Machine Learning with Applications 6 (2021) 100084

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Evolutionary clustering and community detection algorithms for social media health surveillance

Heba Elgazzar *, Kyle Spurlock, Tanner Bogart

School of Engineering and Computer Science, Morehead State University, Morehead, KY 40351, USA

ARTICLE INFO

Keywords:
Unsupervised machine learning
Evolutionary clustering
Community detection
Social networks
COVID-19
Health surveillance

ABSTRACT

The prominent rise of social networks within the past decade have become a gold mine for data mining operations seeking to model the real world through these virtual worlds. One of the most important applications that has been proposed is utilizing information generated from social networks as a supplemental health surveillance system to monitor disease epidemics. At the time this research was conducted in 2020, the COVID-19 virus had evolved into a global pandemic, forcing many countries to implement preventative measures to halt its expanse. Health surveillance has been a powerful tool in placing further preventative measures, however it is not a perfect system, and slowly collected, misidentified information can prove detrimental to these efforts. This research proposes a new potential surveillance avenue through unsupervised machine learning using dynamic, evolutionary variants of clustering algorithms DBSCAN and the Louvain method to allow for community detection in temporal networks. This technique is paired with geographical data collected directly from the social media Twitter, to create an effective and accurate health surveillance system that grows as time passes. The experimental results show that the proposed system is promising and has the potential to be an advancement on current machine learning health surveillance techniques.

1. Introduction

The rise of social media over the 21st century has been unprecedented in the way it has connected modern society together. Its use has become so common throughout the world that it can be considered as a world in and of itself, with people from across the globe having the ability to interact with one another in ways never before experienced. Due to the tremendous amount of information that can be discovered on these platforms, there have been many proposed applications for how it can be most effectively utilized. One that is of highest precedence within the current year is what it is capable of inferring in the way of health around the world, through its utilization with health surveillance techniques.

Health surveillance systems play a key role in putting forth defensive measures to contain an outbreak of disease before it has a chance to propagate to neighbouring areas. Despite how important such a system is, the traditional methods currently in place are often considered to be slow and inconsistent in their collection and distribution of information. Largely this can be attributed to the overall difficulty and the wide range of variables that take place in the reporting scheme of these traditional surveillance systems. In many of these systems such as the National Notifiable Disease Surveillance System (NNDSS) operated by the Centres for Disease Control (CDC), the accurate reporting of cases of a disease are purely limited to a localized regions level of participation in the system (Haston & Pickering, 2019). What this means for the overall quality of the collected data is that some areas may be omitted from the overall statistical information present in the event of an epidemic, and before there is time or information available to appropriately respond the disease has already been allowed to spread (Haston & Pickering, 2019).

It is the intention of this research to explore different means of health surveillance that may prove more effective, or at least aid in creating a more robust system that is capable of deriving beneficial information from a variety of sources. In the case of social media, it is a simple task to request access to the APIs of some of the biggest sites, the one of which considered in this research being Twitter. The following sections will describe an application of collected data from Twitter by considering it as a time-based, temporal network that is capable of expanding without negatively impacting the clustering of previous iterations of the network. This is especially beneficial in the case of social medias, were a dynamic approach most naturally captures the stream of information that enters these sites. Through using modified versions of two novel community detection algorithms:

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: h.elgazzar@moreheadstate.edu (H. Elgazzar), kdspurlock@moreheadstate.edu (K. Spurlock), tpbogart@moreheadstate.edu (T. Bogart).

https://doi.org/10.1016/j.mlwa.2021.100084
Received 23 March 2021; Received in revised form 18 June 2021; Accepted 21 June 2021
Available online 24 June 2021

DBSCAN and the Louvain method, the efficacy of this approach will be tested to determine if it has a positive return on the results of clustering, and what this may mean for this application as an effective health surveillance technique. Additionally, through the format of the collected data this approach will focus on real-time disease mapping through considering the geographical location of potentially infectious individuals. The main contributions of this work are as follows:

- Design and implementation of a social media web scraping system to collect temporal data of potential instances for an infectious disease based upon some metrics. A pipeline has been detailed that is capable of directly connecting unsupervised machine learning techniques to an instantly transmittable, dynamic source of data.
- Design and implantation of evolutionary density-based clustering and evolutionary Louvain method for social media health surveillance and showing how the evolutionary methods make a noticeable difference in the quality that is inferable from one generation of the network to the next. The proposed evolutionary methods has the potential to be an advancement on current machine learning health surveillance techniques.
- The prioritization of geographically identifiable data combined with timestamps has allowed a seamless connection to applying evolutionary methods on information that reflects the real-world.

2. Related work

2.1. Big data and health surveillance

Research conducted by Hay, George, Moyes, and Brownstein (2013) examines the proposed benefits that big data has on disease mapping utilizing a real-time simulation of disease presence within a geographical region. They first examine the difficulties of acquiring this information by hand, listing such an example as the sheer processing time required to accurately map this information manually by using varied disease reports (Hay et al., 2013). Further cementing the idea that the current disease reporting infrastructure has not advanced quickly enough to be able to accurately report on disease, with maps becoming quickly outdated before their spatial information can be put to proper use (Hay et al., 2013) Through considering the elements that make big data a valuable source of information: volume, velocity, and variety, they argue that the temporal collection of such sources would have a notable impact on the processing time of disease mapping (Hay et al., 2013) In their research they examine the effectiveness of search queries and blogging data as a source for measuring disease prevalence within a population, with an additional idea of weighted reliability applied to prevent reporting bias (Hay et al., 2013) With the combined information available from these sources, it is proposed that the data available through this public inquisition would be able to provide an effective baseline for measuring disease spread and its history within a region, allowing for further wide-scale efforts to be more effective at accurately mapping the spread of disease in a timelier manner (Hay et al., 2013). Challenges presented by this suggested framework largely concern big data's potentially novel approach in conveying meaningful information, as well as the ability of machine learning to convert this information into actable or interpretable analyses (Hay et al., 2013). The level of explainability in these models would have to show their equitability in these results as well to find parties willing to accept them as potential health-surveillance avenues (Hay et al., 2013).

Another work by Woolhouse, Rambaut, and Kellam (2015) also examines the potential that supplemental data has in connection with health surveillance, citing that past poor and inefficient reporting of the Ebola virus disease (EVD) as being detrimental to the efforts of reducing its spread (Woolhouse et al., 2015). They again propose that data collected in real-time with geographical components would be the most effective measurement in discovering the transmission network of a disease (Woolhouse et al., 2015). Furthering this idea, they also speak on the sequencing of the virus genome and attaching it to the location in which it occurs to further analyse the dynamics of how such a disease propagates (Woolhouse et al., 2015) By combining these two elements they recognize several additional methods which could be effective in considering this information, such as risk mapping and statistical modelling through methods such as regression (Woolhouse et al., 2015). Challenges to this proposed type of system come largely from the quality and volume of data able to be collected, citing issues related to countries with low-health infrastructure with many rural areas, cultural differences, and a lack of a suitable global health surveillance system for sharing complete measurements of disease instances and impact (Woolhouse et al., 2015).

