CDOWatcher: Systematic, Data-driven Platform for Early Detection of Contagious Diseases Outbreaks

Abdullah M. Albarrak,

Imam Mohammad Ibn Saud Islamic University, Saudi Arabia

Summary

The destructive impact of contagious diseases outbreaks on all life facets necessitates developing effective solutions to control these diseases outbreaks. This research proposes an end-to-end, data-driven platform which consists of multiple modules that are working in harmony to achieve a concrete goal: early detection of contagious diseases outbreaks (i.e., epidemic diseases detection). Achieving that goal enables decision makers and people in power to act promptly, resulting in robust prevention management of contagious diseases. It must be clear that the goal of this proposed platform is not to predict or forecast the spread of contagious diseases, rather, its goal is to promptly detect contagious diseases outbreaks as they happen. The front end of the proposed platform is a web-based dashboard that visualizes diseases outbreaks in real-time on a real map. These outbreaks are detected via another component of the platform which utilizes data mining techniques and algorithms on gathered datasets. Those gathered datasets are managed by yet another component. Specifically, a mobile application will be the main source of data to the platform. Being a vital component of the platform, the datasets are managed by a DBMS that is specifically tailored for this platform. Preliminary results are presented to showcase the performance of a prototype of the proposed platform.

Keywords:

health informatics; syndromic surveillance systems; artificial intelligence; data mining.

1. Introduction

With the continuous outbreaks of contagious diseases and epidemics in the early years of the 21st century [1, 2, 3], it became clear that there is a significant need for innovative solutions to early detect these outbreaks. Early detection of contagious diseases enables rapid responses, thereby reducing the overall destructive impact of such outbreaks [4, 5, 6]. A timely detection of small and localized clusters of unusual diseases outbreaks brings forth reassurance that a large outbreak is not occurring, provided that appropriate actions are followed [7].

In line to fulfill that need, we propose in this paper CDOWatcher: Contagious Diseases Outbreaks Watcher. CDOWatcher is a systematic platform which aims to facilitate the early detection of contagious diseases outbreaks. This platform belongs to a novel category of surveillance systems often termed syndromic surveillance systems which aims to identify any increase of frequency for contagious diseases above the background occurrence of these diseases [8]. Typically, electronic health data (diagnostic and non-diagnostic) are continuously collected and processed by these systems for which health officials (e.g., The Saudi Center for Disease Prevention and Control) monitor outbreaks indicators and signals in real or at least near-real time.

As Figure 1 shows, CDOWatcher platform consists of four inter-connected modules, whereby each module plays a specific role in the process of detecting contagious diseases outbreaks. The end goal of CDOWatcher is to increase access to such information (i.e., detected outbreaks) in real time, thus, allowing robust assessment by decision makers and health officials. Furthermore, CDOWatcher provides automated detection of outbreaks clusters and visualizes these clusters, thereby reducing overall intensive manual analysis of electronic health data.

The rest of this paper is organized as follows: Section 2 presents fundamental concepts related to syndromic surveillance systems, whereas Section 3 presents the proposed platform in detail. In Section 4, a thorough discussion of the proposed platform is presented and its functionality is evaluated and compared against a baseline algorithm. Finally, Section 5 concludes the paper.

2. Background: Syndromic Surveillance Systems

Typical syndromic surveillance systems [9, 8, 10, 11, 12] utilize health data from various sources to detect diseases outbreaks clusters at early stages. The timely detection of outbreaks is pivotal as it enables health officials in controlling and curbing diseases outbreaks efficiently.
While these systems differ based on the type of acquired data [13, 14, 15], electronic syndromic surveillance systems are being proposed and developed more than ever since these systems have the timeliness advantage over other systems. Specifically, electronic syndromic surveillance systems utilize a diverse set of data sources, such as mobile phones which has the advantage of ubiquitous-ness and wide network coverage [14], and web-data sources such as online news articles [15].

Manuscript received November 5, 2022
Manuscript revised November 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.11.11

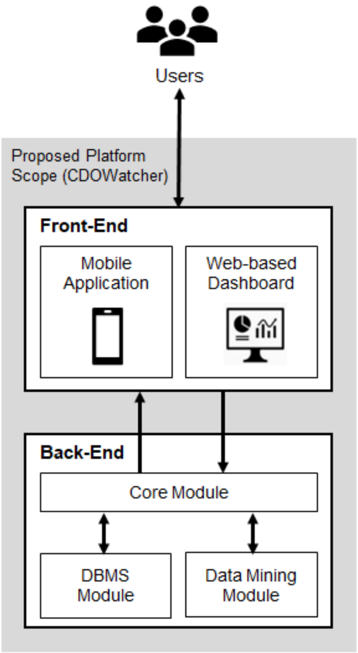


Figure 1: An architectural block diagram of the proposed platform which illustrates the four main modules: web-based dashboard, mobile application, data mining and DBMS modules.

Generally, most traditional syndromic surveillance systems rely on typical data sources (e.g., laboratory tests results and electronic health records [16]), however, the demand to reliably detect diseases outbreaks at the earliest possible stage has further advanced the consideration of real time data sources. Internet activities and social media are among those new data sources which syndromic surveillance systems rely on to accomplish their goal [11, 14, 13]. One approach of syndromic surveillance systems is dependent on dedicated data collection systems, which gives it the advantage of relaying on no more than itself to achieve its goal. Other approaches utilize data sources that are routinely collected for other purposes [7] such as emergency departments log sheets, medical prescriptions and drug purchases [17].

3. Proposed Platform: CDOWatcher

The primary purpose of CDOWatcher is to support the detection of diseases outbreaks at the earliest possible stages. As a secondary functional value, it also designed to support augmenting other syndromic surveillance systems with its collected data systematically.

As illustrated by Figure 1, users interact with the proposed platform CDOWatcher via the front-end modules: a mobile application module and a web-based dashboard module. The mobile application is used as a tool to input data into the platform, whereas the web-based dashboard is responsible for showing the output of the platform after processing.

