Improving Population Estimation From Mobile Calls: a Clustering Approach

Patrizio Dazzi[§], Matteo Dell'Amico[‡], Lorenzo Gabrielli[§], Alessandro Lulli[*§],
Pietro Michiardi[||], Mirco Nanni[§], Laura Ricci[*§]
[*]University of Pisa, Italy {surname}@di.unipi.it
[§]ISTI, CNR, Pisa, Italy {name.surname}@isti.cnr.it
[‡]Symantec Research Labs {name_surname}@symantec.com
[||]EURECOM, Campus SophiaTech, France, {name.surname}@eurecom.fr

***Abstract*—Statistical authorities promote and safeguard the production and publication of official statistics that serve the public good. One of their duties is to monitor the presence of individuals region by region. Traditionally this activity has been conducted by means of censuses and surveys. Nowadays technologies open new possibilities such as a continuous sensing of the presences by leveraging the data associated to mobile devices, e.g., the behaviour of users on doing calls. In this paper first we propose a specifically conceived similarity function able to capture similarity between individuals call behaviours. Second we make use of a clustering algorithm able to handle arbitrary metric leading to a good internal and external consistency of clusters. The approach provides better population estimation with respect to state of the art comparing with real census data. The scalability and flexibility that characterises the proposed framework enables novel scenarios for the characterization of people by means of data derived from mobile users, ranging from the nearly-realtime estimation of presences to the definition of complex, uncommon user archetypes.**

I. Introduction

Nowadays, mobile phones have an unprecedented rate of penetration across the world: most people almost always have mobile devices with them. As a consequence, the information that can be derived from their movements and presence has been successfully exploited on many fields, such as traffic monitoring or tourist movements analysis.

Our goal is to define methodologies, tools, conceptual and technological frameworks supporting the modelling of user behaviour by leverage the information available at the level of the telecom infrastructure (e.g., calls, SMS, etc.). The underlying idea is to characterize the mobility of the user just relying on network link level information (e.g. micro-cell), without requiring any kind of interaction with the software and the specific hardware of the mobile device (e.g., GPS). The ultimate aim is to provide a set of instruments able to estimate the amount of people living in a certain region, the ones that are used to travel into that region (commuters) and the ones occasionally visiting that region (visitors). To conduct this kind of analysis is of paramount importance to rely on tools able to manipulate and extract meaningful information from that data. In this scenario, the definition of a proper clustering algorithms is crucial.

In a previous work by some of the authors of this paper [1], the algorithm adopted for data clustering was K-means. K-means is one of the most popular clustering algorithm and a common choice in many cases, due to its ease of use. As matter of fact, it is not free from weakness. First of all, it requires to pre-define the number of clusters (the K parameter), that in the general case, is not a straightforward choice. Additionally, K-means clusters all the data, not being able to discriminate against noise data, that characterize most of the real-world datasets. As a consequence, this leads to include in the clusters a sensible amount of noise, which affect the quality of the results and the compactness of clusters. A further aspect of K-means that limits its flexibility, relates with the distance metrics adopted, that can not be different from the euclidean one. Beyond the "functional" limitation of K-means, from the non-functional viewpoint, it is very challenging to design scalable distributed clustering algorithms. In fact, albeit K-means is in principle easy to parallelize, it suffers of a large runtime when K is large, and requires a large number of similarity computations.

To overcome the aforementioned limitations and address the issues underpinning the paper, in this work we propose Muchness a framework that is able to estimate the number of residents, commuters and visitors in a given region by exploiting mobile phone data. To this end, this paper provides a set of different contributions:

- *similarity metric*: we defined personalized metrics able to capture similarities on the temporal calling behaviour of the users as well as the number of calls performed;
- *clustering algorithm*: we inject our metrics on an algorithm originally conceived for text clustering [2] and we adapt it to be suitable to our data;
- *real data*: we estimate the population on Tuscany and compare the result with state of the art [1] [3] using real data from Italian national institute of statistics.

This remaining of this paper is organized as follows: Section II introduces the related works, Section III describes the analytical framework while Section IV presents the results we have obtained. Section V details the impact of the research and the future works.

