either during or before the trajectory’s analysis. This survey has a particular focus on data mining techniques, although there are mentions to other distinct approaches considering the manner through which data are modeled, such as state-space models or hybrid methods, where the last ones typically combine template matching techniques with distinct strategies.

Data mining techniques normally consider each trajectory to be an ordered set, since sequence analysis methods or modified versions of the Apriori algorithm are typically used [8, 9]. Frequent patterns or association rules are identified by first arranging the trajectories into small location sets, and then the analysis methods iteratively evaluate the rules with a pre-determined confidence support. At the end, the resulting patterns and association rules are used to classify the locations within the trajectories. Data mining techniques tend to perform less accurately since these models only rely upon the available data, specially, in situations where the dataset does not cover most of all possible cases, or when there exists a high possibility of uncertainty. Furthermore, data mining methods do not usually consider the spatio-temporal distances between the timestamped locations within the trajectories.

For instance, Morzy presented a data mining approach to predict the future location of a moving object [9]. The first step of his approach is extracting association rules from the moving object dataset. Then, for a previously unknown trajectory, this approach uses matching functions to choose the best association rule that fits to this trajectory, and afterwards makes the prediction based on it. This author reports an accuracy of 80% for the system’s best configuration.

State-space models try to identify the differences between spatial sequences through sequence classification models, such as generative Hidden Markov Models (HMMs) [12], discriminative Conditional Random Fields (CRFs) [17, 15], or more complex versions of these two well-known approaches [4, 10]. In opposition to data mining approaches, sequence models can deal with uncertainty, since they are based on probabilities. Both HMMs and CRFs are distinct but valid approaches for the proposed work of trajectory classification.

Asahara et al. proposed a state-space modeling method for predicting pedestrian movement based on the HMM model, namely the mixed autoregressive HMM (MAR-HMM) [3]. Although the HMM is a flexible approach, the authors identified a problem related to the predicted positions, concerning the fact that the individual positions do not depend on the previous ones. Differently from what the HMM model does for the unobservable states (i.e., the Markov Chain), the positions are not dependent on the previous ones due to the nonexistence of direct dependencies between the generated symbols. Before introducing the MAR-HMM, the authors presented the autoregressive HMM (AR-HMM) model to face the exposed limitation. In opposition to a traditional HMM, the AR-HMM model adds a chain between the symbols (i.e., positions) at the cost of considering a larger number of parameters. The MAR-HMM is a special case of the AR-HMM, where the authors dissociate pedestrian stable properties (e.g., age and gender) from the unobservable states of the AR-HMM model, thus creating a second level of states that resembles the mixed Markov-model (MMM) from their previous work [2]. By grouping the unobservable states that share the same stable properties, the top-level states constrain and reduce the number of unobservable states and the transitions between them. Therefore, the parameter estimation is simpler on the MAR-HMM model. The authors compared the proposed method to others, namely the MMM and the AR-HMM. In that comparison, the proposed method outperforms the others by achieving 56.8% accuracy, against prediction rates of about 49.2% and 51.5%, respectively.

Hybrid approaches normally combine different strategies with template matching techniques, which are based on performing a similarity search on sequential data. For finding these similarities in sequences or time-series data, specific metrics evaluate the extracted features against previously identified patterns or templates. The used metrics are mainly adaptations of algorithms used in traditional string matching problems, such as edit distance or longest common subsequence, although dynamic time warping or other heuristic algorithms can also be used [6, 13]. Template matching techniques normally have a long training time and a low sensibility for spatial variation, due to the inclusion of partial searches using subsequences of a given sequence.

Ying et al. proposed a hybrid method for predicting the next location of a given user, that uses clustering to identify templates [21]. Initially, the clustering strategy groups similar users by analyzing common behaviour present in semantic trajectories. Later, using patterns of users in the same cluster, both the geographic and semantic features of the trajectory support the prediction task.

Participants in the Nokia Mobile Data Challenge (MDC) event presented an approach to address the problem of classifying places visited within trajectories [7]. In this event, and for the visited places classification task, the winning method by Zhu et al. showed the vital importance of feature engineering, before using a well-known model [20]. The authors proposed the use of a technique based on conditional features, which involved the evaluation and generation of two types of features, namely the unconditional and the conditional features. Each unconditional feature is a single value (e.g., WiFi strength). As for the conditional features, the correlation and dependency between variables were explored where one or more temporal features conditioned the calculation of a non-temporal one (e.g., WiFi Strength conditioned by WeekDay/Weekend). The technique selected a small set of features though to be more important for obtaining a correct classification with the model by analyzing the
relationship between them. An important conclusion to retain was that the time dependent features being very useful for the place category classification. The models that had better results were based on Gradient Boosted Trees and L1-regularized Logistic Regression, achieving respectively an accuracy of 75.1% and 74.6%.

3. THE PROPOSED METHODS
This paper presents different methods for semantically classifying locations visited in the context of a trajectory. In this section, I start by giving a formal overview about this problem, focusing on the input data and on the information resulting from a classification approach. Then, I explore different methods for executing the classification task, while also analyzing their distinct features.

3.1 Overview
A trajectory is an ordered set of visited locations collected by a GPS receiver. Formally, a trajectory is represented as a sequence \( T = \langle l_0, \ldots, l_n \rangle \), in which \( n \) is the number of locations. Note that the number of locations may vary from one trajectory to another. Within a trajectory, each of the locations is characterized by a tuple \( l_i = \langle \text{latitude}, \text{longitude}, \text{timestamp} \rangle \). The first two fields are the coordinates of the position captured by the GPS receiver, and the last one registers the temporal instant when the visit happened. An entire trajectory dataset can be represented as \( T = \langle t_0, \ldots, t_m \rangle \), where \( m \) is the number of available trajectories.

Using a trajectory dataset as described above, we are only able to identify spatial and temporal patterns. Since the objective of our work is to analyze human behavior, we need to add some information about the categories of the individual locations within these raw data. Nowadays, location-based social networks disseminate this missing information. In a location-based social network (LBSN), people share information about visited locations and, normally, this includes specific contextual information. Some of the existing LBSNs suggest tags to users, in order to represent these contextual elements. A tag usually helps to identify immediately the activities that can be performed in a specific location. For instance a tag can be a category (e.g., Food), which may indicate the presence of a restaurant in a specific location. To include the behavioral information in a trajectory dataset, we use the location’s coordinates (i.e., latitude and longitude) to look for contextual information in a specific LBSN, namely Foursquare. Then, we add a new field to each location in the trajectory dataset, containing the search result for the location’s raw information. Each location in the original trajectory dataset is thus afterwards represented as a tuple \( l_i = \langle \text{latitude}, \text{longitude}, \text{timestamp}, \text{cat} \rangle \), in which \text{cat} refers to the semantic category of each location.

By associating semantic categories to existing trajectory data, and latter through automated analysis, we can perhaps address questions such as (1) what are the categories that are typically visited in sequence within the trajectories, or (2) which categories are most popular, at particular geographic regions or at particular temporal intervals?

3.2 Heuristic Trajectory Classification
For performing trajectory classification, we can use the coordinates of each location to search for behavioral information in a LBSN. Since geopositioning devices collect the locations associated with a timestamp, we can also strengthen an heuristic classification approach with this information. Believing there are activities that are more likely to happen at a particular time of the day, I propose to use time intervals to filter the categories representing the activities. Our common sense determines these time intervals, believing that they can apply to different cultures. In this way, we built a classification heuristic that will try to take full advantage of the available information, for each specific point in a trajectory.

