Research on Online Digital Cultures - Community Extraction and Analysis by Markov and k-Means Clustering

Giles Greenway(✉), Tobias Blanke, Mark Cote, and Jennifer Pybus

Department of Digital Humanities, King's College London, London, UK
giles.greenway@kcl.ac.uk

Abstract. We investigate approaches to personal data analytics that involves the participation of all actors in our shared digital culture. We analyse their communities by identifying and clustering social relations using mobile and social media data. The work is part of our effort to develop tools to create a "social data commons", an open research environment that will share innovative tools and data sets to researchers interested in accessing the data that surrounds the production and circulation of digital culture and their actors. This experiment focuses on the groups of clustered relations that are formed within a user's social data traces. Community extraction is a popular part of the analysis of social data. We have applied the technique of Markov Clustering to the Twitter networks of social actors. Qualitatively, we demonstrate that it is more effective than the Louvain method for finding social groups known to the subjects, while still being very simple to implement. We also demonstrate that traces of cell towers captured using our "MobileMiner" mobile application are sufficient to capture significant details about their social relations by the simple application of k-means.

1 Introduction

Applications, especially social media applications, on mobile devices are frequently given access to information about their users' location, through information about cellular network towers, wireless networks and full GPS access. When this is combined with the media or messages the users choose to send or receive, an application can learn a great deal about them. This data is often used to target advertising or recommend other content or connections with other users. Some studies are able to negotiate with mobile service operators for the use of customers cell tower data, under certain conditions. Others have made use of mobile application data through agreement with social media companies or use of their APIs within terms-of-service. Levandoski *et al.* [1] collected data from the Foursquare service using its public API, but a page on the University of Minnesota website [2] states that this is no longer available at Foursquare's request. Since this data is anonymised, it is not possible to use it in discussions with mobile device users about their attitudes to data and privacy.

R. Guidotti et al. (Eds.): PAP 2017, LNCS 10708, pp. 110–121, 2017.
https://doi.org/10.1007/978-3-319-71970-2_10

One way to secure such data for long-term use without conditions is to collect it from end-users with their full informed consent and participation. To this end, we have collaborated with 20 young programmers recruited via Young Rewired State [3] (YRS). An Android application provisionally called MobileMiner [4] was developed that logs data commonly gathered by other mobile applications, as well as their network activity. The application was developed in public, with the source-code made available under version 2 of the Apache license so the YRS members could be aware of its activities and participate in its development. They were issued with smartphones with the application installed, and used it to log data over 5 months. During this time, we held two hackday events attended by the YRS members. The first had a theme of application development, the other geared towards returning the data to the participants so they could analyse if for themselves [5]. We have already reported on MobileMiner's collection of network socket data [4]. Here, we investigate the insights that can be gained by its capture of the mobile cell towers user's devices connect to. In order to reason about what insights third-party app developers could gain from this data, given they also have the users' generated content, we also analysed the Twitter networks of the YRS volunteers.

The exponential growth of both transactional data and metadata that is generated within social media platforms raises questions in relation to how data is archived and assigned value [6,7]. Commercial services such as Conversocial [8] and SproutSocial [9] provide the means for businesses to analyse the perception of their brands in social media, in order to manage customer complaints and capitalize on positive reviews. Others, like Twitonomy [10] and Simply Measured [11] track the effectiveness of social media campaigns (how far a Tweet is retweeted, for instance) and identifying key influencers. The TrendMiner [12] project also enables the mining of social media data, with the stated goals of enabling political analysis and economic decision-making.

The focus of these tools is seldom directed towards users and how they use social media in their everyday lives. We believe that there is a need for tools that allow social media users to gain awareness of how the data they generate are being used. In light of analysis of this data, users may assess how they might want to change their use of such platforms. To this end, we provide a "Social Data Commons" in the "Our Data Ourselves" project [13]. This kind of work is often dominated by commercial interests who treat the content produced by users as the central point of value in exchange for "free" online services.