Considering heavily on the task of collecting spatially recognizable, temporal data this research presents the application of evolutionary clustering to perform the task of finding patterns within a set of location mapped data. There are many other works besides the ones summarized that additionally detail methods of utilizing search queries and social media data to discover the trends associated with disease occurrences (Chae, Kwon, & Lee, 2018; Dion, Abdelmalik, & Mawudeku, 2015; Shin, 2016). However, the intention of the approach in this work is to specifically hone in on individuals that may be the most insinuated to be a carrier for a virus, in order to offer an additional method of capturing a diseases presence within a region of the world. Subsequently, these methods aim at finding subset communities of these potential disease instances using clustering in order to determine how one region may influence the transmission in another.

2.2. Evolutionary clustering

Ordinary clustering itself stands as an incredibly beneficial tool in machine learning for its ability to detect patterns within a set of data that would not immediately be evident. The clustering problem can be defined as follows: given a set of n instances for $X = \{X_1, X_2, \ldots, X_n\}$, the goal is to assign these points to a number of k clusters for $C = \{C_1, C_2, \ldots, C_k\}$, where points within the same cluster are the most similar to each other in the global scope of the network, and in some way dissimilar to the points in neighbouring clusters determinable based upon some metric of similarity between points (Cole, 1998; Xu & Tian, 2015) This metric of similarity can be derived from many elements of the network but can often be described and utilized with several algorithms as the distance between points, most commonly determined using the Euclidean distance measurement (Xu & Tian, 2015). This process is perfectly suitable for networks that contain a nonchanging value of n, however capturing accurate clusters within a dynamic network requires more complex methods to ensure its efficacy.

Evolutionary clustering is a relatively new technique first proposed by Chakrabarti, Kumar, and Tomkins (2006) that allows for the clustering of temporal networks where the number of instances is not fixed, but instead grows as time passes. The goal of evolutionary clustering is to appropriately account for this growth in determining the clusters for the next generation of the network, using information that is available in the history of said network (Chakrabarti et al., 2006) Some of the proposed benefits that this form of clustering has over its static counterpart can be stated as the following (Chakrabarti et al., 2006):

(1) Future generations of the network will be similar in form to previous generations, providing a consistency between them that is recognizable from generation to generation.
(2) By allowing the clustering algorithm to learn from the structure of past generations a noise reducing effect can be achieved by adjusting cluster assignments in correspondence with previously seen data points.
(3) Temporal smoothing allows even for shifted networks to retain some semblance of their past generations through minor adjustments based on a previous generation's structure.

To accomplish these benefits, there has been many frameworks proposed for how to best implement the networks transition from one time into the next. Chakrabarti et al. (2006) propose a variable C_t referable to as the snapshot quality of the network at a current time *t*. The history cost of the clustering at a time t is then taken to be the distance that C_t has to C_{t-1}, where each of these snapshot qualities are reflective of the overall fitness of the clustering at each time step (Chakrabarti et al., 2006). The summation of the history cost for C_t and C_{t-1} at each time t is then considered to be the overall cost of the network (Chakrabarti et al., 2006). Ideally through this method the snapshot of each clustering should be of high quality by itself, and the history cost between the snapshots should be very low, meaning that the network did not significantly shift from the previous generation (Chakrabarti et al., 2006). A user-defined parameter *cp* in the interval [0, 1] is also utilized to modify to what degree the history cost determines the overall cost between clustering times (Chakrabarti et al., 2006). The inclusion of such a parameter is beneficial in that is further allows this framework to be applied to many different situations, such as the case within the authors' work that history cost dramatically affected the distances weighed when using an algorithm such as agglomerative hierarchical clustering (Chakrabarti et al., 2006).

Another framework for determining subsequent graph generations can be seen in Kim and Han's (2009) research in the improvement of temporal smoothing, sharing similarities to the framework of Chakrabarti et al. (2006). Their method approaches the overall cost function of the temporal network based upon the sum of a snapshot cost and a temporal cost (Kim & Han, 2009). In this cost function, the snapshot cost is determined from a variable CR_0 and CR_t, which compares the clustering result at a time t with an original clustering CR_0 from the original network that does not have temporal smoothing applied (Kim & Han, 2009). In similar form the temporal cost compares CR_t against CR_{t-1}, showing the difference between the clustering of following generations (Kim & Han, 2009). For both costs, the lower the value, the greater the overall quality of each sample (Kim & Han, 2009). Again, a user-defined parameter α in the interval [0,1] manages the trade-off of snapshot cost against temporal cost, with the temporal cost considered as the complement to the snapshot cost (Kim & Han, 2009). The authors found that applying this cost embedding technique to temporal smoothing for density-based networks resulted in a more efficient return of high quality, smoothed clusters at each snapshot of the network (Kim & Han, 2009). Following up to this work, Folino and Pizzuti (2010) examined their genetic algorithm-based approach *DYN-MOGA* against Kim and Han's (2009) work to find significant improvement when examining the normalized mutual information (NMI) to detect differences between true and detected partitions. They consider again a very similar problem with snapshot cost *SC* and temporal cost *TC,* however, provide separate multi-objective optimization for each of these components (Folino & Pizzuti, 2010).

Work conducted by Rossetti and Cazabet (2018) additionally explore methods outside of and containing the initially proposed temporal smoothing for community detection in dynamic networks. They propose three classes of frameworks involving a certain variation on the amount of temporal smoothing applied to each successive iteration of the graph, and the benefits and detriments of each method (Rossetti & Cazabet, 2018). Most importantly for the methods of this research is the second classification, "temporal trade-off community discovery" (Rossetti & Cazabet, 2018). This approach details an incremental model to temporally smoothing the network for increasing values of t and can be considered to encompass the methods mentioned prior (Chakrabarti et al., 2006; Kim & Han, 2009; Rossetti & Cazabet, 2018) The discovered benefits of such a method over ones that utilize complete temporal smoothing from the future and past, and ones that include no smoothing at all is that is more resistant to the instability of community detection of the algorithm it is partnered with (Rossetti & Cazabet, 2018). In the cases where a clustering algorithm is capable of finding several different clustering's within a graph, temporal smoothing adds an additional value of consistency even with faced with variations that could variably alter the previous shape of the network (Rossetti & Cazabet, 2018). The listed drawback of this method suggests that it may fall victim to an "avalanche-effect", by which the smoothing becomes too pronounced and fails to recognize the standard partitions a static algorithm would detect (Rossetti & Cazabet, 2018).

2.3. Potential competitors

Now understanding the general framework proposed with evolutionary clustering, as well as the task at hand with health surveillance, there can be said to be competitors to this work in accomplishing both of these respective components. Both fields of supervised and unsupervised machine learning have been commonly proposed for applications of health surveillance. A framework for combining Twitter-collected data with machine learning models was proposed by (Rodríguez-Martínez & Garzón-Alfonso, 2018) in their work. They dub this system the Twitter Health Surveillance (THS) system, and provide a majority of focus on developing an application platform for health officials to collect, interpret, and store large quantities of potential health-related information from Twitter. They employ a similar data-scraping system to what was used in this work, that collects tweets from a live-stream and assesses them by way of sentiment analysis (Rodríguez-Martínez & Garzón-Alfonso, 2018). Using recurrent neural networks (RNN), they have also included a built-in labelling schema to determine what potential medical conditions or disease a tweet could possibly relate to (Rodríguez-Martínez & Garzón-Alfonso, 2018). They do not provide any additional applications in way of analysing the data, however this framework is relevant since the data collection ideology is similar, however more generalized. This system may in fact provide better long-term support for the proposed connection of evolutionary clustering with health surveillance if the features collected by THS contain maximal instance information like geography, connections, etc.