The main elements of LBSNs are places, where users do and share activities. Thus, we are able to associate the geographic characteristics of these places to behavioral information. Normally, LBSNs provide users two ways of describing their activities in a specific place, namely through comments, and by selecting existing tags. We use the tags as a semantic meaning (i.e., categories) for the locations.

We choose the Foursquare\(^1\) LBSN for collecting behavioral data because of its popularity, verified by both the number of registered users and by the number of indexed places, with basis on the information provided by Foursquare. We considered a set of nine possible categories corresponding to \( \text{cat} \in \{ \text{Shop and Service}, \text{Nightlife Spot}, \text{College and University}, \text{Food}, \text{Outdoors and Recreation}, \text{Professional and Other Places}, \text{Arts and Entertainment}, \text{Residence}, \text{Travel and Transport} \} \). Due to the number of existing locations and due to limitations in the Foursquare’s API, we developed a cache mechanism to avoid similar queries to Foursquare’s API. We also consider adding the same tags for locations within a 70 meters radius of a given point, without performing an additional search.

Considering both the location’s coordinates and the timestamp, the dataset is enriched with the previously mentioned set of categories from Foursquare. The algorithm described next supported the choice of one of these categories to label each GPS location within a trajectory. For each of the trajectories in a given dataset, the following steps are applied:

1. Use the Foursquare API to collect the set \( S \) of all local businesses that are located within a radius of 150 meters from the geospatial coordinates associated to the trajectory point;
2. Return UNKNOWN if \( S = \emptyset \), and consequently discard the current location. We can then re-start the process for the remaining part of the current trajectory;
3. Return the category associated to the closest local business in \( S \), if this category \( \neq \) UNKNOWN;
4. Return Nightlife Spot if the name for the closest local business in \( S \) contains the corresponding keywords presented in Table 1, and if the timestamp associated

\(^1\)https://developer.foursquare.com/
to the trajectory point is within a period of the day between 22:00PM and 05:00AM;

5. Return *Arts and Entertainment* if the name for the closest local business in \( S \) contains the corresponding keywords presented in Table 1.

6. Return *Shop and Service* if the name for the closest local business in \( S \) contains the corresponding keywords that are presented in Table 1.

7. Return *Food* if the name for the closest local business in \( S \) contains the corresponding keywords that are presented in Table 1.

8. Return the category associated to the second closest business in \( S \), if this category exists and if it is \( \neq \) UNKNOWN;

9. Return the most visited category, in the set of local businesses \( S \), if this category \( \neq \) UNKNOWN;

10. Return the category associated to the most visited local business in \( S \), if this category \( \neq \) UNKNOWN;

11. Return *Nightlife Spot* if this category exists in the set of local businesses \( S \), and if the timestamp associated to the trajectory point is within a period of the day between 22:00PM and 05:00AM;

12. Return the category associated to the second most visited local business in \( S \), if this category exists and if it is \( \neq \) UNKNOWN;

13. Return the second most visited category, in the set of local businesses \( S \), if this category exists and if it is \( \neq \) UNKNOWN;

14. Return UNKNOWN, ignoring this location and transforming the annotated part of the current trajectory (i.e., the annotated locations until the current location which is to be discarded) into a new and individual trajectory, if it contains at least two already annotated locations. Then, we re-start the algorithm for the remaining part of the current trajectory (i.e., starting with the next location that is not annotated), which we consider to be also a new and individual trajectory, if it contains at least two already annotated locations.

From the previous algorithm, it is important to notice that we will split a trajectory into two separate ones when there exists one location that was not possible to annotate with a corresponding category. We consider this approach in order to use most of the available data.

### 3.3 Human Trajectory Classification by Using Hidden Markov Models

In this section, we present the proposed methods based on Hidden Markov Modeling.

The Hidden Markov Model (HMM) is a well-known generative probabilistic model for explaining the generation of sequences of observation symbols \( x \) associated with unobserved (i.e., *sequences of hidden classes*) states \( y \).

<table>
<thead>
<tr>
<th>Category</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Nightlife Spot</em></td>
<td>bar, pub, disco, or club.</td>
</tr>
<tr>
<td><em>Arts and Entertainment</em></td>
<td>cinema, movie, theater, or hall.</td>
</tr>
<tr>
<td><em>Shop and Service</em></td>
<td>store, shop, or mart.</td>
</tr>
<tr>
<td><em>Food</em></td>
<td>restaurant, bistro, café, or teahouse.</td>
</tr>
</tbody>
</table>

**Table 1: Category keywords.**

An HMM can be used to model a process, with a given duration \( T \), since it considers that each state \( y_t \) is directly influenced by the previous state \( y_{t-1} \), and linked to a corresponding symbol \( x_t \). The HMM model also considers that each state can emit different symbols with distinct probabilities. In order to explain the generation of a sequence of symbols, the HMM model takes into account three contributing factors. The first is the probability of the sequence starting at a particular state. The second is the probability of choosing a given state after another (i.e., transition probabilities). The last is the probability for the production of the different symbols in each state (i.e., emission probabilities).

Taking the notation of [5], the model can be seen as a tuple \( \lambda = \langle A, B, \pi \rangle \), inside the context of having \( N \) possible states and \( S \) different possible symbols. Respectively, the vector \( \pi = \langle \pi_{y_1}, ..., \pi_{y_N} \rangle \) contains the probabilities of starting a sequence at a state \( y_n \). The matrix \( A = [a_{y_n, y_m}] \) contains the transition probabilities from a state \( y_n \) to a given state \( y_m \). The matrix \( B = [b_{y_n, x_s}] \) contains the probabilities for the emission of symbol \( x_s \) at a state \( y_n \).

Considering an instantiated hidden Markov model as a tuple \( \lambda = \langle A, B, \pi \rangle \), the two problems that are more important to solve, in terms of sequence analysis, are (i) computing the probability of a given sequence \( x \), that is computing \( P(x|\lambda) \), and (ii) computing the most likely sequence of states \( y \) to have generated an observed sequence of symbols \( x \) (i.e., the classifications for symbols in \( x \)), that is, finding the state sequence \( y = \langle y_1, y_2, ..., y_T \rangle \) which maximizes \( P(x|y,\lambda) \) for a given sequence of symbols \( x = \langle x_1, x_2,...,x_T \rangle \).

Since the objective of the work reported in this paper is to provide contextual information about human behaviour, through the observation of trajectories, the focus in terms of hidden Markov modeling is on the second problem. Here, the Viterbi algorithm [19], which is a particular form of dynamic programming, addresses the problem of finding the most likely sequence of states for a given sequence of symbols. For each symbol \( x_t \), the algorithm computes the probability of its emission for every possible state. The Viterbi algorithm starts by calculating the initial probability of the emission of symbol \( x_1 \) in all possible states. Then, for each state transition, it calculates again the emission of symbol \( x_2 \). This step is repeated for each symbol, until step \( T \), where the given symbol sequence ends. Finally, with all possible paths covered, the algorithm finds the path that ends in the most probable state then goes from the end to the start collecting the most probable states, this way obtaining the
most likely state sequence $y = <y_1, y_2, ..., y_T>$. The Viterbi algorithm can also solve the problem by considering costs, instead of probabilities, returning the sequence of states that minimizes the associated cost. Many practical implementations follow this approach, by taking the logarithms of the probabilities as the costs.

