Automatic detection of user trajectories from social media posts

Loris Belcastro, Fabrizio Marozzo*, Emanuele Perrella

University of Calabria, Rende, 87036, Italy

Abstract

Social media represents a rich environment to collect huge amounts of data containing useful information about people's behaviors and interactions. In particular, such information has been widely exploited for analyzing the mobility of people, as geotagged social media posts allow to extract accurate patterns on movements of people. This paper presents AUDESOME (*AUtomatic Detection of user trajEctories from SOcial MEdia*), an automatic method for discovering user mobility patterns from social media posts. In particular, the method includes two new unsupervised algorithms: (*i*) a text mining algorithm, which analyzes social media posts to automatically extract the main keywords identifying the Places-of-Interest (PoI) in a given area; and (*ii*) a geospatial clustering algorithm, which detects the Regions-of-Interest (RoIs) by using both geotagged posts and extracted keywords. We experimentally evaluated the performance of AUDESOME taking into account the following aspects: identification of keywords, detection of RoIs, and extraction of user trajectories. The experiments, performed on a real dataset containing about 3 million of geotagged items published in Flickr, demonstrate that AUDESOME achieves better results than existing techniques.

Keywords: Trajectory mining, RoI mining, Social media analysis, Geospatial clustering, Keyword extraction

1. Introduction

The huge volume of data generated by users on social media, such as Facebook, Twitter and Flickr, can be exploited to extract useful information concerning people's behaviors and interactions (Talia et al., 2015). For example, such data has been used to analyze the collective sentiments of people, understand the behavior of groups of users during global events, or monitor public opinion

*Corresponding author

Email addresses: lbelcastro@dimes.unical.it (Loris Belcastro), fmarozzo@dimes.unical.it (Fabrizio Marozzo), eperrella@dimes.unical.it (Emanuele Perrella)

close to important events. In addition, social media posts are widely exploited for analyzing the mobility of people. In fact, such posts are often tagged with geographical coordinates or other information that allow to discover users' positions and trajectory patterns (Zheng, 2015a).

This paper presents AUDESOME (*AUtomatic Detection of user trajEctories from SOcial MEdia*), an automatic method aimed at discovering user mobility patterns from posts shared on social media. The method takes as input a set of geotagged posts published by social media users (e.g., a set of Flickr photos or tweets) and performs the following operations: (*i*) *keyword extraction*, which automatically extracts the keywords identifying the Places-of-Interest (PoIs) that are located in a given area; (*ii*) *RoI detection*, which detects the Regions-of-Interest (RoIs) in the area, starting from the extracted keywords (for each PoI) and geotagged posts; and (*iii*) *trajectory mining*, which discovers frequent mobility patterns in user trajectories across RoIs. Since the goal is to create an automatic method, we defined two unsupervised algorithms for both keyword extraction and RoI detection through text mining and geospatial clustering techniques.

We experimentally evaluated the performance of AUDESOME taking into account the following aspects: identification of keywords, detection of RoIs and extraction of user trajectories. The experiments, performed on a real dataset containing 3.1 million of geotagged items published in Flickr in the areas of Rome and Paris, demonstrate that AUDESOME achieves better results than existing techniques. For example, on the Rome's dataset our technique achieved very good F_1 scores: *0.84* for the keyword extraction process, *0.79* for the RoI detection, and *0.85* for the trajectory mining. Our method has been compared to the most relevant techniques used in the literature, by achieving a significant improvement of the F_1 score in each step of our method. We also evaluated the execution time on a private cloud infrastructure, using an Apache Spark cluster equipped with 64 CPU cores and 100 GB of memory. In such a context, the total execution time decreases from 10 minutes using 8 cores to 2.5 minutes using 64 cores. For the purpose of using the code of our method and allowing the reproducibility of the experiments, an open-source version of AUDESOME is available at `https://github.com/SCAlabUnical/AUDESOME`.

This paper extends the work presented in Belcastro et al. (2020) in the following main aspects:

(i) The data analysis workflow has been redesigned as a three-step method that returns, as a final result, no longer the RoIs, but the trajectories of users.

(ii) Two accurate text mining and clustering algorithms have been proposed for the keyword extraction and RoI detection operations, which are executed in an unsupervised and almost automatic way. Furthermore, the detailed description and the pseudo-code of such algorithms have been added.

(iii) A way to evaluate the performance of each operation that composes our method (keyword extraction, RoI detection, trajectory mining) has been discussed, proposing an uniform definition of the used metrics (precision, recall, F_1) for the different operations and explaining how they can be

calculated.

(iv) For all the phases of our method, an extensive comparison with the main algorithms existing in the literature has been carried out.

The main novel contributions of AUDESOME with respect to the other techniques can be outlined as follows. We propose a custom implementation of DBSCAN (adapted for RoI mining) and a new heuristic for automatically estimating its parameters, which lead to more accurate results than other existing techniques. Also for keyword extraction, our algorithm is able to find the most representative keywords for the different areas of interest. Most existing techniques rank keywords for each area and establish a fixed threshold (e.g., top-10) to choose the most representative ones. Instead, our algorithm dynamically chooses a variable number of keywords for each area by taking into account both the importance of a keyword in that area and its frequency. Overall, the combination of such novel algorithms leads to better results in extracting user trajectories than existing techniques.

The structure of the paper is as follows. Section 2 introduces the main concepts and problem statements. Section 3 discusses related work. Section 4 describes the method proposed in this paper. Section 5 presents the case studies and performance evaluation. Finally, Section 6 concludes the paper.

2. Problem definition

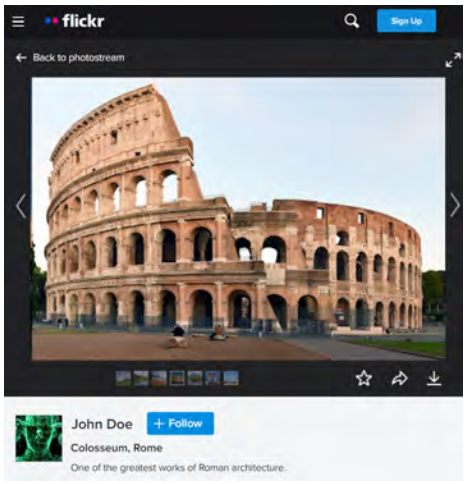
Before discussing the proposed method in detail, it is important to clarify the meaning of some terms that are used throughout the paper, such as Point-of-Interest, Region-of-Interest, and trajectory.

A *Point-of-Interest* or *Place-of-Interest* (*PoI*) is a location that someone finds useful or interesting, such as tourist attractions or business locations. A *Region-of-Interest* (*RoI*) represents the geographical boundaries of the PoI's area (de Graaff et al., 2013), but it is also be defined as "a spatial extent in geographical space where at least a certain number of user trajectories pass through" (Giannotti et al., 2007). A *trajectory* is a sequence of spatial regions (RoIs) followed by a user. Consequently, a *frequent trajectory* is a sequence of spatial regions that emerge as frequently visited by users.

In most cases, data used for extracting trajectories comes from GPS sensors/devices, which provide, at regular intervals, the coordinates of locations where the devices are located. As an example, Yuan et al. Yuan et al. (2011a) analyzed a large dataset containing real-world GPS trajectories generated by 30,000 taxis in Beijing. Although not originally intended for this purpose, also data from social media can be used for extracting trajectories. In fact, *social media items* (or *posts*) are often *geotagged*, which means they contain geographical coordinates or other information that allow to discover the user's position when a post was created. Specifically, a social media item g includes the following fields:

- *text*, containing a textual description of g.

- *tags*, containing is a set of keywords associated with g.
- *coordinates*, consisting of *latitude* and *longitude* of the place from where g was created.
- *userId*, identifying the user who created g.
- *timestamp*, indicating date and time when g was created.



(a) Photo and description.


```
{"id":"2703840000",
 "owner":{ "id":"11235813@N27",
   "username":"John Doe",
   "pro":true, ... }
 "title":"Colosseum, Rome",
 "description":"One of the greatest
   works of Roman architecture.",
 "dateTaken":"2021-14-27T23:25:01",
 "datePosted":"2021-14-27T23:30:36",
 "geoData":{"longitude":12.492373,
    "latitude":41.890251,
    "accuracy":16},
 "tags":["colosseum","rome",
   "italy", "nikon"],
 "originalFormat":"jpg",
 "hasPeople":false,
 "comments":27, "views":270384,
}
```


(b) JSON metadata.

Figure 1: An example of a Flickr post and its metadata.

For example, Figure 1 shows a Flickr post and the metadata that can be extracted from it. As shown, the post contains some specific metadata of the photo, such as the date on which it was taken and posted, the coordinates of the place where it has been taken, the format, a flag indicating if the user is a professional photographer, and so on. The metadata associated with the user's posts can be used to discover his/her movements, but also to obtain useful information for identifying the places visited. Thus, analyzing and aggregating the posts of groups of people, it is possible to find the most visited places and most frequent routes. In addition, textual information contained in such posts allows to discover the names and/or keywords used to refer to a place, the reasons for the visit, the opinion of the users on a place.