Another study in the field of ML-aided health surveillance can be seen with Mackey et al.'s work with an unsupervised biterm topic model (BTM) for assessing self-reporting of COVID-19 instances (Mackey et al., 2020). They again use the Twitter API for live-streaming collection but apply two-level filtering on collected tweets (Mackey et al., 2020). The first filtering collects tweets based on specific COVID-19 keywords, much like will be discussed here in Section 3 (Mackey et al., 2020). The second filtering then looks for specific text instances that could imply self-reporting on COVID-19 symptoms (Mackey et al., 2020). The BTM is then employed for analysing specific textual themes present within these tweets and conversation threads that they exist in to produce topic clusters (Mackey et al., 2020). One significant delineation they provide in this work is the further differentiability in dividing symptom-related tweets from those related to testing and recovery (Mackey et al., 2020). They provide conclusions by way of five categories that define these delineations in tweet content, some of which being first and second-hand self-reporting, results after testing, and recovery discussions (Mackey et al., 2020). There exists evidence of geotagging feature collection in this work as well, however further effort would likely have to be applied to discover accurate geographical information, since it is not always consistent as learned in the collection process for this work. It is worth mentioning that though this work is certainly more thorough in tweet filtering, the application of a double-filtering and category designation system for each collected tweet would seemingly make dynamic processing more difficult for a task like evolutionary clustering. Another potential issue could arise with the cited manual annotation (Mackey et al., 2020). However, given greater hardware capabilities this system seems to provide an incredibly rich array of health-related information that could potentially be used for this task.

Work conducted by Arpaci et al. (2020) also seeks a similar goal as in this research by examining evolutionary clustering potential on Twitter data for the COVID-19 pandemic. Like in the methods proposed by Mackey et al. (2020), the focus on tweet analysis is placed

on examining unigram, bigram, and trigram term associations with pandemic-related buzzwords (Arpaci et al., 2020). The suggested goal of which based on keywords considered is to assess the psychological impact of government restrictions, and general fear regarding the disease from the public-perspective (Arpaci et al., 2020). They utilize evolutionary K-means proposed by Chakrabarti et al. (2006). Initially, k clusters is determined by the typical elbow method, of which for the tested dataset with 43 million total tweets, six clusters were found to be optimal for a single day (Arpaci et al., 2020). As opposed to examining instances based on content to make connections about potential self-reporting instances like in Mackey et al.'s (2020) work, clusters in this case represent the frequency by which terms are used across a 9-day period (Arpaci et al., 2020). Based upon this, they were able to discover that terms like "death, test, spread, and lockdown" were the most prevalent unigram terms over the period, thereby giving some general insight into how the public was responding to the spread of the disease (Arpaci et al., 2020). The proposed application from this information is that it could be used to help government bodies and health organizations better control panic in disease-crises such as this (Arpaci et al., 2020). One potential negative to this is that specific geographical information is not mentioned or considered when clustering these tweets, which may in fact make it more difficult to determine how specific populations are reacting to the disease.

Some recent advances in evolutionary clustering techniques propose new algorithms for dynamic modularity-based community detection, like one utilized for the Louvain method in Section 4. The *C-Blondel* algorithm recently introduced by Seifikar, Farzi, and Barati (2020) is one such method, being based on detecting communities in dynamic networks by building compressed graphs of the network and integrating the Louvain method to discover communities. Rather than the typical association of a snapshot cost *SC* and a temporal cost *TC*, with varying representation, as well as the trade-off parameter α, the evolutionary aspect is satisfied by building-in the communities at $t-1$: $C(G_{t-1})$ and incorporating them into a compressed graph based on previous communities as well as network changes between previous and current snapshots (Seifikar et al., 2020). The result of this compression includes nodes and edges of the history of the network incorporated into a new graph as super nodes and super edges (Seifikar et al., 2020). They additionally seek to solve the issue that temporal smoothing has been considered for when subtle node changes between generations end up destroying the community structure of the network. This is accomplished by way of evaluating network changes that include appearing and disappearing nodes and edges from the network (Seifikar et al., 2020). Degree centrality is heuristically used to determine what nodes could potentially destabilize the network based upon the product of a destruction parameter with average community degree (Seifikar et al., 2020). Bringing this all together, the destruction parameter fulfils the purpose of trading off modularity for execution time, and was found reduce the time complexity of the algorithm while sustaining comparable modularity to other methods *S-Blondel* and *D-Blondel* across three bench-marking datasets (Seifikar et al., 2020). Although not specifically proposed for instant detection from a streaming data source, this method could potentially offer substantial benefits due to compressing information and allowing for modulation for the rate of disappearing nodes/edges.

3. Data collection

The final dataset used for this research was collected over the course of a day in October of 2020. Using the Tweepy library for Python, a collection of an arbitrary amount of 3000 users were mined from Twitter to demonstrate the use of the algorithms within the context of health surveillance (Roesslein, 2009). Every user within the testing set falls within all of the following criteria: their follower to friend ratio suggests they are likely not a bot or media account; they have been flagged by the collection algorithm by tweeting a symptomatic keyword related to COVID-19; their geographical status is enabled on their collected tweet; and they have tweeted within the United States. In addition to these preliminary measurements, five features were collected in total for each of the users. These features included: ID number, location, friend count, follower count, and the time stamp at which their tweet was created. The collected location is originally only displayed as a string value, but by using a public U.S. cities database, each location could be cross-referenced and expanded into 4 extra features, these being: city, state, longitude, and latitude ("United States Cities Database", 2020). The most important features for use in the outlined methods are the coordinates that each tweet has occurred at, as well as the timestamp of each tweet to allow for the use of evolutionary clustering. Outside of initial location-mapping, the only other pre-processing that was performed was removing edges between nodes more than 20 degrees away in the latitudinal direction, since these occurrences would be unlikely to have any correlation with one another, i.e., an individual in New York would not have any foreseeable connection to an individual in California based purely upon location. Longitude was not considered greatly since it may be of interest to examine association between occurrences along the coasts. Additionally, the time stamps were converted to seconds, encoded, and then organized. While this data considered hours and minutes as the means for grouping individuals by timestamp, the following methods would still apply for circumstances including even more diverse windows of time.

When speaking about the characteristics of big data that are most important, there is commonly said to be several "V's" that describe the value of the data (McAfee & Brynjolfsson, 2012). The reason why Twitter was chosen as the platform of choice over another social network, is because its data rather intuitively fulfils itself in the terms of these values, being: volume, velocity, a variety (McAfee & Brynjolfsson, 2012). Its API is easy to interface with and allows for virtually unlimited data collection that can be directly converted to a desired form and passed to other algorithms for analysis. Even with various validifying checks to make sure a user meets the beforementioned criteria; a simple data mining programme is capable of collecting thousands of samples over a couple of hours on a personal computer with average hardware and network specs. This type of data also excels at having various different elements to examine, which has been especially taken advantage of in this research when considering exact city coordinates. While the userbase of this social network may not be as large as some others such as Instagram and Facebook, it still offers a superb means of quickly capturing rich information as it is created.