Twitter has demonstrated to be a rich source for network analysis and communication patterns. It allows us to investigate, for instance, the degree to which users are consumers of content from a few accounts with many followers or whether they use the platform conversationally between other users with reciprocated follower relationships. The fraction of these relationships which have involved sustained conversations and the frequency distributions of their lengths has also been studied [14]. Lastly, the degree to which a user's friends and followers network can be segmented into communities or clusters is of key interest. Members of such clusters will have more friend/follower relationships between

them than with those in other groups. Himelboim *et al.* [15] used NodeXL spreadsheet tool [16] to retrieve tweets containing certain keywords, then cluster the accounts that tweeted them into communities. It was observed that there were far more interactions within communities than between them.

Our experiment uses the Markov Clustering (MCL) graph-clustering algorithm, [17] which is very simple to implement. We apply this algorithm to analyse how sociality develops within the networked relationships around a Twitter account.

2 The MobileMiner App

The Android operating system was pragmatically chosen as the mobile platform for the project; it has the largest share of the smartphone market and requires less investment to release code and apps. Following our co-development approach with YRS, it was essential that the YRS members would be able to contribute to app development. The Apple IOS, for instance, limits these options significantly. The tool-chain for development on Apple's IOS is only available for its MacOS operating system. MacOS can be run on virtual machines without access to Apple hardware, but not without breaching its end-user license agreement. In order for the YRS members to send and receive updates to MobileMiner and install it on their devices, all would require access to Apple hardware. The only other way to distribute the app would be via Apple's I-tunes store, which would have been subject to its approval. In contrast, the Android operating system gives the user the option of installing software from sources other than Google's Play store. The Android development tool-chain or software development kit (SDK) is cross-platform, and released under the Apache open-source license, albeit with restrictions on the distribution of the supplied binary executable files.

The MobileMiner Android app gives users the option to start and stop recording data whenever they choose. It displays the active mobile or wireless network, the id of the current cell tower, and recent app activity in terms of network socket usage (Fig. 1). Newly gathered data is written to a SQLite database on the device at 5 min intervals. Storing each item as it arrived would cause very frequent writing to the device's flash storage, and have a very severe effect on battery life. To avoid accruing mobile data costs to the user, new data is uploaded by an http request to a server every 10 min only if the device has a connection to a wireless network. There are some basic features to examine the collected data on the device. A list of apps ordered by the number of network socket events is provided, and the users may also request a heat-map showing how frequently they visit the cell towers known to the OpenCellID [18] database. The maps are drawn using OpenStreetMaps [19] via the OpenLayers JavaScript library. Users have the option to copy the app's SQLite database to an accessible part of their device's flash storage. Users do not have access to the databases of the apps on their devices by default; root access and the SDK are required. The YRS members were encouraged to access their data using the SQLite module

in the Python standard library during the first hack-day, although SQLite apps are available for examining the data on the device.

Android apps may request GPS location data from two sources, either from the device's sensor or via Google's Play Services API. The latter is combined with location data from cell towers and wireless networks, but requires agreeing with Google's terms and conditions, as does the display of Google's Maps within an app. The precision afforded by such data could allow individuals to identified by their home address. Frequent and repeated use of the GPS sensor would also significantly reduce the device's battery life. It is therefore more attractive to use the locations of cell towers as a proxy for approximate location. Apps using Android's 'coarse location' permission are fed location information based on cell towers and wireless networks, but short of disabling wireless connections on the device, there is no way to limit this to cell towers only. The extra precision of location afforded to the coarse location API by adding wireless network information is too high for it to be captured by the app. MobileMiner uses the Android API to collect the IDs of cell towers as the device connects to them, then finds their locations using data from the OpenCellID database. This contains crowd-sourced measurements of the locations of cell towers, combined with known locations from some mobile network operators that freely publish this information. Unlike collecting cell IDs only during call or SMS events from the logs of mobile operators, [20] these trails of locations are continuous as long as the app is in use, subject to the sparseness of the OpenCellID database.