AUDESOME is designed to process geotagged social media posts by exploiting the metadata they contain to extract user trajectories. The input of our method is a large set of geotagged items gathered from a social media platform (e.g., Flickr and Twitter). AUDESOME is a three-step method for analyzing the metadata of social media posts in order to extract: the *keywords* that identify the most visited places (*PoIs*), the *areas* of such places (*RoIs*), and the most

frequent user trajectories crossing them (*frequent trajectories*). More details are provided in Section 4.

3. Related work

Geographical coordinates and other information used for identifying user positions are often analyzed to discover frequent patterns in user trajectories. In this context, a sequential pattern is a sequence of places (PoIs) visited by users in a similar time interval (Zheng, 2015b). The analysis of frequent sequential patterns in user trajectories is highly valuable in many scenarios, such as: tourism agencies and municipalities can discover the most visited tourist attractions and routes (Cesario et al., 2016); transport operators can discover the places and routes where it is more likely to serve passengers (Yuan et al., 2011b) and crowded areas where more transport facilities need to be allocated (Altomare et al., 2017).

Since AUDESOME is a three-step method for analyzing social media posts, a review of the state-of-the-art in the fields of keyword extraction, RoI mining and trajectory mining is presented in the next sections. Moreover, the main differences with the most related works are discussed.

3.1. Trajectory mining

Giannotti et al. (2007) addressed the problem of trajectory pattern mining under many aspects, providing formal definitions, as well as proposing approaches and algorithms for the different phases of the analysis. In particular, the authors presented a new form of space-time pattern, called *T-pattern*, which represents the set of trajectories that visit the same sequences of places with similar travel times. This pattern formalizes the idea of aggregating user trajectories, not only in terms of sequences of places visited, but also in terms of travel times between them. Specifically, the user trajectories in input are transformed from sequences of points into sequences of RoIs by labeling data during a pre-processing step. Bermingham & Lee (2019) proposed a framework, namely STOSEM, which exploits a Hidden Markov Model (HMM) to convert GPS trajectories into sequences of real-world places from OpenStreetMap. In particular, the framework aims at addressing ambiguity issues that happen when a user position intersects multiple places.

Differently from frequent pattern mining, which applies on trajectories in a similar time interval, the *repetitive pattern mining* applies on trajectories that are repeated at regular time intervals (e.g., day or season). Discovering periodic patterns is really complex, since the moving object does not visit exactly the same location and the period does not exactly have the same value in different occurrences (Mazimpaka & Timpf, 2016). In such a context, Cao et al. (2007) used a predefined period for discovering frequent regions through clustering. Then, frequent regions have been aggregated for discovering periodic patterns in spatio-temporal sequences. Many real data, such as GPS trajectories, are hierarchically structured, but also irregularly sampled (e.g., due to weather conditions or

device malfunctions, interrupted GPS service) and without a spatial semantic meaning. To address these issues, Zhang et al. (2019) proposed an hierarchical clustering method for discovering periodic patterns from irregularly sampled spatio-temporal trajectories. Čavojský et al. (2020) evaluated the Needleman-Wunsch algorithm to address several issues in comparing trajectories due to errors and gaps in GPS data. Pan et al. (2016) proposed a multidimensional trajectory clustering strategy for discovering regular behaviors by considering characteristics such as position, velocity and course.

Other works focused on mining periodic patterns from social media data. Halder et al. (2017) proposed *SPPMiner*, a graph-based mining algorithm for extracting periodic patterns from large volumes of social media posts, which has been used for identifying recurring interactions between users in dynamic social networks, but not for extracting frequent trajectories. Also Cesario et al. (2017) proposed *SMA4TD* (*Social Media Analysis for Trajectory Discovery*), a methodology aimed at discovering behavior rules, correlations and periodic mobility patterns of visitors attending large-scale events, through the analysis of a large number of social media posts. Such a technique has been exploited to analyze Instagram posts for discovering mobility patterns inside the exhibition area, correlations among visits to pavilions and the main flows of origin/destination of visitors (Cesario et al., 2016).

Zhijun et al.Yin et al. (2011) proposed a method for ranking trajectories extracted from geotagged posts published on Flickr, without focusing on mining frequent trajectory patterns. In particular, the authors proposed an algorithm that defines places by clustering post coordinates and extracts trajectories using the PrefixSpan (Pei et al., 2004) algorithm.

3.2. RoI detection

Existing techniques for finding RoIs are based on three main approaches: *predefined shapes*, *density-based clustering* and *grid-based aggregation*.

In the first approach, *predefined shapes* (e.g., circles, rectangles, etc.) are used to define and represent RoIs. For example, Kisilevich et al. (2010a) define RoIs as circles of fixed radius centered on a set of PoIs whose center coordinates are known. Cesario et al. (2015) used rectangles to define RoIs representing stadiums for a trajectory mining study. de Graaff et al. (2013) used Voronoi tessellations to define RoIs starting from a set of geographical coordinates representing PoIs.

In the *density-based clustering* approach, RoIs are obtained by aggregating a set of geographical points or locations. For instance, Altomare et al. (2017) used DBSCAN for detecting regions that are more densely visited based on data from GPS-equipped taxis. Kisilevich et al. (2010b) used a variant of DBSCAN, named P-DBSCAN, to cluster photos taking into account the neighborhood density (i.e., the number of distinct photo owners in the neighborhood) and exploiting the notion of adaptive density for fast convergence towards high density regions. Belcastro et al. (2020) proposed a parallel clustering approach, namely ParCA, to identify RoIs from spatial dataset. ParCA exploits a parallel execution of DBSCAN on subsets of data to generate sub-clusters on each processing node and then merge overlapping sub-clusters to form global clusters.

The *grid-based aggregation* approach discretizes the area under analysis in a regular grid and extracts RoIs by aggregating the grid cells. For example, Lee et al. (2014) argued that rectangular expansion produces RoIs that may contain uninteresting low-density cells. For this reason, they proposed a hybrid grid-based algorithm, called Slope RoI, to mine arbitrary RoI shapes from trajectory data. Shi et al. (2014) mapped geotagged data into grid cells, and then grouped the cells taking into account spatial proximity and social relationship between places. Spyrou et al. (2017) proposed an algorithm that divides a geographical area into cells and then exploits an iterative merging procedure for finding RoIs.

Density- and grid-based algorithms depend on the setting of multiple parameters (e.g., *eps* and *minNumPoints* for DBSCAN, *cellSize* and *minSupport* for Slope RoI). For this reason, it is not easy to find parameters that produce accurate RoIs over different locations with a variety of shapes and data points distributions. To overcome the problem of adjusting the parameters, some automatic clustering techniques have been proposed, such as DSets-DBSCAN (Hou et al., 2016) and TURN∗ (Foss & Zaïane, 2002). Although they demonstrate to work well in certain circumstances, in other cases they lead to poor results because clusters are rarely well-separable in an explicable manner.

Other approaches differ from those mentioned earlier. G-RoI (Belcastro et al., 2018) exploits the indications contained in geotagged social media items (e.g. tweets, posts, photos or videos with geospatial information) to discover the RoI of a PoI with a high accuracy. Starting from a set of predefined keywords identifying a PoI, *G-RoI* iteratively calculates the associated RoI using a density-based criterion.

3.3. Keyword extraction

RoI mining approaches need a method to assign a meaning to each RoI found. There are different ways to perform this task. Yin et al. (2011) assigned a name to each cluster by taking the most frequent keywords in the geotagged items. Ferrari et al. (2011) automatically associated with each RoI the zip code of the data points in the cluster center. Järv et al. (2018) used a hierarchical clustering algorithm, named HDBSCAN, for producing a dendrogram of clusters. Each place is represented as a natural hierarchy of other regions (e.g., a museum may belong to a particular district). Then, using both a PoI database from Foursquare and metadata contained in geotagged items, the authors assigned a semantic mean to each region.

Most used techniques for extracting representative keywords are based on TF-IDF (Ramos et al., 2003), an information retrieval algorithm for finding the most relevant keywords and topics in clusters or subsets of data (Ahern et al., 2007; Zhu et al., 2019). Some researchers exploited TF-IDF to assign a semantic meaning to RoIs obtained through clustering of geotagged social media posts ((Spyrou et al., 2017; Hu et al., 2015)). Specifically, TF-IDF has been used for ranking keywords/tags contained in each cluster, so as to find the most representative ones (e.g., choosing the top-N).

3.4. Main differences with with the most related works

AUDESOME is a three-step method for analyzing the metadata of social media posts in order to extract: the *keywords* that identify the most visited places (*PoIs*), the *areas* of such places (*RoIs*) and most frequent user trajectories crossing them (*frequent trajectories*).

It is worth noting that our work does not propose a new trajectory mining algorithm, but defines a full automatic method for extracting trajectories from social media data, which combines algorithms of keyword extraction, RoI detection, and trajectory mining. Thus, AUDESOME does not place any constraints on the trajectory mining algorithm to be used. In our experiments, once user trajectories across RoIs have been calculated, we used a parallel implementation of PrefixSpan (Pei et al., 2004) for extracting the most frequent ones.

