RESEARCH PAPER

Parallel Extraction of Regions-of-Interest from Social Media Data

Loris Belcastro[1] | M. Tahar Kechadi[2] | Fabrizio Marozzo*[1] | Luca Pastore[2] | Domenico Talia[1] | Paolo Trunfio[1]

[1]DIMES, University of Calabria, Rende, Italy
[2]Insight, UCD, Dublin, Ireland

Correspondence
*Fabrizio Marozzo Email: fmarozzo@dimes.unical.it

Present Address
DIMES, University of Calabria, Via P.Bucci 41C, 87036 Rende, Italy

Summary

Geotagged data gathered from social media can be used to discover Places-of-Interest (PoIs) that have attracted many visitors. Since a PoI is generally identified by geographical coordinates of a single point, it is hard to match it with people trajectories. Therefore, we define an area, called *Region-of-Interest* (*RoI*), represented by the boundaries of a PoI. The main goal of this study is to discover Regions-of-Interest from PoIs using spatial data mining techniques. In this paper we propose a new parallel method for extracting RoIs from social media datasets. It consists of two main steps: i) *automatic keywords extraction and data grouping*, and ii) *parallel RoIs extraction*. The first step extracts keywords identifying the Places-of-Interests; these keywords are used to group social media items according to the places they refer to. The second step uses a Parallel Clustering Approach (ParCA) of spatial dataset to identify RoIs. ParCA exploits a parallel execution of DBSCAN on sub-sets of data to generate sub-clusters on each processing node, and then merge overlapping sub-clusters to form global clusters. ParCA was implemented using the MapReduce model. Experiments performed over a set of PoIs in the city of Rome using social media data show that our approach is highly scalable and reaches an accuracy of 79% in detecting RoIs. On a parallel computer with 50 cores, we obtained a speedup of 52 by processing large datasets divided into 32 splits, compared to the execution time registered when each dataset is not partitioned.

KEYWORDS:
Regions-of-Interest, RoI mining, Social media analysis, Scalability, Parallel Clustering

1 | INTRODUCTION

The huge volumes of digital data produced by social media users can be effectively exploited to extract valuable information concerning human dynamics and behaviors. Extracting useful information from this large amount of data is the main aim of social media analysis. It is used for understanding collective sentiments, the behavior of groups of people and the dynamics of public opinion. Social media posts are often tagged with geographical coordinates or other information (e.g., text, photos) that allows identifying users' positions. Therefore, social media users moving through a set of locations produce a huge amount of geo-referenced data that embed extensive knowledge about human dynamics and mobility behaviors[1].

The analysis of geotagged data permits to determine if users visited or not interesting locations (e.g., touristic attractions, shopping malls, squares, parks), often called Places-of-Interest (PoIs). Since a PoI is generally identified by the geographical coordinates of a single point, it is hard to match it with user trajectories. For this reason, it is useful to define the so-called *Region-of-Interest* (*RoI*) representing the boundaries of the PoI's area[2]. The analysis of user trajectories through RoIs is highly valuable in many scenarios, e.g.: tourism agencies and municipalities can discover the

most visited touristic attractions and tours [3]; transport operators can discover the places and routes where is it more likely to serve passengers [4] and crowed areas where more transport facilities need to be allocated [5].

RoI mining techniques are aimed at discovering Regions-of-Interest from PoIs and other data [6]. The main contribution of this paper is a new methodology based on parallel computations for extracting RoIs from social media data, which is composed of two main steps: i) *Automatic keywords extraction and data grouping*, for finding keywords that identify the places of interests; these keywords will be used to group social media items according to the places they refer to; ii) *RoIs extraction using a parallel clustering approach*, which exploits a parallel DBSCAN [7] implementation on grouped social media items to identify RoIs efficiently. The parallel clustering is based on ParCA [8], a parallel approach for clustering spatial datasets. ParCA proved its efficiency using syntactic and benchmark datasets only. As ParCA is based on the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm, it also suffers for the problem of setting input parameters; it needs to tune the input parameters for the DBSCAN algorithm running locally in order to guarantee accurate final results. However, tuning the parameters needs a ground truth dataset to compare to, which is not given when dealing with real dataset. Therefore, to insure an efficient extraction of the RoIs, in this work we extended and optimized ParCA to deal with real world datasets, and also to choose the appropriate input parameters for the DBSCAN clustering algorithm used. To ensure scalability, the methodology has been implemented using the MapReduce programming model.

Experiments performed over a set of PoIs in Rome using social media data show that our methodology reaches an accuracy of 79% in detecting RoIs by using 1 split only. Using multiple splits, the accuracy slightly decreases (e.g., 77% using 2 data splits for each ParCA task, 74% with 4 data splits, 72% with 8 splits), but scalability significantly increases. For instance, on a parallel machine with 50 cores, we obtained a speedup of 52 by processing large datasets divided into 32 splits, compared to the execution time registered when each dataset is not partitioned.

The remainder of the paper is organized as follows. Section 2 introduces the main concepts and the problem statement. Section 3 describes the proposed methodology. Section 4 presents the experimental evaluation of the methodology on a case study. Section 5 discusses related work. Finally, Section 6 concludes the paper.

2 | PROBLEM DEFINITION

A *Place-of-Interest* (*PoI*) is a specific location that someone finds useful or interesting. Generally, PoIs refer to business locations (e.g., shopping malls) or tourist attractions (e.g., squares, museums, theaters, bridges). In this study the terms *Place-of-Interest* and *Point-of-Interest* are considered similar thus they are used interchangeably through the paper.

To analyze users' behavior, it is useful to understand whether a user visited a PoI or not. Since information on a PoI is generally limited to an address or to GPS coordinates, it is hard to match trajectories with PoIs. For this reason, it is useful to define *Region-of-Interest* (*RoI*) representing the boundaries of the PoI's area [2]. RoIs can be defined as "spatial extents in geographical space where at least a certain number of user trajectories pass through" [9]. Thus, RoIs represent an useful abstraction for partitioning the space into meaningful areas and, correspondingly, to associate a label to a place. In literature, RoIs are also named as *regions of attraction* or *frequent (dense) regions*.

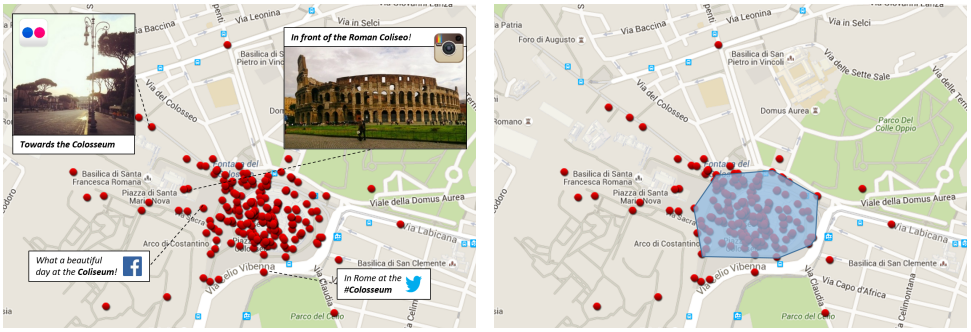
A *geotagged item* is a piece of information (e.g., tweet, post, photograph or video) to which geospatial information were added. Specifically, a geotagged item g includes the following features:

- *text*, containing a textual description of g.
- *tags*, containing the tags associated to g.
- *coordinates* consists 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 geotagged item can be associated to a PoI $\mathcal{P}$ if its text or tags refer to $\mathcal{P}$. For example, Figure 1(a) shows some geotagged items (red points on the map) that refer to the Colosseum in Rome since their text contain keywords such as "Colosseum", "Coliseum" and "Coliseo". By grouping all the items that refer to a PoI and applying a clustering algorithm, a suitable RoI can be obtained. For example, Figure 1(b) shows how the geotagged items that refer to the Colosseum can be grouped so as to define a RoI for the Colosseum (i.e., the boundaries of the PoI's area).