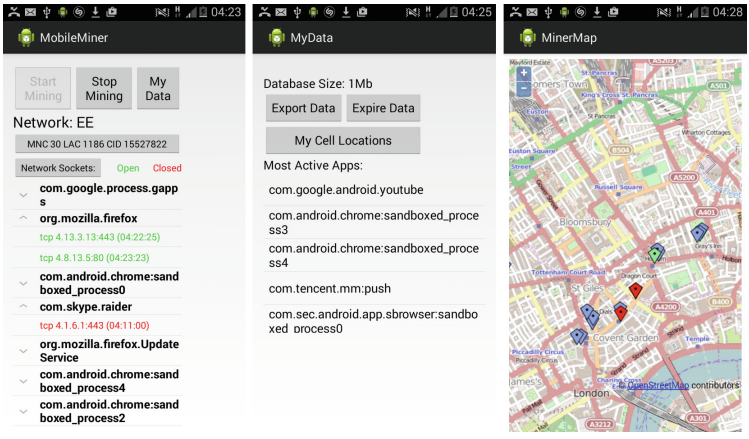

Fig. 1. Screenshots of the MobileMiner app, showing network sockets, apps ordered by total activity, and a heat-map of cell towers.

3 MCL

MCL community detection works by reasoning about random walks on a network, for which an adjacency matrix is created. The columns are normalized,

such that entries for each column node represent the chance of visiting a connected node chosen at random. Squaring the matrix produces columns of probabilities of node occupancy after two such random steps, starting from the column node; this stage is called expansion. The elements of the matrix are then raised to a power called the inflation parameter (2 in this case) and the columns are renormalized. This has the effect of raising higher probabilities and reducing lower ones. The two steps are repeated until the matrix converges, when most of the entries will be close to 0 or 1. The non-zero entries in a row will represent probable starting nodes for a random walk, given that it ended in the row node. Assuming that a random walk starting in a densely connected community of nodes is more likely to remain within it than to leave it, the sets of nodes with non-zero entries for each row represent clusters.

3.1 Acquiring Twitter Data

Before applying MCL to the Twitter accounts of the YRS volunteers, we tested it using that of a digital culture researcher, also an author of this paper. User profiles were extracted with the Twitter API and stored in the Neo4J graph database. The properties returned by the API call were used to form the attributes of each node representing a user. The users' friends and followers were retrieved by repeated API calls; these relationships were modeled by creating directed edges between the nodes in the database. Having seeded the users in the Neo4J graph, the friends and followers were expanded in the following way, using Neo4J's graph query language Cypher:

1. Find set of user nodes who have a friendship relationship with the seed user in any direction.
2. Find the set of user nodes who have a relationship with any users in the previous set.
3. Find the count of outgoing (friends) or incoming (follower) relationships already within the database for each node in this set.
4. Compare the count to the expected number of relationships as described by the "friends_count" or "followers_count" attributes returned by the Twitter API calls.
5. Return the users for which the count is lower, in ascending order of when they were retrieved.
6. Expand these users friend-follower relationships and repeat.

3.2 Applying and Evaluating MCL

A set of Twitter users with bi-directional follower relationships was constructed with another query in order to be analysed with MCL, implemented using NumPy [21]. All such relationships were returned for users within two relationships of the target user. An adjacency matrix was constructed to represent this network, with the diagonal filled with 1s to add a self-loop to each node. Having normalized the columns, the matrix was squared, raised to the power of

an inflation parameter of 2.0 element-wise, and renormalized. The process was repeated until the standard deviation of the absolute differences between successive matrices was less than 10^{-3}. Clusters were extracted by considering rows with more than three non-zero elements. For the first Twitter account examined, the graph matrices was around 7000×7000 in size, with the algorithm converging in around 25 iterations.