Typically, users of the platform are either users who provide data to the platform (i.e., information sources), or consumers of information. For instance, health care workers (HCWs) are those who examine patients and record patients visits to hospitals into health care systems. Hence, it is assumed that HCWs will be using the mobile application to collect patients data. General practitioners (GPs) are also assumed to be collecting patients information via the mobile application since normally they are the ones who are firstly visited by patients when seeking medical care. Additionally, the patients themselves can also enter their data voluntarily into the platform using the mobile application without visiting a hospital or a GP doctor. Though, their input to the platform must follow a particular form and structure that is dictated by the platform.

Generally, health care officials such as Centers for Disease Control and Prevention (CDC) commanders are the typical users of CDOWatcher's web-based dashboard. Using the collected data from the mobile application, CDOWatcher periodically processes these data and provides summarized views of contagious diseases outbreaks in real-time to CDC commanders and health officials using appropriate formats.

The back-end of CDOWatcher, as Figure 1 shows, consists of two main modules: a DBMS module and a data mining module. The former is responsible for storing collected data in structural formats (i.e., tabular formats) that are designed to serve the objective of CDOWatcher, whereas the latter is the module which periodically processes data to detect diseases outbreaks and generates output for the dashboard module. The following subsections discuss in detail each module and its objectives in respect to the overall objective of CDOWatcher.

3.1 Web-based Dashboard Module

This is the front-end of the platform. It is a web-based dashboard that consists of graphical representations of current outbreaks in real-time, enabling informed decisions to be made. Figure 2 shows a prototype of the dashboard. It is divided into two zones: the navigation zone, and the preview zone.

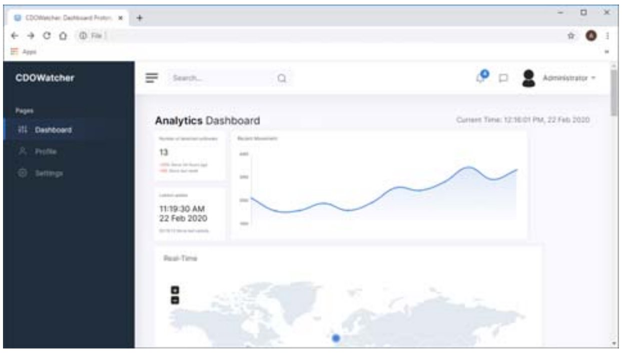


Figure 2: The prototype of the web-based dashboard. It is divided into two zones: preview zone (middle) which is the heart of the platform where output is displayed, and navigation zone (left-side).

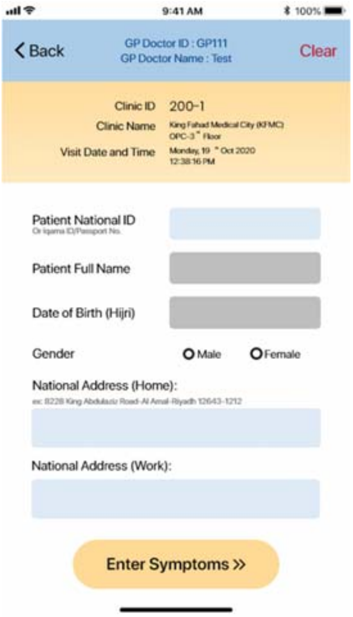
The preview zone is the main area where the output of the platform is displayed. Aligning users needs with the output of the platform is of paramount importance for this platform to have an impact on real world usage. Therefore, this zone contains multiple forms of output that will minimize the need for exhaustive manual analysis and exploration of collected data. In particular, it shows aggregated data (e.g., number of detected diseases clusters within predefined timeframe), specific performance indicators, map visualization of detected clusters, and line charts to visually observe and monitor diseases outbreaks activities. Presumably, this summarized information boosts CDC commanders' judgments in triggering further investigations to prevent and control diseases outbreaks.

The navigation zone is located at the left-hand-side. Users of CDOWatcher can check their profile and change the preview settings by following the appropriate links on this zone. The dashboard will be built using Bootstrap framework, along with Java for server-side programming. Bootstrap is a fully customizable framework used to create HTML and CSS responsive templates, which makes it a suitable candidate to develop the proposed dashboard.

3.2 Mobile Application Module

Data acquisition is the first step to achieve the goal of CDOWatcher. A mixture of diagnostic and non-diagnostic data are either automatically collected or are reported by HCWs via this module. We assume that there are three main health behaviors for which data is collected and stored into the platform:

- **Health behavior 1**: A person is suffering from symptoms which might be related to contagious diseases is seen by a GP doctor or visits a hospital and is assessed by a HCW. The doctor records the person's information and her symptoms using the mobile application. Then, information such as person's address, mobile number, symptoms, etc., are all recorded and saved in the platform.

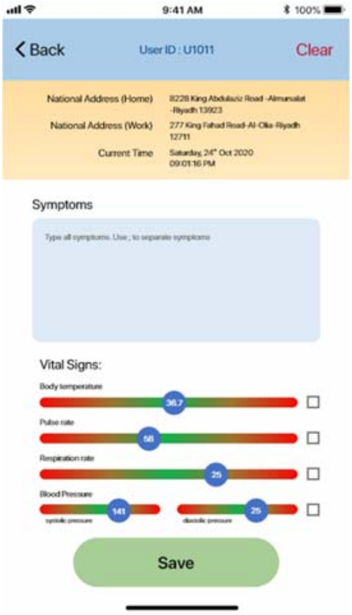


GP Doctor ID : GP111
GP Doctor Name : Test

Clinic ID 200-1
Clinic Name King Fahad Medical City (KFMC) OPC-3rd Floor
Visit Date and Time Monday, 19th Oct 2020 12:38:16 PM

Patient National ID
Or Iqama ID/Passport No.
Patient Full Name
Date of Birth (Hijri)
Gender ○ Male ○ Female
National Address (Home):
ex: 8228 King Abdulaziz Road-Al Amal-Riyadh 12643-1212
National Address (Work):

Figure 3: GP doctor or HCW enters patients' information. Clinic details and visit's timestamp are automatically generated.