The HMM approach has some limitations regarding the capture of correlations. The correlation between two separated states is one of these limitations, since states that are not connected cannot communicate directly with each other. This communication is made through other states that are in the path, represented by the black arrows in Figure 1. Although the first-order Markov model does not capture these kinds of long relations (e.g., in a first-order HMM, a state $y_t$ only depends on state $y_{t-1}$), higher-order models can minimize this problem. For instance, on a three-level model, we have that at a given time $t$, the probability of reaching a given state depends on the current state and on the two previous ones. This is represented in Figure 1 by the blue arrows between states.

The other limitation regards the generation of the symbols, since they depend only on the corresponding state $y_t$. This is a problem, since $P(x_t)$ may be related with the surrounding states of $y_t$. A possible representation for these missed relations is given by the red arrows in Figure 1. Theoretically, in HMMs, one can use a distribution like $P(x_t|y_{1:t-1}, y_t, y_{t+1})$, so that $P(x_t)$ can be influenced by the surrounding values in $y$. However, the principles behind this representation are not as clear as in traditional HMMs.

To apply HMMs to our task, we consider the information available in each trajectory’s location. Therefore, a symbol of the HMM is a representation of a trajectory’s spatio-temporal location and, in this way, a temporal sequence of symbols stands for a trajectory. For building each location’s representation (i.e., the corresponding symbol), we use both the geospatial and the temporal information.

For detailed information about the usual algorithms for solving problems with hidden Markov models, please refer to the tutorial by Rabiner [12].

### 3.3.1 Handling Spatio-Temporal Positions Through a Hierarchical Triangular Mesh

Considering the continuous spatio-temporal data in the Geolife dataset, and considering that HMMs typically use discrete-based representations for modeling the emission probabilities, a consistent method was necessary to represent the continuous spatio-temporal data as discrete labels, according to the proposed task (i.e., symbols that represent visits occurring in a specific area).

The Hierarchical Triangular Mesh (HTM) [16] is a method for dividing a spherical surface, such as the Earth, into triangular regions (i.e., trixels) of a form and size that is roughly uniform. In a very efficient way, this method provides the opportunity to index localized objects in the surface of a sphere (e.g., locations corresponding to coordinates), see Figure 2.

This method is configurable through a resolution parameter, which indicates the number of division iterations that should be considered. Each triangular shape can be divided into 4 new ones, through the midpoints of its edges. Initially, the sphere is divided into 8 triangular shapes (e.g., 4 shapes for each hemisphere). Thus, after the first iteration, I will have $8 \times 4 = 32$ triangular shapes, while after the second one I will have $8 \times 4^2 = 128$ regions, and so on. Therefore, a resolution parameter $r$ can be used to control the number of trixels accordingly to the formula $8 \times 4^r$.

Given that this work focuses on the semantic classification of locations within human trajectories, we believe that considering regions of approximately 740 to 1500 m$^2$ would be appropriate for our task. Since the considered semantic classes include shops or entertainment regions, we considered a resolution of parameter 18 for the HTM approach, which generates triangles with the aforementioned characteristic. Then, with this method, we represent a geoposition (i.e., coordinates of latitude and longitude) through the HTM code of the corresponding triangular region. This HTM code constitutes the geospatial information present in the symbols used by my HMM models.

However, geospatial information does not completely reflect the distinct activities that can occur on the location under consideration. In some cases it is better to rely on temporal information to semantically classify a location. For instance, if in one location there are some shops, an art gallery and a nightclub, you can go shopping or visit an exhibition during the day, and later go to the night club. However, by only considering the geospatial information, we will not be able to classify semantically the activities performed at this location. With temporal information, it is easier to understand which activity occurred, since art galleries usually do not open every day, the same happening with clubs that, normally, receive visits during the night.
We therefore define our model’s symbols as the concatenation of both the geospatial and temporal information of each visit within a trajectory. Particularly, we consider the day of the week and the hour of the day to be the temporal marks that are more important for this task. The reason behind this choice is based on the fact that people usually make some activities according to specific hours, or according to the day of the week, as described in the previous example. Henceforward, and as illustrated in Figure 3, the symbols used in our HMM models can be seen as tuples \(< HTM\ code,\ Day\ of\ the\ week,\ Hour\ of\ the\ day >\), with the day of the week mapped into a number from 1 to 7 (i.e., from Sunday to Saturday) and with the hour of the day mapped to a number from 0 to 23.

### 3.3.2 Using HMMs in a Supervised Setting

One particular problem with the usage of HMMs concerns with learning the model parameters. Considering a supervised setting, one can use annotated data obtained through the proposed heuristic trajectory classification method (i.e., using categories as states) to estimate the beginning, transition and emission probabilities, through the following equations:

\[
\pi_{yi} = \frac{c_{yi}(i \rightarrow j) + 1}{\sum_{z \in Y} c_{zi}(i \rightarrow s) + |Y|} \quad (1)
\]

\[
a_{yi,yj} = \frac{c(i \rightarrow j) + 1}{\sum_{z \in Y} c(i \rightarrow s) + |Y|} \quad (2)
\]

\[
b_{yi,x} = \frac{c(i \uparrow x) + 1}{\sum_{\rho \in X} c(i \uparrow \rho) + |X|} \quad (3)
\]

Consider that the parameters \(c(i \rightarrow j)\) and \(c(i \uparrow x)\) represent the number of existing transitions from state \(i\) to \(j\), and the number of times that symbol \(x\) is observed under state \(i\). After the counting process, and to avoid excluding any possibility (i.e., to avoid the appearance of cases with a probability of zero), we can apply Laplace smoothing to the estimation of the model parameters. As Equation 1 shows, in the case of the initial probabilities, we add 1 to the count of each beginning state, which represents the context of the first visit, and then we sum to the denominator the total number of states. In Equation 2, for each transition we also add 1, but this time we add to the denominator the number of possible transitions to any state (i.e., \(|Y|\)). Finally, Equation 3 presents the smoothing procedure over the emission probabilities. Here, we add 1 to the relative count on a symbol under a given state, and we later add to the division the total number of symbols (i.e., \(|X|\)).

To circumvent the problem of sparse data, we also designed a process to smooth the emission probabilities based on the HTM method. In this procedure, we use two supervised HMMs with distinct geospatial resolutions, where a coarse-grained model smooths a thin-grained one (i.e., the model with a larger number symbols and a higher geospatial resolution). The rational for the implementation of this smoothing strategy is that, for a given timestamp, we can obtain the activities’ distribution (i.e., the context for the visits) over a larger area, and use it as a component for the activities’ distribution of each smaller area within the larger one. In this way, we can construct a probability distribution on the context of visits to symbols that do not appear in the training set. This does not happen when we use only the Laplace smoothing procedure, since for these specific symbols, in the context of a given state, the emission probability will be constant. The following equation shows the strategy used for smoothing the emission probabilities:

\[
b_{yi,x} = (b_{yi,x} \times 0.8) + \left(\frac{b_{yi,x}}{|X| \times 0.2}\right) \quad (4)
\]

This equation has two components, one considering the Laplace smoothing with a weight of 80%, and another representing the smoothing factor of the strategy described before, and with a weight of 20%. The \(b_{yi,x}\) parameter is the emission probability of a coarse-grained symbol in a given state \(y_i\).