3 | METHODOLOGY

This paper proposes a new methodology based on a parallel computations for extracting RoIs from social media data. The methodology is designed to ensure system's scalability, so as to obtain RoIs within a reasonable response time even in presence of Big Data. In addition, it addresses some



(a) Geotagged items (red points on the map) and the content of some of them.

(b) Colosseum RoI.

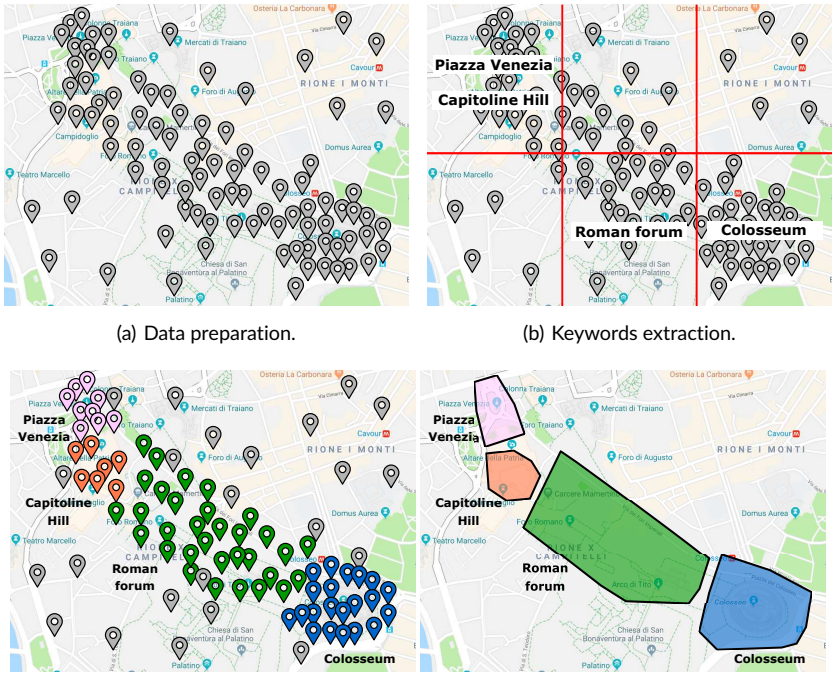
FIGURE 1 Use of the geotagged items, from different social media that refers to the Colosseum in Rome, for extracting a suitable Region-of-Interest.

issues of existing techniques by providing automatic keyword extraction, identification of arbitrary shapes, and easy parameter configuration. Figure 2 depicts the entire workflow of the proposed methodology, which consists of four steps:

- *Crawling*: data are gathered from social media, e.g., by using multiple crawlers that run in parallel. Collected data can be stored on parallel/distributed file systems (e.g., HDFS) for concurrent processing.
- *Data preparation*: a set of functions are used to make data suitable for the subsequent analysis, such as removing all unnecessary information from data (data pruning), adding information coming from external sources (data enrichment), or transforming data values (e.g., data normalization).
- *Automatic keywords extraction and data grouping*: during this step, the keywords identifying the PoIs are extracted; these keywords are then used for grouping social media items according to the places they refer to. Each group of social media items are then processed in parallel during the next step. The keywords extraction step can be possibly skipped if the keywords are provided manually (e.g., by using a PoI database).
- *RoIs extraction*: a data parallel clustering is exploited for extracting RoIs from social media data grouped by keywords. During this step, multiple instances of ParCA run in parallel over the different groups of social media items that have been found at the previous step. Each ParCA instance can work sequentially on one single split, or in parallel on multiple splits. We discuss the performance of the sequential and parallel approaches in Section 4.

FIGURE 2 Main steps of the proposed methodology.

For a sake of clarity and for Reader's convenience, before going into algorithmic details, we explain, through an example, how our methodology works (see Figure3). Starting from a small sample of geotagged social media items referring to the center of Rome (Figure 3(a)), we divided the area under examination into squared cells of equal size (Figure 3(b)). All the geotagged items in our sample contain at least one keyword that identifies a PoI. The automatic keyword extraction algorithm (more details are provided in Section 3.2) is able to extract the most representative keywords in the area. In the example, the extracted keywords are: "Piazza Venezia", "Capitoline Hill", "Roman forum", "Colosseum". Figure 3(c) shows, starting from the extracted keywords, how the geotagged items are divided into groups (data grouping): the pink items refer to Piazza Venezia, the orange to the Capitoline Hill, the green to the Roman Forum, and the blue to the Colosseum. Finally, Figure 3(d) shows how each group of geotagged items is analyzed and clustered for extracting the Region-of-Interest (RoI extraction).



(a) Data preparation. (b) Keywords extraction.

(c) Data grouping. (d) RoI extraction.

FIGURE 3 An example of the methodology applied to an area in the city of Rome.

3.1 | Input data format

The input is a set of files containing geotagged items that can be gathered from different sources (Twitter, Flickr, etc.). Each social media item is represented by a metadata document composed of common fields to all social media platforms (source, item id, date and time, location coordinates, user info) and specific fields of the given source. For example, Listing 1 shows some metadata of a Flickr post that contains specific fields, including the date on which the photo was taken and posted, the format of the photo (e.g., JPEG), a flag indicating that the photo has been posted by a professional photographer, and so on. On the other hand, Listing 2 contains the metadata of a tweet, including also some specific fields of the social media platform (whether it is a retweet or not, the retweet count, the reply details, and so on).

```
{"ID":"43012793876",
 "Owner":{ "ID":"111222333@N00", "USERNAME":"fm84"}
 "Title":"St Peter",
 "Description":"St Peter's church in Rome",
 "DateTaken":"2016-11-20T20:00:01.000",
 "DatePosted":"2016-11-21T22:12:36.000",
 "Comments":25, "Views":2354, "PRO": true,
 "Geodata":{"LNG":12.456661,"LAT":41.90245,
            "Accuracy":16},
 "Tags":["holiday","vatican"],
 "Format":"jpg"
}
```

Listing 1: Metadata of a Flickr post serialized in JSON format.

```
{"Source":"Twitter", "ID":"111222333444555",
 "DateTime":"2015-12-20T23:20:34.000",
 "Location":{"LNG":-0.1262,"LAT":51.5011},
 "User":{"UserID":"12345", "Username":"joe"},
 "InReplyToScreenName":"bill",
 "InReplyToUserId":123456789,
 "InReplyToStatusId":678712345678962848,
 "Text":"@bill That sounds great!",
 "Hashtags":[ "#code", "#mapreduce"],
 "Retweets":0, "IsRetweet":false
}
```

Listing 2: Metadata of a tweet serialized in JSON format.