User ID : U1011

National Address (Home) 8228 King Abdulaziz Road -Almursalat -Riyadh 13923
National Address (Work) 277 King Fahad Road-Al-Olia-Riyadh 12711
Current Time Saturday, 24th Oct 2020 09:01:16 PM

Symptoms
Type all symptoms. Use ; to separate symptoms

Vital Signs:
Body temperature 36.7 □
Pulse rate 58 □
Respiration rate 25 □
Blood Pressure 141 systolic pressure 25 diastolic pressure □

Figure 4: Patient enters her information voluntarily into the platform without visiting a hospital.

- **Health behavior 2**: A person with symptoms which might be related to contagious diseases enters her information voluntarily via the mobile application. Then, her information are systematically imported to the platform.

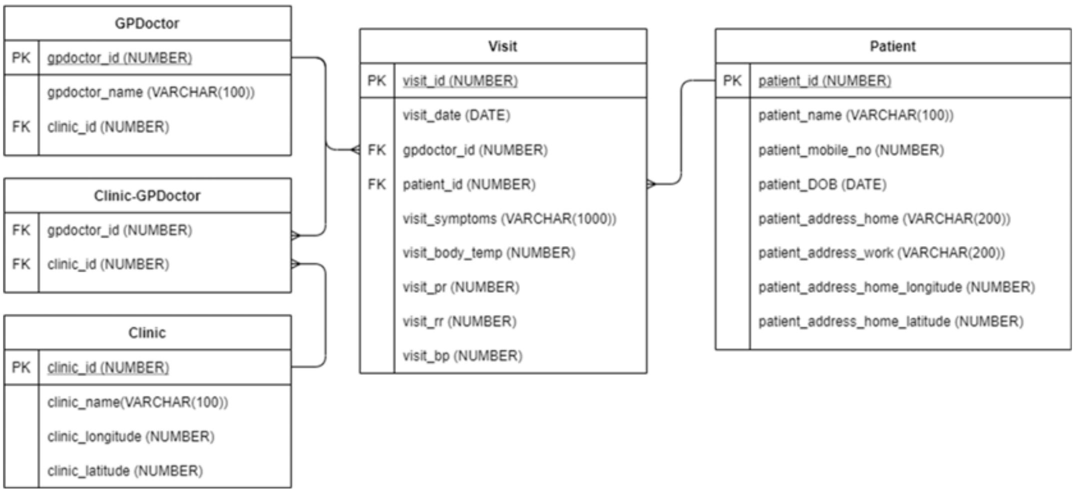


Figure 5: A physical Entity Relationship (ER) data model of the proposed database. It consists of five relationships. A GP doctor can work in multiple clinics, hence, the Clinic-GPDoctor relation represents the many-to-many relationship between these two relations. The Visit relation is the most important one which CDOWatcher depends on to detect diseases outbreaks.

- **Health behavior 3**: A person is admitted into a hospital with contagious diseases symptoms. The person information are recorded on the hospital information system. Through authorized APIs, the information for this particular person are systematically transmitted to the platform.

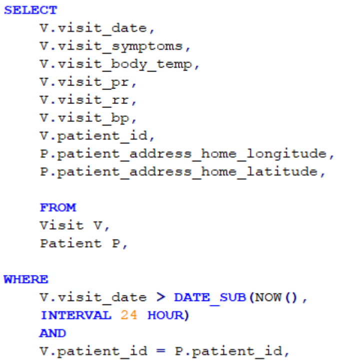
The instantaneous recording of these behaviors (i.e., once health care is sought) enables CDOWatcher to capture diseases outbreaks in near real-time. Clearly, the second behavior follows a crowd-sourcing principal where people voluntarily report their vital signs readings in a relatively fast method. With the ubiquitous of mobile phones, this behavior render itself to be of high potential as a data source [13, 18].

As illustrated by the mobile application prototype in Figure 3, GP doctor or HCW are able to enter patients information after logging in using their credentials and choosing the current clinic. Once the symptoms are typed in and the vital signs are selected, the information will be saved into the database. Also, patients themselves can enter their information voluntarily via the mobile application, as shown in Figure 4.

3.3 Database Management System Module

To accomplish the goal of CDOWatcher, the appropriate datasets must exist. This module is responsible for the persistent storage of data. Accordingly, it acts as the information source and the data central repository for the data mining module discussed next. Note that the database module is populated via the mobile application module, which is discussed above, and prospectively via authorized APIs which are assumed to be available for the platform.

In order to detect contagious diseases outbreaks, diagnostic and non-diagnostic data that are generated when health care is sought are automatically recorded. For instance, when a patient is admitted to a hospital her address and symptoms are recorded. To have a concrete understanding of the recorded data, Figure 5 below shows the Entity-Relationship (ER) diagram for the database module.
The important relation among the five relations in Figure 5 is the Visit relation. It stores spatial temporal information regarding each admitted case, including the geo-location information of the patient's work and home address and timestamp of each hospital visit. Additionally, this relation stores symptoms as raw text data, and stores body temperature, pulse rate, respiration rate and blood pressure as plain numbers. For instance, to retrieve all visits details in the last 24 hour from the database, the query in Figure 6 can be used.

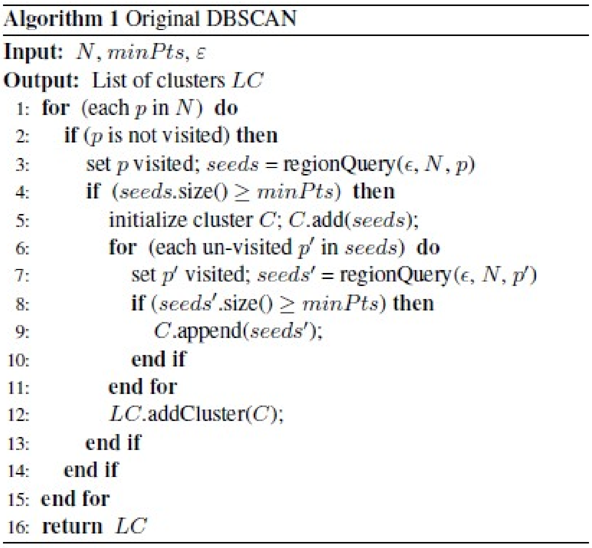


```
SELECT
    V.visit_date,
    V.visit_symptoms,
    V.visit_body_temp,
    V.visit_pr,
    V.visit_rr,
    V.visit_bp,
    V.patient_id,
    P.patient_address_home_longitude,
    P.patient_address_home_latitude,

    FROM
    Visit V,
    Patient P,

WHERE
    V.visit_date > DATE_SUB(NOW(),
    INTERVAL 24 HOUR)
    AND
    V.patient_id = P.patient_id,
```

Figure 6: An SQL query to select all visits details in the last 24 hour. Spatial information are retrieved by joining the two relations Visit and Patient.