II. Background

In this section we present works related to ours that use mobile data as well as few clustering approaches related to our approach. Mobile phones traces have been utilized to monitor the traffic in cities and analyse tourists movements. In particular two popular works focus on this issues for the cities of Rome [4] and Graz [5]. Other works identify places that could be considered as meaningful by mobile users as work and home points [6]. In addition, a plethora of works, for instance the winner of the Nokia Mobile Data Challenge [7], build predictors able to determine the next position of an individual given the current context. The idea of exploiting mobile phone data for estimating density of population has been first investigated by Deville *et al.* [3] that propose a framework called MP. According to such methodology, the density of a population is estimated as a function of the night-time phone calls occurring in a given area. However, a simple rule-based approach to identify the user presence may hinder to derive some more useful information obtainable by conducting a deeper analysis on the calling data to derive the behaviour of users. For instance, it would be cumbersome to define rules able to characterize individuals that are Commuters or Visitors.

To overcome the aforementioned limitations, in a seminal work Furletti *et al.* [8] defined how to build individual profiles based on mobile phone calls. Such profiles characterize the calling behaviour of a user, in different time slots. By analysing these profiles, it is possible to identify three categories of users: Residents, Commuters or Visitors. Sociometer [1] focuses on this characterization to aggregate users having a similar calling behaviour with the K-means clustering algorithm. The centroid of each cluster is compared with predefined archetypes representing the categories of interest, then, each cluster is classified by means of the associated archetype. Hereafter we use the term *exemplar* to refer to the cluster's centroid. This work advances the achievements of Sociometer in the following areas: *(i)* it performs experiments on a large Italian region (Tuscany) instead of focusing on just two cities (Pisa and Paris); *(ii)* it provides a scalable distributed approach which can process a sensibly larger collection of data; *(iii)* it defines a personalized similarity metric that leads to better clustering results; *(iv)* it automatically removes outliers to improve the overall quality and to provide a better estimation of the population; *(v)* it does not require to provide in advance the number of clusters as in K-means.

Since our work is based on a distributed clustering algorithm it is worth to present a brief comparison covering a few of the widely used categories of clustering algorithm. One of the most popular clustering algorithm is K-means that iteratively aggregates data around K centroids. It has three main limitations: the K parameter has to be user-provided, the distance used to measure data points is limited to the euclidean distance, it has a bias on the initial selection of centroids. Moreover, despite parallel and distributed implementations of K-means exist, they suffer of longer running time when K is large due to the large number of comparisons. Another interesting class of clustering algorithm falls in the dbscan family, defined by Ester *et al.* [9]. The underpinning idea is to cluster items that have at least MINPTS neighbours at maximum distance ε. The main advantages against K-means are the following: *(i)* it is not required to know the number of clusters in advantage; *(ii)* the ability to cluster items with complex shapes instead of aggregating items that are simply close (according to the euclidean distance) to a centroid. MR-dbscan [10] has been the first proposal targeting a distributed implementation of dbscan, realized as a 4-stage MapReduce algorithm.

TABLE I: Overview of frameworks to estimate population

Name	Method	Residents	Commuters
MP [3]	rules on each data	yes	no
Sociometer [1]	clustering K-means	yes	yes
Muchness	clustering k-NN based	yes	yes

Recently, it has been proposed a distributed clustering algorithm based on nearest neighbour graphs [2] able to deal with arbitrary similarity metrics. This is at the basis of the approach presented in this paper because it is possible to inject the metrics defined in Muchness. Albeit this clustering algorithm is suitable for our scenario, it needs to be adapted to our case, since the original algorithm was only tested on text data exploiting the JaroWinkler metric. In addition, our approach returns an exemplar for each cluster to help data scientists to recognize the typology of the clusters without checking each element.

III. MUCHNESS: A FRAMEWORK FOR CENSUS

As we stated above, we propose to derive statistics about population by clustering individuals having similar phone calling behaviour. Then, we analyse the clusters and classify each one as resident, commuter or visitor.