From the above literature review, some works use similar approaches, but with some important differences:

- SMA4TD (Cesario et al., 2017) does not deal with keyword extraction, as the RoIs (e.g., areas of the 2014 FIFA World Cup stadiums) are manually defined before analyzing the movement of users. Differently, we propose a clustering algorithm with automatic parameter estimation that is able to automatically identify RoIs with arbitrary shapes.

- SPPMiner (Halder et al., 2017) is a graph-based mining algorithm for extracting periodic patterns from large volumes of social media posts, which has been used for identifying recurring interactions between users in dynamic social networks. However, as stated by the authors, it needs further development for supporting the extraction of trajectories.

- Slope RoI (Lee et al., 2014) discretizes the area under analysis in a regular grid and extracts RoIs by aggregating the grid cells using a heuristic based on density variation. This approach has the problem of finding parameters that produce accurate RoIs over different locations with a variety of shapes and data points distributions. It does not propose any algorithm for assigning a meaning to RoIs or for extracting trajectories.

- Spyrou et al. (2017) presented a RoI mining algorithm that splits the area under analysis into squared cells of equal size and, for each cell, finds the top-10 representative keywords using TF-IDF. The RoIs are obtained by aggregating adjacent cells using the Jaccard similarity measure Stork et al. (2001) calculated between the keyword sets of each cell. This technique is not able to dynamically determine a cut value for the keywords ranked by TF-IDF, nor to detect the presence of multiple RoIs within the same cell. AUDESOME aggregates posts using sets of keywords and, therefore, it is able to define RoIs that can fall within one or more cells, i.e., the extraction of RoIs is totally decoupled from the concept of cell (see Section 4.2).

- G-RoI (Belcastro et al., 2018) proposes a clustering algorithm with a top/down approach for the extraction of RoIs, but it does not deal with

the extraction of keywords, which are provided manually. This limits the applicability of G-RoI, since it requires to know a priori the PoIs and their related keywords. Instead, AUDESOME is able to automatically extract the most relevant keywords from data by exploiting an ad-hoc text-mining algorithm. Moreover, AUDESOME uses a custom-version of DBSCAN, which natively turns out to be very robust in handling data with noise and more accurate in detecting RoIs with different shapes and densities. From the computational point of view, the time complexity of G-RoI is $\mathcal{O}(n^3 \log n)$, while our implementation has a lower complexity of $\mathcal{O}(n^2)$.

- ParCA (Belcastro et al., 2020) exploits a TF-IDF-based approach to extract the most relevant keywords from cells, but does not provide any pseudo-code, description or performance evaluation of the algorithm. Concerning the RoI detection process, ParCA calculates the RoIs through a parallel implementation of DBSCAN, where the *eps* parameter is calculated by considering a fixed percentage of noise in data, as suggested in Ester et al. (1996). This approach is not very effective (Schubert et al., 2017), since it is not able to adapt to RoIs with different shapes, densities and percentage of noise.

Summarizing, the main novel contributions of AUDESOME with respect to the other techniques can be outlined as follows. AUDESOME proposes a custom implementation of DBSCAN (adapted for RoI mining) and a new heuristic that allows to automatically estimate its parameters. Compared to existing techniques used for estimating the parameters of DBSCAN, i.e., fixed noise percentage, L-Curve and DSets-DBSCAN (Hou et al., 2016), our heuristic permits to obtain more accurate RoIs than other existing techniques (e.g., G-RoI, ParCA, and Slope RoI). In fact, the heuristic proposed in AUDESOME dynamically calculates the threshold point (i.e, the point that separates the noise points from those assigned to some clusters) for each RoI, so as to calculate effective clustering parameters for each RoI. Also for keyword extraction, our algorithm it is able to find the most representative keywords for the different areas of the interest. Existing techniques based on TF-IDF rank all the keywords and establish a fixed threshold (e.g., top-10) to choose the most representative ones. Instead, our algorithm is able to dynamically choose this threshold by exploiting both the number and frequency of the keywords found. Also in this case, experiments demonstrated that our approach is able to obtain better results than existing techniques (e.g., top-N TF-IDF). Overall, the combination of such novel algorithms leads to better results in extracting user trajectories than existing techniques.

4. Proposed method

The proposed method performs the following three macro steps (see Figure 2):

- *Keyword extraction*: a text mining algorithm is exploited to identify the main keywords that are used in a certain area. Such keywords, which

identify the PoIs that are located in the area, are used to group social media posts according to the place they refer to.

- *RoI detection*: a clustering algorithm is used to extract RoIs starting from social media posts that have been grouped by keywords. Specifically, the social media posts referring to a PoI are transformed into a series of geographical points (we only keep the coordinates). Such points are then appropriately clustered to define RoIs.

- *Trajectory mining*: starting from the RoIs defined at the previous step, the social media posts are analyzed to extract the trajectories of each user, and thus to obtain the mobility patterns of users.

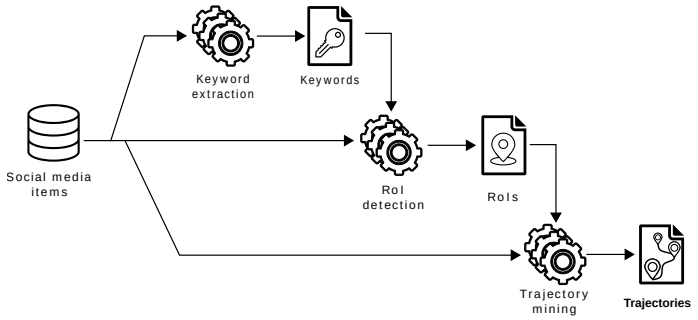


Figure 2: Workflow of AUDESOME.

Overall, the analysis process produces the following intermediate/final outputs: (*i*) the sets of keywords that are used to refer the PoIs in the area under analysis; (*ii*) the RoIs calculated for the different PoIs, which can also be easily displayed on a map; (*iii*) the most frequent trajectories followed by users.

Figure 3 shows an example of outputs produced by AUDESOME in an area of Rome. In particular, starting from the geotagged social media posts generated by four users ($U1$, ..., $U4$), the method identifies five Places-of-Interest and the related keywords (e.g., $\langle$ Colosseum, Colosseo, Coliseo, Colaseum, Coloseu, ...$\rangle$ for the Colosseum). From such places, the associated RoIs are generated to understand whether a user has visited or not a certain PoI. For each user, we can extract his/her movements across places, grouping social media posts by user id and sorting them by date and time. As an example, Figure 3 shows the trajectory of user $U1$: *Colosseum* $\rightarrow$ *Roman Forum* $\rightarrow$ *Circus Maximus* $\rightarrow$ *Tiber Island*. Through the analysis of the trajectories of different users we can discover the most frequent behaviors and trajectories.

In the next two subsections we will describe in detail the novel algorithms that we have proposed for extracting keywords and defining RoIs. For each of them, a formal description and practical examples are provided. Table 1 reports the meaning of the main symbols used to define the pseudo-code of these two algorithms. Concerning trajectory mining, the user's trajectories are transformed

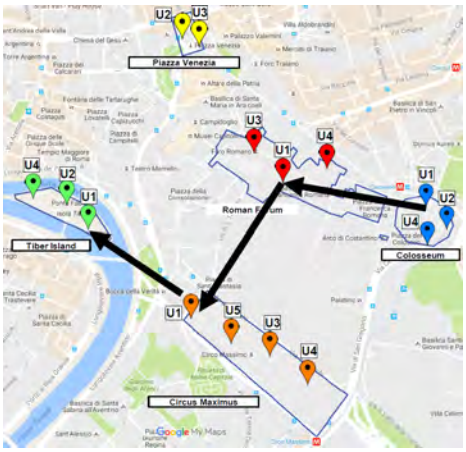


Figure 3: An example of the outputs produced by AUDESOME.

from sequences of coordinates into sequences of RoIs. For this reason, the quality of the RoIs influences the quality of the trajectories obtained.

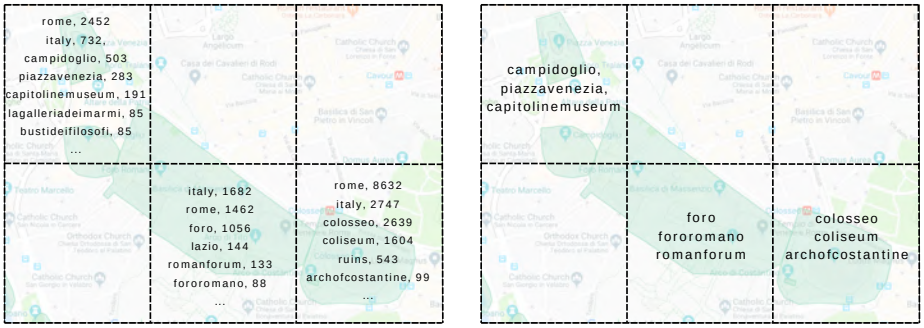
Table 1: Meaning of the symbols used in AUDESOME.