We chose to evaluate our implementation of MCL by comparing it with an out-of-the-box approach and then let the researcher evaluate the results and label them. To this end, we used the popular graph analysis tool Gephi [22], which implements the Louvain clustering algorithm [23]. This algorithm involves successively joining clusters together such that modularity, which measures the density to intra-cluster links to inter-cluster ones, is maximized. The researcher then rated each of their clusters returned on a scale of 1–3, where 1 indicated that the individuals within the cluster were connected by a clearly identifiable common denominator and research goal, 2 that the individuals seemed connected, but the context was not clear, and 3 that the individuals seemed to have no connection. The researcher was not involved with the collection of data or the implementation of MCL, and was unaware of which methods were used to generate the clusters. This semi-automated approach has proven effective for the analysis of clusters [24].

The Twitter account had 7144 others within two reciprocal friend/follower relationships. The MCL algorithm placed 55% of these in clusters with more than three members, the remainder were considered unclustered. Just over 8% of the accounts were in the largest cluster, the next largest had 5%, 5% and 4% of users respectively. This tendency towards large numbers of small clusters suggests that MCL may be over-fitting by partitioning larger clusters into smaller ones. The clusterings are summarized in Table 1.

At the time of capturing the data, the account had 319 friends and 346 followers, with 100 users in both groups. The qualitative cluster labeling exercise resulted in 20% of clusters being given the highest grade of relevancy, these contained 195 Twitter accounts. 37% of clusters received the second highest grade. Given that the vast majority of the Twitter accounts in the network were unknown to the target user, the fraction of clusters with no apparent relevance does not seem unduly high. MCL has clearly been able to extract some meaning from the account's connections, with many clusters clearly recognizable as belonging to specific institutions or conferences. All of the clusters obtained by the Louvain method received the lowest grade of relevance. Qualitatively, MCL clearly extracts meaningful communities as observed by the owner of the Twitter account surveyed, and has the advantage of clarity of application. There is no definitive quantitative definition for graph clusters [25], nor a definitive quantitative method for evaluating them [26].

3.3 Results

The same Twitter-crawling algorithm was applied to the 9 YRS volunteers with Twitter accounts, and MCL and the Louvain method were applied to

Table 1. Fractions of clusters achieving each relevancy score, for MCL and the Louvain method.

	MCL	Louvain
1. Cluster is identifiable and relevant	20%	0%
2. Cluster is not identifiable, but possibly relevant	37%	0%
3. Cluster is neither identifiable nor relevant	43%	100%

all the accounts collected. MCL returned 13 clusters, with all but one of the YRS accounts placed in the largest, with 319 accounts as shown in Table 2. The remaining YRS member was placed in the second largest cluster, with 45 accounts. The Louvain method failed to identify the YRS members as a coherent group, and distributed them across four clusters with 1–3 members in each. If the Twitter communities detected by MCL are to have true social context, it is reasonable to expect them to Tweet about well-defined topics. This is clearly the case, as shown by the hashtags used in the MCL clusters containing the accounts of YRS members in Table 3. Both clusters are dominated by the 2015 UK general election, but the cluster with more YRS members predominately uses tags about the YRS organization and technology. The smaller cluster is mostly concerned with UK politics.

Table 2. Distribution of YRS member Twitter accounts in clusterings by MCL and the Louvain method.

MCL	cluster size	20	26	6	6	5	45	5	319	6	5	14	14	5
	YRS accounts	0	0	0	0	0	1	0	8	0	0	0	0	0

Louvain	cluster size	15	78	7	43	168	67	55	230	24
	YRS accounts	0	1	0	0	0	3	2	3	0

4 Mobile Cell Tower Data

When MobileMiner is in use, it creates a time-series of cell tower IDs, which using the standard Android Java API. Many of the captured cell towers were known to the OpenCellID database. We explore how much information these trails reveal about the users. One of the most prolific YRS users of the app visited and logged 930 unique cell towers in 148 days. The latitudes and longitudes of 253 (27%) of these were known to OpenCellID.

A simple approach to analysing user behaviour is to cluster the cell towers spatially, [20] and then examine the times of occupancy of the clusters. k-Means clustering is best suited to convex clusters in a low-dimensional Euclidean space, these conditions are met by the cell-tower data reasonably well. The k-means

Table 3. Hashtag usage for MCL clusters containing YRS members.