With respect to state-of-the-art approaches, our clustering algorithm provides the following advantages: *(i)* it is scalable and designed for a distributed environment; *(ii)* does not require to know the number of clusters in advance; *(iii)* it is able to handle outliers; *(iv)* it supports arbitrary similarity metrics; Muchness is inspired by two previous works: a k-NN based text clustering algorithm [2] and Sociometer [1] and brings the benefits of both. In the next sections we describe how the data are collected and aggregated, the details of the clustering algorithm and the metrics used to aggregate individuals having similar behaviours.

A. Data description

Telco operators know the micro-cells connecting each of their customers to the network, however, usually they only collect call data records they need for billing purpose. Operatively this means that for each customer they collect information about the cells from which such customer makes calls. Each record consists of a tuple having the anonymous identifier of the user, the call timestamps and the cell id. To perform our experiments, we conducted a spatio-temporal aggregation of call data records within Tuscany (Italy). We manage around 2.6 mln records representing calls generated by about 800k individuals from 115 different municipalities. A municipality is an administrative tessellation of the territory. Our data span between municipalities having a density of population

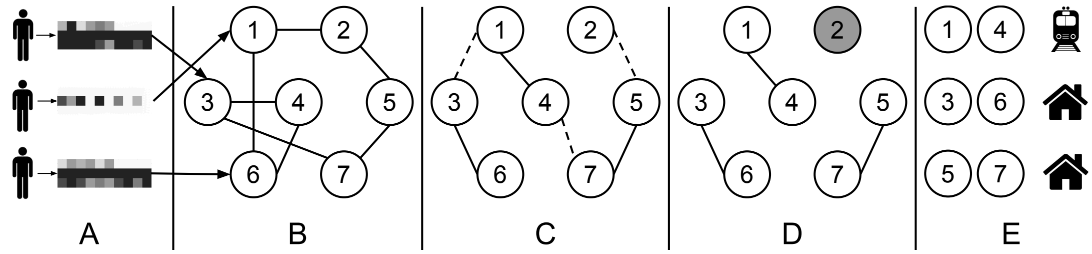


Fig. 1: Muchness analytical process. A : for each individual we assign an ICP. B : each ICP becomes a node in a graph. C : we search for similar nodes and at the end we prune low similarity edges (dashed). D : we search for connected components and we identify outliers (node 2). E : for each cluster we define an exemplar (icons) classified as Resident, Commuter or Visitor.

in the range 6 to 261 individuals per square kilometre. For each user, we compute an Individual Call Profile (ICP), following the approach defined in a paper from Furletti *et al.* [8]. Such approach is based only on the temporal data, considering only the municipalities in which a mobile phone user perform at least one call. Each ICP is a 30-dimensional array in which each position represents a specific time slot of the day (morning, afternoon, evening) discriminating between weekdays and weekends for a total of 5 weeks. A value greater than 0 indicates that the represented user performed at least one call in a specific time slot. The clustering algorithm takes in input the ICPs to provide clusters of individuals and tag such clusters as Resident, Commuter or Visitor. Such information is eventually processed, to estimate the number of residents, commuters and visitors.

B. The clustering algorithm

Our clustering algorithm builds upon the results achieved in a previous work from (a subset of) the authors of this paper. Such work provides a k-NN based text clustering algorithm [2]. In the following of this section we provide a brief description of the main features characterizing such work to help understanding how to choose the correct parameter values, how to introduce specific metrics and help understanding the improvements introduced.

1) The analytical process: Figure 1 gives an overview of the whole analytical process. For each mobile user we build an ICP (see column A). Then, we generate a graph of ICPs. At the bootstrap, we randomly link each node to few other nodes (see column B). Then, the algorithm iterates, starting from the initial graph, adjusting the neighbourhood of each node with most similar nodes. In the following stage, are pruned the edges connecting nodes which similarity is below a given threshold parameter (see column C). The resulting clusters are the connected components [11] derived from the pruned graph (column D). It is worth to notice how in this phase the nodes without neighbours are identified as outliers (Situation represented in Figure 1 by node #2). Finally, for each cluster it is generated an exemplar (column E), used by the automatic classifier to label the clusters as Resident, Commuter or Visitor.