Symbol	Meaning
G	Set of ⟨geotagged items⟩
C	Set of ⟨square cells⟩ representing a subdivision of the area under analysis
O	Map⟨cell, map of ⟨term, occurrences⟩⟩: for each cell it stores the occurrences of terms belonging that cell
L	Map of ⟨term, set of ⟨cells⟩⟩: for each term, it stores the square cells that contain that term
S	Map of ⟨cell, map of ⟨term, score⟩⟩: for each cell, it stores the score of the terms belonging to that cell
K	Sets of ⟨keywords⟩
P	Map of ⟨PoI, set of ⟨geotagged items⟩⟩: it stores the geotagged items that have been associated with each PoI
CL	Set of ⟨clusters⟩, where each cluster is composed of a set of geographical points
R	Map of ⟨PoI, RoI⟩: for each PoI a Region-of-Interest
D	Array of ⟨distances⟩

4.1. Keyword extraction

This algorithm is used to automatically extract the keywords that identify the main PoIs located in a given area. Overall, the algorithm can be divided in three parts: (*i*) the area under analysis is divided into cells and, for each cell, the main keywords (and their occurrence) are extracted by analyzing the geotagged posts whose coordinates fall into that cell; (*ii*) an heuristic based on the L-curve (Hansen, 1992) algorithm is exploited to separate high and low-representative keywords automatically; and (*iii*) the Levenshtein metric (Levenshtein, 1966) is used to group the most representative keywords based on similarity.

Figure 4 shows an example of how the keyword extraction algorithm works. For each cell, the algorithm extracts the keywords contained in the geotagged posts falling into that cell, and calculates their frequency. Then, such keywords are sorted by frequency. A high frequency does not necessarily denote high quality representative keywords, but it is a useful starting point. As an example, in Figure 4(a), the keywords "italy" and "rome" have higher frequency than "colosseum" and "romanforum", although the latter are more representative of the PoIs contained in those cells. Noisy keywords, such as "italy" or "rome", are then removed (see Figure 4(b)) so as to maintain only the most representative keywords for each cell. Then, in Figure 4(c) the remaining keywords are grouped according to their textual similarity, such as "{colosseo, coliseum}".



(a) Keyword discovering.

(b) Keywords extraction.

(c) Keywords grouping.

Figure 4: An example of how the keyword extraction algorithm works.

Algorithm 1 shows the pseudo-code for automatically extracting keywords. The input is composed of a set of geotagged items G and a set of cells C that represents a subdivision of the area under analysis in cells. The output is a set of keywords K that have been found in G. As the first step (lines 1-2), two maps are initialized: O, which stores the occurrences of the terms that have been found in a cell, and L, which stores the list of cells that contain a specific term.

For each geotagged items $g_i \in G$, the algorithm iterates by performing the

following operations (lines 3-7):

- using the coordinates of g_i, it gets the cell c_i where g_i falls into (line 4);
- for each term t_j contained in the textual information of g_i (i.e., title, description and tags), it increases the term occurrence in $O[c_i]$ and adds the cell c_i to $L[t_j]$ (lines 5-7). In particular, during this phase, textual information is preprocessed so as to remove duplicates, normalize all the terms by transforming them in lowercase, and remove stop words using preset lists.

A method based on a discrete L-curve (Hansen, 1992) is exploited for distinguishing between high and low-occurrence keywords in O (line 8). Using the *findThresholdLCurve* method, the items in O are sorted in descending order by number of occurrences, so as to create a bi-dimensional curve composed of an ordered set of Cartesian points $XY = \{(0, Occ_0), (1, Occ_1), ..., (n, Occ_n)\}$. A generic element $p_i \in XY$ represents a term t_i and is graphed as a point (i, Occ_i), where i and Occ_i indicate the index and number of occurrences of t_i, respectively. Since for two consecutive points p_i and p_{i+1} it is always true that $Occ_i \geq Occ_{i+1}$, the curve turns out to be an L-curve. From this curve, the elbow point $E = (e, Occ_e)$ is calculated. The ordinate of E (i.e., Occ_e) is the threshold th_1 used for separating high and low-occurrence terms. A term $t_i = (i, Occ_i)$ is considered as a high-occurrence if $Occ_i \geq th_1$, otherwise as low-occurrence (i.e., $Occ_i < th_1$). Then, all the terms whose occurrences are lower than th_1 are filtered out (line 9).

A new map S is initialized for storing the score assigned to each term that has been discovered in the set of cells (line 10). For calculating such a score, the algorithm iterates over O, which is a set of pairs $F_i = \langle$term, occurrences$\rangle$ containing the occurrences of each term. In particular, the following operations are performed (lines 11-15):

- Calculate the score of each term t_j in the cell c_i by using TF-IDF (line 13). Specifically, this score is calculated by considering the occurrences of the term (o_j), the sum of the occurrences of all terms in c_i, and the number of cells in which the term is present $L[t_j]$.
- A method based on a discrete L-curve is then exploited for distinguishing between high- and low-score terms in the cell c_i (lines 14-15). Specifically, a local threshold for the cell th_2 is calculated for all the terms in c_i (line 14) and all the terms whose score is lower than th_2 are removed in $S[c_i]$ (line 15).

Then, a new set K is initialized for storing the most representative terms (i.e., keywords) in the area under analysis (line 16). For each cell of S, the algorithm iterates on each pair $S_i = \langle$term, score$\rangle$ to find the keywords and their synonyms (lines 17-20). In particular, all similar keywords are grouped into a set k_j (e.g., *colosseum*, *coliseum* and *coliseo*) that is added to K.

ALGORITHM 1: Keyword extraction.

Input : Set of ⟨geotagged items⟩ G, set of ⟨square cells⟩ C
Output: Sets of ⟨keywords⟩ K

$O \leftarrow \emptyset$; /* Map of <cell, map of <term, occurrences>> */
$L \leftarrow \emptyset$; /* Map of <term, set of <cells>> */
for $g_i \in G$ **do**
  $c_i \leftarrow$ findCell(C, $g_i.coordinates$); /* Cell c_i where g_i falls into */
  for $t_j \in terms(g_i)$ **do**
    $O[c_i][t_j]$++; /* Increase the occurrences of the term t_i for c_i */
    $L[t_j] \leftarrow L[t_j] \cup c_i$; /* Add the cell c_i to $L[t_j]$ */
$th_1 \leftarrow$findThresholdLCurve(O); /* Calculate the threshold th_1 */
deleteLowTh1Terms(O, th_1); /* Delete low occurrence terms from O */
$S \leftarrow \emptyset$; /* Map of <cell, map of <term, score>> */
for $< c_i, F_i > \in O$ **do**
  for $< t_j, o_j > \in F_i$ **do**
    $S[c_i][t_j] \leftarrow$ score(o_j, sum($O[c_i]$), $|L[t_j]|, S$); /* Calculate the score of t_i in c_i */
  $th_2 \leftarrow$findThresholdLCurve($S[c_i]$); /* Calculate the threshold th_2 */
  deleteLowTh2Terms($S[c_i], th_2$); /* Delete terms with low scores */
$K \leftarrow \emptyset$; /* Sets of <keywords> */
for $< c_i, S_i > \in S$ **do**
  for $< t_j, s_j > \in S_i$ **do**
    $k_j \leftarrow <$ t_j, synonyms(t_j, O) $>$; /* Group t_i and its synonyms */
    $K \leftarrow K \cup \{k_j\}$; /* Add the keyword k_j to K */
return K

Once the keyword sets in K (i.e., the output of the algorithm) have been generated, they are used to associate social media posts to the set they refer to. In particular, given a set of keywords $K_i \in K$, a geotagged item g_i is assigned to K_i if it contains at least one keyword of K_i. It should be noticed that each set of keywords represents a PoI. As the next step, a map P is used to store the geotagged items associated with each PoI. This map is given as input to the RoI detection algorithm.

4.2. RoI detection

The RoI detection algorithm aims at defining Regions-of-Interest from the geotagged items assigned to the different PoIs. As discussed, geotagged items can be transformed into geographical coordinates (i.e., pairs of ⟨latitude, longitude⟩) that can be aggregated through clustering. Specifically, a customized version of DBSCAN (Ester et al., 1996) has been used, so as to make it suitable for RoI mining and able to set its input parameters in an almost automatic way.

Algorithm 2 shows the pseudo-code for detecting RoIs. The input is a map P, which contains the set of geotagged items associated with each PoI, and a threshold value $th \in (0, 1)$. The output is a map R used to store a Region-of-Interest for each PoI. At the beginning, R is initialized (line 1) and the DBSCAN parameter $minPts$ is set to 5, as usual when working two dimensional

data (Schubert et al., 2017) (more details in Section 4.2.1). The algorithm iterates (lines 3-8) on each PoI p_i in P by performing the following operations:

- Extract the set of coordinates CC_i from the geotagged items G_i associated with p_i (line 4). CC_i is obtained by extracting the coordinates (latitude/longitude) from each geotagged item in G_i.

- Estimate eps_i by using the set of coordinates CC_i. Specifically, eps_i is estimated by using the method *calculateEps*, which receives as parameters CC_i, $minPts$, and th (line 5). Choosing a good *eps* value allows to correctly separate noise points from those that can be assigned to some clusters.

- Execute DBSCAN on CC_i by using $minPts$ and eps_i, generating a set of clusters CL (line 6). Among them the cluster containing the highest number of points c_{max} is selected (line 7).

