2021-10

Spatiotemporal IoT Data Analysis and Prediction using OGC Open Standards in Digital Contact Tracing and Air Quality Prediction Applications

Ojagh, Soroush

UNIVERSITY OF CALGARY

Spatiotemporal IoT Data Analysis and Prediction using OGC Open Standards in Digital Contact Tracing and Air Quality Prediction Applications

by

Soroush Ojagh

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN GEOMATICS ENGINEERING

CALGARY, ALBERTA

OCTOBER, 2021

Abstract

Internet of Things (IoT) devices have become more and more integral to our everyday lives by the advent of sensors, computing and communications technology. The increasing use of IoT devices has caused a demand for the processing and analysis of spatial and temporal data. IoT devices such as smartphones, Bluetooth Low Energy beacons, and air quality sensing systems collect various types of spatiotemporal data used by various applications. This M.Sc. thesis focuses on two research applications of spatiotemporal IoT data analysis.

First, an IoT system is designed to automatically trace human contacts with contaminated places and diagnosed carriers in indoor environments. This designed digital contact tracing system aims to find and notify possibly exposed people at the earliest possible stage to "flatten the curve" of spreading coronavirus. This research proposes a hierarchical graph-based data model that provides contact tracers with different granularity levels in spatial, temporal, and contextual dimensions. The spatial dimension of our data model is designed based on the Open Geospatial Consortium (OGC) IndoorGML standard. In comparison to other digital contact tracing systems, some key differences can be noted for this IoT-based contact tracing system. First, most digital contact tracing applications have been focused on person-to-person interactions. However, both person-to-person and person-to-place types of spreading infectious diseases have been considered in our system. Second, using OGC IndoorGML standard and considering topological relationships between indoor spaces, a validation procedure is designed to detect noisy indoor trajectory points. Third, different types of low-price BLE beacons have been utilized in this application. Finally, various user contexts (*e.g.*, activity type and vulnerability to be exposed by coronavirus) are considered in an enhanced digital contact tracing application. Evaluating the functionality of our proposed system proved that the proposed contact tracing system had recognized 44.98% fewer

possible contacts in comparison to traditional contact tracing applications. Also, 73.53% of noisy indoor trajectory points have been recognized using the proposed validation procedure. In all, the proposed data model and designed IoT architecture provide more flexibility in considering additional spatiotemporal information for indoor applications.

Second, a mixed edge-based and cloud-based framework have been designed to predict air quality using IoT sensors based on deep learning methods. In this application, the most challenging problem is dealing with inconsistent data provided by IoT sensors. The term "inconsistent" refers to the fact that the actual time series from IoT sensors are generally incomplete. Thus, performing predictions on inconsistent spatiotemporal data will provide lower quality results. Hence, data preprocessing comprising filling up missing data should be of equal importance and sometimes more important than providing a prediction model. In this application, an edge-based data preprocessing approach is proposed to predict missing values and aggregate the raw PM2.5 measurements collected by PM2.5 IoT sensors. Both spatial and temporal information has been considered in edge-based preprocessing to improve the estimated values for missing measurements. Also, three different aggregation techniques have been applied by the edge component to reduce data granularity from minutes to an hour. The preprocessed data will then be combined with meteorological data and feed into the Long Short Term Memory (LSTM) technique representing the multivariate deep learning-based prediction model. To evaluate the significance of the proposed preprocessing technique, the prediction model's performance was evaluated on both preprocessed and unprocessed datasets. The proposed preprocessing technique improved air quality prediction accuracy by 40.18 percent on average.

Keywords: Location-Based Services, Internet of Things Devices, Contact Tracing Application, OGC IndoorGML standard, Air Quality Prediction, Deep Learning.

Preface

This thesis resulted in the following publications:

Ojagh, Soroush*, Sara Saeedi, and Steve HL Liang. "A Person-to-Person and Person-to-Place COVID-19 Contact Tracing System Based on OGC IndoorGML." ISPRS International Journal of Geo-Information 10, no. 1 (2021): 2. (doi.org/10.3390/ijgi10010002).

Ojagh, Soroush, Francesco Cauteruccio, Giorgio Terracina*****, and Steve HL Liang. "Enhanced Air Quality Prediction by Edge-Based Spatiotemporal Data Preprocessing." Journal of Computers & Electrical Engineering. (Conditionally Accepted; Manuscript ID: COMPELECENG-D-21-00289R1).

Acknowledgments

I would like to express my profound gratitude to my supervisor Professor Steve Liang for his continuous support and encouragement throughout my research. His ideas, feedback, and vision helped me to shape my research career while he welcomed new ideas and suggestions. I also like to acknowledge helps and resources provided for this research from GeoSensorWeb lab and SensorUp Inc. I would like to extend my appreciation to Dr. Sara Saeedi, who provided invaluable guidance and positive impact throughout my research.

Finally, I wish to acknowledge that my research has been generously supported by the Alberta Innovative Scholarship Award, University of Calgary Eyes High International Doctoral Scholarship, and Chancellor Award.

Dedication

To my parents and brothers, without their support, success would not have been possible

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 General Background

In recent years, advances in developing small, low-cost, low-power sensors, edge-based and cloud-based computing, and communication technologies enabled the Internet of Things (IoT) to be used in various applications, including health [1, 2] and smart cities [3, 4] applications. IoT applications are growing in popularity with human lives in an increasingly pervasive and intimate manner [5]. Connecting 127 new IoT devices on a second basis, it is estimated that 35 million IoT devices will be embedded in human lives by 2021 [6]. Low-priced IoT sensors can provide real-time data that can be utilized in real-time informative decision-making, planning, monitoring, and responding to different events in various applications [7].

There are various challenges in spatiotemporal data analysis [8]. First, the spatial and temporal data model and its use for analysis and prediction paradigms need to be defined [1]. Second, the quality of data provided by IoT sensors influences the quality of results of the spatiotemporal analysis. For example, if there is noisy or missing data in the spatiotemporal dataset, less accurate results from spatiotemporal analysis or prediction would be expected. So, the quality of data needs to be improved using data preprocessing techniques. Third, spatial-temporal analysis imposes considerable computational costs on the IoT system. Therefore, the system architecture must support both edge and cloud computing to provide continuous IoT data collection by sensors. Another critical challenge in a scalable and extensible IoT architecture is using open standards for interoperability [7]. These challenges are different from one use case to the others; therefore, they need to be defined and studied based on various applications, sensors, and computation costs.

The proliferation of the Internet of Things devices has resulted in an exponential increase in the need for processing and analyzing spatial and temporal data. Thus, in order to yield value-added information from the enormous volume of continuous time series collected by IoT devices, spatial-temporal data analyses must be performed. This thesis aims to demonstrate two different applications of spatial-temporal analysis to data collected by IoT devices. In the first application, the analysis of human movement, a time series of location information across space and time, is carried out to identify the spread of infectious diseases and discover the spatial-temporal patterns of safe indoor workplaces [9]. In other words, spatial, temporal, and contextual information is collected by IoT devices and then used to trace human contact with people and indoor places contaminated by the coronavirus.

Predicting the future status of different phenomena such as air quality based on historical data provided by IoT sensors is essential to find the spatial-temporal pattern of air quality and determine safe zones in the city for vulnerable people [10]. So, we focus on air quality prediction in the second application, as another application of spatiotemporal analysis using air quality data provided by IoT air pollution sensors. In conclusion, conducting spatiotemporal analyses on data derived from IoT sensors for health-related purposes can be viewed as a commonality in both applications. The following sections provide a brief background on those applications: a) contact tracing and b) air quality prediction.

1.1.1 Contact Tracing Applications

The exponential increase in the number of people with the coronavirus has motivated many governments and developers to develop different digital contact tracing systems to ease the burden of manual contact tracing on public health departments. For example, both web-based (*i.e.*, TeamSense [11]) and mobile (*e.g.*, COVID Alert [12] and ABTraceTogether [13]) contact tracing

systems are used in Canada to combat the COVID-19 pandemic. The high transmission rate of the novel COVID-19 severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) has led to a global epidemic [4, 5]. Despite the fact that spreading COVID-19 is still under debate among scholars [6, 7], there are two ways in which it can spread. The infection can be transmitted directly by direct contact with an infected person or indirectly by touching contaminated objects or surfaces. Contact tracing is a public health practice that different governments have well-accepted to stop the spread of coronavirus. This can be done manually by human resources or digitally using IoT devices. The time-consuming nature of manual contact tracing (*i.e.*, in-person interviewing the infected individuals) compelled the use of digital contact tracing applications.

Digital contact tracing can be applied to a large scale in a short time to stop the spread of coronavirus by extending the use of mobile technology, scaling up cloud data storage, and increasing the capabilities of physical devices to access the internet [16]. Different governments have begun looking at the possibility of using digital contact tracing to combat COVID-19 by launching smartphone applications such as TraceTogether (Singapore) [17], CovidSafe (Australia) [18] and PACT (East Coast) [19]. The most widely applied approach used by contact tracing applications is the use of Bluetooth communication between nearby smartphones [14]. In other words, most contact tracing applications [14] rely on proximity estimation and the duration of contact between nearby smartphones.

A spatial-temporal analysis of IoT data could also provide some additional benefits in terms of improving the accuracy of digital contact tracing applications. Firstly, the location context can be utilized to enhance the accuracy of contact tracing applications. Incorporating location context is essential because the SARS virus can be transmitted by touching a shared surface previously touched by a carrier [20]. From this perspective, it is also derived that the temporal sequence of

visits to a shared place also plays an important role in enhancing the accuracy of the contact tracing application. The context of location was considered in various studies, such as the SafePaths project for outdoor contact tracing [20]. Several factors contribute to the importance of indoor contact tracing in contact tracing applications. Enclosed indoor environments pose higher risks of community spread than are outdoor environments [8, 10, 21], and people usually spend most of their time in an indoor environment [22, 23]. However, indoor locations and the order of visiting common spaces have not been addressed [12, 20].

Secondly, semantic contexts such as cleaning and disinfection activities need to be taken into account by contact tracing applications to improve their accuracy. The cleaning and disinfection of commonly visited locations can effectively stop the chains of the spread of the coronavirus [26] and should be accounted for in the contact tracing application. As an example, a cleaned indoor place may be less likely to transmit the coronavirus to other individuals after being visited by an infected individual and before being visited by others. However, user contexts have been overlooked in contact tracing applications [1]. A digital contact tracking system that combines spatiotemporal movement trajectories and semantic contexts is required to bridge the existing research gaps. The proposed contact tracing system can find the workplace risk of infection and help vulnerable people find safe places or outbreaks online [7].

1.1.2 Air Quality Prediction

More than 60,000 people die each year due to air pollution, and research and industry are increasingly thinking about how to monitor and predict the air quality. When it comes to human health, air quality misprediction would be too expensive, especially for those patients with asthma or other pulmonary diseases that should not be exposed to contaminated air. Hence, we are in growing need of making the results of the predictive algorithm as reliable as possible by applying

data preprocessing and data cleaning techniques. It is possible to monitor air quality with air quality monitoring stations, such as those that are operated by government agencies or by non-governmental organizations that can then be used for monitoring purposes. Air quality monitoring stations can indeed provide highly trustworthy data on the air quality of an area. However, the high cost of air quality sensors limits its scalability. As an example, only four air quality stations were established within the boundary of Calgary, Alberta, by the Calgary Region Airshed Zone Society (CRAZ)[1]. Air pollution can vary greatly within the same neighborhood due to meteorological variables such as wind direction. So, there is a need for finer granularity in the spatial distribution of air quality monitoring stations. IoT sensors can provide finer granular air quality data at a lower price. In other words, there is a need to use IoT sensors, which are more affordable than air quality stations, to better protect vulnerable populations by providing finer granular air quality data.

Real-time air quality measurements are essential for effective decision-making in real-time, regulatory planning, and understanding trends in air quality over time. However, microclimates factors (*e.g.*, topography and land cover) may rapidly change air quality measures from one street to another. So, air quality prediction can be of the same importance to provide an early warning against harmful air pollutions for sensitive groups with pre-existing respiratory or cardiovascular conditions. Among all air pollutants such as suspended nitrogen dioxide, ozone, sulfur dioxide, and carbon monoxide, particulate matter with a diameter less than 2.5 micrometers known as PM2.5 is recognized as the most dangerous atmospheric pollutant [14]. As a result, predicting PM2.5 has increasingly attracted more attention and has recently become a hot topic among researchers. Low-cost IoT sensors can provide real-time PM2.5 measurements and, consequently, provide a better understanding of local-scale PM2.5 concentration.

[1] https://www.iqair.com/ca/profile/the-calgary-region-airshed-zone

Predicting air quality from data collected from IoT sensors is a challenging task for several reasons. First, since measurements are sometimes missing and noisy, IoT data sets are as inconsistent as other real-world data sets. Predicting air quality from such incomplete data sets is difficult using predictive algorithms. Second, air quality has spatiotemporal patterns that exhibit nonlinear behavior [15]. For example, the level of PM2.5 varies spatially and temporally depending upon meteorological conditions such as wind speed and temperature [16, 17]. Air quality predictions can even be more complicated when other factors such as traffic and human activities are taken into account. Therefore, linear techniques such as linear regression are not suitable for predicting air quality. Thirdly, communications bottlenecks caused by too much data transmitted from IoT sensors to central servers can result in data packet losses, delays, and real-time computation constraints.

To conclude, making air quality prediction as reliable as possible is always necessary because it might affect human health. In order to accomplish this, it is necessary to apply preprocessing and data cleaning techniques to the inconsistent data provided by air quality IoT sensors. Finally, air quality needs to be predicted as a nonlinear, multivariate, and spatiotemporal pattern using the deep learning technique.

1.2 Thesis Rationale

As seen in Figure 1.1, the spatiotemporal data provided by IoT sensors was studied in two different applications for this thesis: a) contact tracing application and b) air quality prediction.

The first application focuses on the spatiotemporal analysis of human location time series using a hierarchical graph-based data model. In more detail, we proposed a graph-based data model that considers the semantics of indoor locations, time, and users' contexts in a hierarchical structure and encoded using the open OGC IndoorGML standard. The proposed graph-based data

model enables us to store and analyze real-time data showing users' movement trajectories in indoor environments. Then, the functionality of the proposed data model is evaluated for a COVID-19 contact tracing application with scalable system architecture.

In comparison, the main focus of the second application is on analyzing air quality data collected by IoT sensors to provide more accurate air quality prediction. In more detail, a mixed edge-based and cloud-based framework with the final goal of PM2.5 value prediction is designed for this application. Taking PM2.5 as a spatiotemporal, multivariate, and non-linear parameter, we could improve the accuracy of PM2.5 prediction using an edge-based preprocessing approach.

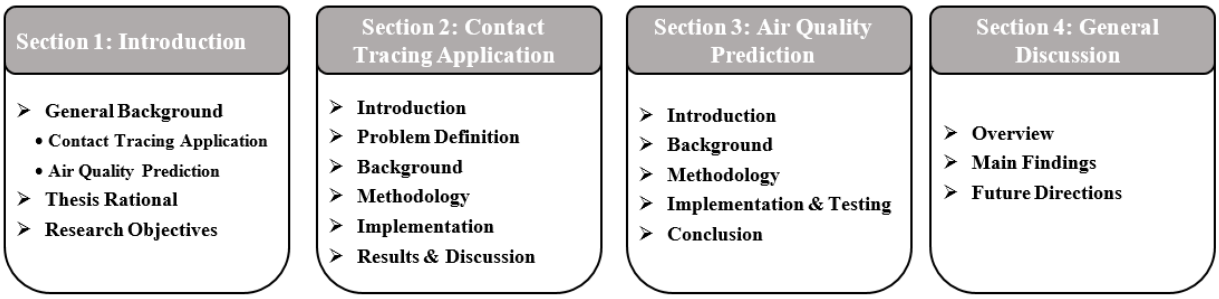


Figure 1.1. An overview of the research outline

The first application is explained in chapter two. As a general overview of this chapter, our motivation, research gaps, and specific contributions are given in subsection 2.2. Subsection 2.3 is assigned to elaborate the research problem using various real-world examples. Next, the literature for closely related subjects, including existing contact tracings applications, movement trajectory segmentation approaches, indoor trajectory modelling and representations, and existing research gaps are explained in subsection 2.4. Our methodology to define a spatiotemporal graph-based data model, including the relationship between spatial, temporal, and contextual spaces, is described in subsection 2.5. Details of our real-world and simulation experiments are then given

in subsections 2.6 and 2.7. Finally, this chapter wraps up by explaining our findings and some future research directions.

The second application is explained in chapter three, where the research motivation, research gaps, and specific contributions are explained in subsection 3.2. Related research in terms of applying different algorithms to predict air quality is briefly reviewed in subsection 3.3. The overall architecture of the applied deep learning methods, including the proposed preprocessing engine and the deep learning predictor, is discussed in subsection 3.4. The statistical information on the real-world data set and other descriptive information about the implementation and testing of the proposed approach are provided in subsection 3.5. We draw conclusions and suggest future directions for research in subsection 3.6. Finally, a general discussion over our main findings and future directions is given in chapter four.

1.3 Research Objectives

The specific objectives in addition to a brief description of the research gaps, key questions and limitations are described in this section.

1.3.1 Spatiotemporal Modeling of person's location and context time series using a hierarchical graph-based data model

A formal spatiotemporal data model as an abstraction of a person's location and context time series is required for any spatiotemporal analysis on movement trajectories. Indoor trajectory data analysis, such as contact tracing, requires a formal spatiotemporal data model as an abstraction of indoor movement trajectory [18]. In this objective, a graph-based data model is proposed for encoding spatial, temporal, and contextual information about users as moving objects in indoor trajectories. Using a graph data structure representing trajectory data allows movement trajectories

to be stored in natural graph form. The first component of the proposed data model defines an indoor spatial space based on the OGC IndoorGML standard.

The proposed hierarchical graph-based data model allows the indoor movement trajectory to be represented at different levels of granularities. The different levels of granularities provided by the proposed model support the aggregation of indoor semantic trajectories over three dimensions: spatial, temporal, and contextual. The flexibility of the OGC IndoorGML standard in providing spatial units with the ability to have their own semantic and topological relationships in a graph-based data structure makes the OGC IndoorGML standard an appropriate alternative to model spatial dimension of indoor trajectories. OGC IndoorGML standard has been used in various researches to model the indoor space for different applications [18, 19]. However, a general data model including the user contextual information and temporal dimension can be mentioned as a research gap in this objective. This part aims to answer the following specific research questions:

- How can different interpretations of an indoor space such as signal coverage, indoor cells, floors, and buildings be considered in a hierarchical graph-based structure?
- How temporal and contextual information can be modelled in a hierarchical structure?
- How can the real-time location of the moving object be modelled in the general data model?

For this objective, we have considered the following scope and limitations:

- The location of the moving object inside the building is going to be identified using the proximity-based Bluetooth Low Energy technology, which has some constraints in accuracy and installation.
- For the implementation purpose, we consider a single BLE beacon per each OGC IndoorGML cell.
- Although there are some options for seamless positioning systems, we have been focused on indoor environments only.

1.3.2 Validating semantic indoor movement trajectories

Missing or unstable RSSI data made by BLE beacons directly leads to noisy semantic movement trajectories. This issue is caused by delayed signal transmissions and interferences from walls and glass doors [20]. Noisy or unstable RSSI signals make the movement trajectories semantically invalid and lead to less reliable spatiotemporal analysis results. Although validating indoor movement trajectories are of great importance for spatiotemporal analysis, this research has remained untouched. Using the proposed data model, we can consider topological relationships among indoor cells as a noise filtering method to extract semantically valid indoor trajectories. So, the spatial topologic relationships extracted from OGC IndoorGML in the proposed data model are used in this objective as a preprocessing technique to filter out semantically invalid trajectory points. This part aims to answer the following specific research question:

- How topological relationships extracted from OGC IndoorGML can be utilized to identify and filter out noisy trajectory points?

For this objective, we have considered the following scope and limitations:

- Although the topological relationships amongst indoor cells are used to filter out semantically invalid trajectory points, reconstructing missing trajectory points is outside the scope of this research.

1.3.3 Developing an IoT Architecture for COVID-19 contact tracing applications

The importance of digital contact tracing applications motivated us to evaluate the functionality of the proposed hierarchical graph-based data model for contact tracing applications. We developed a scalable IoT cloud-based architecture that is able to i) identify the real-time location of a person in an indoor environment using a BLE proximity-based positioning system, ii) store movement trajectories in a standard-based data model in a graph database, and iii) analyze spatiotemporal trajectories in an enhanced contact tracing application. Although there are generally two ways of transmitting the coronavirus (person-to-person and person-to-place), most of the researches [21] has been focused only on person-to-person virus transmission. However, we will evaluate the flexibility of our proposed data model considering both transmission ways of the coronavirus in indoor environments. In addition, the proposed data model enables us to consider user contextual information such as disinfecting activities in an enhanced contact tracing application. In more detail, the proposed enhanced contact tracing application considers both person-to-person and person-to-place methods of coronavirus transmission, sequential order of visiting places, and disinfection history are considered. This analysis shows the flexibility of our proposed data model to consider additional parameters in COVID-19 contact tracing. This part aims to answer the following specific research questions:

- How can a scalable cloud base architecture be developed to support the proposed graph-based data model?

- How can different types of BLE beacons be utilized as a proximity-based positioning system in the cloud-based IoT architecture?
- How can contextual information such as disinfecting activities be considered using our data model in contact tracing applications?
- How can the proposed data model help to enhance contact tracing applications?
- How can the sequential order of visiting indoor places and considering cleaning activities influence person-to-person contact tracing applications?

For this objective, we have considered the following scope and limitations:

- User contexts can be automatically extracted using their IoT devices. However, we have used a manual approach in which users are required to select their contexts from a predefined list of contextual information. So, the automatic extraction of users' contexts such as cleaning activities and job type is outside the scope of this paper.
- Since we did not have any real-world information regarding movement trajectories of diagnosed cases, we assumed some movement trajectories as the movement trajectory of COVID-19 confirmed cases.
- Only 20 real users tested our real-world mobile application. In addition to the real-world dataset, a simulation is developed to generate users' trajectories for 20,000 users.