2) Parameter choice: Our proposed solution requires to specify two parameters: k and ε. k represents the number of neighbours for each node in the graph, it affects both the quality and the execution time of the clustering. In general is acceptable to set a value $\in [5, 10]$ to have a good trade-off between quality and time as suggested in Lulli *et al.* paper [2]. ε is a threshold parameter that drive the edge pruning process to avoid that very different nodes would fall in the same cluster. The clustering algorithm starts with a randomly connected graph and is devoted at connecting each node to its k most similar nodes under a given similarity measure. The similarity measure can be arbitrary. In Section III-C we make a deep discussion on the better metrics to be used for our problem with ICPs. The output is an approximated nearest neighbour graph. This is pruned, based on the threshold parameter ε, to remove low similarity edges. The idea is to keep connections between high similarity nodes and break the connectivity between low similarity nodes.

3) Adapting the algorithm for ICPs analysis: In this section we describe the improvements introduced in the algorithm in order to be suitable for ICPs data. In addition, we introduce new functionalities to help data scientists to investigate the data:

a) Injecting an arbitrary similarity metric: One of the claim of the original algorithm is its ability to accommodate arbitrary similarity measures. However, it has been tested only with text data using the JaroWinkler similarity metric. In this work we define specific similarity metrics that are able to exploit the similarity between the ICPs data.

b) Exemplar definition: Due to the large size of the dataset it is necessary to define an exemplar for each cluster. The exemplar is the first entry point to analyse a cluster by a manual investigation. Recall that the data is a d-dimensional array. We define a d-dimensional array as the $exemplar_s$ for each cluster s. Each position i of the array has the value equals to the average of the values in position i of all the elements of the cluster s.

C. Metrics to capture ICPs similarities

In this section we discuss on the metrics to use for our data. As introduced before, each ICP is a 30 dimensional array representing the calling behaviour of an individual in a municipality. We define the *shape* of an ICP equal to the positions of its array where the values are greater than 0. The shape give an idea about the presence of an individual in the

TABLE II: Similar ICPs extracted by expertises. A comparison of similarity values using: EUC, JAC and EUC+JAC

			EUC	JAC	EUC+JAC
Residents			0.5	1	0.8
Commuters			0.78	1	0.91

territory without considering the amount of calls performed. The Euclidean similarity (EUC) is unable to grasp similarities between ICPs having similar shapes. Due to this, our main idea is to introduce a metrics able to capture the similarities between individual sharing a common shape. Next, we present our metrics to improve the quality of the results obtained by the clustering and an example to exhibit its advantages on ICPs data (Table II).

1) How to capture shapes similarity: A metric able to capture the shape of the array is the Jaccard similarity (JAC). In order to use JAC we modify each array in a boolean array where we set the value 1 in position i if in position i the data has a value greater than 0. However, the JAC takes into account exclusively the shape of the profiles but it loses all the informations about the weights in the array. Therefore we combine the two similarities, the EUC and the JAC. We define the EUC+JAC similarity as follow:

$$\text{EUC+JAC}(a,b) = \alpha\text{EUC}(a,b) + (1-\alpha)\text{JAC}(a.b) \qquad (1)$$

Our goal is to identify the shape of the ICPs, due to this is acceptable to put more weight on the JAC. After a careful analysis we identified in $\alpha = 0.4$ an acceptable configuration.

2) Comparing the metrics, an example: We provide an example supporting our idea in Table II. We select some ICPs with the help of expertises representing two residents and two commuters having similar shapes. Table II represents in the first two columns the ICPs selected and in the last three columns the similarity values using different metrics. The ICPs have a very similar behaviour resulting in similar shapes. For instance, take in consideration the two residents in the first row of Table II. Although some positions have different values, note the color darkness representing the value on a single position of the array, they have an equal shape representing the same calling behaviour. With the EUC we cannot assess that the two ICPs are similar (only 0.5 similarity) however the JAC (giving value 1) suggests that the two ICPs have identical shapes. With our EUC+JAC we can take the benefits of both the metrics and we obtain an high similarity of 0.8. Similar considerations can be applied also to the commuters example.

IV. Experimental evaluation

All the experiments have been conducted on a cluster running Ubuntu Linux 12.04 consisting of 5 nodes (1 master and 4 slaves), each equipped with 128 Gbytes of RAM and with a 32-core CPU, and inter-connected via a 1 Gbit Ethernet network.