Lastly, both \(X\) and \(Z\) respectively correspond to the sets of symbols of the thin-grained and coarse-grained models. Note that we consider only symbols of the dataset. If a larger area contains smaller ones that do not appear in the dataset, then we distribute its probability mass between the existing areas. Another consideration is that, as Equation 3 shows, we only use Laplace smoothing in the coarse-grained model. In order to be able to relate symbols of distinct resolutions, but that share the same timestamp and their intersecting areas, we use the hierarchical structure of the HTM codes, which contain the number of thin-grained symbols that compose a coarse-grained symbol (i.e., this is variable, since we consider only symbols that occur in the dataset).

### 3.3.3 Using HMMs in an Unsupervised Setting

Another approach for addressing the semantic classification of locations, using HMMs, involves the usage of an unsupervised procedure for finding the model parameters. In opposition to the supervised approach, this one does not require any contextual information (e.g., semantic annotations), thus being independent of the heuristic solution. More specifically, given a set of trajectories, only the symbols that represent the spatio-temporal locations of each trajectory are used to support the task of learning the parameters. The goal is to find the parameters that best explain a given set of sequences.
There are several well-known algorithms that can derive efficiently a local maximum likelihood HMM model, such as Baum-Welch. Several works use this algorithm, recognizing it as being very effective in general, resulting in accurate models [14, 1]. The Baum-Welch algorithm is a special case of the expectation-maximization (EM) algorithm, which uses the principle of dynamic programming through the forward-backward algorithm for estimating the HMM parameters.

For describing this algorithm, we consider an HMM model $\lambda = \langle A, B, \pi \rangle$ with the notation presented earlier, within the context of having $N$ possible states and $S$ different possible symbols [5]. Here, $A$ is a matrix with the time-independent transition probabilities between states, $B$ is a matrix that contains the emission probabilities corresponding to the observations of a particular symbol given a state, and $\pi$ is a vector with the initial state probabilities, which is the probability that represents the states observed in the first position of the trajectory. In order to approximate the supervised set of states, we start at position $i$ and consider the model with $9$ possible states (i.e., $N = 9$).

In particular, the Baum-Welch algorithm aims to find $\lambda^* = \max_{\lambda} P(X|\lambda)$, where $X$ is a set of trajectories or one given trajectory. Let us consider $x = \langle x_1, x_2, ..., x_T \rangle$ as a trajectory with $T$ locations. This algorithm starts by initializing the HMM’s parameters with random values and, iteratively, updates them until convergence, or up until reaching a configurable number of steps.

An expectation step computes the probabilities for each state that possibly completes the trajectory’s location information, using the current parameters of $\lambda$ and the Forward-Backward algorithm. In brief, we have that the Forward algorithm calculates $\alpha_i(t) = P(x_1, \ldots, x_t, Y(t) = y_i|\lambda)$, which is the probability of seeing the partial observable sequence $x_1, \ldots, x_t$ ending up in state $y_i$ at time $t$. Efficiently, we can get these probabilities by using the following equations:

$$\alpha_{y_i}(1) = \pi_{y_i} b_{y_i, x_1}$$

$$\alpha_{y_i}(t + 1) = b_{y_i, x_{t+1}} \sum_{j=1}^{N} \alpha_{y_j}(t) \times a_{y_j, y_i}$$

Consider $a_{y_j, y_i}$ as the transition probabilities and $b_{y_i, x_t}$ as the emission probabilities in the previous equations. Later, the Backward algorithm calculates the probability of the remaining and ending part of the sequence $x_{t+1}, \ldots, x_T$, given that we start at position $t$ with the state $y_i$. Similarly to the Forward algorithm, this step calculates recursively the values of $\beta_{y_i}(t)$ through the equations:

$$\beta_{y_i}(T) = 1$$

$$\beta_{y_i}(t) = \sum_{j=1}^{N} \beta_{y_j}(t + 1) a_{y_j, y_i} b_{y_i, x_{t+1}}$$

With the values of $\alpha_{y_i}$ and $\beta_{y_i}$, we are able to calculate $\gamma_{y_i}$, which is the probability of a particular state at position $t$, and $\xi_{y_i, y_j}$, which is the transition probability from a state to another in a specific position $t$.

$$\gamma_{y_i}(t) \equiv P(Y(t) = y_i|X, \lambda) = \frac{\alpha_{y_i}(t) \beta_{y_i}(t)}{\sum_{j=1}^{N} \alpha_{y_j}(t) \beta_{y_j}(t)}$$

$$\xi_{y_i, y_j}(t) \equiv P(Y(t) = y_i, Y(t + 1) = y_j|X, \lambda) = \frac{\alpha_{y_i}(t) a_{y_i, y_j} b_{y_j}(t+1) b_{y_j, x_{t+1}}}{\sum_{j=1}^{N} \sum_{y_j} \alpha_{y_j}(t) a_{y_j, y_i} b_{y_i}(t+1) b_{y_i, x_{t+1}}}$$

A maximization step can then re-estimate the model’s parameters using the values of $\gamma$ and $\xi$. The equations shown below correspond to the update rules:

$$\pi_{y_i} = \gamma_{y_i}(1)$$

$$a_{y_i, y_j} = \frac{\sum_{t=1}^{T-1} \xi_{y_i, y_j}(t)}{\sum_{t=1}^{T-1} \gamma_{y_i}(t)}$$

$$b_{y_i, x_k} = \frac{\sum_{t=1}^{T} \delta_{x_k, x_{t+1}} \gamma_{y_i}(t)}{\sum_{t=1}^{T} \gamma_{y_i}(t)}$$

In the nominator of the equation that gives $b_{y_i, x_k}$, the summation is only made over observed symbols equal to $x_k$, that is, $\delta_{x_k, x_{t+1}} = 1$ if $x_k = x_{t+1}$ and $\delta_{x_k, x_{t+1}} = 0$ otherwise. Until convergence and after updating the parameters $\pi, A$ and $B$, the Baum-Welch algorithm executes a new iteration of the presented procedure.

Due to the characteristics of the trajectory dataset, and with the purpose of improving the model’s accuracy, we apply a similar smoothing strategy to the one described in Section 3.3.2. More specifically, consider another model with symbols of a low HTM resolution. Each one of these symbols relates to one or more higher resolution symbols, since they represent a larger area which includes at least one of them. The estimation of parameters in the lower resolution model occurs in the same way as in the original one, using the algorithm described earlier. The difference lies in the maximization step, where to re-estimate the emission probability $b_{y_i, x_k}$ for a given higher resolution symbol $x_k$, we consider the emission probability of the corresponding lower resolution symbol $x_u$, as follows:

$$b_{y_i, x_k} = (0.8 \times b_{y_i, x_k}) + (0.2 \times b_{y_i, x_u})$$

4. RESULTS

This section presents the results of the experiments executed to validate the proposed approaches. A dataset characterization is also provided, before presenting the obtained results.

4.1 The GeoLife Dataset

The dataset used for testing the proposed approaches is the Microsoft Research GeoLife GPS Trajectories dataset [23,
24, 22]. This dataset is characterized by having been mostly collected in China (i.e., in over 30 cities, within an emphasis on Beijing), although also in some European and American cities, for a period of 4 years (i.e., from April 2007 to October 2011) by 178 users. These data contain 17,621 GPS trajectories of locations visited in sequence, making for a total distance of about 1.2 million kilometers and a total duration of over 48,000 hours. The GPS locations of the trajectories within the dataset were captured every 15 seconds or every 510 meters, and include routine movements as well as leisure activities.