1.3.4 Designing a preprocessing procedure to improve the quality of PM2.5 time series

Missing and noisy values make a vast amount of data collected by IoT sensors as inconsistent as other real-world data sets. Predicting air quality based on such incomplete data set

is an arduous task for predictive algorithms, and the final result could be less reliable. In this objective, we design a preprocessing strategy that can be used on the edge for filling missing sensor data where both temporal and spatial information is taken into account. The temporal-based estimation considers the similarity between time series, whereas the spatial-based estimation looks at the nearest sensors to estimate missing values. Both estimations are finally combined with a weighted average that is dynamically set based on the latest available data from the sensor. Also, the preprocessing step reduces the granularity of data (from minutes to hours), thus significantly reducing the amount of information that must be sent to the cloud for the prediction tasks. This part aims to answer the following specific research questions:

- How can similarity between time series be utilized as a temporal estimation method to predict the missing value?
- How can spatial distance between sensors be used as a spatial estimation method to predict the missing value?
- How does a dynamic weighting approach predict the missing value using spatial and temporal parameters?

1.3.5 Predicting air quality for the next hour using historical IoT data

The negative impact of poor air quality on human health attracts increasing attention in air quality prediction using various techniques. The diverse range of prediction algorithms from linear regression to neural network algorithms is applied to predicting air quality [16, 22-24]. One of the most significant drawbacks of conventional methods is that they require many computing resources [25]. Air quality patterns declined as spatiotemporal, non-linear and multivariate parameters have a complex nature, making them unpredictable with some conventional predicting techniques like linear regression. Recently, deep learning has emerged as an alternative solution

to address the shortcomings of conventional linear methods in air quality prediction [25, 26]. LSTM techniques, which provide the potential of better considering both long- and short-term historical knowledge, have been widely applied to predict air quality. For this objective, the aim is to use an LSTM-based method for air quality prediction on the cloud where, to improve the quality of predictions, (clean) sensed data is coupled with weather information. While this deep learning technique has been employed for air quality prediction in other studies [14, 26-30], the impact of missing data along with the inclusion of weather information have not been investigated so far. This part aims to answer the following specific research questions:

- How can meteorological data be feed into the LSTM model to predict the air quality?
- How can the standard LSTM deep learning technique be utilized to predict the PM2.5 concentration for the next hour?
- How can the proposed preprocessing technique influence the result of the predictive algorithm?

1.4 Statement of Contributions and Clarifications on Publications

The contributions of the authors for each publication are as follows:

1.4.1 A Person-to-Person and Person-to-Place COVID-19 Contact Tracing System Based on OGC IndoorGML

1.4.1.1 Authors' Contributions

I designed the hierarchical graph-based data model of the proposed contact tracing system. I implemented and deployed the hierarchical graph-based data model and indoor trajectory preprocessing algorithm. Also, I designed and performed real-world and simulated experiments. Prof. Steve Liang supervised this project, contributed to the architecture design, and reviewed

experimental results. Dr. Sara Saeedi helped supervise the project, contributed to experimental design and IoT computational framework. I took the lead in writing the paper. All authors provided critical feedback and helped shape the research, analysis, and manuscript. Also, all authors discussed the results and contributed to the final findings.

1.4.1.2 Clarifications Regarding My Contributions

My contributions to this paper are:

1. Taking the lead in writing the paper;
2. Proposing the graph-based hierarchical data model and implementing this data model in Neo4j;
3. Developing a mobile application to collect trajectory points and send them to Amazon cloud;
4. Developing an AWS Lambda function to enrich raw observations and store them in Amazon DynamoDB;
5. Designing real-world and simulated experiments and collecting indoor movement trajectories for the real-world scenario;
6. Developing a Node.js program to simulate indoor movement trajectories;
7. Implementing data analytics on Neo4j using Cypher Query language;
8. Analyzing experimental results;

More details are provided as follows. In this paper, two visualization tools have been used. The first one showing the real-time position of users on the map. I used the SensorUp dashboard, providing real-time visualization for live datastreams. As seen in Figure 2.8, the SensorUp dashboard is connected to the Amazon DynamoDB using AWS Identity and Access Management.

Thanks to SensorUp Inc. to let me use its product to visualize the real-time position of users. In more detail, creating datastreams on Amazon DynamoDB, reading them using the SensorUp dashboard, and also defining 3D indoor models in the dashboard is not done by this research and is done by SensorUp Inc. The second visualization tool is the Neo4j browser. As an example, Figure 2-12 showing the proposed hierarchical graph on the Neo4j browser. It is worth mentioning that I deployed components such as AWS Lambda functions on the Amazon cloud platform provided by the GeoSensorWeb Lab. I also need to mention the GeoSensorWeb lab provides BLE beacons, and I used them as a resource in this research. So, I also need to acknowledge the resources and help provided by GeoSensorWeb lab for this research.

1.4.2 Enhanced Air Quality Prediction by Edge-Based Spatiotemporal Data Preprocessing

1.4.2.1 Authors' Contributions

SensorUp Inc. provided the real-world dataset collected from 40 air quality IoT sensors deployed in Calgary, Canada. I collaborate with Dr. Giorgio Terracina and Dr. Francesco Cauteruccio from the University of Calabria, with Time Series Analysis and Machine Learning expertise. Dr. Giorgio Terracina and Dr. Francesco Cauteruccio implemented edge preprocessing algorithms on the Raspberry Pi 3 emulator as the edge components (Section 3.5.2). Dr. Giorgio Terracina contributed in writing Sections 3.4.1.4 and 3.4.1.5. Dr. Francesco Cauteruccio helped in writing the literature review (Section 3.2). I proposed the design for the preprocessing procedure, including spatial- and temporal-based estimation methods for missing values (Sections 3.4.1.1, 3.4.1.2, and 3.4.1.3). I also implemented the deep learning-based prediction method (Section 3.4.2) on the cloud component. I took the lead in writing the paper. All authors provided critical feedback and helped shape the research, analysis, and manuscript. Also, all authors discussed the results and contributed to the final findings.

1.4.2.2 Clarifications Regarding My Contributions

My contributions to this paper are:

1. Taking the lead in writing the paper;

2. Proposing preprocessing algorithms including temporal- and spatial-based estimation methods;

3. Implementing a python program for recognizing sensors and data aggregation;

4. Implementing the web API to apply an LSTM model for the air quality prediction;

5. Analyzing the experimental results;

More details are provided as follows. This paper proposed using the edge component as the place that preprocessing procedure can be carried on. One possibility is to carry out complex data preprocessing and cleaning tasks on the cloud. However, resorting to cloud-based methods presents some drawbacks. First of all, communication bottlenecks yielded by too much data transmitted from nodes to the central servers may induce packet losses and delays [31]. Moreover, in large sensor networks, simultaneously analyzing all sensor streams would introduce real-time computational constraints. The other possibility is carrying out data preprocessing and cleaning directly on the sensor equipment. This alternative can be either impossible due to the lack of computational power or unreliable due to the necessity of only analyzing local data. So, we decided to choose edge nodes as the place that preprocessing can be implemented.

This paper uses the Raspberry Pi 3 emulator instead of using real edge nodes for several reasons. Firstly, the original dataset for this study was collected from September 2017 to March 2018 by SensorUp Inc., and this paper is dated 2021. So, we were unable to test node edges in a real scenario. Secondly, the contribution of this paper is evaluating the impact of using the proposed preprocessing algorithms on air quality prediction. To this end, we first need to apply

the proposed preprocessing algorithms to estimate missing values. Second, a single prediction model is used to predict air quality using preprocessed and original datasets. Finally, the performance of the prediction model is evaluated to evaluate the importance of the proposed data preprocessing procedure. To this end, we first need to implement the proposed algorithm to estimate missing values. The estimation method can be implemented using various methods (*e.g.*, using a python program to run preprocessing algorithms and obtaining results or using real Raspberry Pi boards). In this paper, we have no contribution to proposing the best way to implement preprocessing algorithms. Testing the real-world implementation of edge nodes is out of our scope and needs further investigation. I proposed the required data preprocessing algorithms to estimate missing values in the original dataset. However, the implementation of edge-based estimation has been done by Dr. Giorgio Terracina and Dr. Francesco Cauteruccio as co-authors of this paper. Dr. Giorgio Terracina and Dr. Francesco Cauteruccio decided to use the Raspberry Pi 3 emulator to implement estimating algorithm. I need to clarify that I have no contribution to implementing the edge-based preprocessing algorithms, and the decision of using Raspberry Pi 3 emulator has been made by Dr. Giorgio Terracina and Dr. Francesco Cauteruccio. I also need to claim that the edge nodes do not exist in the real world, and edge preprocessing algorithms are tested on an emulator. Testing edge nodes on real scenarios are out of the paper scope. In future studies, I am planning to test the proposed algorithms on real edge nodes.

In more detail, I implemented a python program that is responsible for: 1) separating measurements for each sensor and 2) aggregating sensor measurements in the original dataset provided by SensorUp Inc. SensorUp Inc. collects this data, and they deployed 40 IoT air quality sensors in Calgary. As a result, I need to clarify that I have no contribution to data collection and real-world challenges that SensorUp has addressed in this regard. For the first task, the original

dataset provided by SensorUp includes the location information per each measurement. However, it does not include the sensor number. I used location information to differentiate various sensors using their location (latitude and longitude) in the developed program. For the second task, this program takes the original dataset provided by SensorUp and reduces data granularity from minutes to an hour using proposed data aggregation methods (explained in Section 3.4.1.1). The output of the program is the 40 Comma-Separated Values (CSV) files. Each file includes aggregated sensor measurements for a single sensor from September 2017 to March 2018. This process has been carried out offline, and results are fed into edge nodes (*i.e.*, Raspberry Pi 3 emulator). The implementation of k-medoid clustering, calculating DTW distances and estimating missing values is done on the Raspberry Pi 3 emulator by Dr. Giorgio Terracina and Dr. Francesco Cauteruccio.

The result of the edge component in CSV format is then used as the input of the prediction algorithm. For the prediction model, I developed a cloud-based standard LSTM model using Tensorflow (v1.14.0) and Keras (v2.3.1) deep learning libraries developed by FastAPI[1] (v0.16.2). Although other web frameworks such as Django[2] and Flask[3] could be used to implement the web API, I used FastAPI in this paper as a fast, open-source, and highly performant web framework. As a result, I have no contributions to evaluating the performance of different technologies used to develop the web API. The main goal of the cloud component is running the standard LSTM to evaluate the performance of the air quality prediction model. In other words, we implemented a simple web API running on a local computer system that can take the input file (*i.e.*, the output of the simulated edge component) in a CSV file format, run the LSTM prediction model, and finally

[1] https://fastapi.tiangolo.com/
[2] https://www.djangoproject.com/
[3] https://flask.palletsprojects.com/en/2.0.x/api/

show the evaluation result. Figure 1.2 shows different components that have been used in this research and their interactions.

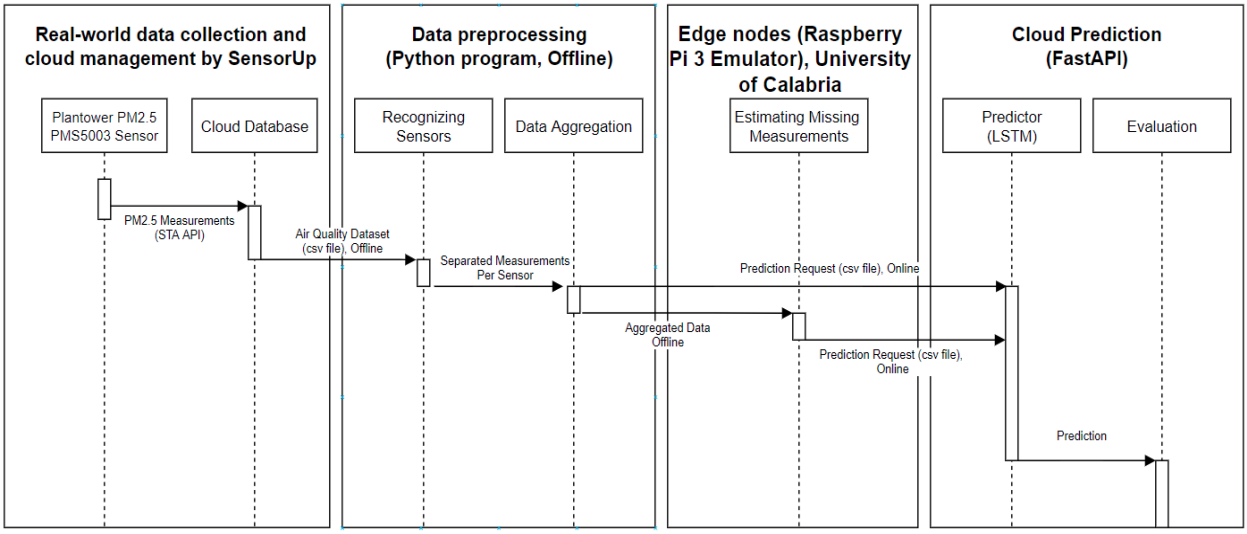


Figure 1.2. A representation of different components that have been used in this study and their interactions

2 A Person-to-Person and Person-to-Place COVID-19 Contact Tracing System Based on OGC IndoorGML

2.1 Abstract

With the wide availability of low-cost proximity sensors, a large body of research focuses on digital person-to-person contact tracing applications that use proximity sensors. In most contact tracing applications, the impact of SARS-CoV-2 spread through touching contaminated surfaces in enclosed places is overlooked. This study is focused on tracing human contact within indoor places using the open OGC IndoorGML standard. This paper proposes a graph-based data model that considers the semantics of indoor locations, time, and users' contexts in a hierarchical structure. The functionality of the proposed data model is evaluated for a COVID-19 contact tracing application with scalable system architecture. Indoor trajectory preprocessing is enabled by spatial topology to detect and remove semantically invalid real-world trajectory points. Results show that 91.18% percent of semantically invalid indoor trajectory data points are filtered out. Moreover, indoor trajectory data analysis is innovatively empowered by semantic user contexts (*e.g.*, disinfecting activities) extracted from user profiles. In an enhanced contact tracing scenario, considering the disinfecting activities and sequential order of visiting common places outperformed contact tracing results by filtering out unnecessary potential contacts by 44.98 percent. However, the average execution time of person-to-place contact tracing is increased by 58.3%.

Keywords: Trajectory Analysis, Graph-Based Data Model, OGC IndoorGML Standard, COVID-19 Contact Tracing

2.2 **Introduction**

High transmissibility of the novel severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) has made the COVID-19 disease a global health crisis [32, 33]. As we approach the end of 2020 and many pieces of research about SARS-CoV-2 have been conducted around the world, transmission ways of coronavirus are still under debate among scholars [34, 35]. According to recent studies [36, 37], three main transmission ways of coronavirus can be reported: 1) contact transmission that defines situations in which the infected person and someone else has direct contact or touch a common surface; 2) through virus-laden droplet transmission with a diameter larger than 5 μm (referred as respiratory droplets), and 3) through the airborne transmission of droplets with a diameter less than 5 μm (referred as droplet nuclei). Depending on the size of SARS-CoV-2-laden droplets, they can either rapidly fall out of the air in the immediate environment around the infected host (*i.e.*, cause contamination on surfaces close to the emission point) or remain suspended in the air and travel over tens of meters [38]. Van Doremalen et al. [39] evaluated the stability of SARS-CoV-2 on various surfaces under ten different experimental conditions and found that the SARS-CoV-2 can survive up to two days on surfaces. Therefore, the spread of COVID-19 can occur directly by being in direct contact with an infected person or indirectly through touching contaminated surfaces.

Amongst the different strategies used to decrease the infection rate of COVID-19, contact tracing is utilized as a public health practice [33, 40]. Using contact tracing, people who have had close contact with an infected carrier in the past 14-21 days (the incubation period for COVID-19) are identified as people who might be at significant risk of infection [41]. This practice can be conducted manually with health authorities by interviewing infected individuals [42]. However, manual contact tracing is a time- and effort-consuming task that requires experienced contact

tracers. So, the rapid viral spread of COVID-19 necessitates utilizing a scalable and digital approach for contact tracing [43]. An increase in the use of mobile technology, scalability of cloud data storage, and capability of physical devices to connect to the Internet using the Internet of Things (IoT) technology can meet the underlying requirements of applying digital contact tracing on a large scale [44].

Due to the interest of different governments in using digital contact tracing to address COVID-19, different smartphone contact tracing applications such as TraceTogether (Singapore) [45], CovidSafe (Australia) [46], and PACT (East-coast) [47] have been launched. A survey of recently introduced COVID-19 contact tracing applications can be found in [42]. Communication between nearby smartphones using their built-in Bluetooth interfaces is the most widely applied approach used by contact tracing applications [42]. Most contact tracing applications [42] only use proximity estimation and duration of contact between nearby smartphones. However, the contact tracing applications are insufficient for accurate contact tracing because they do not consider the impact of location and other contextual information [21].

The accuracy of contact tracing applications can be enhanced by considering the historical location context of users. The SARS-CoV-2 virus can be transmitted by a person touching a common surface that was previously touched by a diagnosed carrier [21]. In other words, when the common spaces were visited, the temporal sequence plays an essential role in enhancing the accuracy of contact tracing applications. The location history of users (provided by GPS) was considered in the SafePaths project in order to enhance the accuracy of contact tracing applications [21]. Similarly, He et al. [40] analyzed the location history of users in a COVID-19 contact tracing application for outdoor environments. However, indoor locations and the sequential order of visiting a common space have been overlooked [21, 40]. Several reasons underlie the importance

of indoor spatiotemporal trajectories for COVID-19 contact tracing applications: Enclosed indoor environments pose higher risks of community spread than outdoor environments [36, 38, 48], and people usually spend a larger part of their lives in indoor environments [49, 50]. Therefore, an indoor location-based contact tracing approach is required to model complex spatiotemporal topology for multiple floors and intrinsic connectivity [51, 52].

To the best of our knowledge, another missing gap in contact tracing applications is the inclusion of semantic contexts such as cleaning and disinfection activities. The disinfection and cleaning of commonly visited locations can effectively stop the chains of coronavirus spread [53] and should be considered by the contact tracing applications. An example scenario explaining the importance of disinfecting activities and the sequential order of visiting a place is given in Appendix A. To bridge the existing research gap, an enhanced digital contact tracing system is required to take the spatiotemporal movement trajectories of users and users' semantic contexts into consideration.

Indoor trajectory data analysis, such as contact tracing, requires a formal spatiotemporal data model as an abstraction of indoor movement trajectory [18]. The indoor raw trajectory is a temporal sequence of the time-stamped geographical coordinates of the moving object [54, 55]. Keeping a record of such precise location information requires more battery power, communication, and computation overheads [54]. Trajectory segmentations try to break down a raw trajectory into fragments that carry semantic meaning to address the challenges above. In this paper, raw movement trajectories are semantically partitioned into spatial fragments called stay points. Stay points are spatial objects which carry particular semantic meaning and contain all of the geographical coordinates located within the stay point, and the moving object remains in it for a length of time that is above a certain threshold [56].

For this study, a graph-based data model is proposed for encoding spatial and temporal dimensions as well as user contexts for indoor trajectories. Using a graph data structure, representing trajectory data allows movement trajectories to be stored in natural graph form using recent graph database technologies [56]. The first component of the proposed data model defines a spatial indoor space based on the OGC (Open Geospatial Consortium) IndoorGML standard [52]. The IndoorGML standard considers an indoor space as a set of non-overlapping cell spaces and models the topological relationships between cells using one or more Node-Relation Graphs (NRGs) [57]. The second component describes the time dimension as the duration of stay in each IndoorGML cell using a temporal hierarchy. The third component, called contextual dimension, shows user contextual information such as user job type, activity type, and how vulnerable they are in terms of being exposed to the SARS-CoV-2 virus. The proposed hierarchical graph-based data model allows the indoor movement trajectory to be represented at different levels of granularities. The different levels of granularities provided by the proposed model support the aggregation of semantic indoor trajectories over three dimensions: spatial, temporal, and contextual. For this research, the following contributions are discussed:

1. A spatiotemporal graph-based indoor trajectory data model is proposed to reflect hierarchies in space, time, and user's contextual information. The OGC IndoorGML standard is incorporated to enrich stay points with topological relations amongst indoor cells. The proposed model can store and analyze semantic indoor movement trajectories regardless of the indoor positioning system type.

2. A COVID-19 contact tracing system is developed and investigated using the proposed data model. In comparison to other contact tracing applications, to the best of our knowledge, this paper is the first research to implement and evaluate both types of SARS-CoV-2 transmissions, namely,

person-to-person and person-to-place. Also, the contact tracing application is further enhanced by including the place's disinfection history based on user's contextual information.

3. The spatial topologic relationships extracted from OGC IndoorGML in the proposed data model is used in a preprocessing technique to filter out semantically invalid trajectory points.

User privacy is a big challenge for contact tracing applications. Using historical location or proximity information can be considered as a threat to user privacy [21]. As mentioned in [58], 90 percent of people can be identified using only four trajectory points. Hence there is always a trade-off between user privacy versus the effectiveness of contact tracing applications. Although the application of privacy protection techniques is outside the scope of this paper, some considerations have been taken into account in order to protect user privacy, as discussed further in Section 2.6. Moreover, the scope of this research focuses on indoor movement trajectories by considering an active BLE beacon for each indoor cell. Although the topological relationships amongst indoor cells are used to filter out semantically invalid trajectory points, reconstructing missing trajectory points is outside the scope of this paper. Also, the automatic extraction of users' contexts such as cleaning activities and job type is outside the scope of this paper.

The rest of this paper is organized as follows: Firstly, the definition of the problems is defined in Section 2.3, followed by a discussion of related works in Section 2.4. The architecture of the proposed system is then described in Section 2.5. Section 2.6 provides details of the proposed data model, whilst the implementation results are illustrated in Section 2.7. Finally, Section 2.8 wraps up this paper with the conclusion, future work possibilities and ongoing problems.