- Calculate the Region-of-Interest associated with p_i as the convex polygon that contains all the points of the cluster c_{max} (line 8).

In the next section, we explain how the *eps* parameter is estimated.

ALGORITHM 2: RoI detection.

Input : Map of ⟨PoI, set of ⟨geotagged items⟩⟩ P, threshold $th \in (0, 1)$
Output: Map of ⟨PoI, RoI⟩ R

1 $R \leftarrow \emptyset$; /* Map of <PoI, RoI> */
2 $minPts \leftarrow 5$; /* minPts is fixed to 5 */
3 **for** $< p_i, G_i > \in P$ **do**
4 $CC_i \leftarrow$ getCoordinates(G_i) ; /* Coordinates of G_i */
5 $eps_i \leftarrow$ calculateEps$(CC_i, minPts, th)$; /* Calculate eps */
6 $CL \leftarrow$ DBSCAN$(CC_i, minPts, eps_i)$; /* Set of ⟨clusters⟩ */
7 $c_{max} \leftarrow$max(CL); /* Get the cluster with the highest number of points */
8 $R[p_i] \leftarrow$convexHull(c_{max}); /* Generate a RoI for p_i */
9 **return** R

4.2.1. Heuristic for choosing DBSCAN parameters

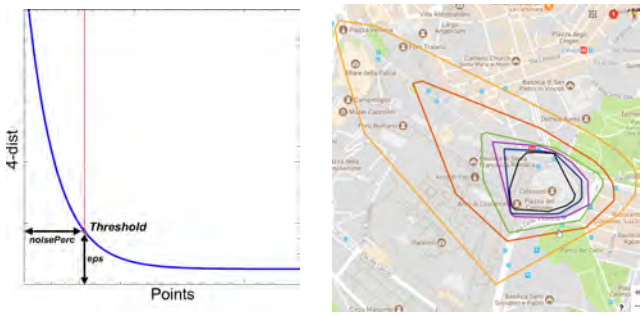
The following parameters must be set to run the DBSCAN algorithm: *eps*, the radius of the neighborhood of a point; and $minPts$, the minimum number of points in a neighborhood that are required to form a cluster. As defined in Ester et al. (1996), such parameters can be calculated using the following procedure:

1. Compute the k-nearest-neighbor distances (k-dist) of all the points and sort them in descending order. As proposed by Ester et al. (1996), the value of k can be set to 4 for bi-dimensional data, while $minPts$ can be set to $k + 1$ (Schubert et al., 2017) thus equal to 5.

2. Choose a *threshold point* (also called "cut-off" point) on the k-distance plot to isolate the noise points. Specifically, if a point has a k-distance value higher than that of the threshold point, it is considered noise and thus discarded; otherwise, it is assigned to some clusters.

To calculate the threshold point, Ester et al. (1996) proposed to plot the k-distance and find the first "elbow" (or "valley"). Such an elbow represents the threshold point, and its k-distance value is used as *eps* value.

Figure 5(a) shows an example of a k-distance plot (with k=4) where the threshold point (*eps*) and percentage of noise points (*noisePerc*) are highlighted. As shown, setting a threshold point is equivalent to defining the noise percentage and vice versa. It should be noted that choosing an accurate noise percentage not only permits to discard noise points, but also to define more accurate RoIs. For example, Figure 5(b)) shows how choosing the noise percentage affects the definition of the RoI of the Colosseum. With a low noise percentage (e.g., 5% for the yellow polygon) we obtain a too large RoI, with medium noise percentage (e.g., 20% for the purple polygon) a quite accurate RoI, with high noise percentage (e.g., 30% for the black polygon) a too small RoI.



(a) Example of sorted k-distance plot (k=4).

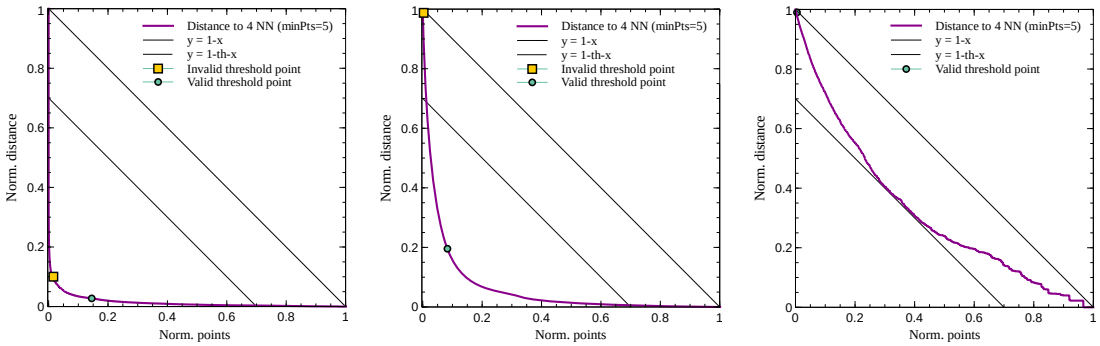
(b) Colosseum's RoIs for different values of *noisePerc*.

Figure 5: How the percentage of noise (*noisePerc*) affects the definition of RoIs (from 5% for the yellow polygon to 30% for the black polygon).

As clearly explained in a recent survey on DBSCAN (Schubert et al., 2017), it is not always easy to identify the threshold point, as it is often not clearly visible in the k-distance plot. Our algorithm analyzes the k-distance plot trying to estimate a good threshold point through an iterative process. In particular, it discards the noise points (i.e., those with higher k-distance values) until the k-distance plot calculated on the remaining points no longer shows an evident elbow point.

Figure 6 shows the k-distance plot in three different iteration phases. Figure 6(a) shows the initial k-distance plot, i.e., a L-curve with an evident elbow. As the points with the highest k-distance value are discarded, the curve flattens more and more. In Figure 6(b), despite having eliminated several noise points, the curve still shows an evident elbow. Finally, in Figure 6(c) the curve has flattened enough to stop the iteration process. Specifically, to understand when the curve is flat enough we use a threshold value th. If all the points are above the line $y = 1 - th - x$, the iteration is stopped (more details below).

Algorithm 3 shows the pseudo-code used to estimate the *eps* parameter. The



(a) Initial k-distance plot. (b) Intermediate k-distance plot. (c) Final k-distance plot.

Figure 6: Procedure for choosing a threshold point.

input is composed of a set of geographical points P, a minimum number of points $minPts$, and a threshold value $th \in (0, 1)$. The output is the estimated value of eps.

At the beginning, the sorted k-distance is calculated on P and stored in an array D (lines 1-5). Given two adjacent points $D[i] = (i; d_i)$ and $D[i + 1] = (i + 1; d_{i+1})$, the distance d_i is strictly greater than d_{i+1}, because the distances are sorted in descending order. Then, the index of the cut-off point cut is set to zero (line 6) and the variable n is set to the number of points (line 7).

The cut-off point is found through an iterative process. Given the index cut of the current cut-off point $D[cut] = (cut, d_{cut})$, the algorithms verifies if all the points in $[D[cut], D[n]]$ are placed above the threshold line $y = 1 - th - x$ (i.e., within a threshold distance th from the line $y = 1 - x$). To understand this, on each point $D[i] \in [D[cut], D[n]]$, the following operations are performed (lines 10-15):

- Normalize $D[i].x$ with respect to $[D[cut].x, D[n].x]$ (line 11);
- Normalize $D[i].y$ with respect to $[D[cut].y, D[n].y]$ (line 12);
- If the normalized point (x_{norm}, y_{norm}) is below the threshold line $y = 1 - th - x$ (line 13), it means that the plot has an elbow. Thus, the variable $isFlat$ is set to false and the inner iteration is stopped (lines 13-15).

If the inner iteration is never broken, i.e., all the points in $\{D[cut], ..., D[n]\}$ are above the line $y = 1 - th - x$ (i.e., $isFlat$ is true), the algorithm returns the distance of the cut-off point $D[cut]$ as eps value (lines 16-17). Otherwise, the algorithm evaluates the next cut-off point (lines 18-19).

The complexity of AUDESOME in detecting RoIs is $\mathcal{O}(n^2)$. In fact, estimating the eps parameter has a complexity of $\mathcal{O}(n^2)$, while the execution of DBSCAN can be reduced to $\mathcal{O}(n \log n)$ by using indexed data structures. Since the two phases are performed in sequence, the final complexity results to be $\mathcal{O}(n^2)$.

ALGORITHM 3: Eps estimation.

Input : Set of ⟨coordinates⟩ P, minimum numbers of points $minPts$, threshold $th \in (0, 1)$
Output : $Eps \in (0, +\infty)$.