Algorithm 1 Original DBSCAN

```
Input: $N$, $minPts$, $\varepsilon$
Output: List of clusters $LC$
for (each $p$ in $N$) do
  if ($p$ is not visited) then
    set $p$ visited; $seeds$ = regionQuery($\epsilon$, $N$, $p$)
    if ($seeds$.size() $\geq minPts$) then
      initialize cluster $C$; $C$.add($seeds$);
      for (each un-visited $p'$ in $seeds$) do
        set $p'$ visited; $seeds'$ = regionQuery($\epsilon$, $N$, $p'$)
        if ($seeds'$.size() $\geq minPts$) then
          $C$.append($seeds'$);
        end if
      end for
      $LC$.addCluster($C$);
    end if
  end if
end for
return $LC$
```

This DBMS module will be developed using the popular open-source relational database management system MySQL Server. The data mining module discussed next depends entirely on the information recorded in this database and frequently submits queries (such as the one above) to generated valuable outputs for the front-end to show.

3.4 Data Mining Module

As mentioned previously in Section 1, the end goal of CDOWatcher is to detect contagious diseases outbreaks in real time, thus allowing prompt actions by decision makers and health officials. This module utilizes several data mining techniques and algorithms that work with different types of data to detect contagious diseases outbreaks in real time. Moreover, since efficiency is a critical factor, those techniques and algorithms must be implemented in their outmost optimized versions thus, allowing at least near-real-time output.

The information sources for this module can be categorized into two types: spatial-temporal type, and raw text type (e.g., medical reports). The former represents an entity by its position (e.g., geolocation information) and its relevant time. These two pieces of data are crucial for detecting diseases outbreaks. The latter is supplementary and is used to augment the detection process.

Having the appropriate datasets, density-based clustering algorithms such as DBScan [19] can be performed to detect diseases outbreaks based on geographical areas. These types of algorithms work well with arbitrary shapes of clusters (which is the case for this project since input data is of spatial temporal type) and is resilient with noise existence in data. However, the dynamic nature of diseases outbreak and their evolving behavior must be considered when implementing such algorithms as these algorithms generally assume offline settings and infrequent updates to the raw input to achieve high performance [20]. Once this assumption (i.e., infrequent updates to data) is excluded, density-based clustering algorithms performances become problematic and will be a major challenge for CDOWatcher to detect outbreaks as they happen with no major delays.

The density-based clustering functionality in CDOWatcher is implemented as it was originally defined in [19]. DBSCAN detects arbitrary shaped clusters with high density using two input parameters: minimum points *minPts*, and minimum distance ε. Using these two parameters, each point $p \in N$ is classified as a point that belongs to a cluster (i.e., a core point or a border point) or not (i.e., noise).

Though, the original algorithm exhibits a quadratic running time complexity of $O(N^2)$. This is clearly visible in Algorithm 1: there are N points and for each one a range query is called, which takes *O(N)* in the worst case. Hence, we propose an approximate version of the clustering algorithm which enables CDOWatcher to run frequently and produce results in near real-time. Specifically, caching based and partition-based optimization techniques were implemented to reduce range query time and also reduce overall number of computations.

Caching-based Optimization: Since CDOWatcher uses Euclidean distance as a distance measure in the range query, by definition, Euclidean distance is symmetric, i.e., $\forall x, y \in N$, we have: $\sqrt{\sum(x_i - y_i)^2} = \sqrt{\sum(y_i - x_i)^2}$.

Therefore, to reduce query time, distance computations for closely-located points are cached in memory to be used later in subsequent calls for the range query function. Note that each distance computation is made up of five float point operations, hence, if a distance computation is served from memory, then five FLOPS of computational overhead will be saved.

Given N points in a 2-D dimensional space $p_1; p_2; \dots; p_N$ as an input, each region query in Algorithm 1 will entail $(N-1)$ distance computations. Consequently, there will be $N \times (N-1)$ distance computations. Exhaustively caching all of these distance computations will require a cache M of size $|M| = N(N+1)/2$ (recall that Euclidean distance is symmetric). Hence, theoretically, the number of distance computations which can be served from memory is almost half of the total number of distance computations. However, having a cache with $O(N^2)$ storage complexity is of impractical nature. Therefore, we propose in CDOWatcher to limit the capacity of $|M| \ll N(N+1)/2$ and have a first-in-first-out data structure (such as a queue) to store these distance computations. The intuition behind this FIFO cache is that data points in the 2-D dimensional space that are located close to each other will most likely be in the seeds of the current point. Once a new distance computation is required, a simple check on the current size of the cache is performed. If the maximum capacity is reached, then the head of the queue is removed.

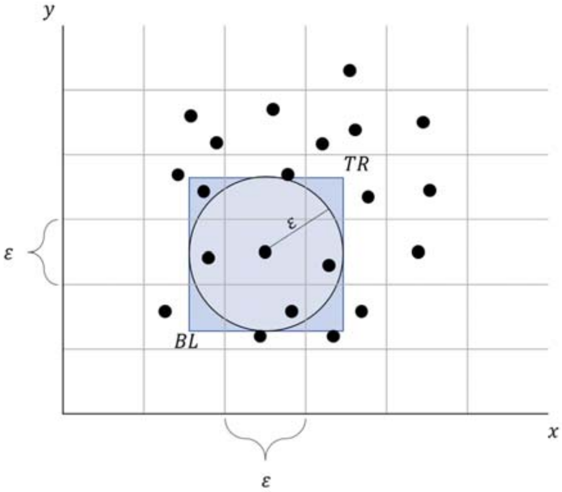


Figure 7: The equal height-width grid *G* is used to create a Quadtree index. A point $p \in N$ can be easily checked if it is a core point using this index in logarithmic time complexity.