4. Evolutionary clustering and community detection algorithms

After the collection and processing of the data is complete, then comes the time to make use of the information it holds by applying the proposed evolutionary clustering techniques to it. The process used can be simply implemented and performed any number of times without significant alterations but to the parameters of the evolutionary algorithms themselves, as will be described in the following sections. Considering this, the flowchart contained within Fig. 1 represents a simplified version of the steps followed to arrive at the conclusion of the overall experiment.

4.1. Evolutionary DBSCAN clustering

Density-based methods of clustering group points based upon their distance to other points neighbouring them (Xu & Tian, 2015). The most popular algorithm in this category is DBSCAN, and already has a history of being utilized effectively with evolutionary techniques applied to it (Elgazzar & Elmaghraby, 2017; Kim & Han, 2009). DBSCAN estimates density level based upon two parameters: *minimum points* (*MinPts*), which denotes the minimum number of points required to form a cluster, and ϵ, a distance measurement that determines the

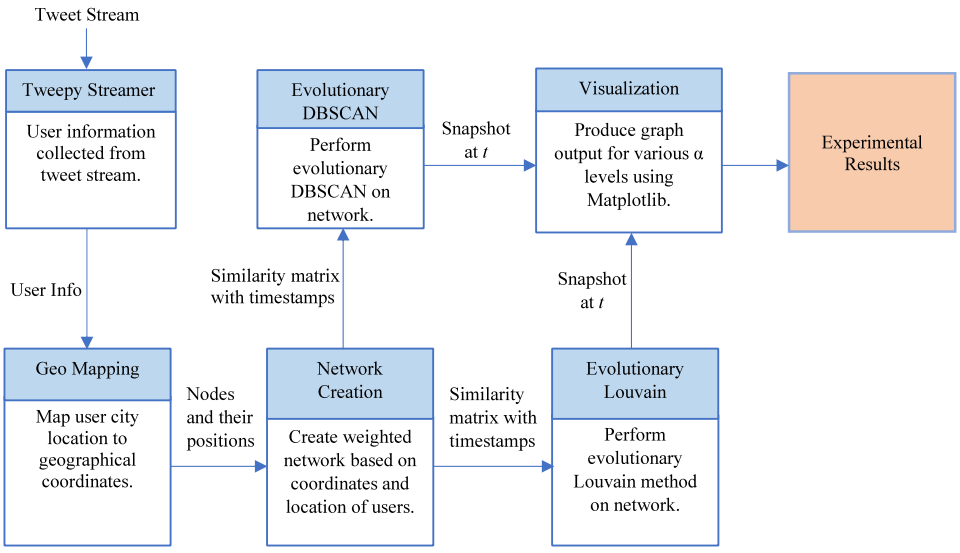

Fig. 1. Process of proposed methods.

radius by which points can be considered part of a cluster (Schubert, Sander, Ester, Kriegel, & Xu, 2017). Points containing more neighbouring points than those required within the radius around it are to be considered as a core point (Schubert et al., 2017). Points outside of the radius of a core point are initially designated to noise, however once a core point is discovered all neighbouring points within its radius are added to its cluster (Schubert et al., 2017). Thus, considering these two prerequisites to form a cluster, in detecting communities the most similar individuals are the ones that exist the closest to each other in the network (Schubert et al., 2017). This type of algorithm is naturally very applicable to situations relating to geographical information, as density is a real-world component of this information. This intrinsic connection to the type of the data explored is one of the reasons DBSCAN was used as a base model for this research. Additionally, DBSCAN is an algorithm that does not require the number of clusters to be specifically passed upon its inception, which further makes it valuable in an evolutionary model when the number of clusters is likely to change as the network expands over time. It has further value through its generality in cluster creation by being able to handle convex and non-convex shapes, again of importance when the shape of clusters and or the network in general is unknown outside of snapshots. The simplified pseudocode describing the regular DBSCAN algorithm can be seen in Fig. 2, assuming the Euclidean distance measurement is used (Elgazzar & Elmaghraby, 2017).

The DBSCAN algorithm starts by considering each point within the network as noise. Iterating through each point, it checks for the neighbours surrounding a point using the given measurement of distance ε. If number of neighbours surrounding the point is greater than or equal to the specified minimum number of points, it is considered to be a core point, and it must be expanded through its child connections. If the number of neighbours is less than the minimum points, it is marked as seen and passed over. When expanding upon the core point, the neighbourhoods of the neighbours are added to the cluster of the core point. Through this an initial core point can expand outwards based upon the neighbourhoods of points within its radius. To prevent one cluster from stealing points from another, a list of visited points must be kept and a control structure utilized at each time a new point is visited. When all points within the network have been visited, points that lay outside of any clusters are considered outliers based on the given value for ε. Fortunately, the algorithm does not require a predetermined value of k for the number of clusters, however multiple variations of ε and *MinPts* must be tested to determine the best fitting clustering distribution.

To apply an evolutionary method to this algorithm, a variation of a temporal radius measurement produced by an author of this work for another study was utilized (Elgazzar & Elmaghraby, 2017). The technique applied is a sort of smoothing of the radius parameter ε, that changes in correspondence with the unique distances present within a snapshot of the network at a time t (Elgazzar & Elmaghraby, 2017). Eq. (1) demonstrates the calculation of this parameter for a given value of t (Elgazzar & Elmaghraby, 2017).

$$\varepsilon_t = \begin{cases} \text{median}(\text{unique}\left(W_t\right))/\beta, & \text{t} = 1 \\ ((1-\alpha) * median\left(unique\left(W_t\right)\right)/\beta \\ \quad +(\alpha) * median(unique\left(W_{t-1}\right))/\beta, & \text{t} > 1 \end{cases} \tag{1}$$

This time-dependent version of ε_t can be considered similar in form to the calculation for smoothing of the entire network proposed by Kim and Han (2009) for density-based clustering methods. There must be a special case when the graph is first instantiated at $t = 1$ since there is no prior snapshot of the network to base a new distance on. At this timestamp, the median distance out of all unique distances in the network is selected and taken as the initial value for ε_1. After a new snapshot of the graph has been taken, the snapshot of the network at its current time t is then considered as well as the history of the median unique distances of the network at the time $t-1$. Two user-defined parameters, α and β, are used to further modulate the value for epsilon independent of the state of the network. The parameter α determines what ratio of the snapshot cost and history cost is to be considered in determining the new value for ε_t. The constant β is an arbitrary parameter that is included to normalize the radius to some degree against a starting high distance value. It accomplishes the similar feat as being able to tweak the value for epsilon manually, however, while still allowing for it to be modulated based upon the included history cost.