$D \leftarrow \emptyset$; /* Array of ⟨distances⟩ */
for ($p_i \in P$)
    $kdist \leftarrow KDistance(p_i, P, minPts)$; /* Compute KNN distances for p_i */
    $D[i] \leftarrow kdist$; /* Set value of p_i in distances set */
$Sort(D)$; /* Sort all k-distances by descending values */
$cut \leftarrow 0$; /* Index of the cut-off point */
$n \leftarrow |P|$; /* Number of points */
while $cut < n$ **do**
    $isFlat \leftarrow true$; /* Variable to understand if the k-dist plot is flat */
    for ($i = cut + 1;\ i < n - 1;\ i$++)
        $x_{norm} = (D[i].x - D[cut].x)/(D[n].x - D[cut].x)$; /* Compute normalized x distance */
        $y_{norm} = (D[i].y - D[n].y)/(D[cut].y - D[n].y)$; /* Compute normalized y distance */
        if $y_{norm} < 1 - th - x_{norm}$ **then**
            $isFlat \leftarrow false$; /* k-dist plot is not flat */
            break; /* Break cycle */
    if $isFlat$ **then**
        return $D[cut]$; /* Return D[cut] as eps value */
    else
        cut++; /* Update index of cut-off point */

5. Performance Evaluation

We experimentally evaluated the performance and scalability of AUDESOME using more than 3 million of geotagged items, published in Flickr from January 2006 to May 2020 in the cities of Rome and Paris. Specifically, we evaluated the following steps that are part of the AUDESOME's workflow: (*i*) extraction of keywords; (*ii*) detection of RoIs; and (*iii*) mining of frequent trajectories. In each step, we carried out a comparison with the main existing techniques in the literature.

5.1. Accuracy

The performance of the AUDESOME's workflow has been evaluated by using the precision and recall metrics. To rank the results, we combined precision and recall using the F_1 score (i.e., the harmonic mean between precision and recall). In the following, we provide a description of how such metrics have been calculated for each step.

5.1.1. Keyword Extraction

We evaluated the performance of AUDESOME in extracting keywords, comparing it with the TF-IDF algorithm (Ramos et al., 2003). The area under

analysis is divided into a set of square cells C of the same size. In particular, after several preliminary experiments, for our case studies we have chosen a cell size of 200 square meters, which is the same value used by Spyrou et al. (2017).

Let $K_{real}(c)$ be the real keywords used to refer the PoIs located in a cell $c \in C$, which can be collected using APIs provided by public services such as Foursquare[1] or GeoNames[2]. Let also $K_{found}(c)$ be the keywords found in c by a keyword extraction algorithm. The precision $Prec_K$ and recall Rec_K of the keyword extraction process are then defined as:

$$Prec_K = \frac{1}{|C|} \sum\nolimits_{c \in C} \frac{|K_{real}(c) \cap K_{found}(c)|}{|K_{found}(c)|} \tag{1}$$

$$Rec_K = \frac{1}{|C|} \sum\nolimits_{c \in C} \frac{|K_{real}(c) \cap K_{found}(c)|}{|K_{real}(c)|} \tag{2}$$

We measured the performance of AUDESOME in extracting the keywords that are most representative of the PoIs contained in each cell's area. Then, we compared such results with those obtained by using different configurations of TF-IDF. It should be noticed that TF-IDF is not able to automatically calculate the number of keywords to be returned, which requires to be manually defined (e.g., return the top N frequent keywords in a cell). Differently, AUDESOME implements an heuristic, based on L-Curve, that allows to automatically estimate the number of keywords to be returned for each cell. For better assessing results, we compared AUDESOME with different configurations of TF-IDF (i.e., top 2, top 5, and top 10 frequent keywords for each cell). Also for AUDESOME, we evaluated the effectiveness of the heuristic used to estimate the number of keywords, comparing it with others based on a fixed cut (i.e., top 2, top 5, and top 10 frequent keywords in each cell).

Table 2: Comparison of keyword extraction algorithms.

Algorithm	Rome			Paris		
	$Prec_K$	Rec_K	F_{1K}	$Prec_K$	Rec_K	F_{1K}
TF-IDF Top-2	0.87	0.87	0.81	0.63	0.61	0.62
TF-IDF Top-5	0.48	0.48	0.52	0.30	0.60	0.40
TF-IDF Top-10	0.24	0.57	0.34	0.28	0.70	0.40
AUDESOME Top2	0.97	0.70	0.81	0.65	0.57	0.61
AUDESOME Top5	0.35	0.64	0.45	0.21	0.67	0.32
AUDESOME Top10	0.34	0.76	0.47	0.27	0.58	0.37
AUDESOME	*0.95*	*0.75*	***0.84***	*0.80*	*0.56*	***0.66***

Table 2 shows the results obtained. In particular, TF-IDF requires a strong cut on the number of keywords to be considered for achieving good results. In fact, considering the top 2 frequent keywords per cell, TF-IDF reaches a F_1 score

[1] https://foursquare.com/
[2] https://www.geonames.org/

of 0.81 in Rome and 0.62 in Paris, while choosing an higher number of keywords per cell (top-5 or top-10), the performance of TF-IDF considerably decreases. Differently, AUDESOME achieved a high quality results using both the approach based on an automatic keyword cut (F_1 score 0.84 in Rome and 0.66 in Paris) and that based on a fixed cut (e.g., F_1 score 0.81 in Rome and 0.61 in Paris with the top 2 approach). Compared to the other techniques, our method leads to an overall improvement of the F_1 score up to 0.26, which is calculated as the difference between the F_1 score obtained by AUDESOME and the worst one of the other techniques (i.e., TF-IDF Top-10). These results show that our algorithm, being able to automatically select a variable number of keywords for each cell, allows to obtain better results than the other techniques.

5.1.2. RoI detection

We compared the performance of AUDESOME in detecting RoIs with four existing techniques: *DBSCAN* (Zheng et al., 2012), *DSets-DBSCAN* (Hou et al., 2016), Slope (Lee et al., 2014), and *G-RoI* (Belcastro et al., 2018). In particular, we used P, two sets of 24 popular PoIs in Rome and Paris, whose real RoIs are known (they can be obtained from online services like OpenStreetMap[3]) and characterized by different shapes and sizes.

Let $R_{real}(p)$ be the real RoI of a generic PoI $p \in P$, $R_{found}(p)$ be the RoI of p found by a RoI detection algorithm, and $A(r)$ the area of a generic RoI r. We define the precision $Prec_R$ and recall Rec_R of the RoI detection process as:

$$Prec_R = \frac{1}{|P|} \sum_{p \in P} \frac{A(R_{real}(p) \cap R_{found}(p))}{A(R_{found}(p))} \tag{3}$$

$$Rec_R = \frac{1}{|P|} \sum_{p \in P} \frac{A(R_{real}(p) \cap R_{found}(p))}{A(R_{real}(p))} \tag{4}$$

Also in this case, the results are ranked by using the F_1 score. As an example, Figure 7 shows the real RoI (green polygon) and that found by a clustering algorithm (red polygon) for the Colosseum. As shown, the RoI found only partially matches the real one, which influences the values of precision and recall.

Figure 7: A comparison between real and found RoI for the Colosseum.

[3]https://www.openstreetmap.org

In the following, we describe how the different techniques involved in the comparison have been configured:

- For *DBSCAN*, a noise threshold has been used so that all the points with a k-distance value higher than it are considered noise. Specifically, we evaluated two configurations: (i) a fixed noise percentage (i.e., 20%), as used in Belcastro et al. (2020); and (ii) a noise percentage estimated by using the L-Curve approach described in Section 4.2.1.
- *DSets-DBSCAN* is able to generate clusters of arbitrary shapes without having to set any parameter input.
- *Slope* requires two parameters: the *cell size* and *minimum support*. After several preliminary tests to find parameter values that perform effectively in all the scenarios, the square cell side has been set to 55 meters and the minimum cell support to 150.
- *G-RoI* requires as input parameter a threshold (i.e., a distance value between 0 and 1), which has been set to 0.27.
- *AUDESOME* requires a threshold parameter, which has been set to 0.27.

It should be noted that only G-RoI and AUDESOME have been designed to return a single cluster (RoI), while the other techniques can find more than one cluster. When more clusters are found, the cluster containing the higher number of points is chosen. Then, the Region-of-Interest is calculated as the convex polygon that contains all the points of the chosen cluster.

Table 3 illustrates the performance (average values of precision, recall, and F_1 score) of the different techniques, calculated on all the PoIs considered in Rome and Paris. DBSCAN achieves acceptable results with the fixed noise threshold (F_1 score of 0.67), while using the L-Curve does not obtain accurate results (F_1 from 0.11 to 0.26). Also DSets-DBSCAN does not bring accurate results, with a F_1 score ranging from 0.44 to 0.49. In fact, it produces a very high precision with a low recall, which means that the RoIs identified by the technique are too small compared to the real ones. Slope obtains slightly better results, with a mean F_1 score of about 0.60. G-RoI obtained good results in both Rome and Paris, with a mean F_1 score of 0.79 and 0.73 respectively. Finally, AUDESOME outperformed all the other techniques by obtaining a F_1 score of 0.79 in Rome and 0.75 in Paris. Thus, our method achieved an overall improvement of the F_1 score up to 0.64 (i.e., in comparison to DBSCAN with L-Curve). These results clearly show that our algorithm is able to define RoIs with different shapes and sizes with great precision, obtaining better results than that of existing techniques.

5.1.3. Trajectory mining

We have experimentally evaluated the performance of AUDESOME in the extraction of user trajectories. In particular, the user's trajectories are transformed from sequences of coordinates into sequences of RoIs. For this reason, the quality of the RoIs influences the quality of the trajectories obtained. In our evaluation,

Table 3: Comparison of RoI detection algorithms.