3.2 | Automatic keywords extraction and data grouping

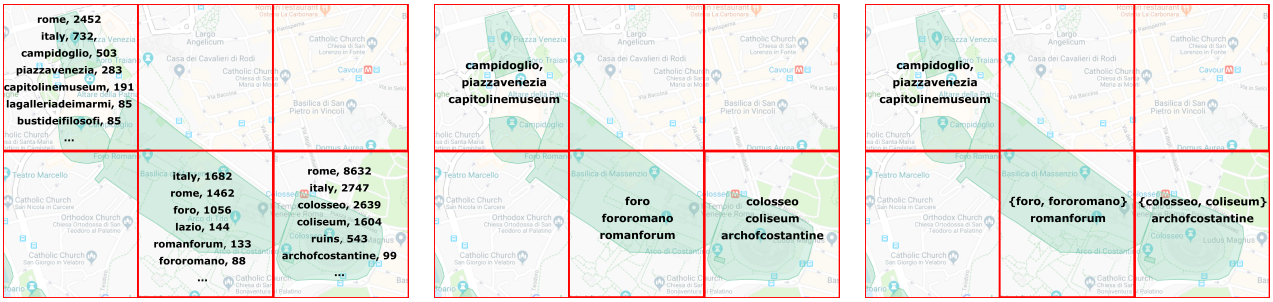
The keyword extraction and data grouping algorithm has been implemented as a MapReduce program, so that, as the dataset size increases, it can be executed in parallel on a Cloud or a HPC infrastructure. The algorithm extracts the most relevant keywords used by social users to tag places-of-interest in a given area. Once extracted, the keywords are grouped by similarity measure to produce the list of keywords that identify each PoI in the area.

The algorithm is composed of three steps:

1. *Keyword discovery*. The area of interest is divided into squared cells of equal size. As shown in Figure 4(a), for each cell, we extract the keywords (and their frequency) contained in the geotagged items posted from that cell. The keywords are then sorted by frequency. A high frequency does not necessarily denote a 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", although "colosseum" is evidently more representative of the PoI contained in the cell. Noisy keywords, such as "italy" or "rome", are then removed in the next step.

2. *Keyword selection*. A method based on a discrete L-curve [10] is exploited for distinguishing between high and low-frequency keywords. Finding the elbow point of the curve permits to distinguish between high and low-frequency keywords. The algorithm takes into account both global high-frequency keywords (i.e., calculated on the whole area) and local high-frequency keywords (i.e., calculated on each cell). Starting from high-frequency keywords in a cell, it discards high-frequency global keywords to produce a list of the most representative keywords for each cell. As shown in Figure 4(b), the algorithm removes all the keywords that do not identify a PoI in the cell (i.e., "italy", "rome", "lazio").

3. *Keyword grouping*. The most representative keywords are grouped according to their textual similarity. In particular, the concept of similarity is based on the Levenshtein distance [11], a commonly-used metric for measuring the distance between two strings. The algorithm outputs a number of sets containing keywords that are similar to each other (see Figure 4(c)) where each *key set* contains the keywords identifying a PoI. During the clustering step, each key set is used to find the associated RoI.



(a) Keyword discovering. (b) Keywords extraction. (c) Keywords grouping.

FIGURE 4 Automatic keyword extraction per cell.

3.3 | Parallel Clustering Approach (ParCA)

Clustering is one of the fundamental tasks in data mining and machine learning. While the concept of clustering is very simple, i.e., assigning similar data objects to the same cluster and dissimilar data objects to different clusters, most existing clustering techniques have high computational complexity and they do not scale well. One way of dealing with this issue is to develop distributed clustering where many processing nodes combine their effort to solve very large problems. Moreover, in many cases the datasets are already distributed, and therefore, it is more efficient to analyze them at their location and communicate only the analysis results. In this case, not only we save the time of communicating all the datasets to a central computing node, but also we avoid data loss and breaches during its transit, mainly for sensitive data.

We developed a Parallel Clustering Approach (ParCA) for spacial datasets [8], which is composed by two main phases: *local clustering phase* and *global clustering phase*. Assume that the dataset is already distributed among the system nodes, during the first phase each processing node executes a clustering algorithm on its local dataset to produce local clusters. The second phase consists of merging local clusters to generate global clusters. The first phase takes advantage of full parallelism, as each processing node clusters only its own sub-dataset. Moreover, one can execute different clustering algorithms on different sub-datasets, as the data collected in each node may be different (heterogeneous). The second phase requires the full cooperation and collaboration of the processing nodes to generate the global clusters (see Figure 5). Therefore, the data exchanged between the nodes has direct effect on the system's performance. Communicating local clusters among the nodes is a very complex task and for very large datasets this may not be feasible. Our parallel clustering approach does not exchange the local clusters entirely. Instead, it exchanges only the local cluster representatives (e.g., cluster contour), which is a very low percentage of the data, and therefore, reduces significantly the network load. There are different approaches of how to extract clusters' representatives. These usually depend on either the nature of the datasets or the clustering techniques. In this paper, we are interested in the cluster contour; the data points that constitute the contour are the representatives of

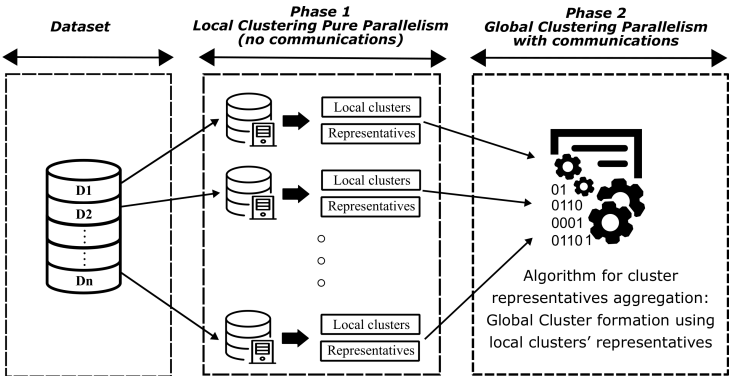


FIGURE 5 Parallel Clustering Approach (ParCA).

the cluster. The contours of the local clusters are extracted during the first phase. There are many efficient algorithms in the literature for contour extraction, with a reasonable complexity of $\mathcal{O}(\mathrm{n}\log\mathrm{n})$, where n is the size of the cluster [12,13,14].

The ParCA is implemented using the MapReduce programming model. This model is well suited for this approach. The first phase is implemented by the mappers. In other words, each mapper implements the local clustering and contours generation of the local clusters. The second phase is implemented by the reducers, by taking the outputs of the mappers, execute the merging algorithm, and generate the global clusters. As the second phase is executed in a hierarchical way to optimize the level of parallelism, we have many reducers that cooperate to produce the final results.

3.4 | RoIs extraction using a parallel clustering approach

Figure 6 describes the parallel clustering approach that has been implemented for extraction RoIs. The input of this step is represented by N datasets, each one containing the points associated with the key set of a given PoI. Each dataset can be divided into M splits, with $\mathrm{M} \geq 1$. Thus, two levels of parallelisms can be exploited:

- N ParCA instances run concurrently, with each instance working exclusively on a single key set.
- Each ParCA instance processes in parallel the M splits of its key set, by producing one local cluster per split; the M local clusters are then merged into a single global cluster which represents the RoI associated with the key set.

The above process is implemented in MapReduce programming framework. In particular, the mappers task is to produce the local clusters, whereas reducers task is to derive the global clusters.