8 YRS accounts		1 YRS account	
Hashtag	Tweets	Hashtag	Tweets
GE2015	275	GE2015	78
tech	214	Eurovision	58
jobs	207	leadersdebate	33
YRS2014	185	bdw2015	24
Haunted	183	BattleForNumber10	21
ghosts	183	BBCQT	18
YRSFoc	181	GBR	15
hackmcr	167	bbcqt	14
yrs2014	156	eurovision	14
Arduino	149	NHTG15	13
FoC2015	141	FoC2015	12
Norwich	133	YRSAmbassadors	11
gamedev	132	depop	11
TG	130	BBCFreeSpeech	10
BigData	112	VoteConverative	9
linux	111	YRS2014	9
YRSHyperlocal	105	DimblebyLecture	9
design	99	endpointcon	9

routine from the SciKitLearn Python package [27] was applied to the normalized location data for increasing values of k clusters, until the mean distances of the points to their assigned cluster centres stopped decreasing appreciably. In this case iteration stopped when the mean distance was greater than an arbitrary fraction of 90% of the previous lowest value.

Figure 2 shows the clustering using simple feature vectors consisting only of latitude and longitude. It is attractive to capture journeys, as well as places. Extended groups of points near the centre of the distribution are split into three clusters. It is likely that these are part of a single journey, with the breaks between the clusters being accounted for by poor reception of the mobile network. To mitigate this, the changes in latitude and longitude between successive points were divided by the time intervals between them to estimate the velocity at each point. The clustering was repeated using feature vectors consisting of both position and velocity, as shown in Fig. 3. Points that are part of the same car or train journey will have similar velocities, and can therefore be assigned to the same cluster. A mobile device might connect to a new cell tower because of network traffic or temporary loss of reception, rather than any change in proximity, so this measure of velocity cannot be expected to be realistic. Such random fluctuations should tend to cancel for clusters consisting of groups of cell

towers that corresponded to places rather than journeys. These remained stable for both clusterings.

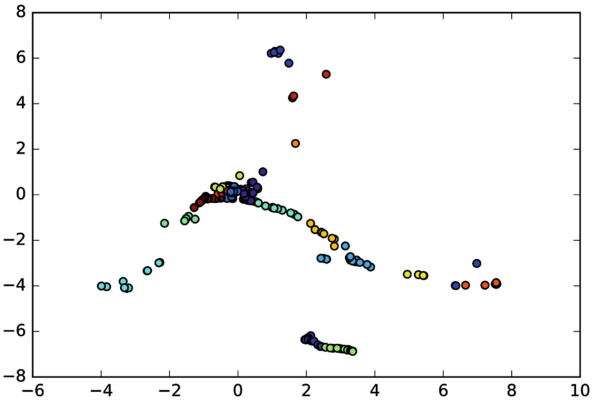

Fig. 2. Cell tower locations with normalized latitude and longitude for a single user, clustered by K-means using positions as feature vectors.

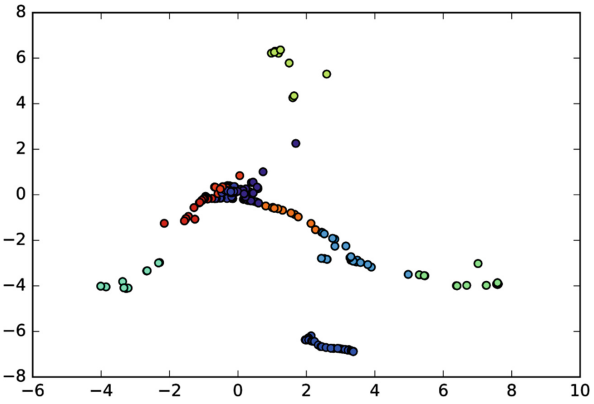

Fig. 3. Cell tower locations with normalized lattitude and longitude for a single user, clustered by K-means using positions and velocities as feature vectors.