Algorithm	Rome			Paris		
	$Prec_R$	Rec_R	F_{1R}	$Prec_R$	Rec_R	F_{1R}
DBSCAN (fixed noise perc.)	0.72	0.63	0.67	0.69	0.66	0.67
DBSCAN with L-Curve	0.15	1.00	0.26	0.06	1.00	0.11
DSets-DBSCAN	0.93	0.29	0.44	0.90	0.34	0.49
Slope	0.69	0.56	0.62	0.59	0.62	0.60
G-RoI	0.78	0.82	**0.79**	0.81	0.66	0.73
AUDESOME	*0.75*	*0.84*	***0.79***	*0.72*	*0.79*	***0.75***

we compared the results obtained using the real RoIs with those obtained through the use of RoI detection algorithms. Specifically, the performance evaluation has been carried out on all the trajectories generated by users.

Let $S = \{s_1, s_2, ...\}$ be a set of sequences of geographical points, where $s_i = (\langle lat_1, lng_1\rangle, \langle lat_2, lng_2\rangle, ...)$ describes the movements of a user in a certain time interval (e.g., in one day). Given a set of RoIs R, each sequence $s_i \in S$ can be converted into a sequence of RoIs $t_i = (r_1, r_2, ...)$. In particular, a point $p_i = \langle lat_i, lng_i\rangle$ is assigned to a RoI r_i if p_i falls into the area of r_i. Otherwise, if p_i does not fall into the area of any RoI, it is not assigned to any RoI.

Then, given a sequence $s \in S$, let $T_{real}(s)$ be the trajectory obtained with the real RoIs, and $T_{found}(s)$ the trajectory obtained with the RoIs extracted by a RoI detection algorithm. The precision $Prec_T$ and recall Rec_T of the trajectory mining process are then defined as:

$$Prec_T = \frac{1}{|S|} \sum\nolimits_{s \in S} \frac{|T_{real}(s) \cap T_{found}(s)|}{|T_{found}(s)|} \tag{5}$$

$$Rec_T = \frac{1}{|S|} \sum\nolimits_{s \in S} \frac{|T_{real}(s) \cap T_{found}(s)|}{|T_{real}(s)|} \tag{6}$$

The intersection $T_{real}(s) \cap T_{found}(s)$ represents the PoIs that the two trajectories have in common.

Figure 8 shows an example of how both precision and recall are calculated. Given a set of real RoIs (circles in blue) and a set of computed RoIs (circles in red) for the PoIs $\{A, B, C, D\}$, the figure illustrates three sequences $\{s_1, s_2, s_3\}$ that describe the movements of different users across such RoIs. In particular, s_1 is converted into the trajectory $A \to B \to D$ only using the real RoIs and in $A \to B$ using the found RoIs. The sequence s_2 is converted into $A \to B \to C \to D$ using the real RoIs and $A \to B \to D$ using the found RoIs. Finally, the sequence s_3 is converted into $A \to B$ using the real RoIs and $A \to B \to C$ using the found RoIs. Then, in this example, the precision is 0.89 and the recall is 0.80.

Table 4 reports the performance of the different techniques during the trajectory mining step. The results confirm that AUDESOME outperformed the other techniques by reaching a mean F_1 score of 0.85 in Rome and 0.87 in Paris. For this step, our method turns out to be the most accurate one in finding

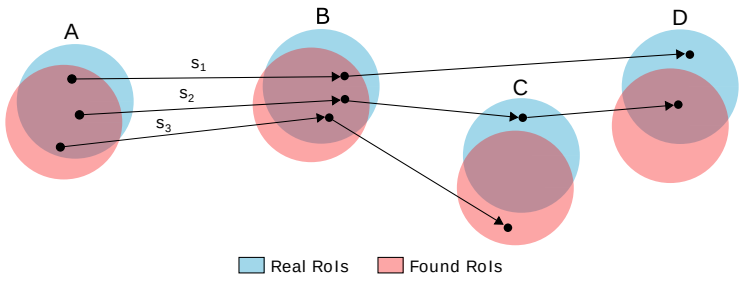


Figure 8: Example of trajectories that have been obtained using real and calculated RoIs.

trajectories, achieving an overall improvement of the F_1 score up to 0.52 (i.e., in comparison to DBSCAN with L-Curve).

Table 4: Comparison of trajectory mining algorithms.

Algorithm	Rome			Paris		
	$Prec_T$	Rec_T	F_{1T}	$Prec_T$	Rec_T	F_{1T}
Traj. + DBSCAN	0.81	0.83	0.82	0.81	0.82	0.82
Traj. + DBSCAN with L-Curve	0.59	0.61	0.60	0.35	0.35	0.35
Traj. + DSets-DBSCAN	0.49	0.43	0.46	0.71	0.67	0.69
Traj. + Slope	0.69	0.67	0.68	0.78	0.76	0.77
Traj. + G-RoI	0.82	0.84	0.83	0.85	0.84	0.84
Traj. + AUDESOME	*0.84*	*0.86*	***0.85***	*0.87*	*0.88*	***0.87***

5.2. Execution time

We evaluated the total execution time and scalability of our method by carrying out several experiments on a private Cloud. In particular, we used a cluster with Apache Spark, equipped with 64 CPU cores and 100 GB of memory. The goal is to assess the scalability of AUDESOME by varying both the number of CPU cores (from 8 to 64) and dataset size.

Starting from D_1, a dataset containing 1.2 million Flickr posts published in Rome and stored in a file of $1.5GB$, we applied a random sampling algorithm to produce four additional datasets D_2, D_4, D_6 and D_8. In particular, D_i contains i*1.2 million posts and it is stored in a file of i*1.5GB.

Figure 9 shows the execution times of the AUDESOME's workflow obtained using different configurations of datasets and number of CPU cores. To better understand the contribution of each workflow's step to the overall performance, the breakdown of the execution time is reported.

As shown, the execution time of each step (keyword extraction, RoI detection, trajectory mining) grows almost linearly with the dataset size. In particular, the keyword extraction is the dominant step that most influences the overall execution time, which takes on average half the overall time. However, increasing the number of cores leads to a significant reduction in overall execution times, which demonstrates the scalability of our method. In particular, increasing from

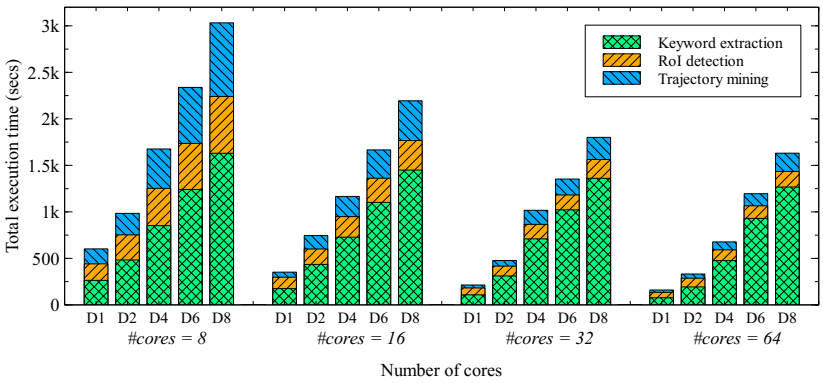


Figure 9: Execution time vs number of cores.

8 to 64 cores, the overall execution time decreases from 10 minutes to 2.5 minutes for the smallest dataset (D_1), from 16.4 to 5.5 minutes for D_2, from 28 to 11 minutes for D_4, from 39 to 20 minutes for D_6. For the largest dataset (D_8), the total execution time decreases from 50 to 27 minutes. It can be noticed that the RoI detection and trajectory mining steps are the parts whose algorithms (implemented in Apache Spark) turn out to be more optimized and scalable, which lead to the greatest time savings. In conclusion, the experimental results demonstrated the effectiveness of our method, in terms of both quality of results and scalability.

6. Conclusion

The widespread use of social media can be exploited to extract useful information concerning people's behaviors and interactions. In this paper, we presented AUDESOME (*AUtomatic Detection of user trajEctories from SOcial MEdia*), an automatic method for discovering user mobility patterns from social media posts. In particular, the method includes two new unsupervised algorithms: (*i*) a text mining algorithm, which analyzes social media posts to automatically extract the main keywords identifying the Places-of-Interest (PoI) in a given area; and (*ii*) a clustering algorithm, which detects the Regions-of-Interest (RoIs) by using geotagged posts and extracted keywords.

We experimentally evaluated the performance and scalability of AUDESOME using more than 3 million of geotagged items published in Flickr from January 2006 to May 2020 in the cities of Rome and Paris. The experiments demonstrated that AUDESOME achieves better results than existing techniques. Our method has been compared to the most relevant techniques used in the literature, by achieving an improvement of the F_1 score up to 0.26 in keyword extraction, 0.64 in RoI detection, and 0.52 in trajectory mining.