As ParCA is based on the DBSCAN algorithm [7], it also suffers for the problem of setting input parameters; one needs to tune the input parameters for the DBSCAN algorithm running locally in order to guarantee accurate final results. However, tuning the parameters needs a ground truth dataset to compare to, which is not given when dealing with real dataset. In the next section we discuss how we extended ParCA to choose the appropriate input parameters for the DBSCAN algorithm.

3.4.1 | Heuristic for choosing DBSCAN parameters

The DBSCAN algorithm needs two key parameters: eps, the radius of a neighborhood with respect to some point; and minPts, the minimum number of points required to form a cluster. These two parameters can be calculated using the following procedure as defined in [7]:

1. Calculate the sorted k-dist plot, which is a plot of the k-nearest-neighbor distances (k-dist), computed for each point, and sorted in descending order [15]. As suggested in [7], for bi-dimensional data, k can be set to 4.
2. Choosing a *threshold point* on k-dist permits to separate noise points from points that are assigned to some clusters. Specifically, all points with a higher k-dist value than threshold are considered noise; otherwise, all points with an equal or smaller k-dist value are assigned to some clusters.
3. The threshold point can be calculated by estimating the noise percentage in the data (*noisePerc*). The k-dist value of the threshold point is used as eps value. Concerning minPts, it can be set as $\mathrm{k}+1$ [15]. Figure 7(a) shows an example of sorted k-distance plot.

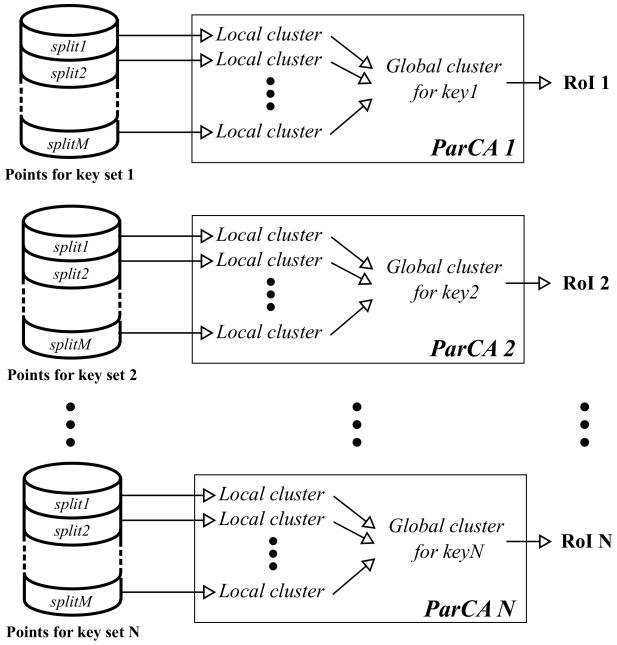


FIGURE 6 RoIs extraction exploiting the ParCA algorithm.

Thus, with fixed the minimum number of points (minPts) and established the noise percentage (noisePerc), we calculate eps and then run DBSCAN on the points collected for a Point-of-Interest. Since DBSCAN calculates one or more clusters on a set of points, we select the points that belong to the largest cluster, and starting from them we return the convex polygon that encloses these points (Region-of-Interest).

As an example, starting from the Colosseum points (i.e., geotagged items that refer to the Colosseum) we plot the sorted k-dist graph, with k=4 (see Figure 7(b)). On the graph we represented various percentages of noise (from 5% to 30%). In Figure 7(c)) we show how noise affects the definition of the Colosseum RoI. With a low noise (e.g., 5% for the yellow polygon) we obtain a large Region-of-interest, as the noise increases the RoI becomes smaller (e.g., 20% for the purple polygon). Figure 7(d) and 7(e) show how the noise affects the definition of Piazza Navona's and St. Peter Basilica's RoIs. In Section 4.2 we describe how to find a good trade-off between precision and recall by analyzing a test set consisting by 20 case studies.

4 | PERFORMANCE EVALUATION

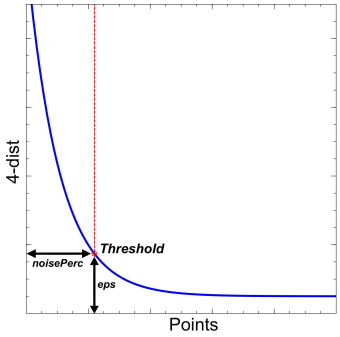
The evaluation was carried out by analyzing a dataset containing about 9 millions of social media items published by Flickr users from 2007 to 2017 referring to the center of Rome. Such data has been collected though public Flickr APIs[1]. During the keyword extraction phase, our methodology was able to find about 210 keywords representing points-of-interest in the area under examination. We used 20 places with known RoIs to facilitate the parameters setting of the clustering algorithm. In particular, Section 4.1 presents the metrics used to measure the accuracy of RoI algorithms. Section 4.2 and Section 4.3 discuss the experiments carried out for evaluating the accuracy and the execution time of the proposed methodology.

4.1 | Performance metrics

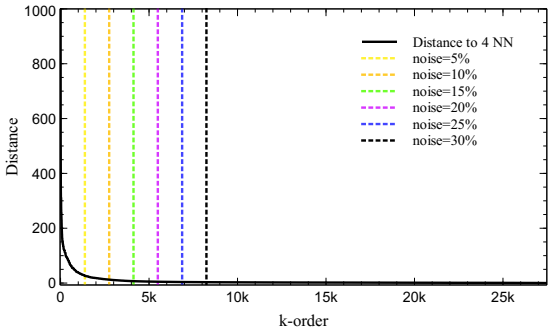
To measure the accuracy of the algorithms for detecting RoIs, we use *precision* and *recall* metrics. As in[2,6], let roi_{real} be the real RoI for a PoI (taken from online services like OpenStreetMap), and let $\text{roi}_{\text{found}}$ be the RoI found by an algorithm. Let us define the true positive area roi_{TP} as the intersection of $\text{roi}_{\text{found}}$ and roi_{real}. Precision Prec and recall Rec are defined as:

$$Prec = \frac{Area(roi_{TP})}{Area(roi_{found})} \qquad Rec = \frac{Area(roi_{TP})}{Area(roi_{real})} \tag{1}$$

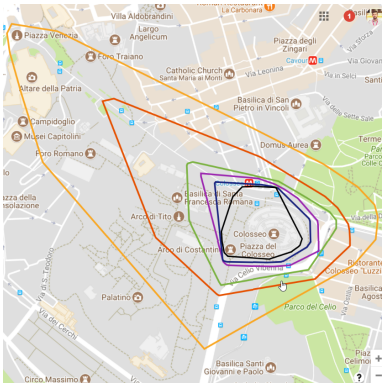
[1]https://www.flickr.com/services/developer/api/



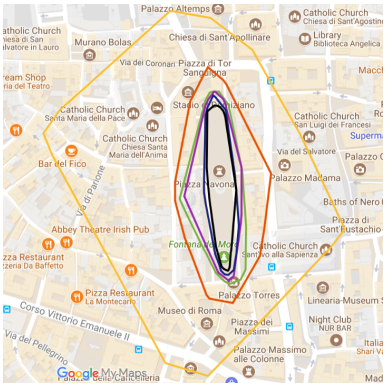
(a) Example of sorted k-distance plot (k=4).