2.3 **Problem Definition**

Consider the situation that the goal is conducting the contact tracing between users in indoor space using Bluetooth Low Energy (BLE). Unique advantages of BLE beacons like being lightweight, low cost, widely supported by smart devices, consuming less power, more flexible, and a higher Received Signal Strength Indicator (RSSI) has attracted many scholars to use them as a dominant indoor localization system [59, 60]. As seen in Figure 2.1(top-left), the topographic space of a physical indoor environment, including four cell spaces, is shown in Euclidian space. Let us consider a situation that a BLE beacon is placed in each indoor cell. Figure 2.1 (bottom-left) shows the signal coverage of each BLE beacon. Connectivity graph among indoor cells and BLE beacons in dual space is shown using IndoorGML NRGs in Figure 2.1 (right).

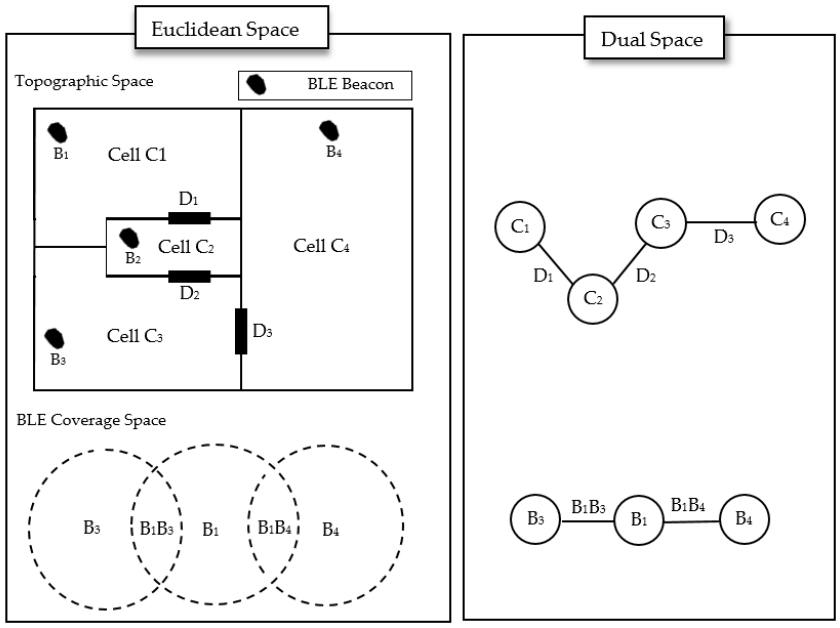


Figure 2.1. Representation of indoor cells and BLE beacon coverage in Euclidian and dual space

The first problem in an efficient indoor trajectory analysis is to model the users' trajectories in a semantic graph-based spatiotemporal data model. Figure 2.2 shows four users' semantic

indoor trajectories considering indoor cells as stay points in user movement trajectories. As an example, user u_1 entered and then exited the indoor cell c_1 in timestamp t_1 and t_3 respectively. In this example, a point will be recorded in the semantic indoor movement trajectory of u_1 as $P_1 = \langle c_1, \Delta_1 \rangle$. In which, Δ_1 is the length of time that u_1 has spent in indoor cell c_1. So, the semantic trajectory of u_1 can be modeled as $\{\langle c_1, \Delta_1 \rangle, \langle c_2, \Delta_2 \rangle, \langle c_3, \Delta_3 \rangle, \langle c_4, \Delta_4 \rangle\}$. The spatiotemporal representation of semantic indoor trajectories for all four users is shown in Figure 2.3.

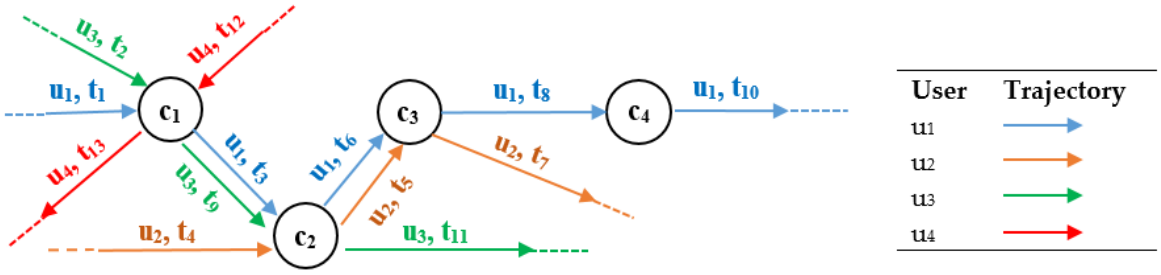


Figure 2.2. Representation of semantic movement trajectories of four users in the indoor environment

The next issue in trajectory modelling is to validate if the sequence of stay points is topologically connected. It is evident that missing or unstable RSSI data directly leads to noisy semantic movement trajectories. This issue is caused by delayed signal transmissions and interferences from walls and glass doors [20]. For example, as seen in Figure 2.1, consider the situation that user u_1 is located in cell c_1 and BLE beacon B_1 is placed in this cell. Consider the situation in which the connection between the BLE receiver (*e.g.*, smartphone) and B_1 is lost for a while. During this time, the BLE receiver will be connected to BLE beacon B_4, considering the signal coverage shown in Figure 2.1. After reconnecting to B_1, user semantic trajectory includes three stay points like $\{\langle c_1, \Delta_1 \rangle, \langle c_4, \Delta_2 \rangle, \langle c_1, \Delta_3 \rangle\}$. However, this movement trajectory is semantically invalid and $\langle c_4, \Delta_2 \rangle$ has made the movement trajectory noisy and semantically

invalid. This semantic trajectory is invalid because there is no connectivity directly between indoor cells c_1 and c_4. So, topological relationships among indoor cells (*i.e.*, NRGs) can be utilized as a noise filtering method to extract semantically valid indoor trajectories.

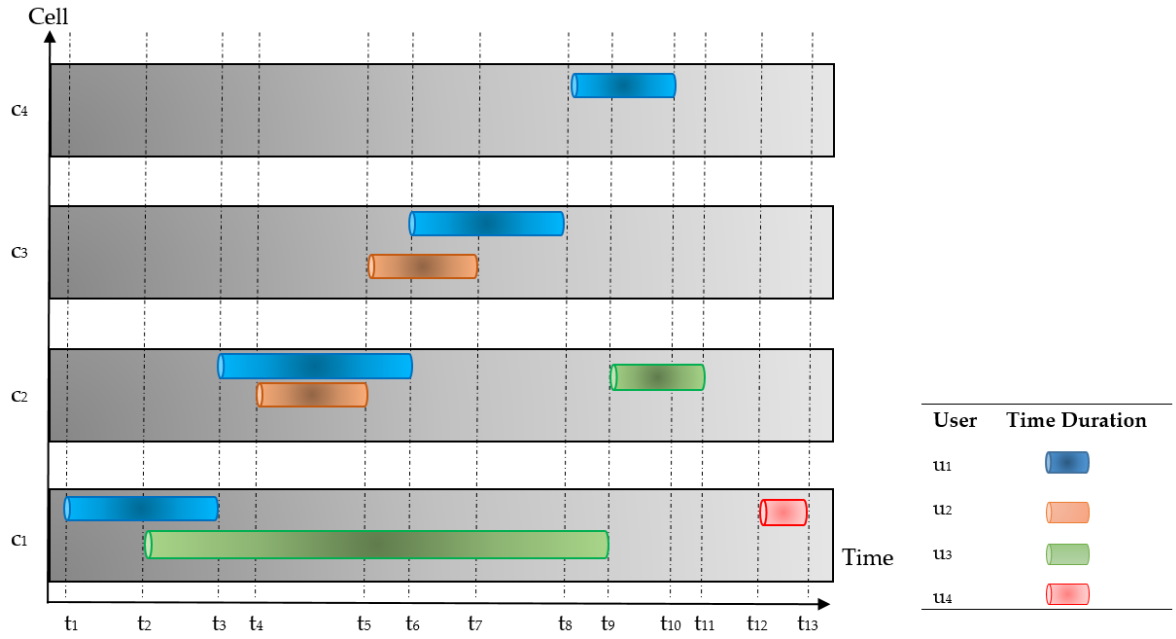


Figure 2.3. Spatiotemporal representation of semantic indoor movement trajectories of four users

The last but not the least challenge in our study is to evaluate contact tracing application as sequential trajectory analysis considering users' contexts (*e.g.*, disinfecting activity). Let us assume user u_1 as an infected person with COVID-19 and u_3 as cleaning staff. In this scenario, different spatiotemporal queries might be of interest to contact tracers. For example, a list of users who visited indoor cells just after those cells being visited by the infected user (*i.e.*, user u_1) and before they have been cleaned by the cleaning staff (*i.e.*, user u_3). As seen in Figure 2.3, all four indoor cells are visited by the infected user. However, only user u_2 has been in contact with the infected user u_1 in indoor cell c_2 before this room being cleaned.

2.4 **Literature Review**

In this section, we categorized the state-of-the-art related to our study into four categories: Current COVID-19 contact tracing applications are briefly reviewed in the first category. Next, existing trajectory segmentation approaches are studied as they require less computation power, communication cost, and are more human-readable. The third category focuses on trajectory representation methods. The fourth category looks over existing data models for indoor environments. Finally, the major differences between this study and other related studies will be summarized at the end of this Section.

2.4.1 COVID-19 Contact Tracing Applications

The exponential increase in the number of people with the coronavirus has motivated many governments and developers to leverage technology to curb the spread of COVID-19 [61, 62]. As the world continues to fight against the COVID-19 pandemic, retracing close contacts of a COVID-19 Confirmed Person (CCP) to find and notify possibly exposed people at the earliest possible stage (*i.e.,* contact tracing) is widely accepted as an available approach to "flatten the curve" [63, 64]. In practice, the laborious and slow process of manual contact tracing (*i.e.,* interview-based contact tracing) and rapid transmission of coronavirus necessitate utilizing scalable digital contact tracing systems [42, 63]. Researchers and developers have proposed different digital contact tracing systems to ease the burden of manual contact tracing on public health departments. To the best of our knowledge, existing contact tracing systems can be categorized into web-based and mobile contact tracing systems. For example, both web-based (*i.e.,* TeamSense [11]) and mobile (*e.g.,* COVID Alert [12] and ABTraceTogether [13]) contact tracing systems are used in Canada to combat the COVID-19 pandemic. In another attempt, BLE-based

wearables are recently proposed by Estimote, Inc. [65], which can be used in a web-based contact tracing system for workplaces. TraceTogether [66] is implemented as a digital contact tracing system by the Singaporean government to help automate the laborious task of manual contact tracing. According to SensorTower [67], this application with more than 3.2 million downloads (*i.e.,* 55.36 percent of the total population in Singapore [68]) from the App Store and Google Play [69] is ranked first among all free applications in this country. As another example, Corona-Warn-App [70] is used as a digital contact tracing application in Germany. This application has received over 22.8 million downloads (*i.e.*, 27.17 percent of the total population in Germany [68]) and ranked first among all free mobile applications in this country [71].

There are currently more than 50 smartphone contact tracing applications utilized in more than 30 countries around the world [72]. Different technologies like BLE, GNSS, RFID, Wi-Fi, and QR codes have been introduced for contact tracing applications. BLE and GNSS technologies can be considered as leading technologies in COVID-19 contact tracing applications [21, 72]. Focusing on coronavirus transmission, obstructions like walls between users can stop virus transmission. Comparing GNSS and BLE technologies, reduced signal strength in BLE technology can represent existing obstructions between users, while GNSS cannot consider (Appendix B). Also, BLE technology is more accurate than GNSS for proximity detection in enclosed places like indoor buildings and underground transit [21]. TraceTogether [66] can be mentioned as the first digital contact tracing application in the world using the BLE proximity technology. Most of the existing contact tracing applications, such as CoEpi [73] and Covid Watch [74], tried to estimate proximity between individuals using BLE proximity technology. A survey of recently introduced COVID-19 contact tracing applications can be found in [42]. Estimating proximity between users can be considered as the heart of contact tracing applications [72]. However, relying only on

proximity among users will not be sufficient because coronavirus can be transmitted by touching a common surface such as a table, keyboard, and door handle. In other words, the person-to-place way of transmitting coronavirus needs to be considered as location-based COVID-19 contact tracing applications.

Berke et al. [21] tried to investigate the location-based COVID-19 by incorporating the location history of users in SafePaths project. In their contact tracing application, the user historical location can be collected by either GPS or BLE technologies. In more detail, GPS 2-dimensions (latitude and longitude) coordinates and time are first mapped to a 3-dimensional geospatial grid. Then, time intervals when two users occupied the same place will be detected using the intersection approach across users' GPS histories. Although authors suggested that their method can be used in the person-to-place way of coronavirus transmission, there is no evidence that the impact of person-to-place virus transmission is considered to the best of our knowledge. To clarify this gap, consider the situation that a common place is visited firstly by user A who is a diagnosed carrier, and after this user left the common place, it is visited by user B. Consider this situation has happened in a short time interval. Intersecting the location at the common time window results in finding user B as a possibly exposed user. So, the sequential order of visiting a common place should be considered in addition to user historical location to model the person-to-place way of transmitting coronavirus among users.

He et al. [40] tried to incorporate users' historical location in COVID-19 contact tracing as a travel companion trajectory mining application for outdoor environments. Authors followed the methodology proposed by Rong et al. [75] to design an efficient index, called the Spatial First Time index, as a similarity metric for trajectory clustering. Their proposed similarity metric was applied to group trajectory segments that are spatially and temporally similar to the query

trajectory (*i.e.*, the trajectory of COVID-19 confirmed case) [40]. Using trajectory clustering, users who have similar trajectories to the query trajectory can be detected. Although they measured users' companionship, they did not study if a location is contaminated with a diagnosed carrier. Also, their proposed system only focused on outdoor environments.

Although researchers and developers have been mostly focused on technological advancements of digital contact tracing systems, broad public participation in digital contact tracing applications is required to halt the pandemic [61]. Hellewell et al. [64] evaluated the impact of isolation and contact tracing to control the pandemic using a mathematical model. According to their results, the majority of outbreaks in regions with the basic reproduction number[1] (R0) less than 1.5 can be controllable if 50% or more of contacts are successfully traced [64]. For example, the estimated basic reproduction number in Singapore is (0.8 - 1.4) [76] and in Germany is (0.9 - 1.3) [77] with a 90% credible interval. According to SensorTower [67], TraceTogether and Corona-Warn-App contact tracing applications are publicly adopted by 55.36 and 27.17 percent of Singapore and Germany, respectively. Although the TraceTogether met the expected adoption (*i.e.*, 50% of the country's population), the Corona-Warn-App lagged behind the expected adoption. Various factors, such as privacy concerns, anonymity, transparency, and concerns about data overusing, affect the public willingness to use digital contact tracing applications [61]. Recent works [61, 63] suggest that automatic contact tracing at any public adoption rate will slow the quick spread of coronavirus. Although a qualitative analysis of the effectiveness and success of digital contact tracing applications is required, we believe that digital contact tracing applications cannot be considered a replacement for manual contact tracing.

[1] The expected number of secondary cases that can be infected by the infected host in a population where all individuals are susceptible

To conclude, user location history and sequential order of visiting a place can better model person-to-person way of coronavirus transmission. However, person-to-place virus transmission is still overlooked at the time of writing this paper. Moreover, user location history has only been considered for outdoor environments. As of December 16, 2020, the disinfection history of commonly visited places is ignored by researchers for both indoor and outdoor environments.

2.4.2 Trajectory Segmentation

The main goal of trajectory segmentation is breaking down a trajectory into fragments that carry semantic meaning, more human-readable, and require less computation power [54]. There are mainly three categories of approaches for trajectory segmentation [54, 55]. The first category takes time interval into account to divide a trajectory into fragments [78]. The second category focuses on trajectory shape and breaks down a trajectory using turning points that maintain the trajectory shape [79]. Finally, partitioning trajectories using each segment's semantic meaning, which is used in our proposed data model, can be considered the most widely applied approach of trajectory segmentation. This approach has been vastly applied in different applications such as transportation [80, 81], tourism [82, 83], and recommender systems [84].

A very natural way of semantic meaning-based trajectory segmentation is splitting a trajectory into segments that show stillness versus movement [54, 55]. Depending on trajectory data analysis, stationary points can either be kept or skipped. For example, Yuan et al. [85] skipped the stationary points in their proposed taxi travel speed estimation approach. However, He et al. [40] have only focused on stationary points in their proposed trajectory clustering approach for COVID-19 contact tracing application. Gómez et al. [56] similarly used the concept of stationary points and assumed users' check-ins extracted from location-based social networks as stationary points that have no duration.

Although there is a large body of knowledge for trajectory segmentation in outdoor applications like transportation, only a few research pieces have been focused on indoor settings. Werner et al. [86] reduced the amount of computation using the Douglas-Peucker algorithm as a trajectory segmentation approach, which is mainly concern with the trajectory shape. Gua et al. [87] improved the accuracy of the map matching process in indoor trajectories considering semantics provided by pedestrian dead reckoning and human activity recognition algorithms. Wang et al. [88] used raw odometer data to extract step length, step count, and heading for the purpose of trajectory segmentation. In our research, the raw indoor trajectory is segmented by the definition of stay points and semantically enriched by OGC IndoorGML cell attributes.

2.4.3 Indoor Trajectory Model

Chen et al. [89] have used the grid partition method to abstract the indoor physical model using hexagonal grids. The IFC data model has been used in this research, and movement trajectories have been generated using the tool Vita [89, 90]. They applied a vertical projection distance approach to transform synthetically generated indoor movement data into determined grids. The OGC has published the IndoorGML standard as a common spatial framework to represent and model indoor spaces for indoor navigation purposes [52]. OGC IndoorGML open standard provides a standard way of abstracting indoor physical environments using a multi-layered graph-based data model. The flexibility of OGC IndoorGML standard in providing spatial units with the ability to have their own semantic and topological relationships in a graph-based data structure makes OGC IndoorGML standard an appropriate alternative to model spatial dimension of indoor trajectories. Alattas et al. [50] analyzed and visualized indoor users' movement data in an evacuation exercise using the extended LADM-IndoorGML. They extracted the location of users by analyzing WiFi logs data collected from the main WiFi network of TU

Delft. LADM-IndoorGML is an integration of OGC IndoorGML and Land Administration Domain Model to determine restrictions, right, and responsibilities of different group of users for each interior space [91]. Alattas et al. [50] used a relational database (*i.e.*, PostgreSQL) to model the 3D geometry of IFC model and technical model of LADM-IndoorGML as an infrastructure to visualize users' movements. Following individual users and monitoring the number of users in each WiFi access point zone were two types of analyzes have been done by Alattas et al. [50].

Kontarinis et al. [18] proposed a hierarchical semantic-enabled symbolic model for indoor movement trajectories of users visited the Louvre Museum located in Paris, France. In [18], the physical model of the Louvre Museum has been abstracted using OGC IndoorGML standard into five symbolic graph layers (*i.e.*, building complex, building, floor, room, and region of interest). Regions of interests have been defined as 51 non-overlapping zones specified by the museum administration and associated with exhibition themes. Raw movement trajectories of visitors have been collected using the BLE beacon infrastructure and smartphone application (app) and then transformed and enriched by considering five symbolic graph layers. So, authors in [18] modelled visitors' indoor movement trajectories as a sequence of their presence in Louver's thematic zones. In [18], the OGC IndoorGML standard has been used to semantically enrich raw trajectories. However, some significant differences between our study and their study are: 1) their proposed data model is focused on museum visitors, while we proposed a general spatiotemporal graph-based model which can be used in COVID-19 contact tracing application; 2) although spatial granularity has been supported in the proposed model by Kontarinis et al. [18], the user contextual information has been overlooked; 3) temporal and contextual granularity will be supported in our proposed model to fully represent the spatiotemporal nature of users' movement trajectories; 4)

our proposed data model is implemented in a graph-based database instead of using a thematic representation of indoor trajectories.

2.4.4 Trajectory Representation

There are three ways to represent movement trajectories using a matrix, tensor, and graph data structure [54]. The matrix representation of movement trajectories is widely applied in recommendation systems. Ojagh et al. [92, 93] transformed GPS trajectories of users into a matrix and then applied a collaborative filtering algorithm to provide users with personalized recommendations. Tensor representation of movement trajectories can be considered a natural extension of matrix-based transformation, with additional information as the third dimension of matrix representation [54]. Zheng et al. [94] extended the location-activity recommendation system to a user personalized recommendation system by adding users to the location-activity tensor representation of users' GPS trajectories.

Using a graph data structure, representing trajectory data allows movement trajectories to be stored in natural graph form using recent graph database technologies. Therefore, graph-based trajectory representation prevents the "impedance mismatch" problem between the data model and storage [56]. Recent advances in graph database technologies increase using a graph data structure in various applications [56, 95, 96]. Hu et al. [97] extracted tourist movement trajectories from users' geo-location data shared on Twitter as social media. In their research, extracted trajectories then turned into graphs using DBSCAN-based clustering. Then, network analytical methods have been applied to extracted graphs to discover tourist movement patterns. Niu et al. [98] constructed a dual graph from movement trajectories considering the transportation network as a complex network. The dual graph has then been utilized in a label-based clustering approach to cluster movement trajectories and addresses the main limitation of distance-based trajectory clustering

methods. Sabarish et al. [99] proposed a hierarchical clustering method based on the graph data structure to identify similar movement patterns of movement trajectories for moving trucks carrying goods. Gómez et al. [56] proposed a spatiotemporal graph data structure and transformed users' movement trajectories from the location-based social network to perform Online Analytical Processing operations on movement trajectories. Guo et al. [87] enhanced the indoor pedestrian trajectory's accuracy using the concept of semantically enriched graph data model extracted from the floor plan.