After enriching the GeoLife dataset with contextual information resulting from an heuristic supported on data collected from Foursquare, there are still locations where the contextual information is unknown. Given the objective to view each location as a staying point, where people perform some activity, we decided to exclude locations without contextual information. For training a model it is important to have a reasonable sample of trajectories. Due to sparse data, if we exclude all the trajectories that include locations with a point corresponding to an unknown context, the trajectory sample is considerably reduced. Therefore, regarding the trajectories that include locations with an unknown context, we decided to use parts of these trajectories, that can still be considered a trajectory (i.e., if these parts contain at least two locations).

Due to the way the data were captured, another problem identified on the dataset concerns with the repeated locations within a given trajectory. Thus, we filter redundant locations through the combination of two techniques, which use the angle of direction, time, and distance measurements. The first technique leverages on the intuition that it is likely for a person to change its activity, when changing the direction of the trajectory. So, as Figure 4 shows, we pick each 3 locations windows within a particular trajectory to evaluate the middle one, and check if there is a change in direction. Considering that the locations within the location window respect a time-series, the change of direction is identified when the angle, formed between the first location and the last one, is outside the threshold of $[160^\circ \text{ to } 200^\circ]$.

With this threshold, we try to include possible GPS inaccuracies and identify if the 3 locations within the trajectory have a consensual direction by drawing them as a vector with two segments. More specifically, when decomposing the vector's segments, there are at least two components from the distinct segments that agree in terms of direction. We use Vincenty's formulae [18] with the location's coordinates to calculate the distances between points and then, by knowing the distances, it is possible to know the location's angle through the Law of Cosines\(^2\). If the angle value is outside of the threshold, we add the location under evaluation to the clean trajectory, since we consider that a change has occurred of direction and possibly also a change of activity. Otherwise the locations agree in terms of direction and, considering the way the locations were collected for this dataset (i.e., every 15 seconds), we perform further analyses to verify if there was an activity change between the first location and the one under evaluation. On this analysis, we start by using a technique that checks for two conditions so we can add the location under evaluation to the clean trajectory. One is if the location under evaluation has a different HTM code than the first location. The other is if they have distinct categories. In addition, the location under evalu-

tion is automatically added to the clean trajectory, if it is the first or the last one of a given trajectory, apart of its known context.

For carrying out the trajectory classification task, we experimented with different geospatial configurations. Table 2 presents the dataset’s characteristics under four different geospatial configurations, from where we can see the influence of the geospatial resolution parameter on the enriched dataset (i.e., GeoLife locations associated to a category from Foursquare). The first segment is about the trajectory samples and the locations within them. We can observe that the number of trajectories is the same for the different geospatial parameters, but the number of locations within these trajectories varies. The higher the geospatial resolution parameter, the more locations within trajectories do exist. The increase of the average and standard deviation, relative to the locations per trajectory, relates directly with this. The second segment tells us how many HTM trixels we used to index the locations of our trajectories, together with the main characteristics of these trixels, namely the area and the positioning range, in our corresponding geospatial resolution models. The third segment shows that the timestamps associated to the dataset remain constant for all characterizations, as the geospatial parameter does not influence this information. Finally, the fourth segment exposes the number of considered symbols in each case. The geospatial resolution influences these values, since they result from the combination of trixels with temporal marks. Table 2 provides an overview on the difficulty associated with the classification task, since it confirms that the data are sparse due to the huge amount of symbols available for each experiment. This fact has a direct effect on model complexity, which is directly proportional to the amount of considered symbols.

### 4.1.1 Results with Supervised HMMs

We experimented with six distinct supervised HMMs, two for each data resolution from Table 2. Table 3 shows the overall results for these experiments, where we use the geospatial resolution parameter to describe the models. For representing the models that include the smoothing strategy relying on a coarse-grained model, we use two values where the smaller one (i.e., 10) is the geospatial resolution parameter of the coarse-grained HMMs. As a first and interesting observation over the results, we can report that the HMM with geospatial resolution of 18 is the most accurate model, and the HMM with geospatial resolution of 19 is the less accurate one.

Another interesting observation is that all models that included the more sophisticated smoothing strategy performed better than the ones without it (i.e., in terms of accuracy, $18_{10} > 19_{10} > 16_{10} > 18 > 16 > 19$). Table 3, and also Table 2, confirms an expected correlation between the geospatial resolution and the accuracy. Considering the existing categories, it is very likely to have visits to small locations (e.g., shops or restaurants) and we correctly expected that a model whose range can include small or medium areas will perform better. More specifically, as Table 2 shows, the area with geospatial resolution of 18 is the one that better fits this scenario, since it can include a regular small business location with its area ranging from 740 to 1534 m$^2$. Table 2 confirms the correlation by showing that, when considering distinct smoothing strategies, the best models are $18_{10}$ and 18. Other models do not possess the same capability, because they either look upon very large or very small locations.

In a different perspective, Table 4 details the results obtained for each state, each one corresponding to one semantic category. The first interesting information that we notice from this table is that the most accurate result corresponds to the Food state in the model $18_{10}$. In opposition, we observe that the less accurate state is the Outdoors and Recreation state in the model 19. Concerning the fact that Food is the most accurate state, we have that this result is expected since the majority of the locations were labeled with this category. Two observations prove this statement, one in the mentioned table, where the Food state is the most accurate one in four different models and is very close to the ones in the remaining experiments, and another in Figure 5, where we show the amount of locations by category present in the dataset filtered with geospatial parameter of 18. Regarding state Outdoors and Recreation having the worst overall result, we were not surprised due to the fact that this category could imply locations with huge areas or where the area varies significantly. Also, Figure 5 indicates that, in the dataset, there are few locations labeled with this category when compared to the other ones. Still in this perspective, it is interesting to report that the Nightlife Spot state achieved the worst results on four models, which it can be explained through the same reasons pointed out when discussing the Outdoors and Recreation state.

**Location distribution by Category**

<table>
<thead>
<tr>
<th>Category</th>
<th>Location Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professional&amp;OtherPlaces</td>
<td>565022</td>
</tr>
<tr>
<td>Residence</td>
<td>114137</td>
</tr>
<tr>
<td>Arts&amp;Entertainment</td>
<td>157165</td>
</tr>
<tr>
<td>Food</td>
<td>125075</td>
</tr>
<tr>
<td>College&amp;University</td>
<td>321208</td>
</tr>
<tr>
<td>Shop&amp;Service</td>
<td>527987</td>
</tr>
<tr>
<td>Outdoors&amp;Recreation</td>
<td>169894</td>
</tr>
<tr>
<td>Travel&amp;Transport</td>
<td>546390</td>
</tr>
<tr>
<td>NightlifeSpot</td>
<td>64297</td>
</tr>
</tbody>
</table>

Figure 5: Location distribution by category present in the dataset, with a geospatial resolution of 18.

We also observe that model $18_{10}$ achieved the majority of the best results by state, contrasting with model 16, that with the exception of the Food and Outdoors and Recreation states obtained the worst ones. Another curious observation is the improvement obtained from the smoothing strategy.
concerning the models with geospatial resolution of 18 and 19, in the context of the Outdoors and Recreation state. This is the perfect example of how the smoothing strategy helps to highlight smaller areas which probably compose the same location.