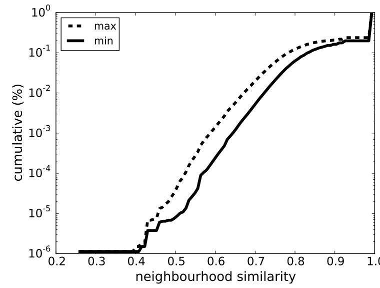


Fig. 2: How to configure Muchness: analysing the distributions of the min and max neighbour similarity (EUC+JAC)

To implement our approach we use Apache Spark [12], our source code is publicly available[1].

To highlight the differences of Muchness with previous approaches we make use of the following competitors:

- Sociometer [1] is the primary competitor, it is the most similar to Muchness because are both based on clustering and designed for the same case study;
- MP [3] targets the same problem, however is not based on clustering and uses rules such as the calling hours to identify if an individual is a resident;
- dbscan, we tried also an implementation[2] of MR-dbscan [10] but we are unable to cluster more than the 10% of the dataset due to memory errors due to the high dimensionality of the ICPs.

A. How to configure Muchness

We start our evaluation helping the reader to understand how to choose the correct values for the parameters k and ε described in Section III-B2.

For what concerns the k value to be used for the k nearest neighbour graph, we refer to the original algorithm for a deeper analysis. However, as suggested by the authors, a value $\in [5, 10]$ is enough to provide a good result. We tested with values 5 and 10 and we obtained really close results in terms of internal clustering evaluation and number of residents and commuters identified.

Next, it is required to define the threshold parameter ε. Recall that this parameter is used to prune all the edges below such value before identifying the clusters. In Figure 2 we show the cumulative distribution of the minimum and maximum similarity in the neighbour list of each node. In particular, the maximum value represents the nodes that became outliers. For instance, setting the threshold parameter equals to 0.8 means that the 90% of the node keeps at least one neighbour in the graph (i.e. they are in the same connected component with other nodes and not outliers). This result refers to the EUC+JAC similarity. The distributions derived by other metrics produce a similar shape and are not included for space constraints. A value around 0.8 represents also the turning point of the

[1] https://github.com/alessandrolulli/knnMeetsConnectedComponents
[2] https://github.com/alitouka/spark_dbscan

TABLE III: Internal clustering evaluation: Compactness and Separation comparisons

	Separation	Compactness
Sociometer	0.77	0.78
Muchness (EUC)	0.67	0.76
Muchness (JAC)	**0.65**	0.85
Muchness (EUC+JAC)	0.72	**0.87**

curve and suggests how to set the threshold parameter. For this reasons, in the following experiments we use $\varepsilon = 0.8$. Note, increasing this value gives a larger number of outliers and a larger number of clusters whereas, a lower value, prunes less edges in the graph and keeps more connectivity resulting in a smaller number of clusters.

Finally, we take in consideration how to set the parameter α of our EUC+JAC similarity described in Section III-C. We tested our approach using multiple values of α and we obtained close results using $\alpha \in \{0.25, 0.55\}$ in terms of clustering quality. In the following experiments we use $\alpha = 0.4$.

B. Internal clustering evaluation

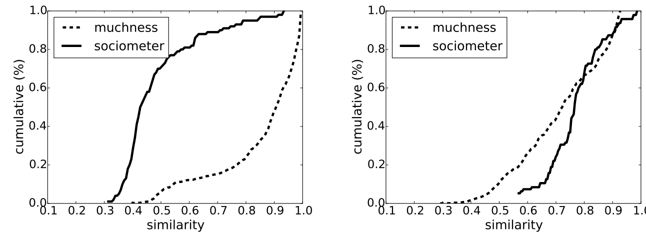
We now evaluate some internal clustering metrics:

- **Compactness**: measures how closely related the items in a cluster are. We obtain the compactness by computing the average pairwise similarity among items in each cluster. Higher values are preferred.
- **Separation**: measures how well clusters are separate from each other. Separation is obtained by computing the average similarity between items in different clusters. Lower values are preferred.

We compare Muchness and Sociometer because are both based on clustering mechanisms (MP is not comparable because does not use clustering). Table III shows the compactness and separation values. Muchness with the EUC metric, the same used by Sociometer, provides almost the same compactness result. However, thanks to its ability to automatically remove outliers it gives a better value of separation. Recall that a lower separation value is better because means that the clusters are more separated.