To conclude related studies, person-to-place location history, the sequential order of visiting a common place, user-related contextual information such as disinfection activities are missing in digital contact tracing applications. Also, the state-of-the-art only focused on outdoor environments. To address aforementioned research gaps, a graph-based data model is proposed to encode spatial and temporal dimensions as well as user’s contexts for indoor trajectories. The first component of the proposed data model defines a hierarchical spatial indoor space. In this model, the raw indoor trajectories using BLE sensors are segmented by the definition of stay points and semantically enriched by OGC IndoorGML cell attributes. The benefit of using OGC IndoorGML is its graph-based structure to model the spatial topology. So, it can be adopted in our proposed data model to represent and store the spatial dimension of movement trajectories in their native graph form. The second component describes the time dimension as the entrance time for each IndoorGML cell using a temporal hierarchy. The third component, called contextual dimension, shows users’ contextual information such as disinfecting activities. We used a graph database to represent the proposed spatiotemporal indoor trajectory model as an efficient approach for contact tracing query processing.

2.5 **Methodology**

In this section, the methodology and preliminary definitions applied for this research are explained. The proposed graph-based hierarchical data model for semantic indoor movement trajectory data analysis is then described. This section is focused on COVID-19 contact tracing. However, the proposed data model can also be used as a general purpose indoor movement trajectory.

2.5.1 Semantic Trajectory Segmentation

Considering the situation in which four BLE beacons are deployed in an indoor environment, the raw indoor movement trajectory of u_1 is shown in Table 2.1: b_i^j indicates the RSSI value measured by the user smartphone for BLE beacon B_i for timestamp t_j. As shown in this table, u_1 smartphone received four RSSI measurements from four BLE beacons (*i.e.*, $B_1, \dots, B_4$) located near it.

Table 2.1. Raw indoor movement trajectory of a sample user

User ID	**Time**	$\mathbf{B_1}$	$\mathbf{B_2}$	$\mathbf{B_3}$	$\mathbf{B_4}$
1	t_0	b_1^0	b_2^0	b_3^0	b_4^0
1	t_1	b_1^1	b_2^1	b_3^1	b_4^1
1	t_2	b_1^2	b_2^2	b_3^2	b_4^2
1	t_3	b_1^3	b_2^3	b_3^3	b_4^3
1	t_4	b_1^4	b_2^4	b_3^4	b_4^4
…	…	…	…	…	…
1	t_j	b_1^j	b_2^j	b_3^j	b_4^j

Table 2.1 shows the indoor movement trajectories for a single user and is called Raw Movement Trajectory (RMT). Given the RMT, the notation of raw indoor movement trajectory can be defined as follows:

Definition 1 (Raw Indoor Trajectory): An indoor raw movement trajectory (*i.e.*, RMT) for the user u_i is a temporal sequence of RSSI values from all of the visible BLE beacons deployed by an indoor positioning system. The measured RSSI values from visible BLE beacons depend on the user geospatial locations. Thus the raw indoor trajectory of a user can be formulized as:

$$RMT = [(u_i, t_0, \{b_i^0 \mid i \subseteq N_0\}), (u_i, t_1, \{b_i^1 \mid i \subseteq N_1\}), \dots, (u_i, t_j, \{b_i^j \mid i \subseteq N_j\})] \quad (1)$$

in which, $N_j \subseteq N$ denotes the number of visible BLE beacons close to the user smartphone at each timestamp j, and N is a set of all BLE beacons ($\{B_1, B_2, \dots, B_N\}$) deployed in an indoor environment. The set $\{b_i^j \mid i \subseteq N_j\}$ represents a set of RSSI measurements (b_i^j) from visible BLE beacons at timestamp j. The order of $t_0 < t_1 < \cdots < t_j$ is considered a natural order in the time frame for geospatial points visited by the user in the RMT.

As seen in Table 2.1, the RMT contains a huge amount of data which makes the trajectory data analysis time-consuming. Also, contact tracers might not be interested in such detailed information regarding the RSSI value for all visible beacons for each timestamp. The proximity zone that the user is located in, and also the amount of time that he/she stayed there might be of greater interest for contact tracers. Hence the raw indoor movement trajectories can be segmented using the semantic concept of "Place of Stay (PoS)". The PoS semantically refers to the proximity zone of the BLE beacon with the highest RSSI value in which the user has spent Δ_k length of time. The notation of PoS is defined as follows:

Definition 2 (Place of Stay): Consider δ_{min} and δ_{max} as the minimum and maximum RSSI values associated with the proximity zone of BLE beacon B_i with the highest RSSI value. All geospatial points with their measured RSSI value belonging to the $[\delta_{min}, \delta_{max}]$ range are defined as the PoS associated to BLE beacon B_i. The length of time (*i.e.,* Δ_k) that the user spent at the PoS is the time difference between the entrance and exit time of the user in this proximity zone.

Also, user contextual information can be considered in semantic indoor trajectories. In the COVID-19 contact tracing application, we denoted job type, activity type, and the level of user vulnerability to COVID-19 virus exposure as user's contexts in each PoS. User contextual information can be manually entered by the user. Table 2.2 illustrates the semantic trajectory segmentation of user u_1.

Table 2.2. Semantic indoor movement trajectory of a user

User ID	Time	Time Duration	Place of Stay	Health Status	Activity Type	Vulnerability
1	t_0	Δ_1	B_1	Healthy	Cleaning	v_1
1	t_1	Δ_2	B_2		Visiting	
1	t_2	Δ_3	B_3		Visiting	
1	t_3	Δ_4	B_4		Visiting	

The notation of semantic indoor movement trajectory is defined as follows:

Definition 3 (Semantic Indoor Trajectory): The semantic indoor movement trajectory (*i.e.,* SMT) is a temporal sequence of visited PoSs including user ID, the time that the user enters the PoS, duration of time stayed in the PoS, and user contextual information in each PoS. The notation of semantic movement trajectory of user u_1 (*i.e.,* SMT_1) according to Table 2.2 is defined as follows:

$$SMT_1 = [(u_1, t_0, \Delta_1, B_1, Healthy, Cleaning, v_1), \dots, (u_1, t_3, \Delta_4, B_4, Healthy, Visiting, v_1)] \quad (2)$$

In which, $B_i \in B_{Dom}$ and B_{Dom} (BLE beacon with the highest RSSI value) is an indicator of the visible set of BLE beacons (*i.e.*, $\{b_i^j \mid i \subseteq N_j\}$) related to the PoS proximity zone.

2.5.2 Multilayered Spatial IndoorGML-Based Data Model

To support interoperability between indoor location-based services, OGC published a XML-based exchange standard indoor data model called IndoorGML in 2014 [100]. The cellular space can be considered the underlying concept of IndoorGML in order to provide an abstraction of the physical indoor environment [101].

Definition 4 (Cellular Space): The cell c is defined as the basic unit type of the indoor primal space of the IndoorGML spatial data model. The cellular space $\mathbb{C}$ is considered the union of non-overlapping cells $c_i \in \mathbb{C}$ which is an abstraction for the given physical indoor space $\mathcal{P}$ [101].

To abstract the indoor physical space using IndoorGML, both geometrical and topological properties need to be defined within the cellular space. Geometrical properties which define spatial extent for cells and their boundaries can facilitate the computation for indoor distance [100]. However, geometrical properties are not necessarily required for many applications (*e.g.*, contact tracing). Using the Poincaré duality thorium three-dimensional cells and their relationships, topographic indoor spaces can be transformed into corresponding dual spaces. In the dual spaces, a topological, or an equivalent adjacency graph, visualizes indoor cells and their adjacency relationships using nodes and edges. Edges in the adjacency graph can be generally classified into navigable (*e.g.*, doors) and non-navigable (*e.g.*, walls) links. As shown in Figure 2.4, a connectivity

graph can be extracted by considering only the navigable links in the adjacency graph. As the connectivity graph illustrated in Figure 2.4 does not represent the geometrical properties of cells, it is called a logical connectivity graph [100].

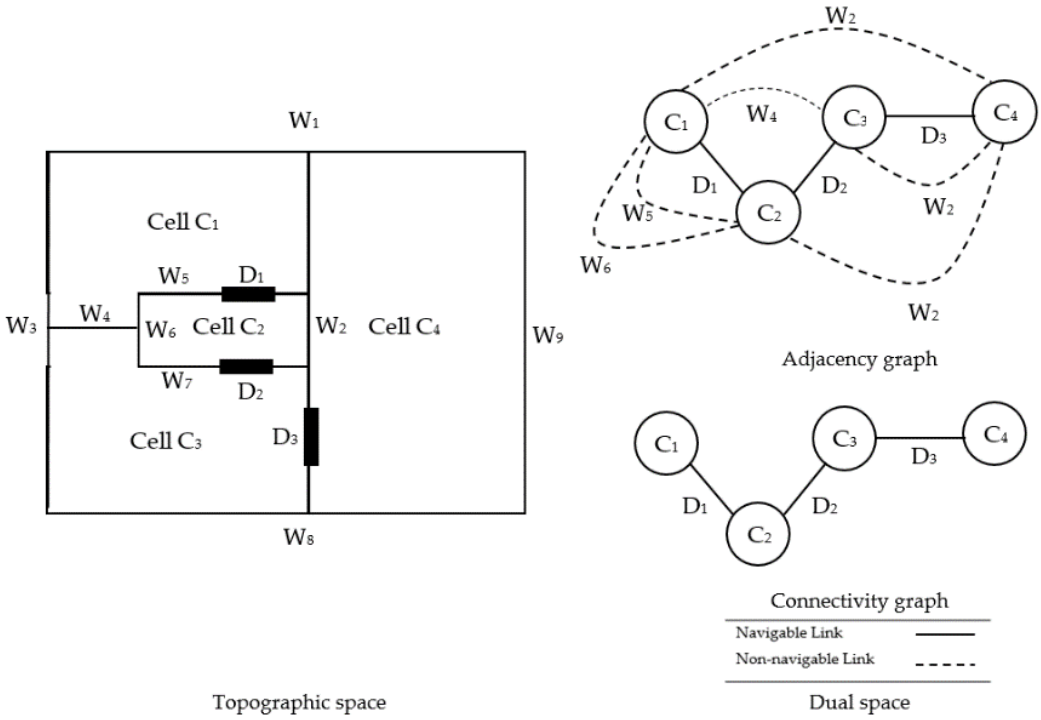


Figure 2.4. Extracting adjacency and connectivity graph from two-dimensional topologic indoor space

IndoorGML offers a mechanism called the multi-layered space model that supports different interpretations for indoor spaces [100, 101]. For example, BLE signal coverage can be defined as a new interpretation for the topological space as shown in Figure 2.5. Each interpretation has its own cellular space. Geometrical and topological properties can be defined for the corresponding cellular spaces. Also, inter-layer types of connections can be used to visualize relationships amongst different cellular spaces. The multi-layered space will be defined as an overlay of interpretations. An example of this is when eight BLE beacons were installed in the topological space as shown in Figure 2.5. As illustrated in Figure 2.5, the BLE signal coverage in

dual spaces can be considered for the extraction of a logical connectivity graph for BLE beacons. The multi-layered space model that includes inter-layer relationships amongst different interpretations is shown in Figure 2.6. Following the same concept, IndoorGML offers a hierarchical graph that covers spatial granularity in cellular spaces [100, 101]. An example would be the situation in which the indoor space illustrated in Figure 2.4 belongs to building *b*. As shown in Figure 2.6, building *b* will be defined as the highest layer of the hierarchical structure offered by the IndoorGML.

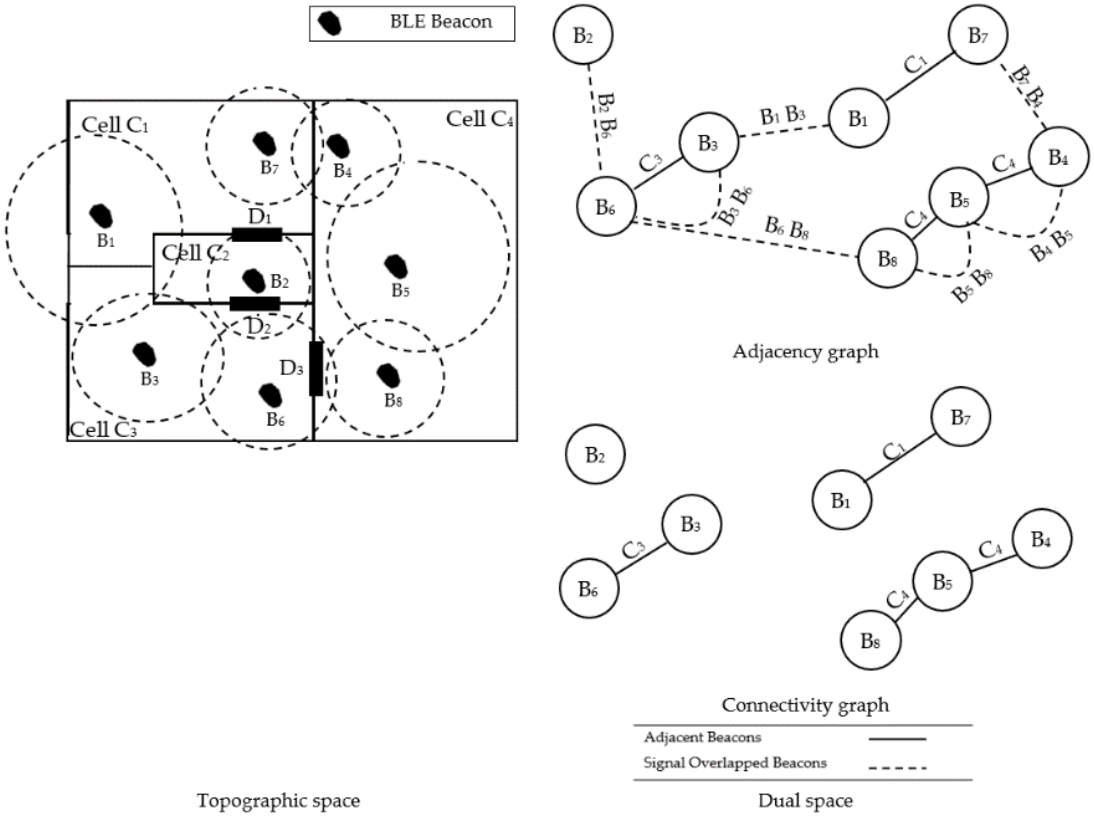


Figure 2.5. Extracting adjacency and connectivity graph for BLE beacons from two-dimensional topologic indoor space

2.5.3 Proposed Graph-Based Indoor Trajectory Modelling

As seen in (2), three main components have been used to define the semantic indoor trajectory. The first component determines the spatial space using the concept of PoS. The second component describes temporal space using a pair of coordinates similar to (t_j, Δ_k) in which t_j indicates the time the user entered the PoS and Δ_k indicates the length of time the user spent in the PoS. The third component, called the contextual dimension, shows user contextual information such as their job type, activity type, and how vulnerable they are in terms of exposure to the SARS-CoV-2 virus.

Three granularity levels will also be defined in order to support aggregation in semantic indoor trajectories: temporal, contextual, and spatial. The temporal hierarchical structure of t_j is defined as $Instant \rightarrow Minute \rightarrow Hour \rightarrow Day \rightarrow Month \rightarrow Year$ in the proposed data model. In addition to temporal spaces, hierarchical structure is also considered for contextual information. Vulnerability is considered as part of the incorporated contextual information for this research. A diverse range of factors influences COVID-19 vulnerability (*e.g.*, age, background medical conditions) [102]. However, for the sake of simplicity, mobile users are only asked to report how vulnerable they think they are in terms of being exposed to the SARS-CoV-2 virus. Vulnerability levels can be entered as an integer value between zero and ten. This integer can then be categorized into high-level and low-level.

The spatial hierarchy of the proposed data model is based on the OGC IndoorGML multi-layered spatial representation. The spatial granularity of the proposed data model is shown in Figure 2.6. The proposed indoor spatial hierarchical structure is defined as $BLE\ Zone \longrightarrow Interior\ Cell \rightarrow Category \rightarrow Floor \rightarrow Building$.

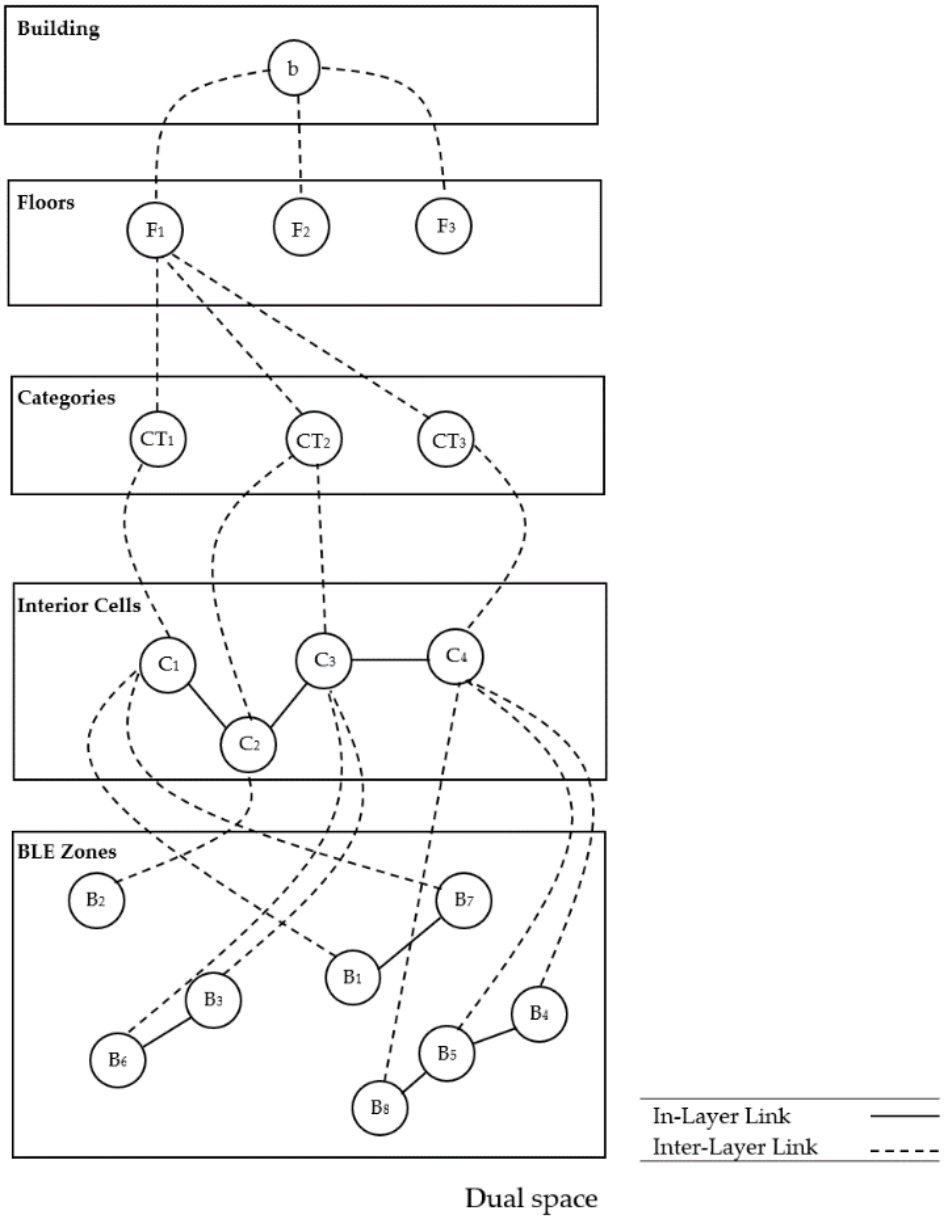


Figure 2.6. An example of hierarchical structure considered for spatial granularity

Spatial hierarchical layers shown in Figure 2.6 from bottom to top are defined as follow:

- The BLE zones layer (as shown in Figure 2.5) is investigated as the finest level of the spatial hierarchy because multiple BLE zones can cover an interior cell;

- The Interior Cells layer divides the indoor space into individual spatial units divided by walls. Interior cells may have different semantic categories: Laboratory, meeting room, personal office, washroom, corridor, stair, elevator, and kitchen;
- The Category layer is defined as a group of interior cells with a similar type of semantic category. For example, the laboratory category of interior cells can be assigned to either chemical or mechanical laboratories;
- The Floors layer is considered a higher spatial granularity of categories. For example, the third floor includes the library, office, and chemical laboratory categories;
- The Building layer is comprised of the various floors associated with it.

An example of modelling semantic indoor trajectory that considers different hierarchical structures (*i.e.*, temporal, contextual, and spatial) is shown in Figure 2.7. In Figure 2.7, circles with dashed blue lines show the trajectory data points in SMT. For example, the central blue dashed point shows the SMT of a user with ID#1 who is a diagnosed carrier. The user has an entrance time of 01: 12: 23Z to the proximity zone of BLE beacon B2 for the duration of 1,177 seconds as shown:$[u_1, 2020-07-25T01{:}12{:}23Z, 1177, B2,\ COVID19, visiting, 8]$ From the proposed data model, we can infer that the user was located at cell ID#123 which belongs to the library category on the third floor of building b1.