An interesting pattern observed in our results is the existing inverse relation between our smoothing strategy and the geospatial resolution, in which our smoothing strategy loses effectiveness when decreasing the considered area’s size. Table 4 exposes this rational through the F1 measure improvement within models of equal geospatial resolution. Almost all states improve for the models with model considering a geospatial resolution of 16, while two states fail to improve on the models with a geospatial resolution of 18, and another three fail to improve on the remaining models. Finally, we identify an interesting pattern when comparing models with the same smoothing approach, but with distinct geospatial resolutions. This pattern is the increase of the F1 measure when comparing a 16 model to a 18 one, and later, a decrease of the same measure when comparing a 18 model to a 19 one. The states not included in this pattern are the ones that have big differences in terms of the F1 measure between models with the same geospatial resolution (e.g., for the models with geospatial resolution of 18, the difference in the F1 measure is 38% on the state Outdoors and Recreation), namely Nightlife Spot, Outdoors and Recreation and College and University. The classification difficulty associated to these states lies with the features present in the considered symbols and on the amount of them. When considering these activities, we expect a sparse existence of multiple locations or different timestamps appearing in our trajectories.

From the obtained results, we conclude that the best model is 18_{10}. Therefore, we will explore in detail several elements of this model, and occasionally from other ones for addressing the proposed questions. Table 5 establishes a more specific evaluation context by presenting a detailed description of the training and test sets for the model 18_{10}. In this table, we divided the information items in the same manner as in Table 2. When looking at the first segment, we can observe that in both sets there are a huge number of distinct locations, leading the model to consider a huge amount of symbols. Another interesting information supporting the initial affirmation, and consistent for both sets, it is the different number of locations that can be on a trajectory and the high average value of the number of locations by trajectory. Moving on to the next segment, we observe data related to the geospatial resolution. Here, as a consequence of the locations considered, we have a huge sum of HTM trixels considered by this model in both sets. The third segment shows the timestamp period considered on both sets, which has a difference of a few days. Temporal marks that we include in the symbols respect these period boundaries. Lastly, we have a huge amount of symbols used for training and testing, since they are a combination of HTM trixels with temporal marks. The most interesting information in Table 5 is the percentage of the new HMM symbols in the test set. This percentage implies that the model has to classify nearly 18% of locations that were not present in the training trajectories.

Besides presenting results related to classification accuracy, we can also analyze other elements, such as the categories that are typically visited in sequence within the trajectories. Figure 6 illustrates the transition probabilities between states. Considering the small average number of categories by trajectory, together with the fact that we had a large chunk of trajectories where there exists only one category, the question of popular transitions between activities could not be addressed in a clear form. Transitions between different types of activities were rarely found.

Figures 7 and 8 address the question about which categories are most popular, at particular geographic regions or at particular temporal intervals. The bar plots are organized in a top-down approach concerning the maximum number of visits by category, where each bar plot illustrates the results for a corresponding category. Note that, in each bar plot, the dark bars represent the HMM classification results and the remaining colored bars represent the annotated data in the heuristic classification approach. Therefore, Figure 7 presents the number of visits per day, where Food is the most popular category and the less popular category is Residence. Figure 7 also shows two interesting behavior patterns, namely the peak of activity in the period between 9 am and 2 pm, and the less data available in the period from 3 pm to 10 pm. Another pattern can be found involving the Nightlife Spot, Outdoors and Recreation and Arts and Entertainment category-state pairs. When observed, the classification peaks are comparable and that could indicate possible classification errors. Notice that the periods from 2 to 3 am and from 5 to 7 am present an equal pattern on Arts and Entertainment and on Nightlife Spot. Also, the Nightlife Spot, Outdoors and Recreation

<table>
<thead>
<tr>
<th>Geospatial Resolution</th>
<th>16</th>
<th>16_{10}</th>
<th>18</th>
<th>18_{10}</th>
<th>19</th>
<th>19_{10}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>42.9</td>
<td>48.2</td>
<td>43.3</td>
<td>56.5</td>
<td>36.9</td>
<td>52.5</td>
</tr>
<tr>
<td>Micro Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision (%)</td>
<td>42.9</td>
<td>48.2</td>
<td>43.3</td>
<td>56.5</td>
<td>36.4</td>
<td>52.6</td>
</tr>
<tr>
<td>Recall (%)</td>
<td>42.9</td>
<td>48.2</td>
<td>43.3</td>
<td>56.5</td>
<td>36.4</td>
<td>52.6</td>
</tr>
<tr>
<td>Macro Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision (%)</td>
<td>45.0</td>
<td>42.6</td>
<td>54.6</td>
<td>49.8</td>
<td>56.4</td>
<td>47.5</td>
</tr>
<tr>
<td>Recall (%)</td>
<td>39.0</td>
<td>51.5</td>
<td>42.7</td>
<td>56.9</td>
<td>39.1</td>
<td>52.9</td>
</tr>
<tr>
<td>Correctly =100%</td>
<td>30.7</td>
<td>32.2</td>
<td>24.4</td>
<td>32.7</td>
<td>26.1</td>
<td>29.1</td>
</tr>
<tr>
<td>Classified (%) ≥ 90%</td>
<td>31.5</td>
<td>33.0</td>
<td>32.2</td>
<td>38.3</td>
<td>29.0</td>
<td>35.6</td>
</tr>
<tr>
<td>Trajectories ≥ 80%</td>
<td>36.3</td>
<td>39.5</td>
<td>37.8</td>
<td>46.9</td>
<td>33.4</td>
<td>41.6</td>
</tr>
<tr>
<td>Trajectories ≥ 70%</td>
<td>41.5</td>
<td>45.9</td>
<td>43.2</td>
<td>52.7</td>
<td>37.4</td>
<td>45.1</td>
</tr>
</tbody>
</table>

Table 3: Results of different supervised experiments.
Table 4: Results for each of the individual HMM states.