Partition-based Optimization: CDOWatcher also utilizes a specially created gird G which enables avoiding calling the range query all together. Initially, the 2-D dimensional space is partitioned into cells of equal height and width, i.e., $H = W = \varepsilon$, giving a total of $\frac{1}{\varepsilon^2}$ cells. Then, each point $p \in N$ is assigned to a cell in G if its coordinates are contained within that cell. This grid G is then indexed by a spatial Quadtree index [21].

Based on the algorithm above, if *seeds.size()* is less than of *minPts*, then there is no need to run the range query. Hence, CDOWatcher checks that in logarithmic time complexity (instead of linear) using the created Quadtree index on G as follows: for any $p \in N$, the maximum Euclidean distance ε forms a circle centered on p with diameter 2ε . The enclosing square of that circle has the following top-right and bottom-left corner points (refer to Figure 7):

$$TR(x + \varepsilon, y + \varepsilon)$$
$$BL(x - \varepsilon, y - \varepsilon)$$

All cells (partially) covering this square (i.e., 8-neighborhood connected cells) retrieved by applying a binary search algorithm on the Quadtree index. An upper bound UB of the ε-neighborhood of p is estimated by adding up the number of points in each retrieved cell. Similarly, a lower bound LB of ε-neighborhood of p is estimated to be the same number of points in the cell that contains p. With these two bounds, for any given $p \in N$, if $LB > minPts$ then there is no need to execute the range query since the actual ε-neighborhood of p will be less than or equal to *UB*.

The proposed platform, CDOWatcher, will also utilize text mining techniques to swift through thousands of medical reports that are recorded when a patient is admitted to a hospital in order to identify any symptoms related to known contagious diseases [15, 22]. While it might seem rational to directly search these reports and apply text similarity methods to identify contagious diseases, it is rather the case that these reports need to be normalized beforehand to minimize noise in data and maximize the accuracy of search results [23]. Natural Language Processing (NLP) techniques such as tokenization, lemmatization and named entity recognition [24] are all adequate techniques to be performed on raw medical reports to support achieving the platform goal.

3.5 CDOWatcher

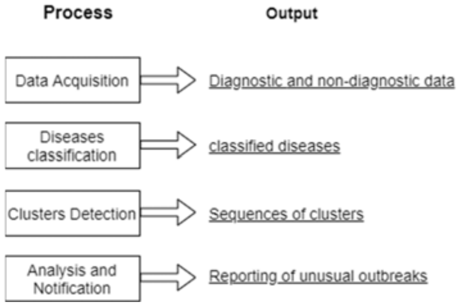


Figure 8: Overall process view of CDOWatcher. It consists of four main processes where the output of each one is considered as an input to the following process.

The proposed platform, CDOWatcher, incorporates the functionality of all above modules in a systematic order to detect contagious diseases outbreaks in real-time. Figure 8 depicts the process view of CDOWatcher. The first process is thoroughly discussed and illustrated above in Sections 3.2 and 3.3.

Classifying diseases (i.e., the second process) using collected diagnostic and non-diagnostic data involves applying NLP techniques and advanced similarity measures. While it is commonly known that laboratory tests are the most reliable and accurate tests to diagnose diseases (e.g., a PCR test for 2019- nCoV [25, 26]), the role of data mining techniques should not be neglected as they consistently show in literature promising results for identifying diseases based on health care data [27, 28, 29, 30]. With the assumption of predefined symptoms for contagious diseases and the existence of definitive vital signs readings which characterize these diseases, it is possible and practical to identify a disease using data mining techniques with high confidence. For instance, the respiratory contagious disease 2019-nCoV correlates with specific symptoms such as fever, coughing, shortness of breath, and diarrhea [31, 32]. We defer investigating diseases classification based on diagnostic and non-diagnostic data for future work.

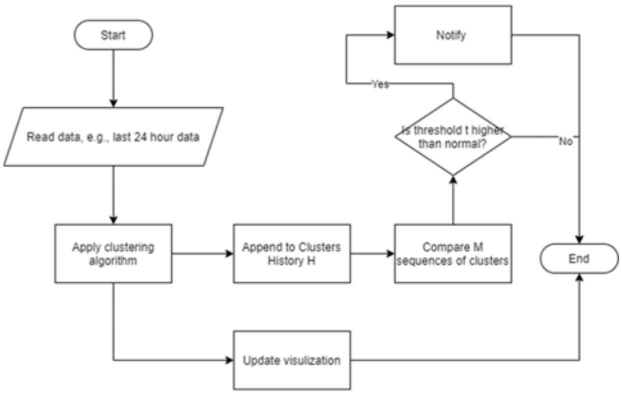
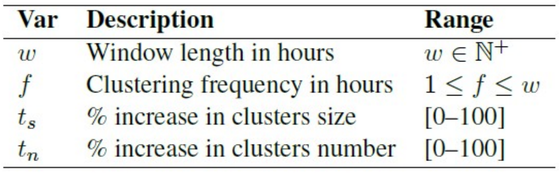

Start
Read data, e.g., last 24 hour data
Apply clustering algorithm
Append to Clusters History H
Compare M sequences of clusters
Is threshold t higher than normal?
Yes
No
Notify
End
Update visulization

Figure 9: Detailed flowchart illustrating the processing pipeline of CDOWatcher

Clusters detection and reporting of unusual outbreaks (third and fourth processes) are best explained using Figure 9 and Table 1. As illustrated in Figure 9, CDOWatcher initially reads a window of size w of collected data. Then, it applies a clustering algorithm to detect the number of clusters and their respective sizes. The detected clusters are then visualized on the dashboard, and they are also appended to the history of detected clusters H. Next, to perform the actual detection of outbreaks, CDOWatcher reads and compares the recently detected m successive clusters in H to find out if there is an abnormal behavior that warrants a notification. Specifically, CDOWatcher compares those m successive clusters in terms of clusters size and number with baseline estimations (i.e., normal ranges) t_s and t_n, as shown in Table 1. If computed percentages overshoot the baseline estimations, then CDOWatcher issues a warning on the dashboard. This whole process is executed based on the frequency f set by the user. Note that m is bounded by $2 \leq m \leq f$.