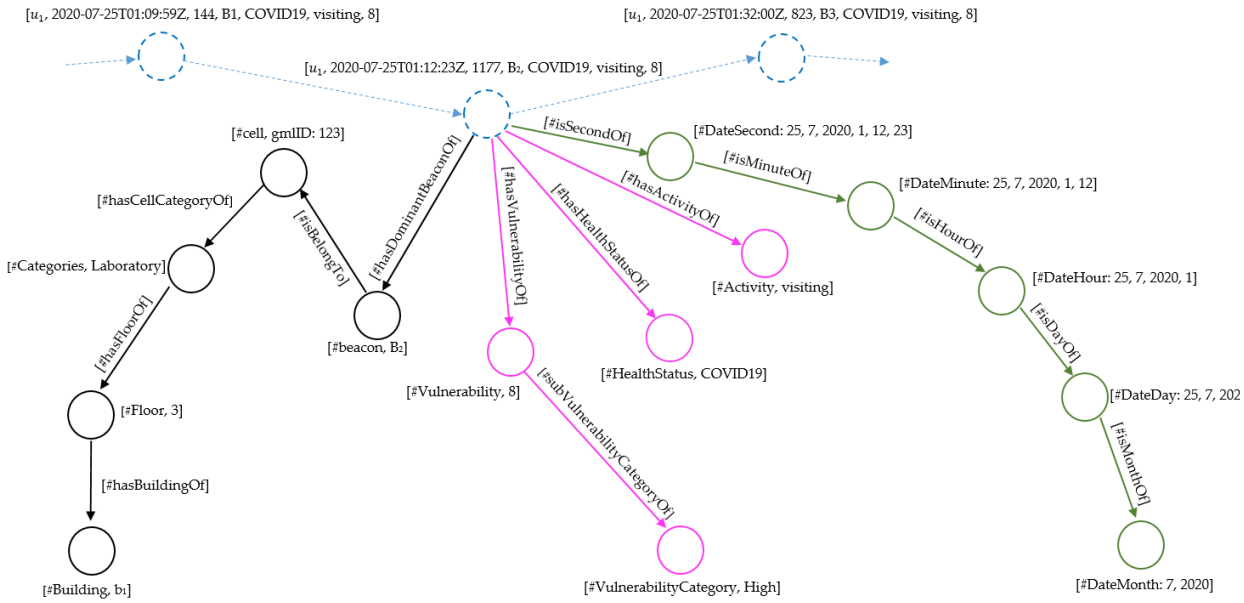


Figure 2.7. An example of the proposed graph-based indoor trajectory modelling

2.6 **System Architecture**

To evaluate the proposed data model in contact tracing application, cloud-based architecture is developed. The architecture includes three layers (Figure 2.8): "Data Collection", "Cloud Data Storage and Management", and "Visualization". We have adopted "cloud Data Storage" and "Visualization" layers from SensorUp[1] architecture. The data collection layer (*i.e.*, a smartphone Android app) is mainly responsible for measuring RSSI values from different visible BLE beacons. The user raw indoor movement trajectories are then segmented in the smartphone app to reduce the communication cost. Smartphone app provides users with both "offline" on device and "online" cloud storage. The user can choose to be in an "online" or "offline" mode. When a user is entered in the proximity zone of a BLE beacon in online mode, the smartphone app

[1] https://sensorup.com/

will start counting seconds. When the user leaves the dominant BLE beacon's proximity zone, a record will be sent to the cloud layer. The record consists of different types of information, including the dominant BLE beacon information (*i.e.,* dominant BLE beacon Identification (ID) and average RSSI), temporal data (the date and time of the user entered and exited the proximity zone and the length of time that user has stayed there), and user-related contexts (*i.e.,* user's health status, activity type, and vulnerability level as entered by the user). In "offline" mode, such a similar record is stored internally in the smartphone's database (*i.e.,* SQLite database). In "offline" mode, whenever the users willing to share their movement trajectory, a bulk data transfer mechanism is applied to share all stored trajectory points with the cloud server.

The second layer is the cloud data storage and management layer developed using Amazon Web Services (AWS). This layer is mainly responsible for handling users' and end-users' identity and access management and data storage. As shown in Figure 2.8, Amazon AWS Cognito[1] is used as a fully managed service for authentication and authorization of smartphone apps. AWS Cognito supports identity and access management using Cognito User Pool and Cognito Identity Pool as standard authentication and authorization services. Amazon AWS Lambda[2] is applied in this layer as a server-less and event-driven computing service. As an instance, an AWS Lambda is triggered to enrich the indoor trajectory as soon as a record from an authenticated smartphone app user is received. A data record is taken as an array of trajectory data point records in JavaScript Object Notation (JSON) format in this enrichment process. For each trajectory point, a mapping function will be applied to map the captured BeaconID to a unique OGC IndoorGML cell in the proposed data model. Then, the output is published to AWS IoT as a standard GeoJSON record. Amazon

[1] https://aws.amazon.com/cognito/
[2] https://aws.amazon.com/lambda/

AWS IoT[1] service is used in this research as a managed cloud service to support cloud data sharing for billions of devices and trillions of messages.

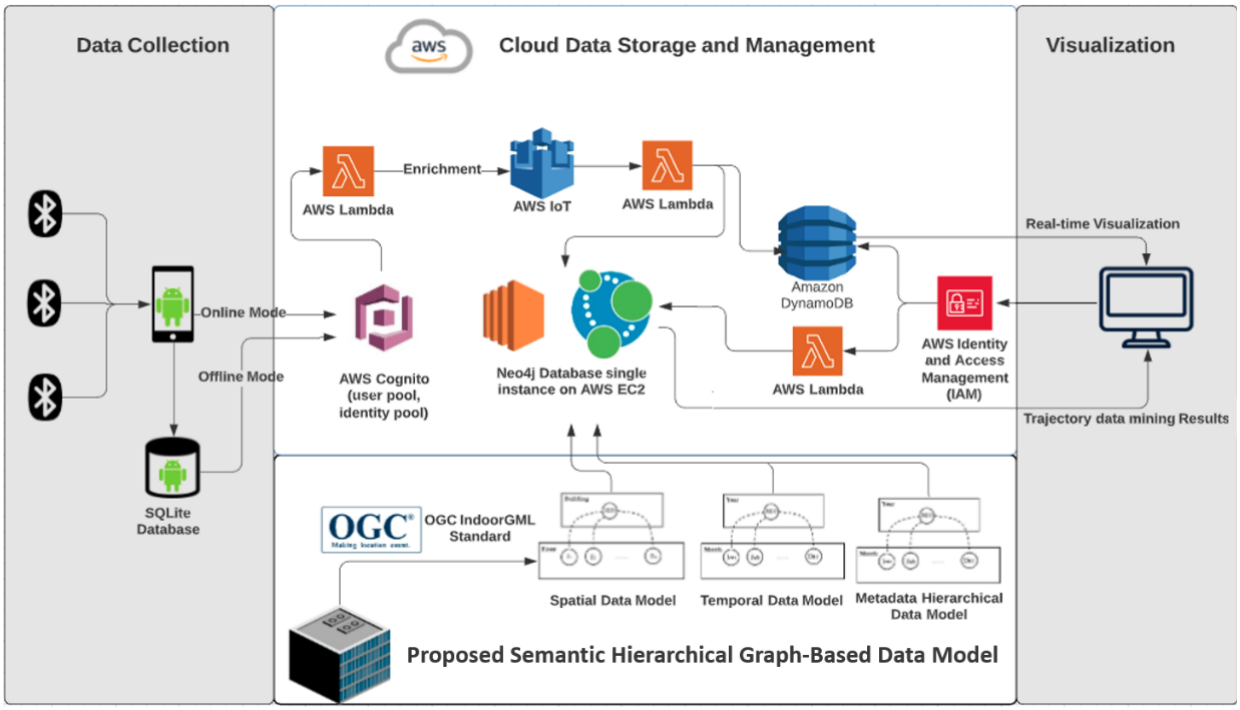


Figure 2.8. System architecture

The data records published on AWS IoT Core will then be stored in Amazon DynamoDB[2] and the Neo4j graph database using another AWS Lambda. Amazon DynamoDB is used as a fully managed No-SQL scalable database offered by Amazon. An instance of the Neo4j database is also deployed in Amazon EC2[3] as the most popular open-source graph database according to DB-Engines ranking[4]. The Neo4j graph database can natively support graph data storage, including nodes and relationships among nodes. The reason behind using two databases in this research is that Amazon DynamoDB is used to visualize enriched trajectory data using SensorUp Explorer

[1] https://aws.amazon.com/iot/
[2] https://aws.amazon.com/dynamodb/
[3] https://aws.amazon.com/ec2/
[4] https://db-engines.com/en/system/Neo4j

web dashboard. While the Neo4j database is used to store and manage the spatiotemporal trajectory for the proposed semantic indoor trajectory data model. The proposed semantic hierarchical graph-based data model implemented by the Neo4j database is responsible for importing hierarchical data: spatial, temporal, and contextual.

Finally, the last layer (*i.e.*, visualization) provides end-users (*e.g.*, contact tracers and building managers) with visualization tools to interact with the proposed system. AWS Identity and Management (IAM)[1] is used in this research to manage end-users' access to AWS.

2.7 **Implementation**

2.7.1 Real-World Data Sets

To evaluate the proposed indoor movement trajectory data model, a real-world experiment is designed. A smartphone app is developed in this experiment, and cloud data storage and management have been set up. A total number of 20 users are asked to install the smartphone app and collect data by their Android smartphones. Four of the users have randomly selected as CCP and two users as cleaning staff. Calgary Centre for Innovative Technology (CCIT) building located in the University of Calgary (UofC) Campus is selected as the test area of our experiment. Figure 2.9 shows the 3rd floor plan of CCIT building, BLE beacons' locations, and connection between indoor cells. For this experiment, 41 BLE beacons from 6 different BLE beacon manufacturers are utilized in 41 indoor cells. Table 2.3 lists the details of different types of BLE beacons used in our experiment, as shown in Figure 2.10.

[1] https://aws.amazon.com/iam/

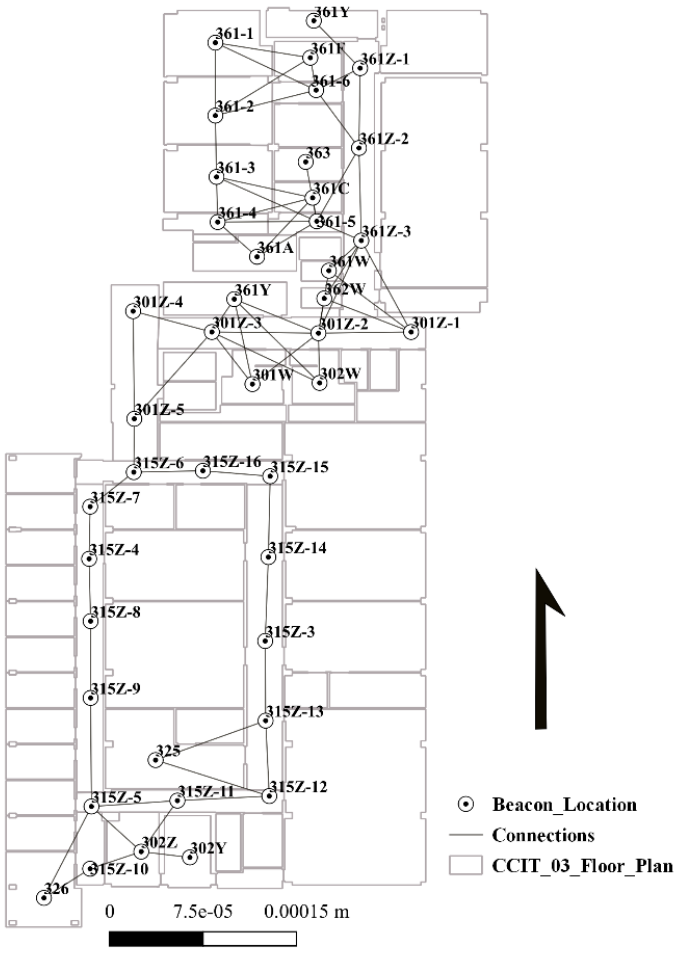


Figure 2.9. Floor plan of the third floor of CCIT building

Figure 2.10. Six different types of BLE beacons from three different BLE manufacturers used in real-world experiment

Table 2.3. Details of BLE beacons used in real-world experiment

BLE Beacon Type	Manufacturer	No. of BEL Beacons in Experiment
Bluetooth Beacon	Estimote[1]	8
IBKS PLUS	Accent Systems[2]	8
IBKS 105	Accent Systems	9
IBS CARD	Accent Systems	7
RadBeacon Dot	Radius Networks[3]	5
RadBeacon Chip	Radius Networks	4

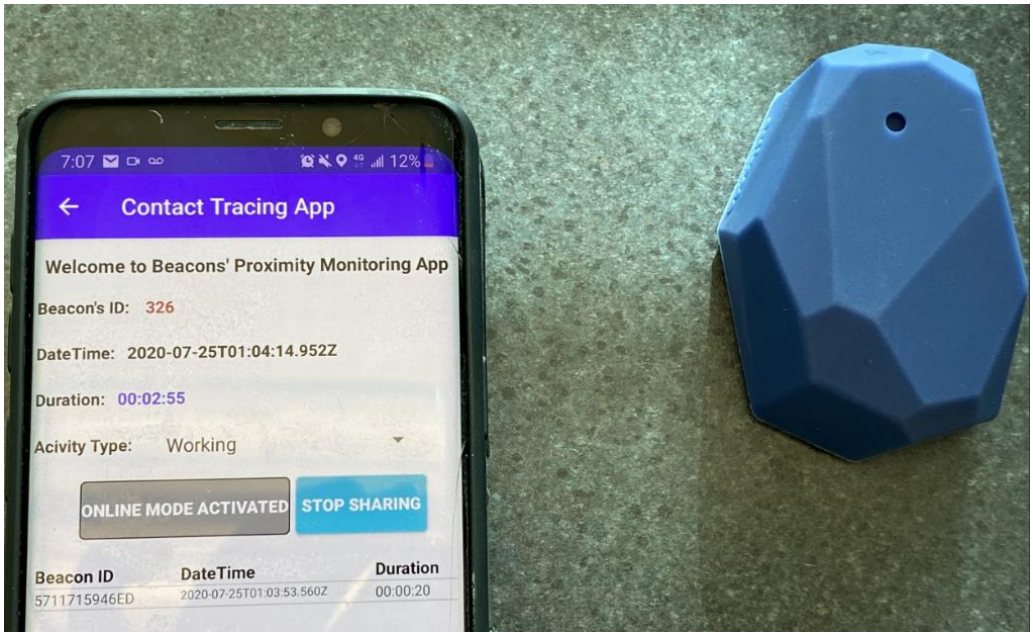
The smartphone app's main responsibilities are measuring RSSI values for all visible BLE beacons, conducting trajectory segmentation, and pushing data records to the cloud data storage. Figure 2.11-a and Figure 2.11-b show the User Interface (UI) of the developed smartphone app when it is capturing RSSI of a Bluetooth Estimote and an IBKS PLUS BLE beacon, respectively. In real-world experiment, all users were asked to spend four hours on 25 July 2020 in the CCIT building. A GeoJSON payload showing a single enriched PoS captured by the developed smartphone app is shown in Figure C1. In this experiment, a total of 582 PoS records were received in the AWS IoT Core. The ultimate goal of our real-world experiment is to evaluate the functionality of the proposed graph-based data model in contact tracing application.

2.7.2 Validating Semantic Indoor Movement Trajectories

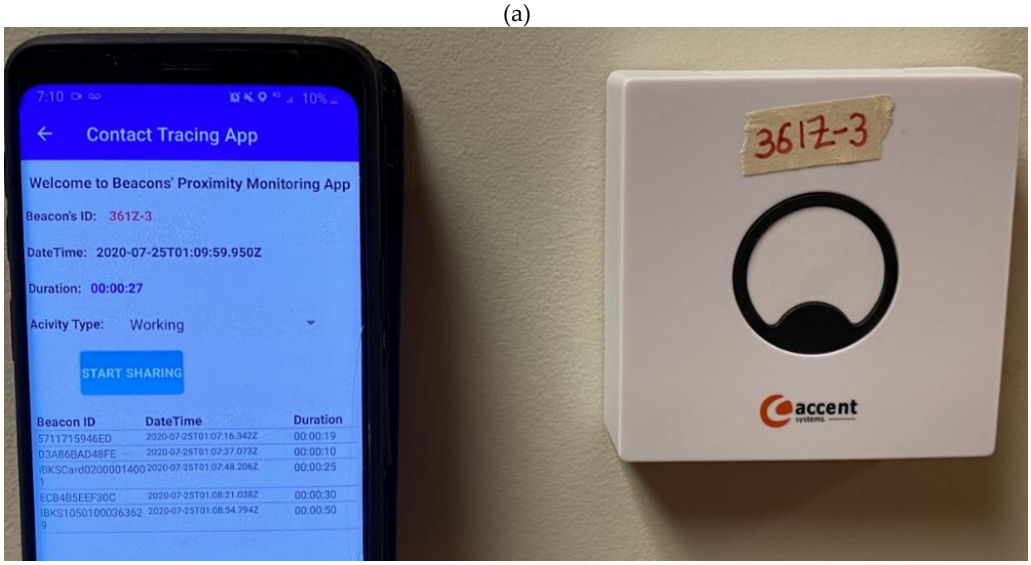
In the real-world experiment, we designed an experiment to directly detect semantically invalid indoor movement trajectory caused by additional, missing or unstable RSSI data. Considering topological relations between indoor cells (*i.e.*, logical connectivity graph) extracted

[1] https://estimote.com/
[2] https://accent-systems.com/
[3] https://www.radiusnetworks.com/

from OGC IndoorGML, they can largely eliminate semantically invalid trajectory segments. In this research, a logical connectivity graph in dual space is considered in the AWS Lambda function to filter semantically invalid trajectory segments. The applied algorithm is shown in Appendix E.6. Figure 2.12 shows the semantic hierarchical spatial data model and connections (in the Neo4J database) of CCIT building, including OGC IndoorGML cells, in-layer, and inter-layer relations.



(a)



(b)

Figure 2.11. The User interface of the developed app on Samsung Galaxy S9 smartphone capturing the proximity of a user from a) Bluetooth Estimote beacon b) IBKS PLUS beacon

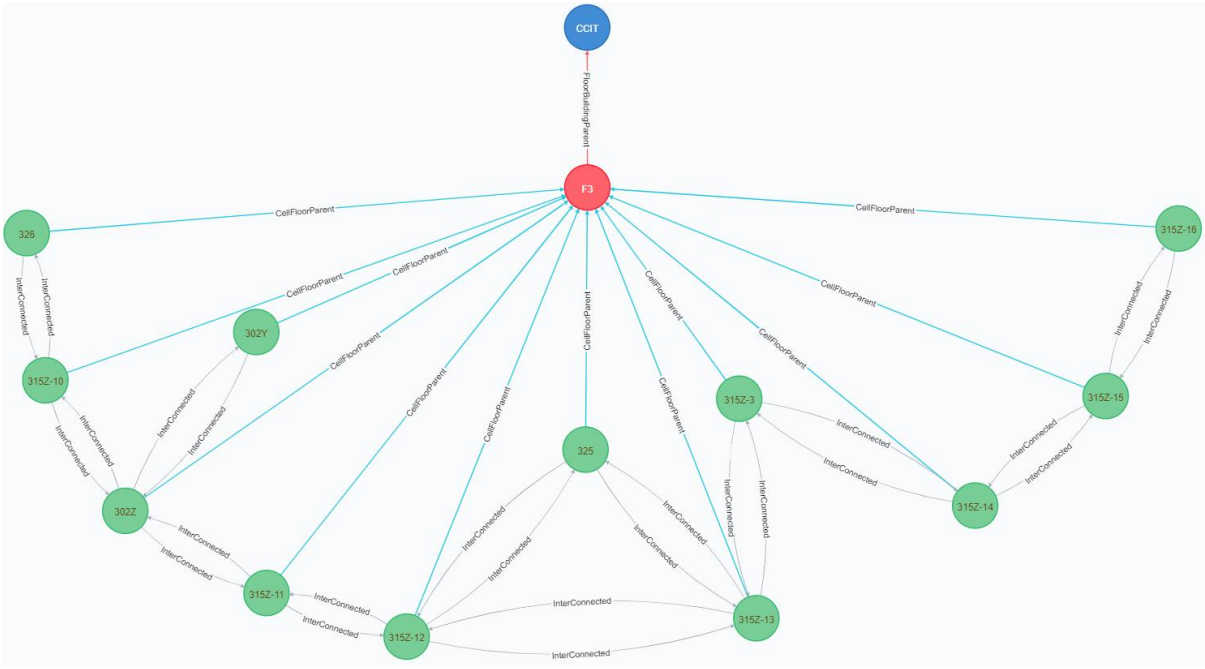


Figure 2.12. Hierarchical spatial graph-based representation of the 20 selected building including OGC IndoorGML cells (green circles), floor (red circle), building (blue circle), inter-layer topological connections among IndoorGML cells (gray arrows), intra-layer topological connections among IndoorGML cells and their corresponding floor (light blue arrows), and intra-layer topological connections floor and corresponding building (red arrow)

2.7.3 Simulation Data Sets

In addition to the real-world experiment, a simulation is developed to generate users' trajectories for the same building setting. To generate synthesized indoor trajectories, a CLI (Command-Line Interface) application with Node.js is developed. The source code of the developed Node CLI application is publicly available on GitHub[1]. In this application, the logical connectivity graph between indoor cells extracted from OGC IndoorGML is considered to synthesize semantically valid indoor movement trajectories. The Random Walk technique, as a stochastic approach, is used to synthesize random indoor movement trajectories for 20,000 users.

[1] https://github.com/soroushojagh/Indoor_Trajectory_Data_Analysis

This simulated dataset contains 453,640 PoS records for all 20,000 users to evaluate the proposed method using a higher number of users. However, as we didn't have the plan of many buildings at the UofC Campus, we consider all trajectories in the same building for a two-week time period. In this simulated dataset, we considered 20% and 10% of users as CCP and cleaning staff types, respectively. So, there are 4,000 CCP users and 1,000 cleaning staff users in total in the simulated dataset. Simulated indoor movement trajectories of all 20,000 users are publicly available in GitHub[1] for further trajectory analysis.

2.7.4 Data Privacy

The Amazon user and identity pools were used as a fully managed service for user authentication and authorization respectively. For the Amazon user pool, a unique Identification (ID) token was assigned to each smartphone user in order to keep the user anonymous. With regards to the Amazon identity pool, user access to back-end services were managed based on their authorization. Also, users have full control over what data they are willing to share with the cloud. To be more precise, they can stop sharing information with the cloud whenever they choose not to. Also, by using user authorization controls, no user is allowed to trace the location of other users. Any user requests for possible contact with diagnosed carriers will receive only the results of the trajectory analysis. Since no further information will be provided for users regarding the time and place they were potentially exposed, the trajectory and ID token of diagnosed carriers will remain anonymous.