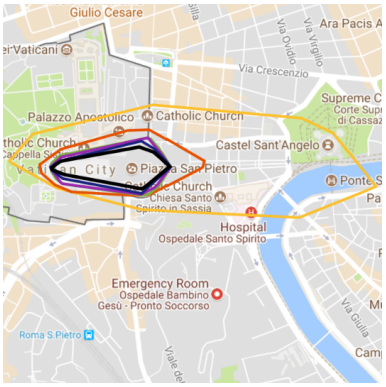
(b) Sorted k-distance plot (k=4) for the geotagged items related to the Colosseum.



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



(d) Piazza Navona's RoIs for different values of noisePerc.



(e) St. Peter Basilica's RoIs for different values of noisePerc.

FIGURE 7 How the percentage of noise (noisePerc) affects the definition of RoIs.

A $\text{roi}_{\text{found}}$ larger than roi_{real} produces a high recall and a low precision, whereas $\text{roi}_{\text{found}}$ smaller than roi_{real} produces a low recall and a high precision. If $\text{roi}_{\text{real}} \subseteq \text{roi}_{\text{found}}$ then $\text{roi}_{\text{TP}} = \text{roi}_{\text{real}}$ and therefore the recall is 1 but the precision is lower than 1. On the other hand, if $\text{roi}_{\text{found}} \subseteq \text{roi}_{\text{real}}$ the precision is 1 but the recall is lower than 1. To rank the results, we combine precision and recall using the F_1 score:

$$F_1 = \frac{2 \cdot Prec \cdot Rec}{Prec + Rec} \tag{2}$$

4.2 | Accuracy evaluation

Table 1 illustrates the clustering performance (Precision, Recall, and F1 score) for different values of the noise in the data (10%, 20%, and 30%). The last row of the table reports mean values computed over the 20 PoIs that have been considered. When precision is higher than recall, the RoI identified by the algorithm is smaller than the real one. On the contrary, when precision is lower than recall, it means that the RoI identified is larger than the real one. To choose the optimal value for the noise percentage (noisePerc) in the data, we used the F1 score, which is the harmonic mean of the precision and recall. The results reported in the table show that considering a noise percentage of 20% we obtained more accurate results. In fact, using noisePerc $= 20\%$ produces the highest mean F1 score of 79%.

Table 2 illustrates the performance (Precision, Recall, F1 score) of our parallel clustering approach taking into account a variable number of splits for the data associated to each PoI. In particular, we have considered six scenarios to compare, from sequential (i.e., 1 split) to parallel approach with an increasing number of splits (from 2 to 32). Both approaches have been tested using all the 20 PoIs that have been considered. The results show that the average F1 score, calculated on the RoIs identified by ParCA, slightly decreases as the number of splits increases. In fact, as the number of split increases, each parallel instance of DBSCAN works on less and less data, which leads to a lower accuracy. In particular, comparing the parallel clustering approach with the sequential one, the F1 score decreases from 79% to 77% using two splits, and up to 69% using 32 splits. In the area under analysis, the proposed algorithm identifies the 20 RoIs shown in Figure 8.

PoI	noisePerc=10%			**noisePerc=20%**			noisePerc=30%		
	Prec.	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*
Campo de' Fiori	0.10	1.00	0.19	0.49	1.00	0.65	0.94	0.76	0.84
Capitoline Hill	0.17	1.00	0.29	0.48	1.00	0.65	0.81	0.86	0.84
Circus Maximus	0.31	1.00	0.48	0.77	0.95	0.85	0.82	0.63	0.71
Colosseum	0.27	1.00	0.43	0.77	0.90	0.83	0.96	0.62	0.75
Mausoleum of Hadrian	0.22	0.99	0.36	0.50	0.85	0.63	0.80	0.62	0.70
Our Lady in Trastev.	0.10	1.00	0.18	0.44	0.98	0.61	0.85	0.88	0.86
Palazzo Montecitorio	0.23	1.00	0.37	0.66	0.94	0.78	0.93	0.59	0.72
Pantheon	0.40	1.00	0.57	0.98	0.70	0.81	1.00	0.36	0.53
Piazza Colonna	0.21	1.00	0.34	0.56	0.99	0.72	0.88	0.89	0.89
Piazza del Popolo	0.54	0.98	0.70	0.86	0.91	0.88	0.99	0.46	0.63
Piazza di Spagna	0.23	1.00	0.38	0.52	1.00	0.68	0.90	0.82	0.86
Piazza Navona	0.46	1.00	0.63	0.84	0.86	0.85	1.00	0.52	0.69
Piazza Venezia	0.19	1.00	0.33	0.46	1.00	0.63	0.61	0.72	0.66
Roman Forum	0.56	1.00	0.72	0.83	0.84	0.83	0.96	0.50	0.66
St. Mary Major	0.72	0.97	0.83	0.97	0.55	0.70	1.00	0.22	0.37
St. Peter's Basilica	0.65	1.00	0.79	0.96	0.89	0.92	0.99	0.65	0.79
Trastevere	0.51	0.95	0.67	0.85	0.76	0.80	0.81	0.43	0.56
Trevi Fountain	0.08	1.00	0.15	0.71	1.00	0.83	1.00	0.61	0.76
Vatican Museums	0.57	0.97	0.72	0.79	0.76	0.78	0.85	0.27	0.41
Villa Borghese	0.84	0.87	0.85	0.91	0.55	0.68	1.00	0.07	0.13
Mean Values	0.37	0.99	0.54	0.72	0.87	**0.79**	0.91	0.58	0.70

TABLE 1 Precision, recall, and F1 score of our clustering approach over 20 PoIs in Rome on 1 data split.

N. of split	*Precision*	*Recall*	*F1*
1	0.72	0.87	0.79
2	0.68	0.89	0.77
4	0.62	0.92	0.74
8	0.59	0.94	0.72
16	0.57	0.95	0.71
32	0.54	0.97	0.69

TABLE 2 Precision, recall, and F1 score of our clustering approach by varying the number of splits over 20 PoIs in Rome.

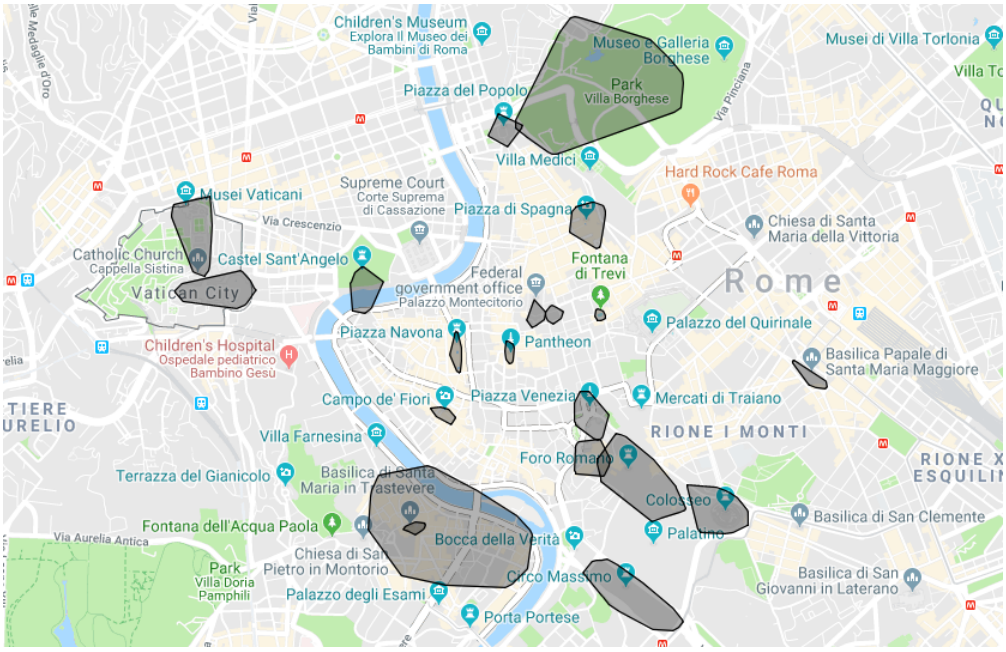


FIGURE 8 City of Rome: 20 RoIs identified by our methodology.