4.1 Temporal Analysis

The sequences of known cell tower locations for each user were clustered by k-means using positions and estimated velocities as feature vectors and each cell tower in a sequence was labeled with its assigned cluster. For each cluster,

the number of days between its first and last visit and the number of days with at least one cell tower connection were determined. The total number of measurements in the cluster on weekdays and weekends were multiplied by $\frac{7}{5}$ and $\frac{7}{2}$ respectively. If the weekday product was greater than 5 times the weekend product, the cluster was deemed to have been mostly visited during weekdays, if the weekend product was greater than 5 times the weekday product, the cluster was considered to have been mostly visited during weekends. The histograms of the hour of each cell tower connection in each cluster were found. The earliest and latest hours where the number of measurements was at least one fifth of the maximum values were used to estimate the range of time when each cluster tended to be occupied. The name of each cluster's location was found by querying OpenStreetMaps [19] API with the latitude and longitude of its centre.

For the user whose data was plotted in the previous figures, this approach yielded a cluster active during all hours of the day for all days of the week, that correctly corresponded to the district where they lived. Another, active during weekdays from 09:00 to 19:00, plausibly connected to a daily commute, was centred close to the school they attended. During some runs of the k-means algorithm, the OpenStreetMaps API correctly identified the school by name. It is likely that this would been achieved this more reliably if MobileMiner recorded full GPS locations, as many commercial apps do. There were three other clusters of note, each in a UK city, with visits on two days each. Given that the user was attending a school, over 16 years of age, and the three cities hosted major universities, a reasonable assumption would be that they were attending open days, then interviews at the universities. This was further reinforced by the appearance of these universities in the logs of wireless networks, and confirmed when the user was interviewed at the second hack-day [5].

4.2 Predicting User Behaviour

The basic approach of examining the distribution of times of measurements within the spatial clusters yielded detailed and accurate information about users' behaviour. However, it would be attractive to predict behaviour as well as summarize it. The cluster identified as a school was occupied as late as 19:00, rather late for the UK school system. This may have been caused by an after-school activity regularly held on certain days of the week. Cluster occupancy was modeled using SciKitLearn's [27] implementation of Random Forests. A sequence of over 15400 cell tower IDs and their hour and day of occupancy for a single user were split into training and test data-sets in the ratio 4 to 1. Using the hour and day of the week as features allowed the cell tower ID to be predicted on only 20% accuracy. This made no attempt to use the physical locations of cell towers or deal with fluctuations in connections. The cell IDs were then replaced with their cluster-labels from the position and velocity based k-means clustering. Since only cells known to the OpenCellID database could be used, the data-set was reduced to a size of 5670 cell connections over 100 distinct days.

94% of cluster visits were to two clusters, with 62% and 32% of visits each. This time the same split between training and test data produced a Random

Forest classifier able to predict cluster occupancy for a given time and day with over 99.9% accuracy. Obviously, the data in this study is very heavily skewed, but the classifier could not achieve this by only correctly identifying the two most popular clusters. A dummy classifier that merely reproduced the class distribution of the training data could only achieve around 50% accuracy, naive Bayes scored 75% using the same features. Gatmir-Motahari *et al.* [28] conducted a study using a large amount of network operator data, achieving over 90% accuracy in place prediction having carefully considered factors including time of day, day of the week and conventional working hours. They concluded that their subjects may have led particularly regular lifestyles, others from more rural and less affluent areas may have been harder to predict. Song *et al.* [29] quote 93% as a more reasonable limit to predictability for cell tower traces. The random forest classifier may have over-fit significantly, and might not cope well with changes in routine. However, it does serve to demonstrate to subjects that their behaviour can be modeled with quite sparse and coarse data.

5 Conclusions

While small datasets from crowd-sourced collections of location and social media data cannot produce the insights of those from datasets where access is granted by service providers, they can give a good indication to participants of what insights are possible. MCL has shown how social media can identify groups with which users associate, and their social context. The knowledge gained about users by social media platforms is enhanced by their use on mobile devices when access to location data is granted. A simple two-step process of clustering cell tower trails followed by supervised learning of cluster occupancy over time is sufficient to demonstrate this; actively engaging participants in their attitudes to their data and privacy.