Table 1: User supplied parameters



Var	Description	Range
w	Window length in hours	$w \in \mathbb{N}^+$
f	Clustering frequency in hours	$1 \leq f \leq w$
t_s	% increase in clusters size	[0–100]
t_n	% increase in clusters number	[0–100]

4. Experiments and Discussion

To evaluate the usefulness of the proposed platform and showcase its functionalities, a prototype was designed and developed based on the aforementioned details. The performance of the prototype depends heavily on the clustering step as clustering is performed frequently. Hence, in the following section we experimentally compare the efficiency of the clustering optimization techniques proposed in Section 3.4.

4.1 Experiments Setup

We evaluated the following two optimization techniques:

- **DBSCAN+CO**: optimizes distance computations by caching closely located points to be used later in subsequent calls for the range query function.
- **DBSCAN+PO**: prunes range query calls for points that are guaranteed to have less points than *minPts* in their ε-neighborhood.

We also used four different datasets in our evaluation, all of which contain two-dimensional data points:

- **Synthetic (SYN):** A synthetic dataset which consists of $N = 1200$ data points and forms three non-overlapping clusters.
- **R15 [33]:** This dataset contains $N = 600$ points distributed over 15 clusters.
- **D31 [33]:** Similarly, this dataset contains $N = 3100$ points distributed over 31 clusters.
- **UCI Yeast [34]:** This dataset contains $N = 1484$ data points, some of which are overlapping with other data points. Out of the eight dimensions in the Yeast dataset, we extracted the first two dimensions because we are evaluating datasets with two-dimensional data points.

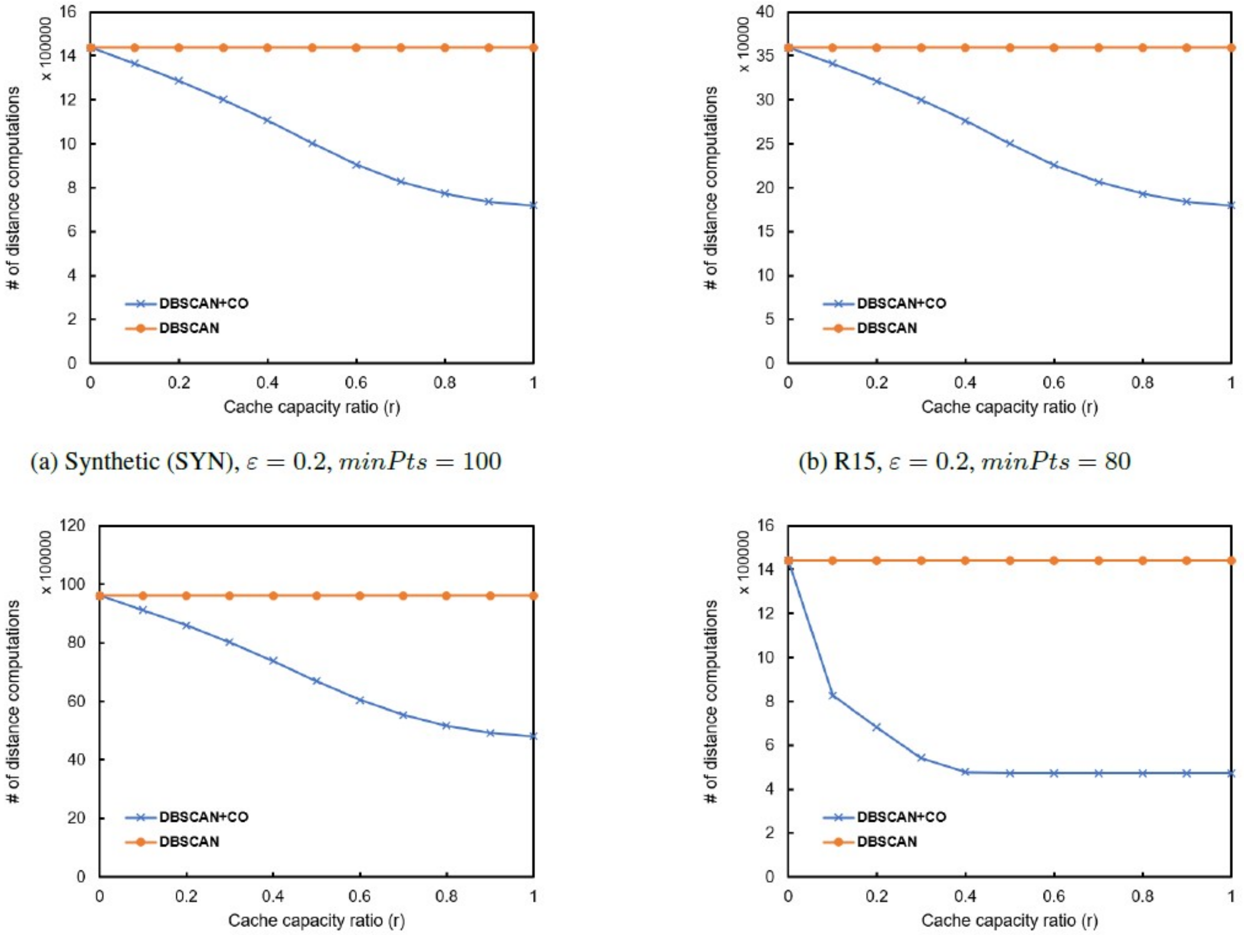
To accurately compare the efficiency of our proposed optimization techniques, we opted to use the following two performance indicators:

- Total number of distance computations: recall that each region query in Algorithm 1 entails $(N - 1)$ distance computations.
- Number of pruned queries: i.e., total number of times the region query in Algorithm 1 is pruned.

4.2 Results

In the first set of experiments, we evaluated the efficiency of the caching-based optimization technique (DBSCAN+CO) against the traditional DBSCAN algorithm. Specifically, we measured the total number of distance computations against the cache capacity as a ratio r of total number of cached computations.