<table>
<thead>
<tr>
<th>Geospatial Resolution</th>
<th>16</th>
<th>16₁₀</th>
<th>18</th>
<th>18₁₀</th>
<th>19</th>
<th>19₁₀</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nightlife Spot</strong></td>
<td>Precision (%)</td>
<td>49.5</td>
<td>18.7</td>
<td>70.6</td>
<td>17.5</td>
<td>73.5</td>
</tr>
<tr>
<td></td>
<td>Recall (%)</td>
<td>11.0</td>
<td>53.8</td>
<td>18.8</td>
<td>55.1</td>
<td>19.2</td>
</tr>
<tr>
<td></td>
<td>F1 (%)</td>
<td>18.0</td>
<td>27.8</td>
<td>29.7</td>
<td>26.6</td>
<td>30.4</td>
</tr>
<tr>
<td><strong>Arts and Entertainment</strong></td>
<td>Precision (%)</td>
<td>47.3</td>
<td>36.3</td>
<td>55.7</td>
<td>42.8</td>
<td>53.3</td>
</tr>
<tr>
<td></td>
<td>Recall (%)</td>
<td>31.8</td>
<td>50.2</td>
<td>32.4</td>
<td>51.2</td>
<td>28.6</td>
</tr>
<tr>
<td></td>
<td>F1 (%)</td>
<td>37.3</td>
<td>42.1</td>
<td>41.0</td>
<td>46.6</td>
<td>37.2</td>
</tr>
<tr>
<td><strong>Shop and Services</strong></td>
<td>Precision (%)</td>
<td>54.8</td>
<td>51.1</td>
<td>69.2</td>
<td>64.7</td>
<td>75.3</td>
</tr>
<tr>
<td></td>
<td>Recall (%)</td>
<td>34.2</td>
<td>38.5</td>
<td>37.2</td>
<td>51.2</td>
<td>30.9</td>
</tr>
<tr>
<td></td>
<td>F1 (%)</td>
<td>42.1</td>
<td>43.9</td>
<td>48.4</td>
<td>57.2</td>
<td>43.8</td>
</tr>
<tr>
<td><strong>Food</strong></td>
<td>Precision (%)</td>
<td>61.3</td>
<td>61.5</td>
<td>74.3</td>
<td>71.1</td>
<td>78.3</td>
</tr>
<tr>
<td></td>
<td>Recall (%)</td>
<td>45.6</td>
<td>46.1</td>
<td>40.6</td>
<td>56.5</td>
<td>31.1</td>
</tr>
<tr>
<td></td>
<td>F1 (%)</td>
<td>52.3</td>
<td>52.7</td>
<td>52.5</td>
<td>63.0</td>
<td>44.5</td>
</tr>
<tr>
<td><strong>Outdoors and Recreation</strong></td>
<td>Precision (%)</td>
<td>27.9</td>
<td>38.6</td>
<td>9.4</td>
<td>50.8</td>
<td>7.6</td>
</tr>
<tr>
<td></td>
<td>Recall (%)</td>
<td>42.3</td>
<td>58.0</td>
<td>67.5</td>
<td>58.3</td>
<td>75.1</td>
</tr>
<tr>
<td></td>
<td>F1 (%)</td>
<td>33.6</td>
<td>46.3</td>
<td>16.5</td>
<td>54.3</td>
<td>13.7</td>
</tr>
<tr>
<td><strong>College and University</strong></td>
<td>Precision (%)</td>
<td>22.8</td>
<td>44.3</td>
<td>40.2</td>
<td>42.0</td>
<td>43.4</td>
</tr>
<tr>
<td></td>
<td>Recall (%)</td>
<td>81.0</td>
<td>76.5</td>
<td>72.6</td>
<td>79.1</td>
<td>66.4</td>
</tr>
<tr>
<td></td>
<td>F1 (%)</td>
<td>35.6</td>
<td>56.1</td>
<td>51.8</td>
<td>54.9</td>
<td>52.5</td>
</tr>
<tr>
<td><strong>Residence</strong></td>
<td>Precision (%)</td>
<td>37.3</td>
<td>29.2</td>
<td>49.8</td>
<td>31.0</td>
<td>49.3</td>
</tr>
<tr>
<td></td>
<td>Recall (%)</td>
<td>30.3</td>
<td>52.5</td>
<td>33.2</td>
<td>54.7</td>
<td>30.0</td>
</tr>
<tr>
<td></td>
<td>F1 (%)</td>
<td>33.5</td>
<td>37.5</td>
<td>39.8</td>
<td>39.6</td>
<td>37.3</td>
</tr>
<tr>
<td><strong>Professional and Other Places</strong></td>
<td>Precision (%)</td>
<td>54.1</td>
<td>50.7</td>
<td>67.8</td>
<td>61.8</td>
<td>73.0</td>
</tr>
<tr>
<td></td>
<td>Recall (%)</td>
<td>36.1</td>
<td>42.8</td>
<td>36.3</td>
<td>51.4</td>
<td>30.1</td>
</tr>
<tr>
<td></td>
<td>F1 (%)</td>
<td>43.3</td>
<td>46.4</td>
<td>47.3</td>
<td>56.1</td>
<td>42.6</td>
</tr>
<tr>
<td><strong>Travel and Transport</strong></td>
<td>Precision (%)</td>
<td>50.2</td>
<td>53.1</td>
<td>54.3</td>
<td>66.9</td>
<td>53.6</td>
</tr>
<tr>
<td></td>
<td>Recall (%)</td>
<td>39.3</td>
<td>45.5</td>
<td>46.0</td>
<td>54.8</td>
<td>40.2</td>
</tr>
<tr>
<td></td>
<td>F1 (%)</td>
<td>44.1</td>
<td>49.0</td>
<td>49.8</td>
<td>60.3</td>
<td>46.0</td>
</tr>
</tbody>
</table>

Realizing that there is a strong relation between the timestamp and the activity performed by a person, our matching method uses the considered time features (i.e., Day of the week and Hour of the day) to identify the category (i.e., the information that represents the context of an activity) that best represents a state’s classified locations.

Regarding classification errors, as Figure 7 illustrates and as we can then confirm through Table 7, the model 18₁₀ confused the category **Professional and Other Places** with **College and University**. Moreover, Table 7 reveals that **Food** visits have an important weight on classification errors, but a higher rate of incorrect classifications involves the **Nightlife Spot**, **Residence** and **College and University** categories.

4.1.2 Results with Unsupervised HMMs

When using an unsupervised approach, one does not consider the semantic information for training the model. This way, the categories have no direct relation with the states, and thus we cannot directly compare HMM states against categories, to evaluate the performance of our models.

In order to measure the performance of our unsupervised models, we choose to train them considering 9 different states for two obvious reasons. First, we want to be consistent and compare this approach with the supervised one, which considered an equal number of states due to the number of available categories. Second, by considering 9 states we can then build a relatively simple method that solves this problem by matching each category to one of the model’s states.

Realizing that there is a strong relation between the timestamp and the activity performed by a person, our matching method uses the considered time features (i.e., Day of the week and Hour of the day) to identify the category (i.e., the information that represents the context of an activity) that best represents a state’s classified locations.

Our matching method to identify the category-state pairs uses a two-step algorithm. Using the time features, the method first calculates all possible similarities between categories and states, and it then proposes a map of category-state pairs.

Initially, we have 4 matrices containing the number of locations, that relate each considered time feature with the categories or states (e.g., State by Day of the week matrix). Our matching method uses the Root Mean Square Deviation measure to look for the similarity between a category and a state. With this measure, we compare each state to all categories and build a matrix containing insights into how, through the time analysis, the locations were previously annotated and how the unsupervised model classified them. The equations bellow show how we adapt this measure to the time features:

\[
RSMD_{\text{Day of the week}} = \sqrt{\frac{\sum_{\text{day}=1}^{7} (\hat{y}_{\text{day}} - \hat{c}_{\text{day}})^2}{7}}
\]

\[
RSMD_{\text{Hour of the day}} = \sqrt{\frac{\sum_{\text{hour}=1}^{24} (\hat{y}_{\text{hour}} - \hat{c}_{\text{hour}})^2}{24}}
\]
In order to adapt the Root Mean Square Deviation measure to the time features, when evaluating the difference between a category \( y \) and a state \( y \), we start by observing the respective Day of the week matrices. For each day, the two \( \text{Day of the week} \) matrices respectively contain the number of locations classified as belonging to a given state \( y_{\text{day}} \) and annotated with a certain category \( y \). As Equation 15 shows, we calculate the difference associated with the days of the week for the given category \( y \) and \( y \). Regarding the two elements that we are evaluating, and for normalization purposes, we also keep the maximum and minimum number of locations, respectively, the \( \text{day}_{\text{max}} \) and \( \text{day}_{\text{min}} \). Afterwards, we repeat the same calculations for the hours of the day, as presented in Equation 16. Finally, as shown in Equation 17, we obtain the balanced difference between a category \( y \) and \( y \), which will be considered in the second step of our matching method.