References

1. Levandoski, J., Sarwat, M., Eldawy, A., Mokbel, M.: LARS: a location-aware recommender system, pp. 450–461 (2012)
2. Wayback machine archive of the University of Minnesota website. http://web.archive.org/web/20161202171606/http://www-users.cs.umn.edu/sarwat/foursquaredata/
3. Young rewired state website. https://yrs.io/
4. Blanke, T., Greenway, G., Pybus, J., Cote, M.: Mining mobile youth cultures. In: 2014 IEEE International Conference on Big Data, pp. 14–17 (2014)
5. Pybus, J., Coté, M., Blanke, T.: Hacking the social life of big data. Big Data Soc. **2**(2) (2015). https://doi.org/10.1177/2053951715616649
6. Beer, D., Burrows, R.: Popular culture, digital archives and the new social life of data. Theor. Cult. Soc. **30**(4), 47–71 (2013)
7. Pybus, J.: Social networks and cultural workers. J. Cult. Econ. **6**(2), 137–152 (2013)
8. Conversocial website. http://www.conversocial.com/
9. Sproutsocial website. http://sproutsocial.com/

10. Twitonomy website. http://www.twitonomy.com/
11. Simplymeasured website. http://simplymeasured.com/
12. Preotiuc-Pietro, D., Samangooei, S., Cohn, T., Gibbins, N., Niranjan, M.: Trendminer: an architecture for real time analysis of social media text. In: 6th International AAAI Conference on Weblogs and Social Media (ICWSM 2012), June 2012
13. Big social data project website. http://big-social-data.net/
14. Bruns, A.: How long is a tweet? Mapping dynamic conversation networks on Twitter using Gawk and Gephi. Inf. Commun. Soc. **15**(9), 1323–1351 (2012)
15. Himelboim, I., McCreery, S., Smith, M.: Birds of a feather tweet together: integrating network and content analyses to examine cross-ideology exposure on Twitter. J. Comput. Med. Commun. **18**(2), 40–60 (2013)
16. Nodexl website. http://nodexl.codeplex.com/
17. Van Dongen, S.: Graph clustering via a discrete uncoupling process. SIAM J. Matrix Anal. Appl. **30**(1), 121–141 (2008)
18. Opencellid website. https://opencellid.org/
19. OpenStreetMap Contributors: Planet dump (2017). https://planet.osm.org, https://www.openstreetmap.org
20. Isaacman, S., Becker, R., Cáceres, R., Kobourov, S., Martonosi, M., Rowland, J., Varshavsky, A.: Identifying important places in people's lives from cellular network data. In: International Conference on Pervasive Computing, pp. 133–151 (2011)
21. van der Walt, S., Colbert, S.C., Varoquaux, G.: The NumPy array: a structure for efficient numerical computation (2011)
22. Bastian, M., Heymann, S., Jacomy, M.: Gephi: an open source software for exploring and manipulating networks (2009)
23. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theor. Exp. **2008**(10), P10008 (2008)
24. Ingersoll, G.S., Morton, T.S., Farris, A.L.: Taming Text. Manning Publications, Shelter Island (2013)
25. Schaeffer, S.E.: Graph clustering. Comput. Sci. Rev. **1**(1), 27–64 (2007)
26. Brandes, U., Gaertler, M., Wagner, D.: Experiments on graph clustering algorithms. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 568–579. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39658-1_52
27. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
28. Gatmir-Motahari, S., Zang, H., Reuther, P.: Time-clustering-based place prediction for wireless subscribers. IEEE/ACM Trans. Netw. **21**(5), 1436–1446 (2013)
29. Song, C., Qu, Z., Blumm, N., Barabási, A.L.: Limits of predictability in human mobility. Science **327**(5968), 1018–1021 (2010)