While parameter β is not mandatory or can be subjective, there exists some contention on how to determine the value for α, since this may have a significant impact on how the network transforms as t grows. For temporal smoothing applications, some works examine the normalized mutual information (NMI) of clustering results for different α-levels to find the best solution (Folino & Pizzuti, 2010; Kim & Han, 2009). In Folino and Pizzuti's 2010 work, this is treated as a traditional genetic algorithm optimization problem, with solution fitness defined as a community score of the current snapshot cost. A work by Xu, Kliger, and Hero (2013) proposed a framework *AFFECT* for determining the optimal α at each snapshot by letting α vary with respect to time t. In this work α is static, and assessment is performed by examining

```
DBSCAN
Inputs:
    W[nxn]: a symmetric, weighted similarity matrix of the network.
    ε: the radius distance surrounding a point.
    MinPts: the number of minimum points within a radius to form a cluster.

Initialize all points in W to noise.
Keep a list of points seen within W.

For each point p in W:
    If p not seen:
        Mark p as seen.
        NNp = NearestNeighbours(W, p, ε)        #Discover the neighbourhood of p
    If size of NNp =< MinPts:                   #Point is noise
        Leave p as a noise point.
    Else:                                       #Point is a core point
        Start a new cluster on p.               #Start a new cluster
        For each point q in NNp:                #Expand the cluster
            If q not seen:
                If q is labelled as a noise point:
                    Add q to the cluster of p.
                Else:
                    NNq = NearestNeighbours(W, q, ε)
                    If size of NNq >= MinPts:
                        Add NNq to the cluster of p.
```

Fig. 2. DBSCAN algorithm.

representative generations for several α-levels. Selection of the most viable parameter could be installed later for on-line purposes given current snapshot characteristics, or assessed ante-hoc for toy-datasets.

Since only the value for the radius measurement changes between each snapshot, the ordinary DBSCAN algorithm can still be utilized when determining the cluster assignments at each timestamp. Fig. 3 contains the pseudo code of the evolutionary DBSCAN algorithm used in this research. In the case of applying this directly onto data that has come straight from the stream of tweets on social media, only minor alternations are needed to convert the dynamic network into a form that can then be processed by the previous DBSCAN algorithm.

4.2. Evolutionary Louvain method

The Louvain method is a partitional, agglomerative community detection algorithm created by Blondel et al. that discovers clusters within a network based upon the optimization of a measurement known as modularity (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008; Dugué & Perez, 2015). Modularity is an optimization function that is used to further weight the value of edges within a network based upon the probability that an edge may appear in a similarly constructed network with the same vertex degrees (Dugué & Perez, 2015). Through this, edges that seem the most unlikely to appear from vertices of small degree are valued higher for their rarity than the edges that exist between vertices with high degrees, and thus are expected to be the most ordinary (Dugué & Perez, 2015). This sense of value can be seen in Eq. (2), which depicts the definition of modularity Q for a partition C in an undirected graph $G = (V, E)$ (Blondel et al., 2008):

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta(c_i, c_j) \tag{2}$$

Where $m = |E|$, A_{ij} is the weight of the edge between vertices i and j, d_i is the degree of vertex i, c_i is the community of vertex i, and the function $\delta(u, v)$ is defined as 1 if $u = v$, and 0 if otherwise (Blondel et al., 2008). It can then be seen then that the weight of a connection A_{ij} is diminished as $d_i d_j$ approaches infinity, representing the prioritization of rare low degree vertex connections over high degree connections.

Now considering this modularity, the Louvain method first starts from a single partition of the graph in which all nodes are divided into their own communities (Blondel et al., 2008). Then for each vertex i in the network, each of its neighbouring vertices j are examined to compute the change in modularity that would occur should vertex i be moved to the community of vertex j (Blondel et al., 2008). This change in modularity ΔQ of moving a vertex i to a new community C is described in Eq. (3) in its reduced form, and is defined as follows (Dugué & Perez, 2015):

$$\Delta Q = \frac{d_i^C}{2m} - \frac{\left(\Sigma_{total}\right) d_i}{2m^2} \tag{3}$$

With the differences from Eq. (2) being that d_i^C is the degree of vertex i within community C, and Σ_{total} is the total number of incident edges of the community C (Dugué & Perez, 2015). After computing this change in modularity for each pair of vertices i and j, the vertex i is then placed in the community of j in which the largest positive gain in modularity is acquired (Blondel et al., 2008). In the case where the modularity gained is not positive, then vertex i will remain in its current community (Blondel et al., 2008). This process is repeated until there exists no additional community changes that would increase the modularity further (Blondel et al., 2008). The algorithm then repeats this process on a copy of the network with the vertices now represented by the communities of the previous partition (Blondel et al., 2008; Meo, Ferrara, Fiumara, & Provetti, 2011). The computation of modularity change by moving vertices to new communities and subsequently transforming communities into the new vertices is then repeated until a maximum modularity is obtained for the network (Blondel et al., 2008; Meo et al., 2011). Fig. 4 contains the pseudocode for the Louvain method algorithm utilized (Aynaud, 2020; Blondel et al., 2008).

There is a number of community detection algorithms besides the Louvain method that are based on optimizing modularity. Especially in the case of recent methods in dynamic networks beforementioned C-Blondel by Seifikar et al. (2020) or CSLM by Chaudhary and Singh (2019). Louvain however is still a popular approach within this sub-type of community detection algorithms, despite being a greedy optimizer. Some principal benefits proposed when using this algorithm come from its intention as a faster way to approximate the modularity through heuristics, which is considered a computationally difficult task when trying to discover completely (Blondel et al., 2008). Blondel et al. (2008) was able to accomplish this within logarithmic time, which proposes exceptional capability for live-streaming analysis purposes. Another benefit is that no prior information regarding number of clusters or required neighbourhood size be considered when using

Evolutionary DBSCAN

Inputs:
X = a dataframe containing columns of x, y, and timestamps.
T_Labels = a list containing the unique labels of the times within X.
α = parameter acting as the ratio between current distance and history distance.
β = constant to scale radius.
MinPoints = the minimum number of points required to create a cluster.

Keep list of seen times.

For each time in T_Labels:
Mark current time as seen.
X' = Select current snapshot from X where time instances have been seen.
W_t = Find similarity matrix of X'.

If time == 1:
ε_t = median(unique(W_t))/β
Else:
ε_t = ((1-α)*median(unique(W_t))/β+(α)*median(unique(W_{t-1}))/β

Keep a history of the network with $W_{t-1} = W_t$
Call DBSCAN(W_t, ε_t, MinPoints)

Fig. 3. Evolutionary DBSCAN algorithm.

this algorithm. For this particular type of application in being able to incorporate potentially thousands of new samples into the network at each time t, time complexity for the algorithm should be of the most concern to avoid lagging behind the flow of data. In addition, temporal smoothing, as it name implies, has been considered a means to "smooth" out evolving communities over time for greedy algorithms like Louvain method, where single point differences could produce significantly different results otherwise (Rossetti & Cazabet, 2018).

Based on several smoothing archetypes described by (Rossetti & Cazabet, 2018) in their survey work, the goal for applying these methods onto a temporal network seeks to keep the partitions the same while seeking to further maximize modularity through the addition of new vertices. For this application, a version of explicit temporal smoothing has been considered, of which is not too unlike the method utilized for density-based approaches on creating a history-based radius (Elgazzar & Elmaghraby, 2017; Kim & Han, 2009). In this case there are no parameters that may need to be adjusted as time progresses, so the option that was considered was to more greatly incentivize certain links to persist throughout the history of the network as it is gradually scaled to completion. Eq. (4) depicts the equation for temporal smoothing applied to the network.

$$W_t = \begin{cases} W_t, & t = 1 \\ (1-\alpha) * W_t + (\alpha) * W_{t-1}, & t > 1 \end{cases} \quad (4)$$

Through the usage of this, communities will not be greatly affected by an influx of new points over time since the points of past partitions are being prioritized over them. This method provides a means of slowly tweaking the modularity as new information becomes available. In a similar fashion to the evolutionary radius measurement, again a user-defined parameter α is utilized for tweaking the ratio between the snapshot of the network at time t and the history of the network at a previous time $t-1$. While this community detection algorithm is not as especially relevant as a density-based approach when considering geographical locations, it still provides functionality as dividing up collected areas into larger, but less defined sections of connected points. The pseudocode for the described implementation of the evolutionary Louvain method can be seen in Fig. 5.