[1] https://github.com/soroushojagh/Indoor_Trajectory_Data_Analysis/tree/master/Data/User_Trajectories

2.7.5 Data Visualization Tool

For the visualization purpose of this research, SensorUp Explorer is used as a spatiotemporal web dashboard developed by SensorUp Inc. A short demonstration video of live trajectory data visualization in SensorUp Explorer and Amazon DynamoDB for this research's real-world experiment is shown in Video S1.

2.7.6 Storing Semantic Indoor Trajectories

Considering the concept of semantic indoor trajectory (2), a temporal sequence of PoSs is stored in the Neo4j graph database. Each PoS reflects a node with relation to user context and proximity zone. This node is labelled as a Check-in type with temporal information and user-related metadata properties. As an example, consider the situation that user u_1 entered the proximity zone of B_1 in time t_1 and stayed there for Δ_1 seconds and finally left this proximity zone in time t_2. In this example, a Check-in type of node like ch_1 is created in the Neo4j database. This node has two relationships. The first relationship shows the relation of ch_1 with the user 1 who has created such a PoS. While, the second relationship shows the relation between ch_1 with the OGC IndoorGML cell hierarchy. This node also has temporal properties, including entrance time, duration of stay, and exit time. The indoor semantic trajectory of u_1 in a real-world experiment is shown in Figure D1. Details of the number of PoSs, nodes, and relationships stored in the Neo4j graph database in both real-world and simulated experiments are summarized in Table 2.4.

Table 2.4. Details of stored semantic indoor movement trajectory in the Neo4j graph database

No. of Users	No. of PoSs	No. of Relations	Total No. of Nodes
20	488	1,149	551
200	4,815	9,803	5,058
2,000	47,783	95,739	48,826

20,000	453,640	907,453	473,683

2.7.7 Contact Tracing Application

In this section, a list of spatiotemporal trajectory data queries for the COVID-19 contact tracing application was selected. Each of the spatiotemporal queries were executed in graph databases with different data sizes. It is worth mentioning that Cypher Graph Query Language was used for this research as a declarative graph query language for the Neo4j database.

Query 1 (Contaminated cells by a CCP): In this query, the goal is finding possibly contaminated geospatial cells visited by a single CCP. According to [103], it is assumed that only people who were in close contact with the user for longer than 15 minutes would be possible infected. Accordingly, a cell is contaminated if a CCP has visited it for more than 15 minutes. The Cypher code for this spatiotemporal query can be found in Appendix E1.

Query 2 (Contaminated cells by all CCPs): This query is an aggregation on Query 1 for all CCPs. There are four, 40, 400, and 4,000 CCPs in our real-world and simulated databases, to be more precise.

Query 3 (Temporally constrained contaminated cells by all CCPs): In this query, the list of contaminated cells (*i.e.,* coming from Query 2) is filtered by a selected time window. For example, the Cypher code to find all contaminated cells visited by CCPs from 2020-07-25T02:29:52.461Z to 2020-07-25T02:58:59.461Z can be found in Appendix E2.

Query 4 (Contact tracing for a single CCP): This query uses the contact tracing method proposed by [21, 40] to analyze person-to-person contacts for 15 minutes duration of time in a commonly visited cell. A list of possibly infected users by considering their contacts with a single CCP user is reported.

Query 5 (Contact tracing for all CCPs): This query is an aggregation on Query 4 for all CCPs. This query will prove that our proposed data model is able to consider all CCPs instead of a single CCP. So, a list of possibly exposed users who have been in close contact with each of CCPs for longer than 15 minutes will be reported in this query. The algorithm of this query is presented in Table 2.5. The Cypher code for this query can be found in Appendix E3.

Table 2.5. Contact tracing algorithm (Query 4)

Algorithm 1: Contact Tracing

Input: SMT for all CCP and all ordinary users

Output: A list of possibly infected users

Initialize:

1. $cCells[] \leftarrow Visitedcells\ by\ all\ CCPS for\ 15\ minutes(Query\ 2)$
2. $oCells[] \leftarrow Visitedcells\ by\ ordinary\ users for\ 15\ minutes$
3. $pInfectedUsers \leftarrow []$
4. ***For Each*** $ordinary\ user\ As\ u_o$:
5. ***For Each*** $oCell \in oCells$:
6. ***If*** $(oCell \in cCells)$
7. ***If*** $(\Delta_{CCP} \cap \Delta_{u_o} \geq 15\ minutes)$
8. $pInfectedUsers \leftarrow u_o$
9. ***RETURN*** $pInfectedUsers$

Query 6 (Temporally constrained contact tracing for all CCPs): This query is similar to Query 5 and the results of Query 5 are filtered for a specific time window. For example, a list of possibly infected users who have been in close contact with all of CCPs within a selected time window from 2020-07-25T02:29:52.461Z 2020-07-25T02:58:59.461Z is reported in this spatiotemporal query.

Query 7 (Contaminated cells considering cleaning activity): This query is similar to Query 2 filtered by cleaning activities. Visiting a contaminated place by a cleaner in a sequential order is assumed as a disinfected place (*i.e.*, not-contaminated status) in our proposed method. For example, if a CCP user visits a cell and then it is cleaned by a cleaning user, it is assumed that this cell will not be classified as a contaminated cell to transmit the coronavirus further. The Cypher code for this query can be found in Appendix E4.

Query 8 (Enhanced contact tracing): In this query, the person-to-place way of coronavirus transmission, the sequential order of visiting places, and the disinfection history of places will be incorporated in the contact tracing application. This query shows the flexibility of our proposed data model to consider additional parameters in COVID-19 contact tracing. The algorithm of this query is presented in Table 2.6. The Cypher code of this query can be found in Appendix E5.

Table 2.6. Enhanced contact tracing algorithm (Query 8)

Algorithm 1: Contact Tracing
Input: SMT for a single CCP and all ordinary users
Output: A list of possibly infected users
Initialize:
1. $cCells[] \leftarrow Visited\ cells\ by\ CCPS for\ 15\ minutes (Query\ 2)$
2. $t_c \leftarrow\ CCP\ EntracnceTime$
3. $oCells[] \leftarrow Visited\ cells\ by\ ordinary\ user\ u_o\ for\ 15\ minutes$
4. $t_o \leftarrow\ Ordinary\ user\ EntracnceTime$
5. $dCells[] \leftarrow Disinfected\ cells\ by\ all\ disinfecting\ users\ u_d$
6. $t_d \leftarrow\ Disinfecting\ user\ ExitTime$
7. $pInfectedUsers \leftarrow []$

8. ***For Each*** $ordinary\ user$:
9. ***For Each*** $oCell \in oCells$:
10. ***If*** $(oCell \in cCells)$:
11. ***If NOT***$[oCell \in dCell$ **AND MAX** $(t_c|t_c < t_o) <$ **MAX** $(t_d|t_d < t_o)$ **AND MAX** $(t_d|t_d < t_o) < t_o]$:
12. $pInfectedUsers \leftarrow u_o$
13. ***RETURN*** $pInfectedUsers$

2.8 Results and Discussion

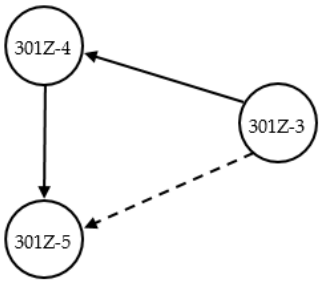
2.8.1 Validating Indoor Real-World Trajectories

Extracting semantically invalid indoor movement trajectories is evaluated as the third aim of this research. In our real-world experiment, all 20 users were asked to keep track of all visited BLE beacons in their proximity zone using unique BeaconIDs that were written on each BLE beacon (as shown in Figure 2.11). This reported BeaconIDs data was used as ground truth indoor movement trajectories. For the real-world experiment, a total of 582 PoSs was detected by all users. After comparing PoS records stored on Amazon DynamoDB and ground truth trajectories, 34 PoS records were determined to be invalid PoSs. Invalid PoS records were caused by missing or unstable RSSI values measured by the smartphone app. After applying our developed preprocessing algorithm in an AWS Lambda function (as demonstrated in Appendix E.6), 31 PoSs were recognized as invalid PoS records. The results of applying the preprocessing algorithm on the indoor movement trajectory in real-world experiments are shown as a confusion matrix in Table 2.7. From Table 2.7, it can be concluded that the preprocessing algorithm detected 73.53 percent of the semantically invalid PoS records. The missing false negative cases in our preprocessing algorithm were caused by multiple connections (links) between the PoS records.

After removing all of the 31 noisy PoSs reported by the developed preprocessing algorithm, another 551 nodes were loaded into the Neo4j graph database. For instance, there are three topological connections between the cells in the building test area of our experiment (Figure 2.9): "301Z-3", "301Z-4", and "301Z-5". In the ground truth trajectory, the user moved from '301Z-3" to '301Z-5". However, the smartphone measurements showed this trajectory: [301Z-3, 301Z-4, 301Z-5]. Therefore, based on ground truth trajectory, "304Z-4" is an invalid trajectory point. However, the proposed preprocessing method cannot detect this point since it is topologically connected to the other two trajectory points. Figure 2.13 shows these cells and trajectory points for both the ground truth and experiment. "301Z-4" is invalid with regards to the ground truth.

Table 2.7. Confusion matrix for preprocessing indoor trajectory method

		Ground-truth trajectory	
		Valid nodes	**Invalid nodes**
Preprocessed trajectory	**Valid nodes**	551	34
	Invalid nodes	554	31



	Indoor movement trajectory segment
Ground-truth	$[\langle 301Z-3\rangle, \langle 301Z-5\rangle]$
Experiment	$[\langle 301Z-3\rangle, \langle 301Z-4\rangle, \langle 301Z-5\rangle]$

Figure 2.13. Representation of an existing situation in which the developed preprocessing algorithm cannot detect invalid trajectory point: dashed lines shows ground-truth and solid line shows collected data from our experiment

2.8.2 COVID-19 Contact Tracing Results

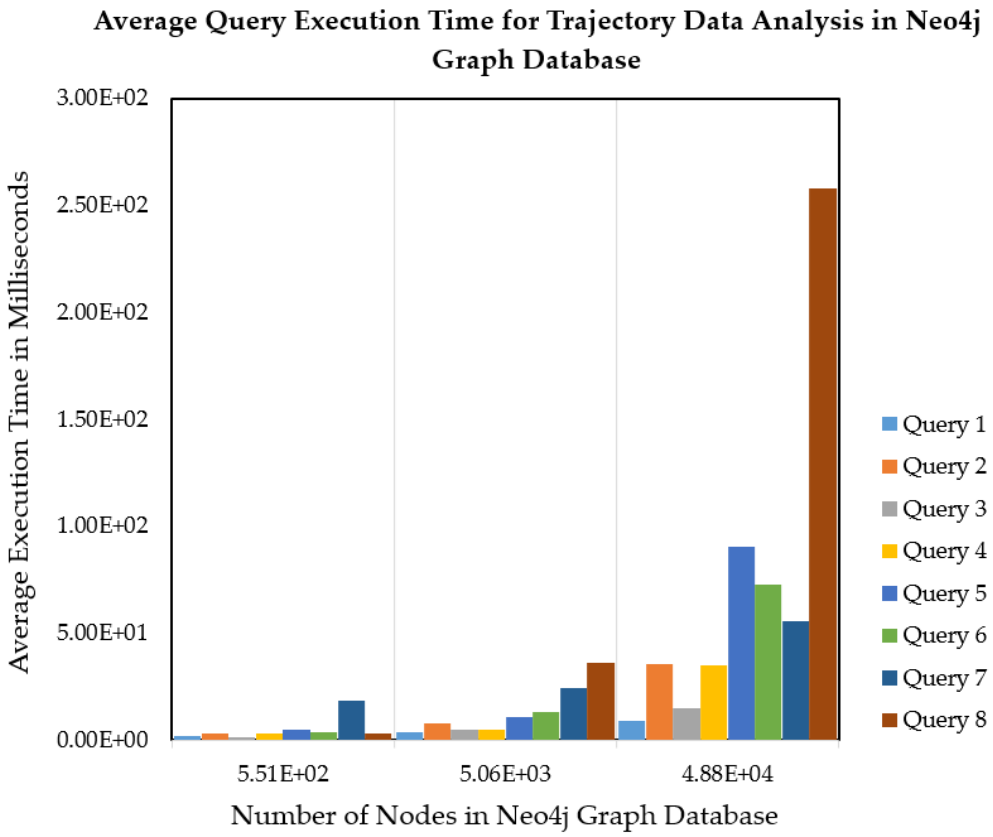
To evaluate the paper's second contribution, the proposed semantic graph-based data model's functionality is evaluated in contact tracing applications (discussed in Section 2.7). Each query was executed a hundred times on four different Neo4j graph databases with a different number of nodes. The first database consists of real-world movement trajectories collected by 20 users with 551 nodes. Simultaneously, the rest three databases are simulated trajectories, including 200, 2,000, and 20,000 users with 5,058, 48,826, and 473,683 nodes, respectively. To show the graph database query execution time for various nodes, the performance results of Query 4 are reported in Table 2.8 as an example. Detailed information, including minimum, average, and standard deviation of all query execution on all four databases are listed in Table E1. For visualizing the performance results, the average and standard deviation of the first three databases are represented in Figure 2.14.

Table 2.8. Query 4 (contact tracing) execution time for different size of the graph databases

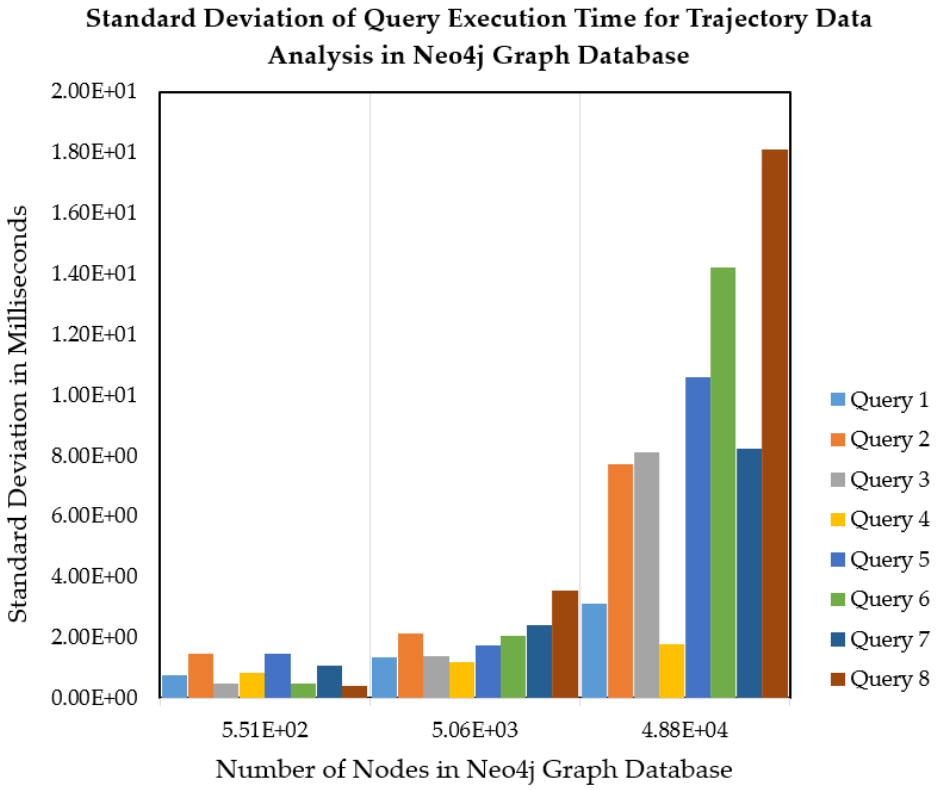
No. of Nodes	Query 4 Execution Time (ms)		
	Minimum	Average	Standard Deviation
551	2.00E+00	2.98E+00	8.24E-01
5058	3.00E+00	5.02E+00	1.17E+00
48826	6.30E+01	9.02E+01	1.77E+01
473683	3.61E+02	4.28E+02	3.47E+01

According to the study done by Silva, F. D., [104], graph size plays an essential role in query execution time. As the general trend, it can be seen that query execution time for all queries increases with the increasing number of nodes. Looking further into average query execution times reveals increasing the number of nodes 100 times for Query 1, 2, 3, 5, and 7 leads to a rise in the

average query execution time by almost less than ten times. While this statistic for Query 6 and 8 is almost 20 and 80 times, respectively.



(a)



(b)

Figure 2.14. Representation of executing trajectory data analysis for 100 times in the Neo4j graph database with 551, 5,058, and 48,826 nodes: a) Showing average query execution time in milliseconds for different trajectory data analysis, b) Representing standard deviation of query execution time in milliseconds for different trajectory analysis.

As seen in Figure 2.14-a, the average execution time for Query 8 (person-to-place contact tracing) is relatively more extensive than other queries. For Query 8, the average query execution time for databases with 0.5k and 5k nodes is less than 20 milliseconds. An increasing number of nodes from almost 5k to 50k results in a rise of almost less than six times for all queries in different sizes of databases. Similarly, as seen in Table E1, increasing the number of nodes from 50k to 500k results the same for all queries except Query 8. For Query 8, an increasing number of nodes from 50k to 500k results in a considerable rise by almost 38 times of average execution time. So, it can be concluded that Query 8 would be more sensitive than other queries to the number of nodes as it is based on the person-to-place contact tracing application.

Looking further into details shows that queries focused on a group of users (*e.g.*, Query 2) require more average execution time than similar queries that are focused only on a single user (*e.g.*, Query 1 with focus on a single CCP). It can also be concluded when the number of query trajectories increases 400 times, the required query execution time increases almost ten times. Also, it can be seen that applying temporal constraints on queries (*e.g.*, Query 3) leads to less required query execution time. Considering temporal indexing for trajectory type of nodes is the underlying reason for this reduction in average query execution time.

As seen in Figure 2.14-b, the standard deviation of query execution time increases by increasing the number of nodes for all queries. The largest standard deviation between all queries in three different databases is for Query 8 with 18.1 milliseconds. It can be concluded that Query 8 has the lowest precision for databases with 5k and 50k nodes. However, this Query has the highest precision in the database with 0.5k nodes. Moreover, although the standard deviation of Query 4 slowly increases with an increase in the number of nodes, it has the lowest rate of change among all queries. So, it can be concluded that Query 4 has the lowest sensitivity with regard to the size of databases.

2.8.3 Enhanced COVID-19 Contact Tracing Results

As discussed earlier, the use of user location history and an overlapping time window of 15 minutes are proposed for the state-of-the-art digital tracing app [21, 40]. For this research, Query 5 is designed to consider all of the aforementioned factors in the people-to-people contact tracing application. Query 8 is designed to consider the people-to-place method of coronavirus transmission, sequential order of visiting places, and disinfection history of places. Those queries were conducted in the Neo4j databases with different number of nodes. Experimental results show that the number of reported possible COVID-19 infected users decreased in Query 8 in comparison

to Query 5 (Figure 2.15). Query 8 successfully filtered 44.98 percent of users who were reported by Query 5 after applying the disinfecting history of the rooms. However, the average execution time of Query 8 increased by 58.3 percent (Table 2.9).

Table 2.9. Comparison of average execution time for Query 5 (contact tracing) and Query 8 (enhanced contact tracing)

Spatiotemporal Queries	**Average Query Execution Time in Milliseconds**		
	Minimum	**Average**	**Standard Deviation**
Query 5	9.00E+00	1.70E+01	4.58E+00
Query 8	8.76E+10	9.91E+01	7.48E+00

In this experiment, we considered 10 percent of the users as providing cleaning activities. In another experiment, a different number of cleaning users is evaluated to show the importance of disinfecting activities in coronavirus transmission. A simulated dataset with 20,000 users is evaluated by considering three different percentages (*i.e.,* 5 percent, 10 percent, and 20 percent) of users as cleaners. As shown in Figure 2.16, disinfecting activities reduce the number of possible COVID-19 infected users in Query 8 by 20.06 percent, 32.34 percent, and 48.16 percent when 5 percent, 10 percent, and 20 percent of the users are considered as cleaning users. It can be concluded that the sequential order of disinfecting activities has a considerable effect on the COVID-19 contact tracing application. In other words, the task of conducting the COVID-19 medical test for the number of possibly exposed users can be decreased by considering disinfecting activities. Our proposed graph data model provides the ability to incorporate this factor for the contact tracing application.