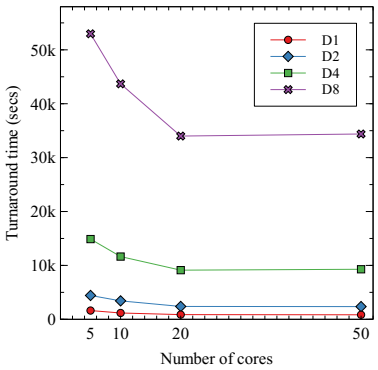
4.3 | Execution time evaluation

We experimentally evaluated the proposed methodology by running the analysis on a private cloud infrastructure. Specifically, we used a cluster equipped with 50 CPU cores and 100 GB of memory. The goal of the evaluation is to assess the scalability of the methodology by varying the number of data splits. The following performance parameters have been considered:

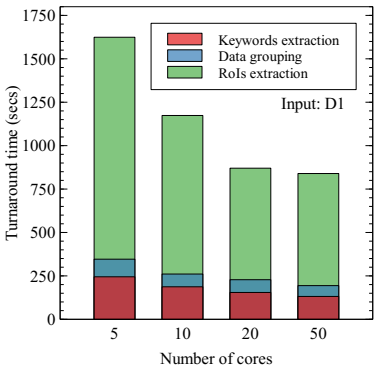
- *Turnaround time*: the amount of time elapsed from the submission of an application to its completion;
- *Data-splitting speedup*: the ratio of the turnaround time using 1 split per ParCA instance, by the turnaround time obtained using n splits per ParCA instance.

Both turnaround time and speedup have been evaluated by considering a variable number of CPU cores (from a minimum of 5 to a maximum of 50). As mentioned before, the evaluation was carried out by analyzing a dataset containing about 9 millions of social media items published by Flickr users from 2007 to 2017 referring to the center of Rome. In order to perform a more complete scalability analysis, we randomly sampled the original dataset to generate four datasets D1, D2, D4, D8 that contain 1.5GB, 3GB, 6GB and 12GB of data, respectively.

Figure 9 shows the turnaround time obtained using ParCA on a single data split, using from 5 to 50 CPU cores. In particular, Figure 9(a) shows the turnaround times of the application for the four datasets. For the smallest dataset (D1) the turnaround time decreases from 27 minutes using 5 cores to 14 minutes using 50 cores. For D2, the turnaround time decreases from 74 to 39 minutes. For D4, the turnaround time decreases from 4.1 to 2.5 hours. Finally, for the largest dataset (D8), the turnaround time ranges from 14.7 to 9.5 hours. From the figure it can be noticed that when more than 20 cores are used the decrease of the turnaround time is negligible. Figure 9(b) illustrates the breakdown of the turnaround time for dataset D1. The RoIs extraction (i.e., the execution of ParCA) is the dominant step that most influences the overall turnaround time, which does not significantly decreases passing from 20 to 50 cores. This is mainly due to the RoI mining step that is not able to exploit a large number of cores, because the clustering tasks are very heterogeneous in terms of execution times. This heterogeneity is due to imbalance in processing the geotagged points associated to each PoI. For example, the number of geotagged items associated to popular places like “Colosseum” or “St. Peter’s Basilica” is much higher than the number of items associated to other PoIs. For this reason, the turnaround time is bound to the execution time of the slowest task instances.



(a) Turnaround time vs number of core (1 data split).



(b) Breakdown of the turnaround time for D1 (1 data split).

FIGURE 9 Turnaround using ParCA.

Dividing data to be clustered into splits allows ParCA to reduce the RoI mining step execution time significantly. Figure 10 shows the turnaround time with different combinations of the dataset sizes and the number of splits by using 50 cores. For instance, notation D2/S8 in the x-axis means dataset D2 with 8 splits. In all configurations, only the RoI extraction step is affected by the number of splits used. This is because parallelism on the key sets is exploited by the ParCA algorithm. For example, for the smallest dataset (D1) the ParCA execution time decreases from 14 minutes using 1 split (about 840 secs in Figure 9(b)) to 6 minutes using 32 split (360 secs). For the largest dataset (D8), the ParCA execution time passes from 9.8 hours (35280 secs) to 11 minutes (660 secs).

The scalability can be evaluated through the graphs in Figure 11, which illustrate the data splitting speedup obtained by ParCA using from 1 to 32 data splits, for each of the four datasets considered. The maximum number of splits was chosen as a good compromise between scalability and accuracy of the results. In fact, using a higher number of splits, each clustering algorithm works on less and less data, thus leading to a lower accuracy. Each graph presents the speedup obtained with a different number of cores. By analyzing the four graphs, we can notice that the speedup depends on the number of splits, the number of cores, and the dataset size. With the smallest datasets (D1), the speedup achieved is never higher than 3, even using a large number of splits and cores. In this case, using as low as 4 splits and 5 cores is enough to achieve the maximum speedup (see Figure 11(a)). On the other hand, the highest speedup (about 52) is obtained when processing the largest dataset (D8) using 32 splits and 50 cores (see Figure 11(d)). From Figure 11(c), we observe that even if we reduce the number of cores to 20, the speedup remains very good (about 41) using 32 splits on the same dataset. This shows the positive impact of increasing the number of splits, even when they exceed the number of cores available for processing. Overall, the experimental results show that the scalability of the system is a function of both the number of splits and the number of cores. The former has a greater influence on the latter, in the sense that the use of a higher number of cores is effective in improving speedup only if the number of splits is also increased.

We conclude the evaluation by providing some data about the throughput achieved during the experiments. Given a large number of combinations that can be obtained by varying the number of splits and the number of cores used, we provide throughput values for two representative configurations: i) D1/S1 using 5 cores, and ii) D1/S32 using 50 cores. In the first configuration, the throughput was 0.12 tasks/sec, while in the second configuration it was 16.69 tasks/sec. This variability in the throughput values is indeed due to the different values of the number of splits and parallelism level that can be found.