5. Experimental results

In applying these methods, several libraries available for Python have been utilized for various purposes. For extra clustering functionality the sci-kit: learn clustering library was used with modifications to implement the algorithms (Pedregosa et al., 2011). For visualization of network snapshots, the Matplotlib library was used to provide graphing and colour mapping to the resulting clusters (Hunter, 2007). The NetworkX library was also used for its visualization and when using the evolutionary Louvain method algorithm (Aynaud, 2020; Hagberg, Swart, & Chult, 2008). Lastly, the Pandas library was used for data frame functionality to store the dataset and to convert timestamps to an ordinal format (McKinney, 2010).

5.1. DBSCAN results

For testing the evolutionary DBSCAN algorithm on the data, three values of α were considered; the parameter used to modulate the ratio of current information and past information influence. These values were $\alpha = 0$; for the static algorithm, $\alpha = 0.50$; for a balance between present and past; and $\alpha = 0.80$, where the history is significantly relied upon to determine the new value for ε_t. The scaling constant $\beta = 6$ was utilized for all cases of α and is used as a source of normalization for the radius. This value for β for testing purposes was taken to be the standard deviation of the latitudinal positions for the collected instances. This coefficient may as well be $\beta = 1$ for most implementations, however the median radius of the first few generations should be taken into consideration if it is sufficiently high enough to include all possible points, which would likely not result in an understandable graph for this application. The additional parameter of *MinPts* = 2 was utilized for each test of the algorithm, and is considered the default value. It must also be considered in the context of this work that this test sequence is only an offline representation of these methods, since it is necessary to show the influence of different parameters on the effectiveness the method has on the same data. Figs. 6, 7, and 8 show the outputs of using this algorithm at three snapshots with $\alpha = 0$, $\alpha = 0.50$, and $\alpha = 0.80$, respectively.

In comparing these graphs, it can be seen that the inclusion of the network's history does make a difference however small it may be in the scope of this network. While the snapshot at the middle set of timestamps at 414 seems to be largely unchanged based on the inclusion of the history, the noise reduction present in the first 100 timestamps as well as in 829 show that the history does make a difference in better encompassing noise to a respective cluster. As discussed in the prior paragraph regarding evolutionary clustering, it can also be seen that the graph does not change significantly between the passage of time. Instead, it is noticeably consistent and evident as to how the snapshot at $t = 100$ has been built into $t = 829$.

Louvain Method

```
Inputs:
        W[nxn]: a weighted non-directed or classical graph of the network.

Communities = {}                                #Keys are nodes, values are communities
Initialize all points in W to their own community.
Variable to hold modularity and compare across partitions.

Define ModularityChange (p, q, Communities, W):
        NewCommunity = Communities[q]
        DegreeCommunityP = 0
        For each key in Communities:
                If Communities[key] equals NewCommunity and W[p, key] does not equal 0:
                        DegreeCommunityP += 1
        IncidentEdges = incident(q)
        DegreeP = degree(p)
        m = edges(W)
        Compute ΔQ
        return ΔQ

While Modularity shows increase:
        For each point p in W:                  #Moving nodes to new communities
                BestModChange = 0
                BestCommunity = Communities[p]
                For each point q in neighbourhood of p:
                        NewModChange = Modularity(p, q, Communities, W)
                        If  NewModChange > BestModChange:
                                BestModChange = NewModChange
                                BestCommunity = Communities[q]
                Communities[p] = BestCommunity

        Calculate modularity for new communities.
        If Modularity does not increase from past partition:
                Break loop.
        Else:                                   #Create new graph with communities as nodes
                NewGraph = addnodesfrom(unique(Communities.values))
                For node1, node2, edge in W:
                        Com1 = Communities[node 1 of edge]
                        Com2 = Communities[node 2 of edge]
                        If edge between Com1 and Com2 does not exist:
                                NewEdge(Com1, Com2)
                                NewEdge.weight(WeightOfEdge(node1, node2))
                                Add NewEdge to NewGraph
                        Else:
                        Com1Com2Edge.weight += WeightOfEdge(node1, node2)
                W = NewGraph

Return(dictionary of highest modularity partition)
```

Fig. 4. Louvain method algorithm.

5.2. Louvain method results

In a similar practice to the beforementioned DBSCAN experiment, three different values for α; the ratio between snapshot cost and history cost, were tested with $\alpha = 0$, $\alpha = 0.50$, and $\alpha = 0.80$. Additionally, it is worth mentioning that this is again an offline approach to the utilization of this method, as for testing purposes it would not have been practical to stream new information into the algorithm and try to compare between completely different sets. The following Fig. 9 shows the results of the static Louvain method with $\alpha = 0$ and representative time slots 100, 414, and 829. Fig. 10 then shows the evolutionary Louvain method at work with $\alpha = 0.80$ and the same time slots. The snapshots at $\alpha = 0.50$ were not included simply for the preservation of space, however their modularity for each time mark either ranked lower than the static method, or slightly less than the modularity of the $\alpha = 0.80$ execution.

For the utilized dataset there did not appear to be any exceptional improvements to modularity as the ratio of history cost was increased, however it is evident that through considering this history the algorithm utilized partitions it would have not considered previously. The snapshots at the 100 and 414 marks are the most interesting occurrence of this since the evolutionary method returned markedly different results with a noteworthy increase in modularity. It is worth noting that for a network of sample size 3000 this method also boasted a fast computation for each of the snapshots as well, with the algorithm consuming the vertices of multiple new time slots in under a second. While this speed would no doubt suffer with sufficiently larger quantities of data, at this size it would be suitable to consider streamed information the second it is inputted from a social network.

6. Conclusion

In this work a pipeline has been detailed that is capable of directly connecting unsupervised machine learning techniques to an instantly transmittable, dynamic source of data. In data collection the prioritization of geographically identifiable data combined with timestamps has allowed a seamless connection to applying evolutionary methods on information that reflects the real-world. This approach to data production could even be further expanded upon by utilizing additional sources of information. Such as through connecting a cross-validation

Evolutionary Louvain Method
Inputs: X = a dataframe containing columns of x, y, and timestamps. T_Labels = a list containing the unique labels of the times within X. α = parameter acting as the ratio between present and history influence. Keep list of seen times. For each time in T_Labels: Mark current time as seen. X' = Select the current snapshot from X where time instances have been seen. W_t = Find similarity matrix of X'. If time == 1: $W_t = W_t$ Else: $W_t = (1-\alpha)*W_t+(\alpha)*W_{t-1}$ Keep the history of the network with $W_{t-1} = W_t$ Call LouvainMethod(W_t)

Fig. 5. Evolutionary Louvain method.