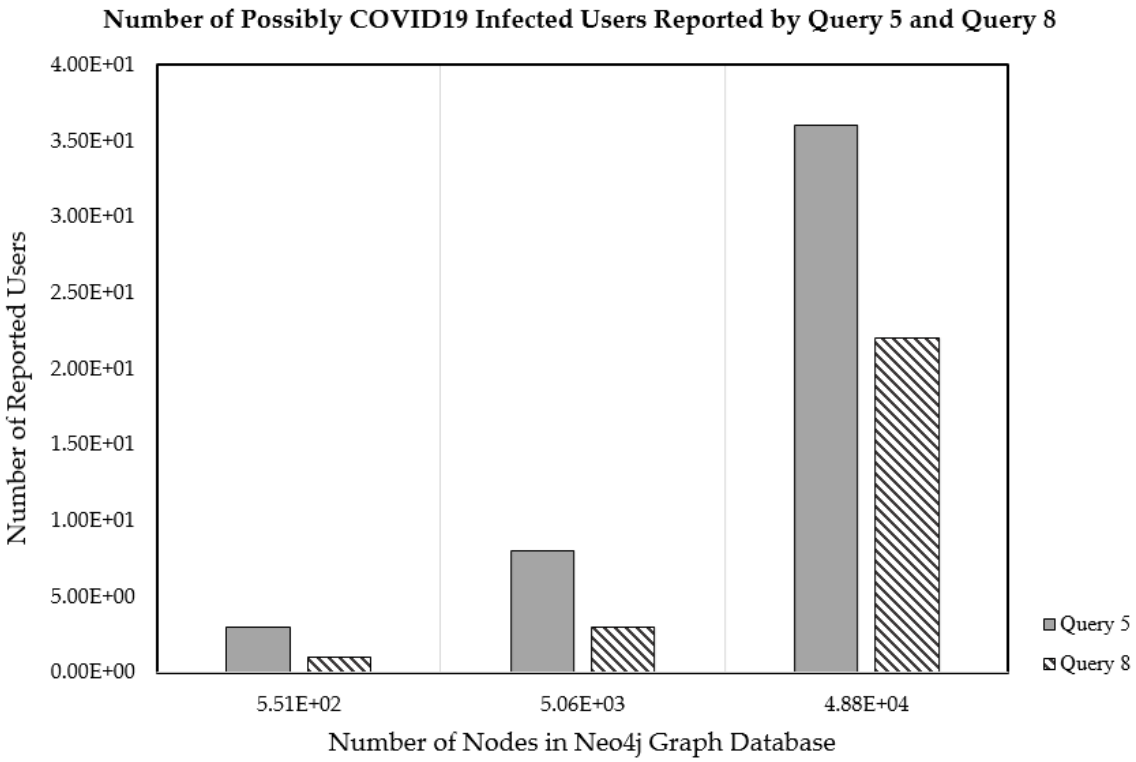


Figure 2.15. Number of possible COVID-19 infected users reported by different contract tracing applications in the Neo4j database with different number of nodes

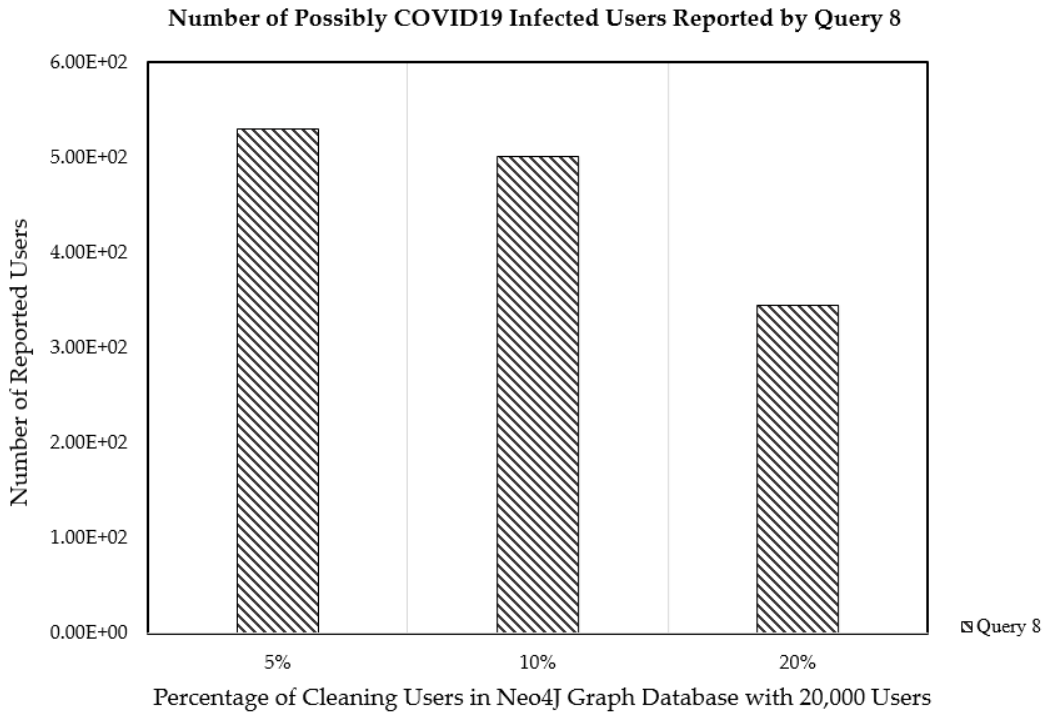


Figure 2.16. Number of possible COVID-19 infected users reported by Query 8 in the Neo4j database with 20,000 users and different percentages: 5 percent, 10 percent, and 20 percent of cleaning users

2.9 **Conclusion and Future Work**

This paper introduces a graph-based semantic indoor trajectory data model that can be utilized in different indoor trajectory analyses. The OGC IndoorGML standard and its multi-layer space model are incorporated in the proposed data model for the semantic segmentation of raw indoor movement trajectories and hierarchical representation of cell spaces in a building (*i.e.*, BLE beacon coverage, rooms, category of rooms, floors, and buildings). Three spatial, temporal, and contextual hierarchical structures were considered in the proposed data model in order to support different granularity levels for trajectory data representation. The digital COVID-19 contact tracing problem was selected as a use case for this research in order to prove the functionality of the proposed data model for trajectory data analysis. There is a large body of research

concentrating on contact tracing applications in person-to-person scenarios for outdoor settings. Hence, this paper focuses instead on indoor settings and both person-to-person and person-to-place scenarios in order to expand state-of-the-art digital contact tracing.

Two experiments were designed to evaluate the main contribution of this research. A smartphone app was developed to collect raw movement trajectories from 20 users for the first real-world experiment. 41 BLE beacons of various types were deployed in a building at the UofC Campus with the assumption that at least one beacon was deployed in each room. Amazon Cloud Web Services was incorporated in order to implement scalable data storage and data management in the Amazon cloud. Taking the logical connectivity graph extracted from OGC IndoorGML into consideration, a filtering algorithm was proposed to clean up the trajectory data. Using the proposed filtering algorithm in the real-world experiment, 73.53 percent of the semantically invalid trajectory points were detected and filtered. In order to further evaluate the performance of the proposed data model, three simulated datasets were generated with 200, 2,000, and 20,000 users and a logical connectivity graph in dual spaces considered. The evaluation results of contact tracing applications in both real-world and simulated experiments illustrated that the proposed graph-based data model could be effectively applied even for the most complicated contact tracing queries. The average query execution time of all of the contact tracing applications in the real-world experiments was less than five milliseconds with an average standard deviation of less than one millisecond. However, the average query execution time increased when the number of nodes in the simulated experiments increased.

For this research, the COVID-19 contact tracing application is selected to evaluate the proposed data model's functionality in indoor environments. The effectiveness of a digital contact tracing system depends on various factors such as public adoption [105]. Different factors such as

privacy, the government's level of enforcement to use the system, and transparency in data storage and re-use can influence the public adoption of a digital contact tracing system [61, 105]. Although assessing the effectiveness of digital contact tracing systems is out of our research scope, evaluating the success of contact tracing systems is required for future pandemics. Also, we focused only on indoor environments as they are the most complicated type of physical environments. A seamless positioning system providing seamless outdoor and indoor location information could be an exciting topic for future study. The proposed filtering algorithm in this research detects semantically invalid trajectory points but cannot improve the trajectories using possible logical connection. Further research is required to develop a trajectory reconstruction approach based on the IndoorGML connectivity graph, beacon coverage, and traverse time between cell spaces. In this research, a BLE-based proximity positioning system is deployed in an indoor environment to determine users' location for indoor spatiotemporal trajectories. Environmental factors such as indoor furniture cause reflecting and blocking the signal and impose inaccuracies on proximity estimations. So, evaluating the accuracy provided by the proximity positioning system is on hold for future work. User privacy and data secrecy protection is another direction for future research as well, especially in relation to user privacy in the cloud. In order to apply the proposed contact tracing application to a large-scale product that can be adopted by the public, detailed, scalable user privacy research needs to be conducted. Although privacy protection is outside this paper's scope, basic authentication and security authorization preserving techniques and user ID anonymization were applied to the proposed contact tracing application. Various user contexts (*e.g.,* cleaning activities and job type) can be automatically extracted without human intervention [93, 106]. Although user contexts are manually selected in this research, investigating

automatic context extraction approaches can improve the scalability of the proposed systems and is on hold for future work.

Appendix A

Considering the importance of disinfecting activities to reduce the risk of being exposed to the virus [35], individuals are strongly recommended by health organizations to sanitize the immediate space around them after each use [107, 108]. As an example scenario, the importance of disinfecting activities in common areas and public spaces such as lobbies is shown in Figure A1. In Figure A1-a, the transmission of SARS-CoV-2-laden droplets from an infected host is shown. As illustrated in Figure A1-b, the immediate vicinity of the infected host is contaminated by droplets and then sanitized using a disinfectant wipe in Timestamp2. However, as shown in Figure A1-c, there are surfaces that are still contaminated and allow the susceptible host to be exposed by the virus in Timestamp3. As shown in Figure A1-d, the well-trained cleaning staff disinfects the contaminated objects using the right equipment (*e.g.,* electrostatic spray) in Timestamp4. As seen in this scenario, disinfecting activities and temporal sequence of visiting common areas are required to be considered in digital contact tracing. In this example, considering the disinfecting activities and temporal sequence of visiting a common area, the susceptible host needs to be notified by the digital contact tracing system. If we assume that Timestamp4 (*i.e.,* Figure A1-d) occurred before Timestamp3 (*i.e.*, Figure A1-c) in the example scenario, there is no need to notify the susceptible host in the digital contact tracing system.

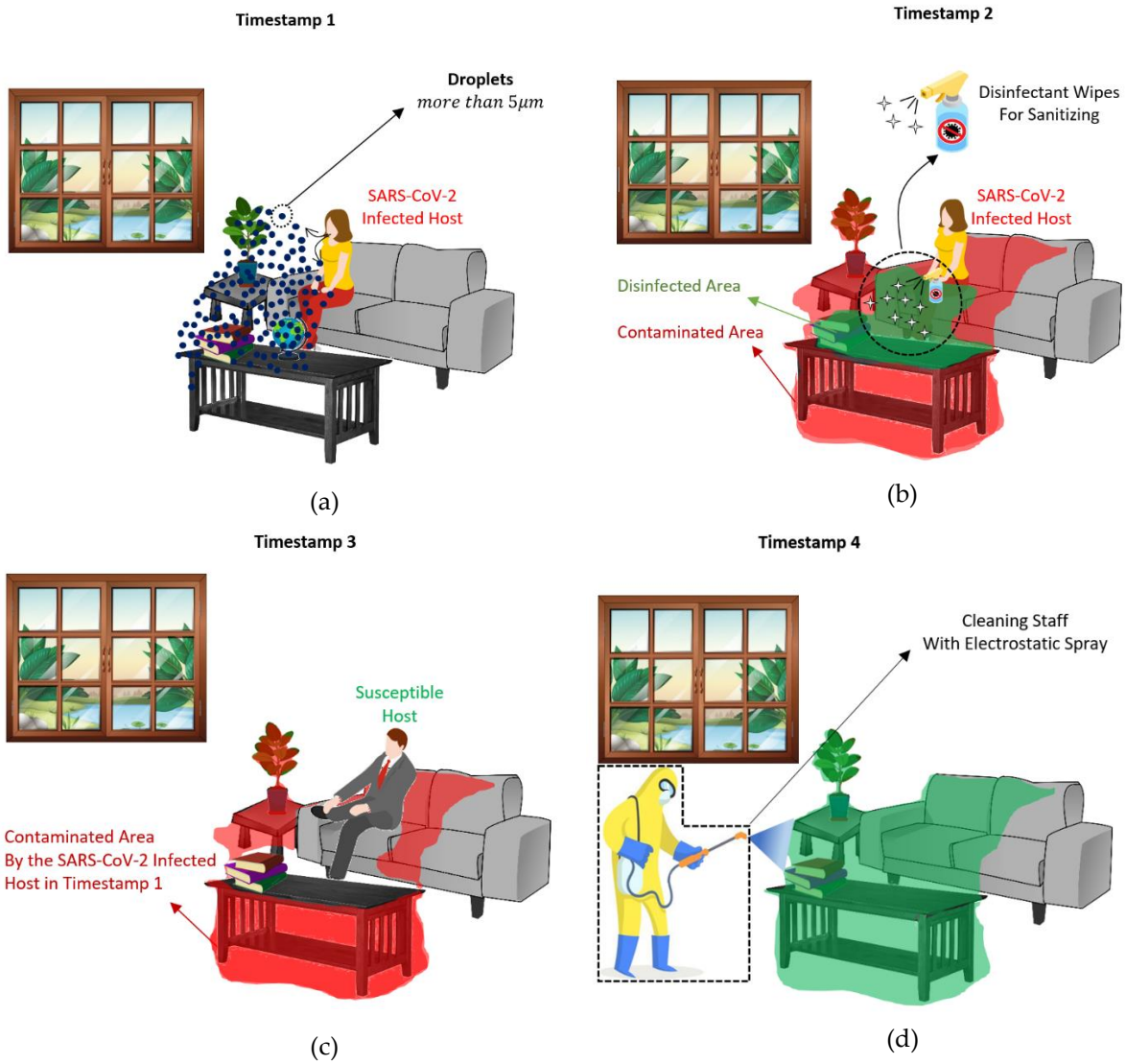


Figure A1. The impact of disinfecting activities to prevent further spread of the SARS-CoV-2 virus: a) Showing transmission of SARS-CoV-2-laden droplets caused by an infected host; b) Showing possibly contaminated surfaces in the immediate vicinity of the infected host and disinfected areas after sanitizing; c) Showing objects that are still contaminated and allow the susceptible host to be exposed by the virus, and d) Showing disinfected area after sanitizing contaminated objects by cleaning staff

Appendix B

In the context of COVID-19 spread, there might exist many situations where individuals are located close to each other while physically separated by obstructions such as walls and glasses. As shown in Figure B1, two individuals are located close to each other but physically separated (*i.e.*, a user is inside a building while the other is in a bus stop). Considering SARS-CoV-2

transmission ways, physical obstructions in between users can stop virus transmission. Existing obstructions in between BLE beacon and receiver (*e.g.*, smartphone) attenuate the radio signal. For more information on signal attenuation and the impact of different materials on RSSI values, interested readers can refer to the study conducted by Çaliş et al. [109]. As shown in Figure B1, reduced signal strength in BLE technology can represent existing obstructions between users. In contrast, GNSS cannot consider existing physical obstructions among individuals [21].

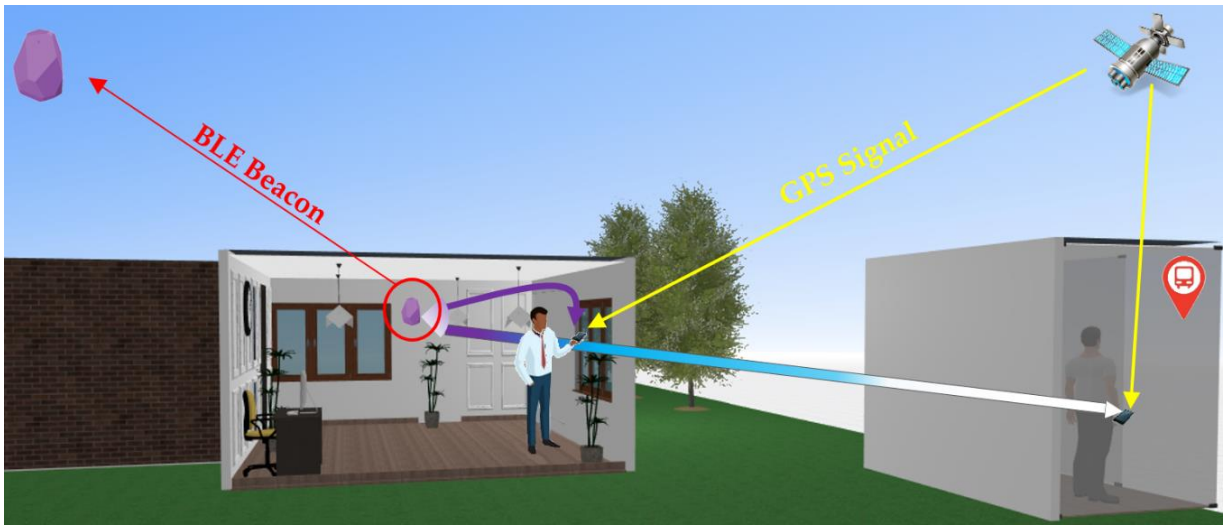


Figure B1. Representation of the difference between BLE and GNSS technology to consider physical obstructions among; The gradient color schematically illustrates radio signal strength in BLE technology (*i.e.*, purple and white colors represent the strongest and weakest radio signal strength)

Appendix C

A JSON payload showing a single POS record captured by the developed smartphone app and received in the cloud AWS IoT Core is shown in Figure C1.

```
{
  "geojson": {
    "geometry": {
      "coordinates": [
        -114.11484917,
        51.07921792,
        8.6
      ],
      "type": "Point"
    },
    "properties": {
      "JobType": "Cleaning",
      "UserID": "3029936e-e185-412c-a544-b0ef85849f1a",
      "VulnerabilityLevel": 8,
      "ActivityType": "Working",
      "BeaconCellName": "361-4",
      "BeaconID": "IBKS10501000363638",
      "BeaconLatitude": 51.0792179,
      "BeaconLongitude": -114.11484917,
      "BeaconAltitude": 8.6,
      "EntranceDateTime": "2020-07-25T00:42:57.461Z",
      "EntranceGPSTime":1911602559461,
      "Duration": 331,
      "ExitDateTime":"2020-07-25T00:48:28.461Z",
      "ExitGPSTime":1911602890461,
      "RSSI": -65
    },
    "type": "Feature"
  }
}
```


Figure C1. A single JSON payload showing the POS record of one of the users received in the AWS IoT Core

Appendix D

Related experimental images for this research are shown in this Appendix.

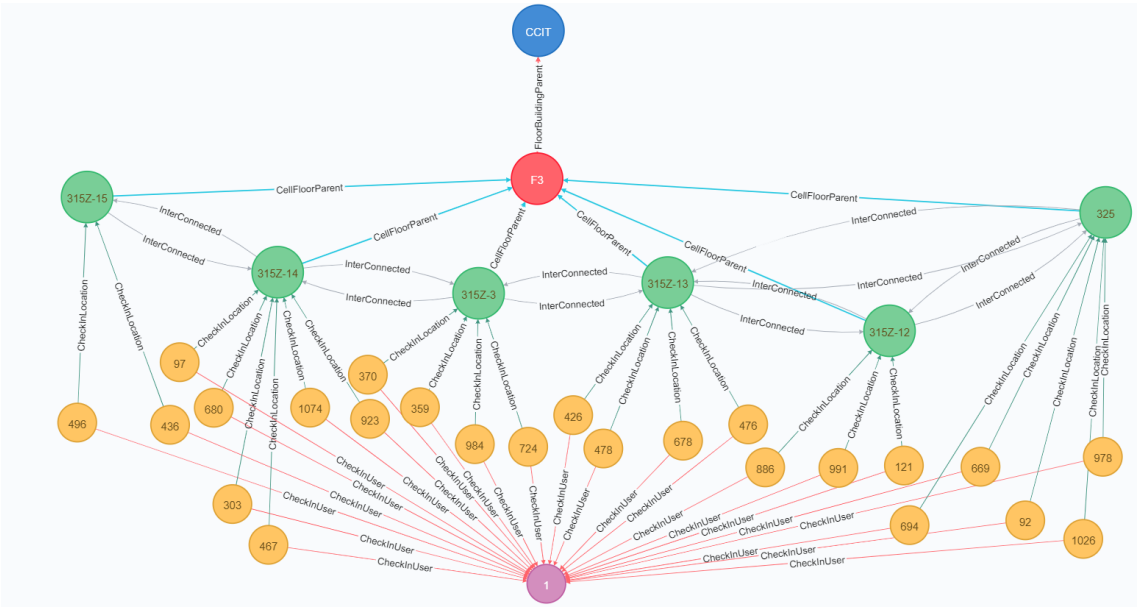


Figure D1. An example of user trajectory modelled by the proposed graph- based semantic indoor trajectory in a real-world experiment using Neo4j graph database for user a single user (*i.e.*, user 1). Nodes are illustrated as circles with different colors: Blue for building, red for floor, green for cells, yellow for trajectory (the number shows the duration of time in seconds for each check-in), and purple for user nodes

Appendix E

This Appendix is focused on the Cypher code representation of the indoor trajectory queries in the Neo4j graph database.

E.1 Query 1

The Cypher code of Query 1 and execution of this query in Neo4j is represented in this subsection.

Cypher Code

```
MATCH (c:Cell)<-[]-(ch:CheckIn)-[]->(u:User)
WHERE u.UserID = '2' AND ch.Duration > 900 //Duration of 15 minutes
MATCH (c)-[:CellFloorParent]->(f:Floor)
MATCH (f)-[:FloorBuildingParent]->(b:Building)
RETURN DISTINCT c.CellID AS Infected_Geospatial_Zone, f.FloorID AS
Infected_Floor, b.BuildingID AS Infected_Building
```

Execution in the Neo4j with 551 Nodes

"Infected_Geospatial_Zone"	"Infected_Floor"	"Infected_Building"
"315Z-4"	"F3"	"CCIT"
"315Z-6"	"F3"	"CCIT"
"315Z-15"	"F3"	"CCIT"
"315Z-14"	"F3"	"CCIT"
"315Z-8"	"F3"	"CCIT"
"315Z-3"	"F3"	"CCIT"
"315Z-16"	"F3"	"CCIT"
"315Z-9"	"F3"	"CCIT"
"315Z-7"	"F3"	"CCIT"

Figure E1.The results of executing Query 1 in the Neo4j graph database with 551 nodes and 1,149 relationships: the first column of results shows contaminated indoor cells, while the second and third columns show the floor and building in the hierarchical graph data model

E.2 Query 3

The Cypher code of Query 3 is represented in this subsection.

Cypher code

```
MATCH (c:Cell)<-[]-(ch:CheckIn)-[]->(u:User)
WHERE u.UserHealthStatus = 'COVID19' AND ch.ExitTime > 1911608974461 AND ch.ExitTime < 1911610721461 //GPS Epoch Time of 2020-07-25T02:29:52.461Z and 2020-07-25T02:58:59.461Z
MATCH (c)-[:CellFloorParent]->(f:Floor)
MATCH (f)-[:FloorBuildingParent]->(b:Building)
RETURN DISTINCT c.CellID AS Infected_Geospatial_Zone, f.FloorID AS Infected_Floor, b.BuildingID AS Infected_Building
```

E.3 Query 5

The Cypher code of Query 5 is represented in this subsection.