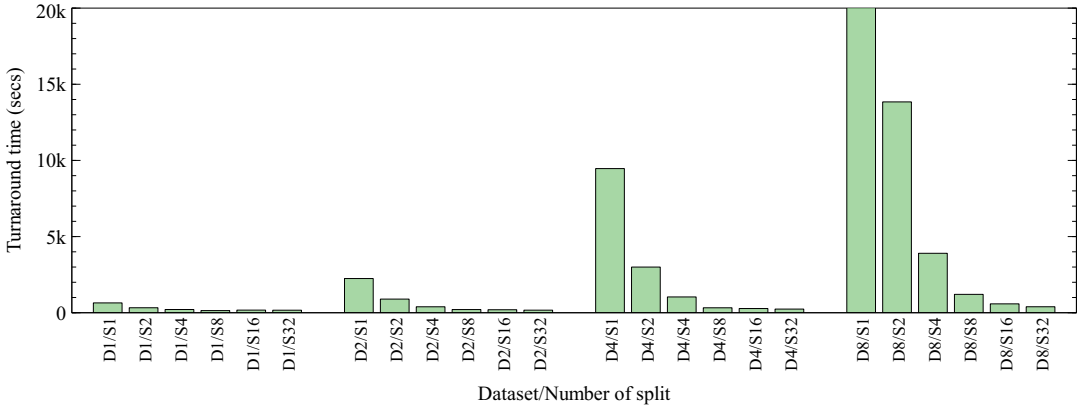
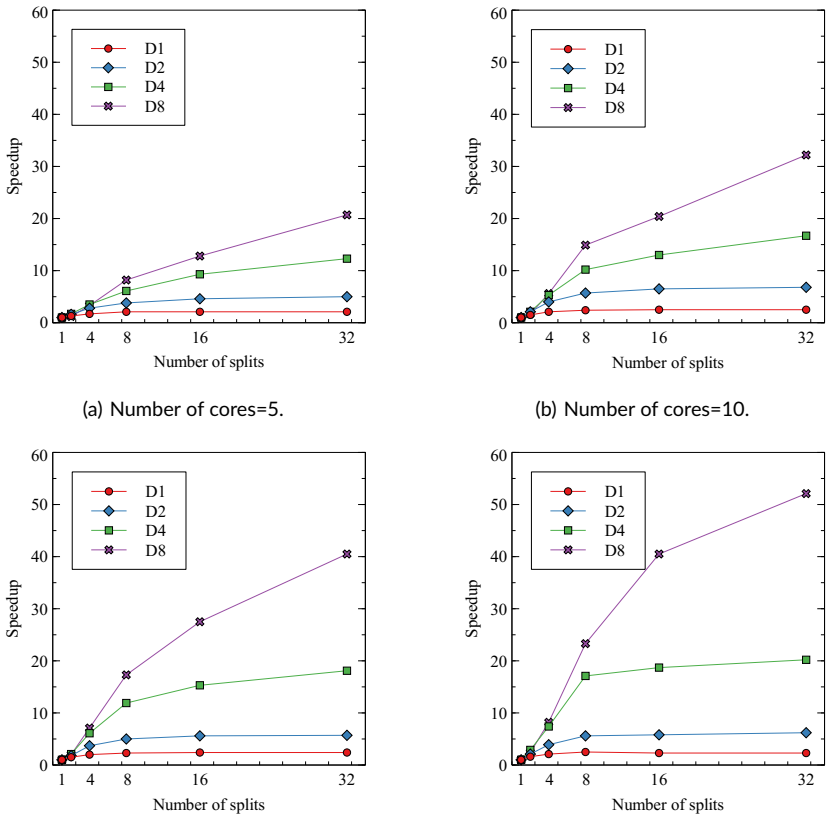


FIGURE 10 Turnaround time vs number of splits using ParCA on 50 cores.



(a) Number of cores=5.

(b) Number of cores=10.

(c) Number of cores=20.

(d) Number of cores=50.

FIGURE 11 Speedup vs number of cores using a ParCA with splits.

5 | RELATED WORK

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

Predefined shapes. According to this approach predefined shapes (circles, rectangles, etc.) are used to represent RoIs. For example, Kisilevich et al.[16] defined RoIs as circles of fixed radius centered on a set of PoIs whose center coordinates are known. Spyrou and Mylonas[17] used circular RoIs to extract popular touristic routes from Flickr. Specifically, circular shapes are used to translate a trajectory of geospatial points into a sequence of RoIs. Cesario et al.[18] used rectangles to define RoIs representing stadiums for a trajectory mining study. In particular, the RoI of a stadium is the smallest rectangle enclosing the stadium's area. De Graaff et al.[2] use Voronoi tessellations to define RoIs starting from a set of geographical coordinates representing PoIs.

Density-based clustering. With this approach, RoIs are obtained by clustering a set of geographical locations. For instance, Zheng et al.[19] used DBSCAN[7] to discover tourist attraction areas from a set of Flickr photos. DBSCAN was adopted for three main reasons: i) it tends to identify regions of dense data points as clusters; ii) it supports clusters with arbitrary shape; iii) it has a good efficiency on large-scale data. DBSCAN was also used by Altomare et al.[5], with the goal of detecting the regions that are more densely visited based on data from GPS-equipped taxis. Kisilevich et al.[20] 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. Density-based approaches need a method to assign a meaning to each RoI found. There are different ways to perform this task. Zheng et al.[19] and Yin et al.[21] assign a name to each cluster by taking the most frequent keyword in the geotagged items. Ferrari et al.[22] automatically associate to each RoI the zip code of the data points in the cluster center. Järv et al.[23] 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.

Grid-based aggregation. This approach discretizes the area under analysis in a regular grid and extracts RoIs by aggregating the grid cells. For example, Giannotti et al.[9] divide an area into grid cells and count the trajectories passing through each cell. Grid cells whose counters are above a certain threshold are expanded to form rectangular shaped RoIs. Cai et al.[24] 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. Cesario et al.[25] split the EXPO 2015 area in a grid and associated grid cells to PoIs representing pavilions, in order to discover the behavior and mobility patterns of users inside the exhibition. Shi et al.[26] map geotagged data into grid cells, and then group the cells taking into account spatial proximity and social relationship between places. Spyrou et al.[27] proposed an algorithm that divides a geographical area into cells and then exploits an iterative merging procedure for finding RoIs. In particular, the merging procedure exploits a similarity metric based on the metadata contained in social media items. The authors have conducted an experimental evaluation on a dataset of Flickr photos referring to the center of Athens. For a qualitative evaluation of the proposed algorithm, the authors conducted a survey for assessing the satisfaction of real-life users.

Hybrid approaches. Other approaches combine some aspects of the techniques mentioned above. As an example, G-RoI[6] exploits the indications contained in 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 manually defined keywords identifying a PoI (which permits to group data for each PoI), *G-RoI* iteratively calculates the associated RoI using a density-based criterion. The experimental results show that G-RoI is more accurate in identifying RoIs than main techniques based on predefined shapes, density clustering, and grid-based aggregation.

The techniques discussed previously present some issues. For example, *predefined shapes* represent a naïve solution to the RoI mining problem, because they are not able to handle PoIs having RoIs with different sizes and shapes. *Density-based* techniques may fail to distinguish regions that are very close to each other or that have different densities. *Grid-based aggregation* techniques discretize the area in a regular grid and then aggregate the grid cells using different aggregation policies. Thus, it may be hard to find a setting for identifying multiple RoIs with different characteristics in the same area. *Hybrid* techniques, like G-RoI, lead to better results. However, G-RoI does not addresses some issues: i) automatic keyword extraction, since in G-RoI the list of keywords is an input parameter of the algorithm; and ii) data clustering in parallel, since G-RoI iteratively runs a sequential algorithm until a RoI is found, which leads to scalability problems as input data grows.

The methodology proposed in this paper addresses all the issues discussed above: i) it is able to identify RoIs of arbitrary shapes regardless of the proximity and density of the different regions; ii) it requires only one parameter, i.e., the noise percentage (see Section 3.4.1), which configures the RoI algorithm for finding regions with different densities; and iii) it exploits a parallel clustering approach, based on the MapReduce programming model, which allows it to identify RoIs by ensuring high scalability.