After getting the differences between each state and all categories, and after building the matrix holding these values, our method identifies the matching state-category pairs. One starts by looking for the minimum value within the matrix. When the method finds this value, it identifies and associates as a pair, both the corresponding state and category. Until each state has a unique corresponding category, our method iterates over the remaining part of the matrix, that is, it does not consider both the line and the column corresponding to the elements that previously formed a pair. The rational behind this choice is to avoid cases, where two distinct categories could relate with the same state. This would happen due to the dataset’s characteristics, since the amount of locations associated with each category is very different.

Similarly to what was performed for the supervised approach, we experimented with six different unsupervised HMMs, and with two of them for each data characterization (i.e., geospatial configuration). Table 4.1.2 presents the obtained results for these experiments. Although the results from the unsupervised experiments had a worst performance when compared to the supervised ones, they confirm that the smoothing strategy works when considering larger areas in our models. In these results, we observe again that the HMM model with the geospatial resolution of \( 18_{10} \) is the most suited one, scoring 33.5% of accuracy, and the HMM with geospatial resolution of \( 19_{10} \) is the less accurate one. The models that obtain worst results are the ones with higher geospatial resolution parameter (i.e., \( 19 \)).

### Table 5: The training and testing datasets used in the case of model \( 18_{10} \).

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Training set</th>
<th>Testing set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectories</td>
<td>120 837</td>
<td>40 280</td>
</tr>
<tr>
<td>Locations</td>
<td>2 779 076</td>
<td>937 599</td>
</tr>
<tr>
<td>Distinct Locations</td>
<td>2 628 632</td>
<td>917 030</td>
</tr>
<tr>
<td>Locations by Trajectory</td>
<td>2 to 1 243</td>
<td>2 to 1 001</td>
</tr>
<tr>
<td>( \mu ) Locations/Trajectory</td>
<td>22.99</td>
<td>23.28</td>
</tr>
<tr>
<td>( \sigma ) Locations/Trajectory</td>
<td>47.24</td>
<td>47.75</td>
</tr>
<tr>
<td>( \mu ) Categories/Trajectory</td>
<td>2.54</td>
<td>2.55</td>
</tr>
<tr>
<td>( \sigma ) Categories/Trajectory</td>
<td>2.06</td>
<td>2.06</td>
</tr>
</tbody>
</table>

\[
\text{Difference} = \left( \frac{\text{RSMD}_{\text{day of the week}}}{\text{day}_{\text{max}} - \text{day}_{\text{min}}} \times 0.5 \right) + \left( \frac{\text{RSMD}_{\text{hour of the day}}}{\text{hour}_{\text{max}} - \text{hour}_{\text{min}}} \times 0.5 \right)
\]

5. CONCLUSIONS

In this paper, I proposed and compared different approaches for the semantic classification of visited locations within human trajectories. My approaches consist on a heuristic method, and two others methods based on Hidden Markov Models (HMMs), either relying on supervised or unsupervised settings. I also surveyed the current state-of-art of in terms of different techniques and methods to solve similar classification or prediction problems. From the experiments reported in this document, I can enumerate three main conclusions:

First, I argue that the GeoLife dataset is not directly suitable for the evaluation of this task, since data collection, instead of being directed to the activities, was temporally oriented (e.g., with devices collecting data every 5 seconds or 500 meters). Therefore, and specially for the unsupervised setting, for achieving more accurate results one would need more and different data collected with identical time features.

The most accurate model is the one whose area range is more diverse, in order to cover different locations where different activities can occur. Also, through the analysis of Figures 7 and 8, I realize that taking into account the temporal marks as unobservable parameters was not a relevant choice.
Table 6: Results of different unsupervised experiments.

<table>
<thead>
<tr>
<th>Geospatial Resolution</th>
<th>16</th>
<th>16_{1D}</th>
<th>18</th>
<th>18_{1D}</th>
<th>19</th>
<th>19_{1D}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>21.6</td>
<td>24.1</td>
<td>21.7</td>
<td>33.5</td>
<td>20.0</td>
<td>19.0</td>
</tr>
<tr>
<td>Micro Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision (%)</td>
<td>21.6</td>
<td>24.1</td>
<td>21.7</td>
<td>33.5</td>
<td>20.0</td>
<td>19.0</td>
</tr>
<tr>
<td>Recall (%)</td>
<td>21.6</td>
<td>24.1</td>
<td>21.7</td>
<td>33.5</td>
<td>20.0</td>
<td>19.0</td>
</tr>
<tr>
<td>Macro Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision (%)</td>
<td>14.2</td>
<td>11.7</td>
<td>11.7</td>
<td>4.7</td>
<td>7.5</td>
<td>9.2</td>
</tr>
<tr>
<td>Recall (%)</td>
<td>11.8</td>
<td>11.5</td>
<td>11.3</td>
<td>11.1</td>
<td>11.0</td>
<td>11.0</td>
</tr>
<tr>
<td>Correctly ≥ 100%</td>
<td>6.6</td>
<td>9.0</td>
<td>0.8</td>
<td>9.4</td>
<td>1.6</td>
<td>0.09</td>
</tr>
<tr>
<td>≥ 90%</td>
<td>6.7</td>
<td>9.1</td>
<td>1.0</td>
<td>9.7</td>
<td>3.2</td>
<td>0.09</td>
</tr>
<tr>
<td>≥ 80%</td>
<td>7.7</td>
<td>10.0</td>
<td>1.6</td>
<td>11.1</td>
<td>5.3</td>
<td>0.09</td>
</tr>
<tr>
<td>≥ 70%</td>
<td>9.4</td>
<td>10.9</td>
<td>2.2</td>
<td>12.9</td>
<td>6.4</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Table 7: Confusion matrix of the supervised model 18_{1D} concerning the visit classifications (i.e., the proposed states) and their categories.

As a final conclusion, I can assert that for achieving good results in this classification task, there is no need for using highly sophisticated sequential data models. Once by adding a novel smoothing strategy to a well-known approach, we have that the results improved 13.2% in the accuracy of the most accurate model, and we observed a global maximum of 15.6% on another model considering a smaller area size.

An interesting idea for future work relates to addressing the same classification task with discriminative models, such as Conditional Random Fields (CRFs) or Support Vector Machines (SVM). I would also like to experiment with higher-order HMMs, which can take into account not only the previous state but multiple previous states.

Finally, I would also like to test our approaches with datasets that contain different characteristics or regions. This way, I could study and compare distinct human or cultural behavior patterns, and the geographic impact on them.

Acknowledgments

I have a lot to thank my advisor, Prof. Bruno Emanuel da Graça Martins, for the constant support, comprehension and availability in these months of work. I want also to thank
Figure 6: Probabilities associated to the supervised model 1810.

him for the opportunity to participate on the Services for Intelligent Geographical Information Systems (SInteliGIS) project.

This work was supported by the Fundação para a Ciência e a Tecnologia (FCT), through the project grant with reference PTDC/EIA-EIA/109840/2009 (SInteliGIS).

6. REFERENCES


Figure 7: Number of visits per hour for each category.

Figure 8: Number of visits per day for each category.