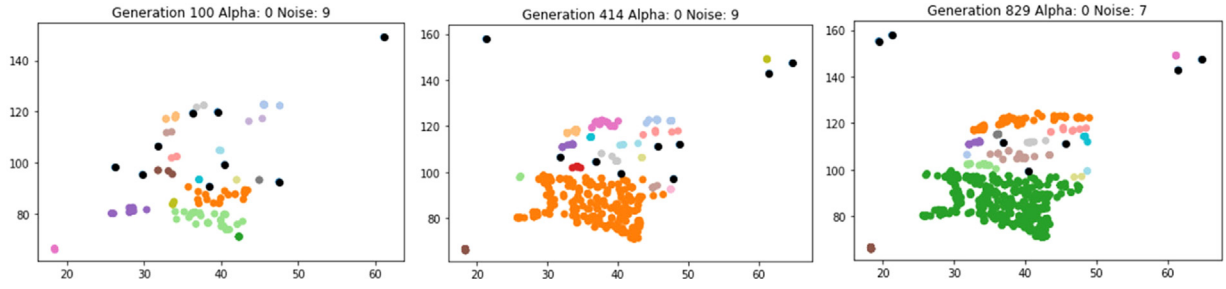

Fig. 6. Static DBSCAN.

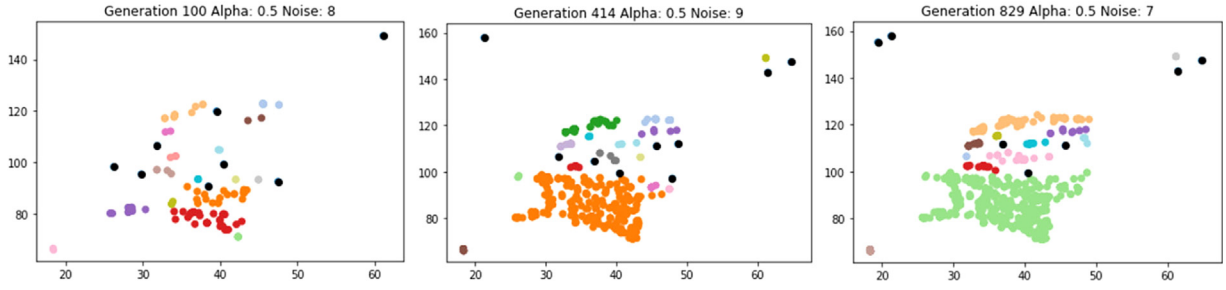

Fig. 7. Evolutionary DBSCAN with $\alpha = 0.50$.

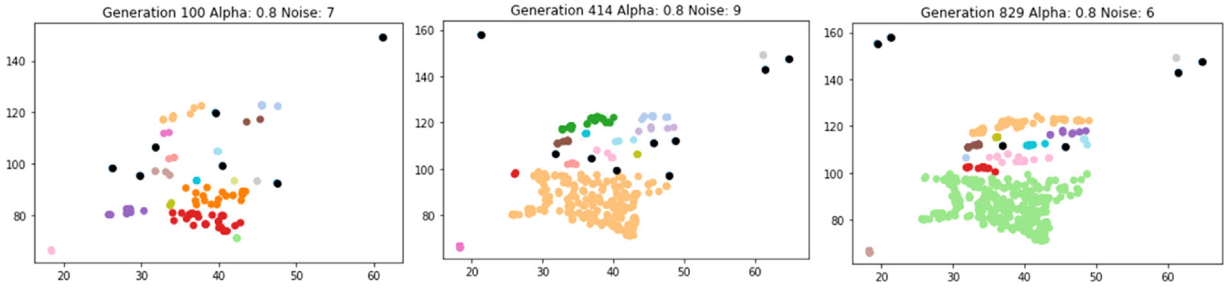

Fig. 8. Evolutionary DBSCAN with $\alpha = 0.80$.

scheme to properly assess the quality of clusters at a given snapshot of the network, as well as tweet sentiment analysis to further evaluate the adequacy of samples.

The results have shown in this case for the size of data used that evolutionary clustering does make a slight, but noticeable difference in the quality that is inferable from one generation of the network to

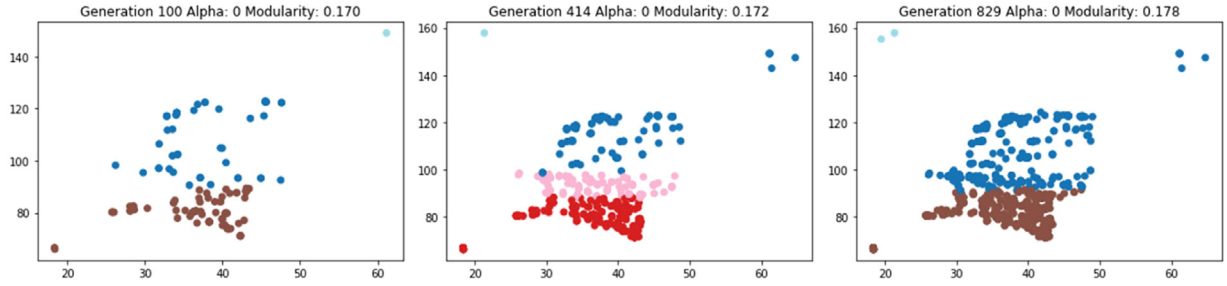

Fig. 9. Static Louvain method.

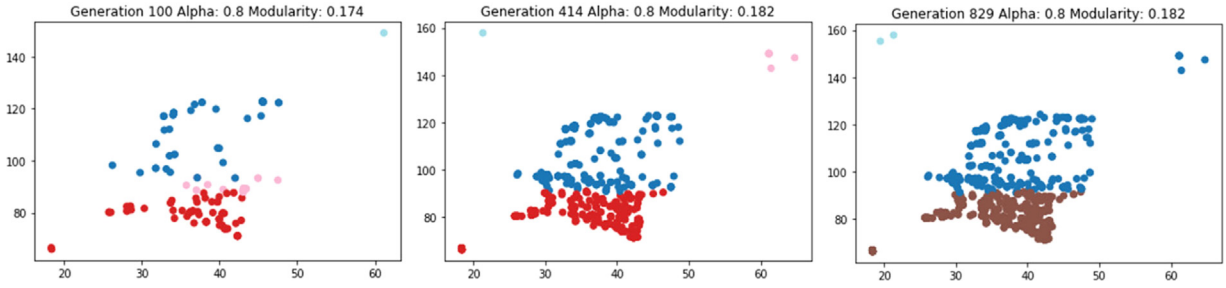

Fig. 10. Evolutionary Louvain method with $\alpha = 0.80$.

the next. This is especially evident when considering smaller sizes of data that have more room for adjustment, which can be seen in both methods at the middle time slot of each network. Despite detailing this method as a beneficial online approach, this testing was still conducted offline as to assess each methods effectiveness. The application of this work would be much better applied to an online system model, in which pre-processed data would be allowed to flow into the clustering algorithms without considering a middle transition. However, this transition could be easily applied from collection, to transformation, to clustering method.

A model of this system would be further benefited still by being applied to a more reliable source of information, like connecting the methods used here to actual indicative cases of disease to directly model disease occurrences the moment they are discovered. An example of this would be of the type mentioned in the work conducted by Woolhouse et al. (2015) mentioned in the opening paragraphs. Applying this method in such a way would of course require a much more thorough approach when trying to accurately depict real world circumstances, but as a framework these methods provide an applicable means of reaching that point. As the use of temporal and spatial data related to disease occurrences coalescence into combination with machine learning practices, it is undeniable that current health surveillance infrastructure will become even more valuable as a means of preventing the future spread of disease.