6 | CONCLUSION

RoI mining techniques are aimed at discovering Regions-of-Interest (RoIs) from Places-of-Interest (PoIs) and other data. Existing RoI mining techniques are based on the use of *predefined shapes*, *density-based clustering* or *grid-based aggregation*. This paper proposes a new parallel methodology for extracting RoIs from social media data, which is composed by two main steps: i) *automatic keywords extraction and data grouping*, for finding keywords that identify the places of interests; these keywords are used to group social media items according to the places they refer to; ii) *RoIs extraction using a parallel clustering approach*, which, starting from grouped social media data, exploits a parallel clustering approach (ParCA) to identify RoIs efficiently, which is based on a parallel implementation of DBSCAN.

To ensure an efficient extraction of the RoIs, ParCA was optimized to deal with real world datasets and also to choose the appropriate input parameters for the DBSCAN clustering algorithm it is based on. In addition, to ensure scalability, the algorithm has been implemented using the MapReduce programming model. Experiments performed over a set of PoIs in Rome using social media data show that our methodology reaches an accuracy of 79% in detecting RoIs. Using multiple splits, the accuracy slightly decreases (e.g., 77% using 2 data splits for each ParCA task, 74% with 4 data splits, 72% with 8 splits), but scalability significantly increases. For instance, using a parallel machine with 50 cores, we obtained a speedup of 52 by processing large datasets divided into 32 splits, compared to the execution time registered when each dataset is not partitioned.

References

1. Talia Domenico, Trunfio Paolo, Marozzo Fabrizio. *Data Analysis in the Cloud*. Elsevier; 2015. ISBN 978-0-12-802881-0.
2. Graaff Victor, By Rolf A., Keulen Maurice, Flokstra Jan. Point of Interest to Region of Interest Conversion. In: SIGSPATIAL'13:388–391ACM; 2013; New York, NY, USA.
3. Belcastro Loris, Marozzo Fabrizio, Talia Domenico, Trunfio Paolo. ParSoDA: High-Level Parallel Programming for Social Data Mining. *Social Network Analysis and Mining*. 2019;9(1).
4. Yuan Jing, Zheng Yu, Zhang Liuhang, Xie XIng, Sun Guangzhong. Where to Find My Next Passenger. In: UbiComp '11:109–118ACM; 2011; New York, NY, USA.
5. Altomare Albino, Cesario Eugenio, Comito Carmela, Marozzo Fabrizio, Talia Domenico. Trajectory Pattern Mining for Urban Computing in the Cloud. *IEEE Transactions on Parallel and Distributed Systems*. 2017;28(2):586-599.
6. Belcastro Loris, Marozzo Fabrizio, Talia Domenico, Trunfio Paolo. G-RoI: Automatic region-of-interest detection driven by geotagged social media data. *ACM Transactions on Knowledge Discovery from Data*. 2018;12(3).
7. Ester Martin, Kriegel Hans-Peter, Sander Jörg, Xu Xiaowei. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: KDD'96:226–231AAAI Press; 1996.
8. Ghanem S., Kechadi T., Tari A. K.. New approach for distributed clustering. In: Proceedings 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services:60-65; 2011; Fuzhou, China.
9. Giannotti Fosca, Nanni Mirco, Pinelli Fabio, Pedreschi Dino. Trajectory Pattern Mining. In: KDD '07:330–339ACM; 2007; New York, NY, USA.
10. Hansen Per Christian. Analysis of Discrete Ill-Posed Problems by Means of the L-Curve. *SIAM Review*. 1992;34(4):561-580.
11. Levenshtein Vladimir I. Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady, vol. 10: :707–710; 1966.
12. Chaudhuri A.Ray, Chaudhuri B.B., Parui S.K.. A Novel Approach to Computation of the Shape of a Dot Pattern and Extraction of Its Perceptual Border. *Computer vision and Image Understranding*. 1997;68:257-275.
13. Melkemi Mahmoud, Djebali Mourad. Computing the shape of a planar points set. *Elsevier Science*. 2000;33:1423–1436.
14. Duckhama Matt, Kulikb Lars, Worboysc Mike, Galtond Antony. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Elsevier Science Inc. New York, NY, USA*. 2008;41:3224-3236.
15. Schubert Erich, Sander Jörg, Ester Martin, Kriegel Hans Peter, Xu Xiaowei. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems (TODS)*. 2017;42(3):19.

16. Kisilevich Slava, Keim Daniel, Rokach Lior. A Novel Approach to Mining Travel Sequences Using Collections of Geotagged Photos. In: Lecture Notes in Geoinformation and Cartography, vol. 0: Springer Berlin Heidelberg 2010 (pp. 163-182).

17. Spyrou Evaggelos, Mylonas Phivos. Analyzing Flickr metadata to extract location-based information and semantically organize its photo content. *Neurocomputing*. 2016;172:114 - 133.

18. Cesario Eugenio, Congedo Chiara, Marozzo Fabrizio, et al. Following soccer fans from geotagged tweets at FIFA World Cup 2014. In: ICSDM 2015 - Proceedings 2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services:33-38; 2015.

19. Zheng Yan-Tao, Zha Zheng-Jun, Chua Tat-Seng. Mining travel patterns from geotagged photos. *ACM Transactions on Intelligent Systems and Technology (TIST)*. 2012;3(3):56.

20. Kisilevich Slava, Mansmann Florian, Keim Daniel. P-DBSCAN: A Density Based Clustering Algorithm for Exploration and Analysis of Attractive Areas Using Collections of Geo-tagged Photos. In: COM.Geo '10:38:1–38:4ACM; 2010; New York, NY, USA.

21. Yin Zhijun, Cao Liangliang, Han Jiawei, Luo Jiebo, Huang Thomas S. Diversified Trajectory Pattern Ranking in Geo-tagged Social Media. In: SDM:980–991SIAM; 2011.

22. Ferrari Laura, Rosi Alberto, Mamei Marco, Zambonelli Franco. Extracting Urban Patterns from Location-based Social Networks. In: LBSN '11:9–16ACM; 2011; New York, NY, USA.

23. Jarv P., Tammet T., Tall M.. Hierarchical Regions of Interest. In: 2018 19th IEEE International Conference on Mobile Data Management (MDM), vol. 00: :86-95; 2018.

24. Lee Ickjai, Cai Guochen, Lee Kyungmi. Exploration of geo-tagged photos through data mining approaches. *Expert Systems with Applications*. 2014;41(2):397 - 405.

25. Cesario Eugenio, Iannazzo Andrea Raffaele, Marozzo Fabrizio, et al. Analyzing social media data to discover mobility patterns at EXPO 2015: Methodology and results. In: 2016 International Conference on High Performance Computing and Simulation, HPCS 2016:230-237; 2016.

26. Shi Jieming, Mamoulis Nikos, Wu Dingming, Cheung David W.. Density-based Place Clustering in Geo-social Networks. In: SIGMOD '14:99–110ACM; 2014; New York, NY, USA.

27. Spyrou Evaggelos, Korakakis Michalis, Charalampidis Vasileios, Psallas Apostolos, Mylonas Phivos. A Geo-Clustering Approach for the Detection of Areas-of-Interest and Their Underlying Semantics. *Algorithms*. 2017;10(1):35.

How to cite this article: L. Belcastro, F. Marozzo, L. Pastore, T. Kechadi, D. Talia, and P. Trunfio (2019), Parallel extraction of Regions-of-Interest from Social Media Data, *Concurrency and Computation: Practice and Experience*, *2019;xx:x–x*.