When Muchness is used with the metrics that take in consideration the shape of the ICPs (JAC and EUC+JAC) it is able to provide clusters having an higher compactness with respect to Sociometer. This result confirms that taking in consideration the interval of time when two different individuals perform a call is meaningful. In particular, using the EUC+JAC similarity we obtained the best compactness value, for this reason, in the following experiments we use EUC+JAC similarity.

Finally, Figure 3 shows the distribution of, respectively, compactness and separation for EUC+JAC. The 80% of the clusters identified by Sociometer have a compactness value lesser than 0.8, instead with Muchness only the 20%. Also, with Muchness the 50% of the clusters have a separation lesser than 0.7, instead with the Sociometer the 30%. This confirms that the majority of the clusters identified by Muchness are more separated with respect to Sociometer.



(a) Compactness(higher is better) (b) Separation(lower is better)

Fig. 3: Internal clustering evaluation: Compactness and Separation distribution (EUC+JAC)

C. Comparing with Official Statistic Bureau

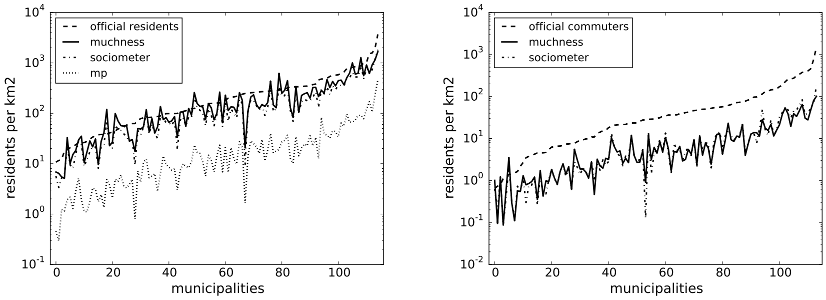
In this Section we evaluate how Muchness is capable of providing an indicator about the number of residents in a municipal area by comparing the results with two state of the art methods: MP and Sociometer. In addition, we evaluate also the number of estimated commuters against Sociometer. Note, the MP method is limited and specialized in providing only the number of residents and does not provide a functionality to estimate commuters. All the estimation made with mobile phone data are rescaled using the market share of our provider.

The results are compared using official census statistics provided by Italian national institute of statistics (Istat). In these data we have the number of residents and commuters of the 115 municipalities under exam.

First we analyse the overall number of residents, commuters and visitors. Table V presents the results. MP provides a number of residents considerably lesser with respect to Muchness, Sociometer and in particular the real data. Muchness using the JAC similarity gives a result really close to Sociometer, otherwise using the EUC+JAC similarity Muchness is able to identify a larger number of residents. In addition using the Sociometer the 60% of the clusters are classified as residents and the size of the clusters is approximately the same. Instead, Muchness (EUC+JAC) provides just one big cluster of residents of nearly the 97% of the total number of residents. We think this is a remarkable result, since this data should be analysed by data scientists, it is useful to have a method able to correctly aggregate near all the residents in a unique cluster.

Next, we compare these results with the real census provided by Istat. Figure 5 depicts the number of residents identified. On the Y axis is presented the density of the population estimated. On the X axis the municipalities ordered by the lower to the higher dense. All the methods have spikes in the same municipalities. This suggests that although the methods are based on different ideas: MP defines rules, Sociometer and ours on clustering, all identify similar behaviours on the data. It is evident that MP is always under estimating the density with a bigger error with respect to Sociometer and Muchness. In particular we divided the error on the estimations in 4 areas having different population density.

Table IV presents the median error on the estimations. Again, MP is providing the estimation affected by the larger



(a) Residents. (b) Commuters.