This study can provide an important contribution in the fields of automatic RoI and trajectory mining, especially when the areas of interest are not known

a priori or when they change over time (e.g., periodic or seasonable RoIs). Since such a method is designed to deal with social media data, it can enable to discover previously unknown places of interest, routes, and mobility patterns of users, which have a great value for many business sectors, such as transportation and tourism. For example: tourism agencies and municipalities can discover the most visited tourist attractions and routes; transport operators can discover the places and routes where it is more likely to serve passengers and crowded areas where more transport facilities need to be allocated.

Our technique is primarily designed to identify RoIs within urban areas. To properly work, it requires large amounts of textual and geotagged data, which are often difficult to obtain due to low use of some social media platforms in some areas of the world, or to imposed restrictions in data acquisition. Furthermore, since data collected from social media platforms could be unreliable, it is necessary to validate the results before applying them in decision-making processes. In particular, the statistical significance of the collected data should be demonstrated to evaluate the representativeness of them in the considered application context.

Future research efforts will be devoted to improving the RoI detection algorithm by involving other semantics in the clustering process, such as textual and image similarity. In such a way, it could be possible to (*i*) identify hidden RoIs that are not detected by spatial clustering based on geographical proximity; and to (*ii*) involve in the analysis social media items that are not geotagged, but which still contain information to extract useful knowledge on user behaviors.

Data and code availability statement

The data that support the findings of this study are publicly available. In particular, this data was gathered using Flickr APIs available at `https://www.flickr.com/services/api/`. For the purpose of using the code of our method, an open-source version of AUDESOME is available at `https://github.com/SCAlabUnical/AUDESOME` along with a small sample of Flickr posts and instructions for running tests.

References

Ahern, S., Naaman, M., Nair, R., & Yang, J. H.-I. (2007). World explorer: visualizing aggregate data from unstructured text in geo-referenced collections. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries* (pp. 1–10).

Altomare, A., Cesario, E., Comito, C., Marozzo, F., & Talia, D. (2017). Trajectory pattern mining for urban computing in the cloud. *IEEE Transactions on Parallel and Distributed Systems*, *28*, 586–599.

Čavojský, M., Drozda, M., & Balogh, Z. (2020). Analysis and experimental evaluation of the needleman-wunsch algorithm for trajectory comparison. *Expert Systems with Applications*, (p. 114068).

Belcastro, L., Kechadi, M., Marozzo, F., Pastore, L., Talia, D., & Trunfio, P. (2020). Parallel extraction of regions-of-interest from social media data. *Concurrency and Computation: Practice and Experience*, .

Belcastro, L., Marozzo, F., Talia, D., & Trunfio, P. (2018). G-roi: Automatic region-of-interest detection driven by geotagged social media data. *ACM Transactions on Knowledge Discovery from Data*, *12*.

Bermingham, L., & Lee, I. (2019). Mining place-matching patterns from spatio-temporal trajectories using complex real-world places. *Expert Systems with Applications*, *122*, 334 – 350. doi:`https://doi.org/10.1016/j.eswa.2019.01.027`.

Cao, H., Mamoulis, N., & Cheung, D. W. (2007). Discovery of periodic patterns in spatiotemporal sequences. *IEEE Transactions on Knowledge and Data Engineering*, *19*, 453–467.

Cesario, E., Congedo, C., Marozzo, F., Riotta, G., Spada, A., Talia, D., Trunfio, P., & Turri, C. (2015). Following soccer fans from geotagged tweets at FIFA World Cup 2014. In *Proc. of the 2nd IEEE Conference on Spatial Data Mining and Geographical Knowledge Services* (pp. 33–38). Fuzhou, China. ISBN 978-1-4799-7748-2.

Cesario, E., Iannazzo, A., Marozzo, F., Morello, F., Riotta, G., Spada, A., Talia, D., & Trunfio, P. (2016). Analyzing social media data to discover mobility patterns at expo 2015: Methodology and results. (pp. 230–237).

Cesario, E., Marozzo, F., Talia, D., & Trunfio, P. (2017). Sma4td: A social media analysis methodology for trajectory discovery in large-scale events. *Online Social Networks and Media*, *3-4*, 49–62.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* KDD'96 (pp. 226–231).

Ferrari, L., Rosi, A., Mamei, M., & Zambonelli, F. (2011). Extracting urban patterns from location-based social networks. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks* (pp. 9–16).

Foss, A., & Zaïane, O. R. (2002). A parameterless method for efficiently discovering clusters of arbitrary shape in large datasets. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* (pp. 179–186). IEEE.

Giannotti, F., Nanni, M., Pinelli, F., & Pedreschi, D. (2007). Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 330–339).

de Graaff, V., de By, R. A., van Keulen, M., & Flokstra, J. (2013). Point of interest to region of interest conversion. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* SIGSPATIAL'13 (pp. 388–391). New York, NY, USA: ACM.

Halder, S., Samiullah, M., & Lee, Y.-K. (2017). Supergraph based periodic pattern mining in dynamic social networks. *Expert Systems with Applications*, *72*, 430–442.

Hansen, P. C. (1992). Analysis of discrete ill-posed problems by means of the L-Curve. *SIAM Review*, *34*, 561–580.

Hou, J., Gao, H., & Li, X. (2016). Dsets-dbscan: A parameter-free clustering algorithm. *IEEE Transactions on Image Processing*, *25*, 3182–3193.

Hu, Y., Gao, S., Janowicz, K., Yu, B., Li, W., & Prasad, S. (2015). Extracting and understanding urban areas of interest using geotagged photos. *Computers, Environment and Urban Systems*, *54*, 240–254.

Järv, P., Tammet, T., & Tall, M. (2018). Hierarchical regions of interest. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)* (pp. 86–95). IEEE.

Kisilevich, S., Keim, D., & Rokach, L. (2010a). A novel approach to mining travel sequences using collections of geotagged photos. In M. Painho, M. Y. Santos, & H. Pundt (Eds.), *Geospatial Thinking* (pp. 163–182). Springer Berlin Heidelberg volume 0 of *Lecture Notes in Geoinformation and Cartography*.

Kisilevich, S., Mansmann, F., & Keim, D. (2010b). P-dbscan: A density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos. In *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application* COM.Geo '10 (pp. 38:1–38:4). New York, NY, USA: ACM.

Lee, I., Cai, G., & Lee, K. (2014). Exploration of geo-tagged photos through data mining approaches. *Expert Systems with Applications*, *41*, 397 – 405.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. (pp. 707–710). volume 10 of *Soviet physics doklady*.

Mazimpaka, J. D., & Timpf, S. (2016). Trajectory data mining: A review of methods and applications. *Journal of Spatial Information Science*, *2016*, 61–99.

Pan, X., He, Y., Wang, H., Xiong, W., & Peng, X. (2016). Mining regular behaviors based on multidimensional trajectories. *Expert Systems with Applications*, *66*, 106 – 113. doi:`https://doi.org/10.1016/j.eswa.2016.09.015`.

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., & Hsu, M.-C. (2004). Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on knowledge and data engineering*, *16*, 1424–1440.

Ramos, J. et al. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (pp. 133–142). New Jersey, USA volume 242.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, *42*, 19.

Shi, J., Mamoulis, N., Wu, D., & Cheung, D. W. (2014). Density-based place clustering in geo-social networks. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* SIGMOD '14 (pp. 99–110). New York, NY, USA: ACM.

Spyrou, E., Korakakis, M., Charalampidis, V., Psallas, A., & Mylonas, P. (2017). A geo-clustering approach for the detection of areas-of-interest and their underlying semantics. *Algorithms*, *10*, 35.

Stork, D. G., Duda, R. O., Hart, P. E., & Stork, D. (2001). Pattern classification. *A Wiley-Interscience Publication*, .

Talia, D., Trunfio, P., & Marozzo, F. (2015). *Data Analysis in the Cloud: Models, Techniques and Applications*.

Yin, Z., Cao, L., Han, J., Luo, J., & Huang, T. S. (2011). Diversified trajectory pattern ranking in geo-tagged social media. In *SDM* (pp. 980–991). SIAM.

Yuan, J., Zheng, Y., Xie, X., & Sun, G. (2011a). T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge and Data Engineering*, *25*, 220–232.

Yuan, J., Zheng, Y., Zhang, L., Xie, X., & Sun, G. (2011b). Where to find my next passenger. In *Proceedings of the 13th International Conference on Ubiquitous Computing* UbiComp '11 (pp. 109–118). New York, NY, USA: ACM.

Zhang, D., Lee, K., & Lee, I. (2019). Mining hierarchical semantic periodic patterns from gps-collected spatio-temporal trajectories. *Expert Systems with Applications*, *122*, 85 – 101. doi:`https://doi.org/10.1016/j.eswa.2018.12.047`.

Zheng, Y. (2015a). Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.*, *6*, 29:1–29:41.

Zheng, Y. (2015b). Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *6*, 1–41.

Zheng, Y.-T., Zha, Z.-J., & Chua, T.-S. (2012). Mining travel patterns from geotagged photos. *ACM Trans. Intell. Syst. Technol.*, *3*, 56:1–56:18.

Zhu, Z., Liang, J., Li, D., Yu, H., & Liu, G. (2019). Hot topic detection based on a refined tf-idf algorithm. *IEEE Access*, *7*, 26996–27007.