As Figure 10a, Figure 10b and Figure 10c illustrate, increasing r leads to more savings of distance computations. For instance, in Figure 10a, Figure 10b and Figure 10c, 23% savings of distance computations is achieved by setting $r = 0.4$. The most interesting case is illustrated in Figure 10d as 66% savings of distance computations is achieved by setting $r = 0.4$. The reason behind the great savings shown in the Yeast dataset is attributed towards the similarity in data points in the dataset: 34% of points in the dataset are similar.



(a) Synthetic (SYN), $\varepsilon = 0.2$, $minPts = 100$

(b) R15, $\varepsilon = 0.2$, $minPts = 80$

Figure 10: Caching-based optimization results on four different datasets

Hence, our proposed algorithm DBSCAN+CO is able to efficiently utilize the similarity within the dataset to greatly reduce the total number of distance computations. Though, in datasets with rare similarity among data points (e.g., R15 and D31 datasets), DBSCAN+CO provides little benefits over the baseline algorithm. One possible modification to mitigate this limitation is to introduce controlled similarity among the data points which would yield approximate results. However, we leave that for future work as it is out of this work's scope.

In the second set of experiments, we evaluated the efficiency of the partition-based optimization technique against the baseline. As a performance indicator, we are measuring the number of pruned region queries for both algorithms. As Table 2 shows, DBSCAN+PO outperforms the baseline algorithm in number of pruned queries, resulting in much better performance, across all four datasets. Specifically, DBSCAN+PO is able to prune 23%-34% more queries than the baseline algorithm using the Quadtree index as described above in Section 3.4. Note that the baseline algorithm also prunes queries, but differently than DBSCAN+PO. Namely, if a point *p* is visited in the outer or inner for-loop, then the algorithm skips calling the region query for that point. DBSCAN+PO takes this one step further and uses the Quadtree index it maintains to prune points using a simple check without any notable overhead.

In the third experiment, we tested the scalability of our proposed optimization techniques. Using the Birch3 [35] synthetic dataset, we increased the size *N* from 1000 points to 100,000 and measured number of distance computations for both techniques. Figure 11 indicates that both techniques exhibit a linear increase in number of computations, relative to *N*, outperforming the baseline algorithm.

Table 2: Partition-based optimization results.

Dataset:	Synthetic (SYN)	R15	D31	UCI Yeast
$minPts$	80	40	80	100
ε	0.2	0.2	0.08	0.2
# probed points	1200	600	3100	972
# pruned points (DBSCAN+PO)	955	320	610	864
# pruned points (DBSCAN)	736	277	397	594
$\|G\|$	25	25	100	25

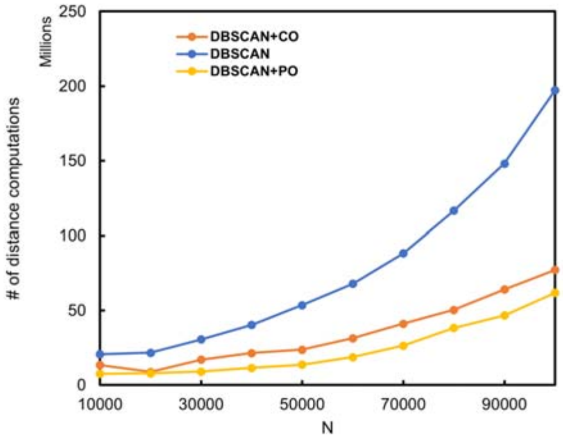


Figure 11: Scalability of proposed techniques, with *r* = 0.3, *minPts* = 0.06%*N*, ε = 0.2

5. Conclusion

Developing effective solutions for public health monitoring is ongoing research that is targeted by new performance demands. This paper proposed a platform to achieve early detection of contagious diseases outbreaks, to allow for prompt and robust prevention management of those outbreaks. Consisting of multiple modules, CDOWatcher (Contagious Diseases Outbreaks Watcher) utilizes data mining algorithms and techniques to detect clusters of outbreaks and gather data through a mobile module which provide an instantaneous recording of diagnostic and non-diagnostic data. We showed preliminary performance results of the platform to demonstrate the optimization techniques embedded in CDOWatcher, which consistently outperformed the baseline algorithm.

Acknowledgments

This research is funded and supported by the Deanship of Scientific Research, Imam Mohammad Ibn Saud Islamic University, Saudi Arabia, Grant No. (20-12-18-011).

References

[1] W. H. Organization, Managing epidemics: key facts about major deadly diseases. World Health Organization, 2018.

[2] Y.-C. Wu, C.-S. Chen, and Y.-J. Chan, “The outbreak of covid-19: An overview,” Journal of the Chinese Medical Association, vol. 83, no. 3, p. 217, 2020.

[3] H. Alahdal, F. Basingab, and R. Alotaibi, “An analytical study on the awareness, attitude and practice during the covid-19 pandemic in riyadh, saudi arabia,” Journal of Infection and Public Health, vol. 13, no. 10, pp. 1446 – 1452, 2020.

[4] Y. K. Dwivedi, D. L. Hughes, C. R. Coombs, I. D. Constantiou, Y. Duan, J. S. Edwards, B. Gupta, B. Lal, S. K. Misra, P. Prashant, R. Raman, N. P. Rana, S. K. Sharma, and N. Upadhyay, “Impact of COVID-19 pandemic on information management research and practice: Transforming education, work and life,” Int. J. Inf. Manag., vol. 55, p. 102211, 2020.

[5] S. Binder, A. M. Levitt, and J. M. Hughes, “Preventing emerging infectious diseases as we enter the 21st century: Cdc’s strategy.,” Public Health Reports, vol. 114, no. 2, p. 130, 1999.

[6] M. M. Wagner, F.-C. Tsui, J. U. Espino, V. M. Dato, D. F. Sittig, R. A. Caruana, L. F. McGinnis, D. W. Deerfield, M. J. Druzdzel, D. B. Fridsma, et al., “The emerging science of very early detection of disease outbreaks,” Journal of public health management and practice, vol. 7, no. 6, pp. 51–59, 2001.