Fig. 5: Comparing with Official Statistic Bureau: municipalities estimations

	Residents $\times km^2$			
	<50	50 - 100	100 - 150	>150
MP	93%	91%	92%	94%
Sociometer	39%	39%	49%	52%
Muchness	**24%**	**29%**	**42%**	**47%**
	Commuters $\times km^2$			
Sociometer	**83%**	84%	86%	89%
Muchness	84%	**83%**	**81%**	**87%**

TABLE IV: Comparing with Official Statistic Bureau: median estimation errors

TABLE V: Comparing with Official Statistic Bureau: number of Residents, Commuters and Visitors

	Residents	Commuters	Visitors
MP	74 021	N/A	N/A
Sociometer	405 845	21 549	2 224 575
Muchness (EUC)	137 121	7 148	2 231 323
Muchness (JAC)	407 020	12 394	2 175 692
Muchness (EUC+JAC)	432 047	15 187	2 037 022

error. Muchness and Sociometer provide similar result for the denser regions where the volume of data is larger and the clustering have more informations, slightly better results for Muchness. Instead, for less dense regions, in particular for < 50 individuals per km^2 and the range $50 - 100$, Muchness provides the 10% less error with respect to Sociometer. Finally we compare the commuters estimations of Muchness and Sociometer. Also in this case the results are compared using real census data. Both approaches give approximately the same result, in terms of estimation errors, in all the scenarios.

V. Conclusions

We implemented a framework for estimating the population in a territory using the mobile phone data. Respect to the state of the art, we presented personalized similarity metric to capture similarities between individual call profiles, overcoming the limitations of existing approaches which do not use the shape of the profiles (in particular for Residents and Commuters). We make use of a clustering algorithm able to handle arbitrary similarity metric. We adapt it to make it suitable to our data and we define an exemplar for each cluster. We showed, through a detailed experimental campaign that our approach is able to provide better clustering compactness and separation with respect to state of the art approaches thanks to the ability to automatically remove outliers. Also, we showed that we provide a better approximation of the population density within the Italian region of Tuscany and we are able to cluster the majority of the Residents in just one big cluster.

Our next step it is to provide a study of the scalability of the approach and to handle data of a bigger region. Implications of our research are to provide to the public administration a tool, which powered by a continuous stream of phone data, is able to provide useful information to improve the achievements of public services such as transportation and security of the territory.

Acknowldegments

This work is partially supported by the European Community's H2020 Program under the scheme 'INFRAIA-1-2014-2015: Research Infrastructures', grant agreement #654024 'SoBigData: Social Mining & Big Data Ecosystem'. (http://www.sobigdata.eu).

References

[1] L. Gabrielli *et al.*, "City users' classification with mobile phone data," in *Big Data, 2015 IEEE International Conference on*. IEEE, 2015, pp. 1007–1012.
[2] A. Lulli *et al.*, "Scalable k-nn based text clustering," in *Big Data, 2015 IEEE International Conference on*. IEEE, 2015, pp. 958–963.
[3] P. Deville *et al.*, "Dynamic population mapping using mobile phone data," *Proceedings of the National Academy of Sciences*, vol. 111, no. 45, pp. 15 888–15 893, 2014.
[4] F. Calabrese *et al.*, "Real-time urban monitoring using cell phones: A case study in rome," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 1, pp. 141–151, 2011.
[5] C. Ratti *et al.*, *Mobile landscapes: Graz in real time*. Springer, 2007.
[6] R. Ahas *et al.*, "Using mobile positioning data to model locations meaningful to users of mobile phones," *Journal of Urban Technology*, vol. 17, no. 1, pp. 3–27, 2010.
[7] V. Etter *et al.*, "Where to go from here? mobility prediction from instantaneous information," *Pervasive and Mobile Computing*, vol. 9, no. 6, pp. 784–797, 2013.
[8] B. Furletti *et al.*, "Use of mobile phone data to estimate mobility flows. measuring urban population and inter-city mobility using big data in an integrated approach," in *Proceedings of the 47th Meeting of the Italian Statistical Society*, 2014.
[9] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, 1996, pp. 226–231.
[10] Y. He *et al.*, "Mr-dbscan: An efficient parallel density-based clustering algorithm using mapreduce," in *Parallel and Distributed Systems, 2011 IEEE International Conference on*. IEEE, 2011, pp. 473–480.
[11] A. Lulli *et al.*, "Cracker: Crumbling large graphs into connected components," in *Proc. of IEEE ISCC*, 2015.
[12] "Apache spark," https://spark.apache.org.