CRediT authorship contribution statement

Heba Elgazzar: Conceptualization, Methodology, Software, Writing - review & editing, Investigation, Visualization, Supervision, Project administration, Funding acquisition. **Kyle Spurlock:** Data curation, Methodology, Software, Writing - original draft, Investigation, Visualization. **Tanner Bogart:** Data curation, Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was sponsored by a research grant from Morehead State University, Morehead, KY, USA.

References

Arpaci, I., Alshehabi, S., Al-Emran, M., Khasawneh, M., Mahariq, I., Abdeljawad, T., et al. (2020). Analysis of Twitter data using evolutionary clustering during the COVID-19 pandemic. *Computers, Materials & Continua*, *65*(1), 193–204, https://doi.org/10.32604/cmc.2020.011489.

Aynaud, T. (2020). *Python-Louvain X.Y: Louvain algorithm for community detection*. GitHub, https://github.com/taynaud/python-louvain.

Blondel, V. D., Guillaume, J. D., Lambiotte, R. D., & Lefebvre, E. D. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), 10008–10020. http://dx.doi.org/10.1088/1742-5468/2008/10/p10008.

Chae, S., Kwon, S., & Lee, D. (2018). Predicting infectious disease using deep learning and big data. *International Journal of Environmental Research and Public Health*, *15*(8), 1596–1616. http://dx.doi.org/10.3390/ijerph15081596.

Chakrabarti, D., Kumar, R., & Tomkins, A. (2006). Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD International conference on knowledge discovery and data mining - KDD 06* (pp. 554–560). http://dx.doi.org/10.1145/1150402.1150467.

Chaudhary, L., & Singh, B. (2019). Community detection using maximizing modularity and similarity measures in social networks. *Smart Systems and IoT: Innovations in Computing*, 197–206, https://doi.org/10.1007/978-981-13-8406-6_20.

Cole, R. (1998). *Clustering with genetic algorithms* (pp. 1–110). University of Western Australia, Retrieved November 16, 2020, from uwa.edu.org.

Dion, M., Abdelmalik, P., & Mawudeku, A. (2015). Big data and the global public health intelligence network (GPHIN). *Canada Communicable Disease Report*, *41*(9), 209–214. http://dx.doi.org/10.14745/ccdr.v41i09a02.

Dugué, N., & Perez, A. (2015). *Directed Louvain : Maximizing modularity in directed networks* (pp. 1–15). Université D'Orléans, Retrieved November 16, 2020, from hal.archives-ouvertes.fr.

Elgazzar, H., & Elmaghraby, A. (2017). Network science algorithms for mobile network analytics. In *SoutheastCon* (pp. 1–7). http://dx.doi.org/10.1109/secon.2017.7925320.

Folino, F., & Pizzuti, C. (2010). Multiobjective evolutionary community detection for dynamic networks. In *Proceedings of the 12th annual conference on genetic and evolutionary computation - GECCO '10* (pp. 535–536). https://doi.org/10.1145/1830483.1830580.

Hagberg, A., Swart, P., & Chult, D. S. (2008). Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in science conference* (pp. 11–16). Retrieved November 16, 2020, from www.osti.gov.

Haston, J. C., & Pickering, L. K. (2019). Cdc's disease surveillance system critical for public health. Retrieved November 16, 2020, from https://www.aappublications.org/news/2019/03/08/mmwr030819.

Hay, S. I., George, D. B., Moyes, C. L., & Brownstein, J. S. (2013). Big data opportunities for global infectious disease surveillance. *PLoS Medicine*, *10*(4), Article e1001413. http://dx.doi.org/10.1371/journal.pmed.1001413.

Hunter, J. D. (2007). Matplotlib: a 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. http://dx.doi.org/10.1109/mcse.2007.55.

Kim, M., & Han, J. (2009). A particle-and-density based evolutionary clustering method for dynamic networks. *Proceedings of the VLDB Endowment*, *2*(1), 622–633. http://dx.doi.org/10.14778/1687627.1687698.

Mackey, T., Purushothaman, V., Li, J., Shah, N., Nali, M., Bardier, C., et al. (2020). Machine learning to detect self-reporting of symptoms, testing access, and recovery associated with covid-19 on twitter: retrospective big data infoveillance study. *JMIR Public Health and Surveillance*, *6*(2), https://doi.org/10.2196/19509.

McAfee, A., & Brynjolfsson, E. (2012). Big data: the management revolution. *Harvard Business Review*, *90*(10), 60–68, Retrieved November 16, 2020, from hbr.org.

McKinney, W. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in science conference (Vol. 445)* (pp. 51–56). http://dx.doi.org/10.25080/majora-92bf1922-00a.

Meo, P. D., Ferrara, E., Fiumara, G., & Provetti, A. (2011). Generalized louvain method for community detection in large networks. In *2011 11th international conference on intelligent systems design and applications*. http://dx.doi.org/10.1109/isda.2011.6121636.

Pedregosa, F., Varoquaux, G, Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: machine learning in python (M. Braun, Ed.). *Journal of Machine Learning Research*, *12*, 2825–2830, Retrieved November 16, 2020, from jmlr.org.

Rodríguez-Martínez, M., & Garzón-Alfonso, C. C. (2018). Twitter Health surveillance (THS) system. In *Proceedings :. IEEE international conference on big data. IEEE international conference on big data 2018* (pp. 1647–1654). https://doi.org/10.1109/bigdata.2018.8622504.

Roesslein, J. (2009). Tweepy documentation. Retrieved November 16, 2020, from https://docs.tweepy.org/en/v3.5.0/.

Rossetti, G., & Cazabet, R. (2018). Community discovery in dynamic networks. *ACM Computing Surveys*, *51*(2), 1–37. http://dx.doi.org/10.1145/3172867.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). DBSCAN Revisited. *Revisited. ACM Transactions on Database Systems*, *42*(3), 1–21. http://dx.doi.org/10.1145/3068335.

Seifikar, M., Farzi, S., & Barati, M. (2020). C-blondel: an efficient louvain-based dynamic community detection algorithm. *IEEE Transactions on Computational Social Systems*, *7*(2), 308–318, https://doi.org/10.1109/tcss.2020.2964197.

Shin, S., Seo, D., An, J., Kwak, H., Kim, S., Gwack, J., et al. (2016). High correlation of middle east respiratory syndrome spread with Google search and Twitter trends in Korea. *Scientific Reports*, *6*(1), 1–7. http://dx.doi.org/10.1038/srep32920.

United States Cities Database. (2020). SimpleMaps. https://simplemaps.com/data/us-cities.

Woolhouse, M. E., Rambaut, A., & Kellam, P. (2015). Lessons from ebola: improving infectious disease surveillance to inform outbreak management. *Science Translational Medicine*, *7*(307), http://dx.doi.org/10.1126/scitranslmed.aab0191.

Xu, K. S., Kliger, M., & Hero, A. O., III (2013). Adaptive evolutionary clustering. *Data Mining and Knowledge Discovery*, *28*(2), 304–336, https://doi.org/10.1007/s10618-012-0302-x.

Xu, D., & Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, *2*(2), 165–193. http://dx.doi.org/10.1007/s40745-015-0040-1.