[7] F. Mostashari and J. Hartman, “Syndromic surveillance: a local perspective,” 2003.

[8] J. W. Buehler, R. S. Hopkins, J. M. Overhage, D. M. Sosin, and V. Tong, “Framework for evaluating public health surveillance systems for early detection of outbreaks,” Morbidity and Mortality Weekly Report, vol. 53, no. RR-5, 2004.

[9] K. J. Henning, “What is syndromic surveillance,” MMWR supplements, vol. 53, pp. 5–11, 2004.

[10] D. M. Sosin, “Draft framework for evaluating syndromic surveillance systems,” Journal of urban health, vol. 80, no. 1, pp. i8–i13, 2003.

[11] L. Samaras et al., “Predicting epidemics using search engine data: a comparative study on measles in the largest countries of europe.,” BMC public health vol. 21, 2021.

[12] P.W. Yoon, A. I. Ising, and J. E. Gunn, “Using syndromic surveillance for all-hazards public health surveillance: Successes, challenges, and the future:,” Public Health Reports, vol. 132, 2017.

[13] J. P and Y. S., “Early warning of epidemics: towards a national intelligent syndromic surveillance system (nisss) in china,” BMJ Glob Health, vol. 5, no. 10, 2020.

[14] E. Christaki, “New technologies in predicting, preventing and controlling emerging infectious diseases,” Virulence, vol. 6, no. 6, pp. 558–565, 2015.

[15] R. Brixtel, G. Lejeune, A. Doucet, and N. Lucas, “Any language early detection of epidemic diseases from web news streams,” in IEEE International Conference on Healthcare Informatics, ICHI 2013, 9-11 September, 2013, Philadelphia, PA, USA, pp. 159–168, IEEE Computer Society, 2013.

[16] N. Rostamzadeh, S. S. Abdullah, and K. Sedig, “Visual analytics for electronic health records: A review,” Informatics, vol. 8, no. 1, 2021.

[17] H. E. Hughes, O. Edeghere, S. J. O’Brien, R. Vivancos, and A. J. Elliot, “Emergency department syndromic surveillance systems: a systematic review.,” BMC Public Health, vol. 20, no. 1, pp. 1– 15, 2020.

[18] M. I. B. Osfoor, H. A. Alfaleh, and A. M. Albarrak, “Dynamicform: A mobile-based solution for covid-19 hospital assessment form,” in 4th IEEE International Conference on Computer Applications and Information Security (ICCAIS), 2021.

[19] M. Ester, H. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA (E. Simoudis, J. Han, and U. M. Fayyad, eds.), pp. 226–231, AAAI Press, 1996.

[20] J. Gan and Y. Tao, “DBSCAN revisited: Mis-claim, unfixability, and approximation,” in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015 (T. K. Sellis, S. B. Davidson, and Z. G. Ives, eds.), pp. 519–530, ACM, 2015.

[21] J. Tayeb, O¨ . Ulusoy, and O. Wolfson, “A quadtree-based dynamic attribute indexing method,” Comput. J., vol. 41, no. 3, pp. 185–200, 1998.

[22] R. Steinberger, F. Fuart, E. van der Goot, C. Best, P. von Etter, and R. Yangarber, “Text mining from the web for medical intelligence,” in Mining Massive Data Sets for Security - Advances in Data Mining, Search, Social Networks and Text Mining, and their Applications to Security, Proceedings of the NATO Advanced Study Institute on Mining Massive Data Sets for Security, Gazzada (Varese), Italy, 10-21 September 2007, vol. 19 of NATO Science for Peace and Security Series - D: Information and Communication Security, pp. 295–310, IOS Press, 2007.

[23] N. Azam and J. Yao, “Comparison of term frequency and document frequency based feature selection metrics in text categorization,” Expert Syst. Appl., vol. 39, no. 5, pp. 4760–4768, 2012.

[24] R. Collobert, J.Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa, “Natural language processing (almost) from scratch,” J. Mach. Learn. Res., vol. 12, pp. 2493–2537, 2011.

[25] K.-S. Yuen, Z. W. Ye, S.-Y. Fung, C.-P. Chan, and D.-Y. Jin, “Sars-cov-2 and covid-19: The most important research questions,” Cell and Bioscience, vol. 10, 2020.

[26] O. Vandenberg, D. Martiny, O. Rochas, A. van Belkum, and Z. Kozlakidis, “Considerations for diagnostic covid-19 tests,” Nature Reviews Microbiology, vol. 19, Mar. 2021.

[27] M. A. N. Banu and B. Gomathy, “Disease forecasting system using data mining methods,” in 2014 International Conference on Intelligent Computing Applications, pp. 130–133, 2014.

[28] R. H. Abiyev and M. K. S. Ma’aitah, “Deep convolutional neural networks for chest diseases detection.,” Journal of Healthcare Engineering, vol. 2018, no. 2018, p. 4168538, 2018.

[29] M. K. S. Ma’aitah, R. Abiyev, and I. J. Bush, “Intelligent classification of liver disorder using fuzzy neural system,” International Journal of Advanced Computer Science and Applications, vol. 8, no. 12, 2017.

[30] D. Zhang et al., “The ai index 2021 annual report,” AI Index Steering Committee, Human-Centered AI Institute, Stanford University, Stanford, 2021.

[31] Y.-H. e. a. Jin, “A rapid advice guideline for the diagnosis and treatment of 2019 novel coronavirus (2019-ncov) infected pneumonia,” Military Medical Research, vol. 7, 2020.

[32] H. C. et al., “Clinical features of patients infected with 2019 novel coronavirus in wuhan, china,” Lancet, vol. 10, 2020.

[33] C. J. Veenman, M. J. T. Reinders, and E. Backer, “A maximum variance cluster algorithm,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 9, pp. 1273–1280, 2002.

[34] D. Dua and C. Graff, “UCI machine learning repository,” 2020.

[35] T. Zhang, R. Ramakrishnan, and M. Livny, “Birch: A new data clustering algorithm and its applications,” Data Mining and Knowledge Discovery, vol. 1, no. 2, pp. 141–182, 1997.