Cypher code

```
MATCH (c:Cell)<-[]-(ch:CheckIn)-[]->(u:User)
MATCH (u2:User)<-[]-(ch2:CheckIn)-[]->(c2:Cell)
WHERE u.UserHealthStatus = 'COVID19' AND u <> u2 AND  c = c2 AND ((ch.EntranceTime > ch2.EntranceTime and ch.EntranceTime < ch2.ExitTime) OR (ch2.EntranceTime > ch.EntranceTime AND ch2.EntranceTime < ch.ExitTime)) AND
(ch.Duration*1000 + ch2.Duration*1000 - ABS(ch2.EntranceTime-ch.EntranceTime)-ABS(ch2.ExitTime-ch.ExitTime))/2 > 900*1000
RETURN DISTINCT u.UserID AS COVID19_User,u2.UserID AS Possibly_Infected_User,c2.CellID AS Infected_Geospatial_Zone
```

E.4 Query 7

The Cypher code of Query 7 is represented in this subsection.

Cypher code

```
MATCH (c:Cell)<-[]-(ch:CheckIn)-[]->(u:User)
MATCH (c2:Cell)<-[]-(ch2:CheckIn)-[]->(u2:User{JobType:"Cleaning_Staff"}
)
WHERE u.UserHealthStatus = 'COVID19' AND u<>u2 AND c = c2 AND ch.ExitTime < ch2.EntranceTime
MATCH (c)-[:CellFloorParent]->(f:Floor)
MATCH (f)-[:FloorBuildingParent]->(b:Building)
RETURN DISTINCT c.CellID AS Infected_Geospatial_Zone, f.FloorID AS Infected_Floor,b.BuildingID AS Infected_Building, u.UserID AS COVID19_User
```

E.5 Query 8

The Cypher code of Query 8 is represented in this subsection.

Cypher code

```
MATCH (c:Cell)<-[]-(ch:CheckIn)-[]->(u:User)
MATCH (u2:User)<-[]-(ch2:CheckIn)-[]->(c2:Cell)
MATCH (uc:User{JobType: "Cleaning_Staff")<-[]-(ch3:CheckIn)-[]->(c3:Cell)
WHERE u.UserHealthStatus = 'COVID19' AND NOT u2.JobType = "Cleaning_Staff" AND NOT
u2.UserHealthStatus = "COVID19" AND u<>u2 AND u2 <> uc AND c = c2 AND c =c3 AND NOT
(ch3.EntranceTime < ch.EntranceTime AND ch3.ExitTime < ch2.EntranceTime)
RETURN u2.UserID As Possibly_Infected_Users
```

E.6 Semantically Valid Trajectory Extraction

The applied algorithm to extract the semantically valid indoor movement trajectory is represented in this section.

***Algorithm 1*:** Extracting semantically valid indoor movement trajectory

Input:

A temporal sequence of indoor cells c for a user as a moving object having size = n :
$S = [\langle c_0, t_0\rangle, \langle c_1, t_1\rangle, \dots, \langle c_n, t_n\rangle]$ in which $t_0 < t_1 < \cdots < t_n$
The adjacency matrix $M_{j\times j}$ showing connectivity between indoor cells in which j is the size of indoor cell:

$$M_{i,j} = \begin{cases} 1 & if\ there\ is\ an\ inter-layer\ connection\ between\ cell\ c_i\ and\ c_j \\ 0 & otherwise \end{cases}$$

Output: Semantically valid trajectory $\hat{S}$ having size = m

Initialize:

$\hat{S} \leftarrow []$
$\hat{S}_0 \leftarrow S_0$
$counter \leftarrow 1$
while $counter <$
    if $M_{\hat{S}[m-1]\times S[counter]} == 1$
        $\hat{S}[m] \leftarrow S[counter]$
    $counter++$
End-while
Return $\hat{S}$

Table E1. Details of query execution time for trajectory queries in Neo4j graph database with different sizes

Spatiotemporal Queries	No. of Nodes	Query Execution Time in Milliseconds		
		Minimum	Average	Standard Deviation
Query 1	551	1.00E+00	2.02E+00	7.74E-01
	5058	2.00E+00	3.82E+00	1.33E+00
	48826	4.00E+00	9.05E+00	3.13E+00
	473683	4.00E+00	1.36E+01	5.96E+00
Query 2	551	1.00E+00	3.05E+0	1.46E+0
	5058	5.00E+00	7.90E+00	2.13E+00
	48826	2.20E+01	3.55E+01	7.72E+00
	473683	3.11E+02	3.49E+02	2.01E+01
Query 3	551	1.00E+00	1.49E+00	5.00E-01
	5058	3.00E+00	4.91E+00	1.37E+00
	48826	2.00E+00	1.51E+01	8.10E+00
	473683	2.00E+00	3.12E+01	1.50E+01
Query 4	551	2.00E+00	2.98E+00	8.24E-01
	5058	3.00E+00	5.02E+00	1.17E+00
	48826	6.30E+01	3.52E+01	1.77E+01
	473683	3.61E+02	4.28E+02	3.47E+01
Query 5	551	3.00E+00	5.02E+00	1.46E+00
	5058	8.00E+00	1.07E+01	1.73E+00
	48826	1.60E+01	9.02E+01	1.06E+01
	473683	1.04E+02	1.46E+02	2.53E+01
Query 6	551	3.00E+00	3.43E+00	4.95E-01
	5058	1.00E+01	1.29E+01	2.05E+00
	48826	4.80E+01	7.24E+01	1.42E+01
	473683	4.96E+02	5.29E+02	1.85E+01
Query 7	551	1.70E+01	1.85E+01	1.06E+00
	5058	2.10E+01	2.44E+01	2.41E+00
	48826	4.20E+01	5.58E+01	8.23E+00
	473683	6.00E+01	9.01E+01	1.83E+01
Query 8	551	2.00E+00	3.00E+00	8.37E-01
	5058	3.10E+01	3.62E+01	3.54E+00
	48826	2.30E+02	2.58E+02	1.81E+01
	473683	9.19E+03	1.03E+04	6.26E+01

3 Enhanced Air Quality Prediction by Edge-Based Spatiotemporal Data Preprocessing

3.1 Abstract

The negative impact of poor air quality on human health attracts increasing attention in air quality prediction. PM2.5 can be considered as the most dangerous air pollutant that affects human health the most. In addition, technological advances, such as those involving the Internet of Things (IoT) for monitoring air quality, have made it possible to monitor air quality for a lower cost. However, missing values and noisy data make nonlinear data provided by air quality IoT sensors less reliable and more complicated than data provided by air quality monitoring stations, which are equipped with more expensive air quality sensors. In this study, we propose a mixed edge-based and cloud-based framework with the final goal of PM2.5 value prediction. In particular, the edge components are in charge of enhancing the quality of data in terms of missing and noisy values and of reducing the amount of data that must be sent to the cloud. The cloud component is in charge of data prediction by applying deep learning methods that combine sensors data and weather information. As a result, the proposed approach can be applied not only for static data but also for real-time data provided by air quality IoT sensors. In order to validate the proposed approach, we evaluate the quality of predictions using both original and preprocessed data on a real-world dataset from air quality sensors distributed in Calgary, Canada. Obtained results show an average improvement of 40.18% of the prediction accuracy on Mean Absolute Percentage Error by using the proposed preprocessing technique.

Keywords: Air Quality IoT Sensors, Data Preprocessing, Long Short Term Memory, Dynamic Time Warping, PM2.5 Prediction

3.2 Introduction

Air quality prediction can be considered one of the essential knowledge that is beneficial not only for industry, governments, researchers, and consultants but also for the public to protect themselves from pulmonary diseases [110]. Based on statistics, air pollution is the primary cause of around 60,000 annual deaths worldwide [111]. Among all air pollutants such as suspended nitrogen dioxide, ozone, sulfur dioxide, and carbon monoxide, particulate matter with a diameter less than 2.5 micrometers known as PM2.5 is recognized as the most dangerous atmospheric pollutant [14]. As a result, predicting PM2.5 has increasingly attracted more attention and has become a hot topic among researchers recently.

The rapid development of sensing technologies, primarily in terms of the Internet of Things (IoT) and air quality sensors, allows gathering a massive volume of the atmospheric air pollutants data. We witness that millions to billions of small sensors and actuators have been embedded in real-world objects and connected to the Internet. Constructing permanent air quality monitoring stations in a metropolitan area can be more expensive and time- consuming than installing IoT sensors to track air quality. Due to the low price of IoT air quality sensors, we can expect better distribution and finer granularity in the collected air quality data. However, missing values and noisy data make the data collected by IoT sensors less reliable than the data collected by a monitoring station.

Missing and noisy values make a vast amount of data collected by IoT sensors as inconsistent as other real-world data sets. Predicting air quality based on such incomplete data set is an arduous task for predictive algorithms, and the final result could be unreliable. When it comes to human health, air quality mis-prediction would be too expensive, especially for those patients with asthma or other pulmonary diseases that should not be exposed to contaminated air. Hence,

we are in growing need of making the results of the predictive algorithm as reliable as possible by applying data preprocessing and data cleaning techniques. Although data preprocessing has not been thought to be of such importance in similar studies, smoothing noisy data and predicting missing values is of great importance in terms of data provided by IoT air quality sensors.

One possibility is to carry out complex data preprocessing and cleaning tasks on the cloud. However, resorting to cloud-based methods presents some drawbacks. First of all, communication bottlenecks yielded by too much data transmitted from nodes to the central servers may induce packet losses and delays [31]. Moreover, in large sensors networks simultaneously analyzing all sensors streams would introduce real-time computational constraints. On the other hand, carrying out data preprocessing and cleaning directly on the sensor equipment can be either impossible, due to the lack of computational power, or unreliable, due to the necessity of analyzing local data only.

In this work we design an edge-based strategy for preprocessing and filling missing sensors data where, as one of the main innovation of the work, both temporal and spatial information are taken into account in order to fill missing data. Edge nodes are equipped with light computing power. Sensors send raw data to the associated edge node which is in charge of both preprocessing (aggregating and cleaning) the raw data and to deal with missing data. It is worth pointing out that this architecture allows to distribute the computational requirements from one single cloud server to a set of edge nodes; moreover, the preprocessing step reduces the granularity of data (from minutes to hours), thus significantly reducing the amount of information that must be sent to the cloud for the prediction tasks.

As far as the air quality prediction task is concerned, it is worth pointing out that it is a spatiotemporal variable with nonlinear patterns [15]. For example, nitrogen dioxide varies spatially and temporally based on anthropogenic emissions like burning fossil fuels and meteorological

conditions such as wind speed and temperature [16, 17]. Different parameters, including human activities, weather conditions, traffic flow, pollutant emission, and deposition, can be considered underlying reasons to make the spatial distribution and temporal trend of air quality more complicated [27]. As a result, simple linear techniques such as linear regression are not very suited to predict air quality, and they cannot always be used as a preprocessing tool to smooth noisy values or automatically fill missing values.

The diverse range of prediction algorithms from linear regression to neural network algorithms is applied to predicting air quality [16, 22-24]. One of the most significant drawbacks of conventional methods is that they require many computing resources [25]. The complex nature of air quality patterns also makes them unpredictable with some conventional predicting techniques like linear regression. Recently, deep learning has emerged as an alternative solution to address the shortcomings of conventional linear methods in air quality prediction [25, 26]. Deep learning techniques applied in the domain of air quality prediction can be categorized into Stochastic Gradient Descent (SGD), Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNN), Recurrent Neural Network (RNN) and Auto-Encoders (AE) [25, 112]. Among all the applied deep learning algorithms in air quality forecasting, Long Short-Term Memory (LSTM, a type of RNN) and AE have shown better performance than other algorithms [27, 28, 110].

In this study, an LSTM-based method is applied for air quality prediction on the cloud where, in order to improve the quality of predictions, (clean) sensed data is coupled with weather information. While this deep learning technique has been employed for air quality prediction in other studies [14, 26-30], the impact of missing data along with the inclusion of weather information have not been investigated so far. Moreover, in order to provide a realistic evaluation of our approach, a real-world dataset provided by air quality sensors distributed in Calgary,

Canada, is exploited for the experiments. It is worth pointing out that the main focus of our current research is the evaluation of the impact of data preprocessing and additional information on the quality of predictions; we will address other performance issues related to latency, load balance, and related topics in future works.

We design a mixed edge-based and cloud-based evaluation strategy for the prediction of air quality, specifically for values of PM2.5. Summarizing, the main contributions and implications of this paper can be summarized as follows:

1. The edge-based component exploits both temporal and spatial information in order to clean raw data and fill in missing values.
2. The cloud-based component implements a standard LSTM-based deep learning method to predict PM2.5 values, where air quality is considered as a nonlinear, multivariate, and spatiotemporal pattern and is coupled with weather information in order to improve the quality of predictions.
3. A real-world use case from a network of air quality IoT sensors in Calgary, Canada is exploited to evaluate the proposed approach.
4. The impact on prediction quality of inconsistent data including missing values, and the corresponding correction technique, is evaluated.

This paper is organized as follows: in Section 3.3 related research in terms of applying different algorithms to predict air quality is briefly reviewed. The overall architecture of the applied deep learning methods, including the proposed preprocessing engine and the deep learning predictor, is discussed in Section 3.4. The statistical information on the real-world data set, evaluation index, and other descriptive information about the implementation and testing of the

proposed approach are provided in Section 3.5. Finally, in Section 3.6 we draw some conclusions and suggest future directions for research.

3.3 Literature Review

The task of air quality prediction, thanks to its importance, rapidly gained a lot of attention from the academic community. More in detail, the amount of work regarding air quality prediction that considers the PM2.5 air pollutant represents a consistent share of the work presented in literature. Undoubtedly, all such studies focus on the main task of prediction while considering different aspects and exploring several methods and technologies. To better navigate among the work available in the literature and to better differentiate our literature review, we categorize the corresponding research studies into two groups, namely *(i)* knowledge-based and *(ii)* data-driven approaches. The former group includes methods which put beside raw data retrieved from sensors several other properties and knowledge about the analyzed context. In the latter group, research focuses on the perspective of the data and the methods used to elaborate it; thus they can be categorized in works adopting linear and nonlinear models, statistical and deep learning approaches. It is worth pointing out that these two groups share several works, considering that the problem is often analyzed through different point of views. In the following, we first describe in detail the related work among the two mentioned groups. Then, in order to provide a bird's-eye view of the discussed studies and to better position our work among them, we present a Table comparing our proposed approach to the most related ones.

3.3.1 Knowledge-Based Approaches

Spatial correlation between monitoring stations and the long-term temporal dependencies between air quality and weather patterns play an important role in accurately predicting air quality.

Wang and Song [30] trained a model based on deep LSTM combined with a regression technique to predict air quality by considering meteorological and air quality historical data. The spatial correlations between monitoring stations were captured by adopting Granger causality to model relative stations and areas in their study. Linear regression, regression tree, deep neural network, and feedforward neural network were used as baselines to evaluate the result obtained from LSTM applied in this research. They proved that the LSTM outperforms other baseline methods to predict air quality by considering spatial relations between monitoring stations. In the study done by Soh et al. [28], a predictive framework has been proposed to forecast PM2.5 within the next 2-day period. They combined artificial neural network, convolutional neural network and the LSTM to take spatiotemporal relations of air quality into account. In their study, elevation information has been innovatively involved in the deep learning predictive model to consider the terrain around the desired location to forecast air quality. It is worth pointing out that there are several other approaches considering external knowledge [14, 16, 25], which are described in the following.

3.3.2 Data-Driven Approaches

Linear and non-linear models. Air quality patterns have been considered in previous studies as linear or nonlinear patterns. For example, looking at air quality as a linear pattern, Donnelly et al. [16] applied a multiple linear regression model, in combination with a time series model and a nonparametric kernel regression model, to prepare a 2-day-ahead prediction of Nitrogen Dioxide (NO_2) concentration. The model has been applied both in an urban and a rural site, showing a better performance in the former where concentrations were very high. In case of major changes in the monitored emission source, the model would require undergoing a calibration process. Similarly, auto regression moving average can be considered as one of the other widely applied models to deal with air quality as linear patterns. Kadilar et al. [113] applied a seasonal

Autoregressive Integrated Moving Average (SARIMA) model to predict Sulphur Dioxide (SO_2) levels. The obtained prediction results show a discrete confidence before a manual selection of the model parameters. Kumar and Jain [22] adapt a stationary stochastic autoregressive moving average modelling approach to forecast concentration of several air pollutants in Delhi, India, obtaining satisfactory results in short-term air quality prediction. The ability of different nonlinear techniques, including multilayer perceptron network (MLPN), generalized regression neural network (GRNN), radial basis function network (RBFN), and multivariate polynomial regression was compared to partial least-square regression as linear method to predict air quality in Lucknow city, India [114]. The performed sensitivity analysis on nonlinear techniques revealed that particulate matter (PM) was the most influencing parameter compared to other air pollutants, including SO_2 and NO_2. Although all of the studied models do not consider missing or corrupted data, it has been shown that nonlinear techniques, and mostly ANN, could relatively better predict air pollutants than partial least square regression, and thus indicating these models as more suitable for the task.

Statistical learning approaches. In the context of statistical-based learning approaches, the work of Liu et al [24] proposes a new model of collaborative forecasting using Support Vector Regression (SVR) for urban Air Quality Index (AQI) prediction in China. In addition to few air pollutants concentration, including PM2.5, SO_2 and NO_2, they also considered minimum and maximum temperature, weather, wind direction and power as variables in the prediction. The approach is collaborative thanks to the fact that different air quality datasets, coming from a different set of cities which are combined together, are used in the learning model. A SVR technique is also used by Kok et al. [115], in combination with a deep learning approach, while

multiple linear regression models are exploited in several other studies [16, 23] , which are discussed in the following.

Machine learning and deep learning. State-of-the-art results have been efficiently acquired by methods employing deep learning technologies, and a number of researchers have been inspired to approach air quality prediction using such techniques. Indeed, predicting and forecasting air quality can be mentioned as one of the areas in which deep learning has achieved outstanding success [110]. Also, deep learning techniques have been applied by different scholars in several topics related to air quality, including spatiotemporal interpolation, prediction and feature extraction [26]. As an example of the usage of deep learning approaches, Li et al. [27] apply a stacked auto-encoder model to extract non-linear spatiotemporal regional air quality features. They use learned representations extracted from a deep learning approach to simultaneously predict all monitoring stations' air quality by using a regression model, thus successfully proposing an approach that couples deep learning and statistical methods. Very similarly, Ong et al. [111] carry out air quality prediction by applying a deep RNN as a pre-training techniques for an auto-encoder model. Another RNN is proposed by Rao et al. [25], which is based on LSTM to predict air quality, on a hourly basis, considering 12 air pollutants in Visakhapatnam, India. Qi et al. [26], instead, embed feature extraction and semi-supervised learning in different layers of their proposed deep learning scheme to interpolate, predict and select simultaneously related features in a comprehensive framework. An orthogonal approach to air quality prediction is introduced by Zhang et al. [29]: they apply CNN technique to predict crowds' movement based on spatiotemporal trajectories collected by taxi drivers in the city of Beijing, China. A framework based on RNN to interpolate missing values and predict air pollutants is presented by Fan et al. [116]; here, spatial and temporal correlations between local air quality conditions are considered in the prediction

process. A thorough comparison between a RNN model and two very accurate baseline models, considering feed forward neural network and gradient boosting decisions trees, is carried out by Fan et al. [116]. They show how the RNN model can better predict air quality than the two considered baselines. Du et al. [14] concentrate on forecasting PM2.5, as the air pollutant which has the most significant impact on human health, by employing a deep learning model. More in detail, they propose a hybrid deep learning model based on CNN and LSTM in order to extract local trend features and long temporal dependencies, respectively. Forecasting SO_2 pollution incidents is the objective of Sánchez et al. [117]. They employ two methods, namely Elman ANN and ARIMA models, and also introduce a hybrid method combining both to study the SO_2 concentration near a coal-fired power station. A RNN model for analysis and forecast of PM10 and PM2.5 is introduced by Biancofiore et al. [23]. In their work, the authors employ three different methods, including a RNN model, to analyze three years of data collected in an urban area of the Adriatic coast in Italy. In the context of machine learning, and especially deep learning approaches, a important role is played by IoT. In fact, IoT can provide data through smart interconnected devices, thus it can be represented as the inevitable component to perform smart analyses in different contexts, such as smart cities [118, 119]. In other words, (big) data provided by IoT has made air quality forecasting a research hotspot in recent years [14, 120]. As an example, Kok et al. [115] apply a LSTM technique as a particular type of RNN to predict air pollutants collected by IoT air quality sensors. They evaluate the results of deep learning with the SVR technique in predicting O_3 and NO_2 in Aarhus, Denmark, and Brasov, Romania, and show how the former technique can predict air pollutants more accurately than the latter. IoT is also the context considered by Ahmed et al. [121]. In their work, they propose an IoT enabled air quality monitoring system which employs the fog computing paradigm for data processing and resource

optimization. Their distributed fog architecture can help in filtering data and reduce the traffic to the cloud by only sending relevant data. The architecture consists of a fog layer between the cloud and the sensor layers. The fog layer receives the data from the sensor layer: this data undergo different processes, such as data cleaning and outlier detection, in order to be analyzed and optimized. Then, the selected data is received by the cloud layer which allows for long-term storage and advanced data analytic.

Our proposed approach. Air quality patterns, declined as spatiotemporal, non-linear and multivariate parameters cannot be easily and accurately predicted by conventional and linear techniques. After having examined the literature, we observe how deep learning techniques have been widely applied in the field of air quality prediction. Neural network models, such as RNN, can better predict the air quality. This is particularly true while using LSTM techniques, which provide the potential of better considering both long- and short-term historical knowledge. Plus, the integration of air quality IoT sensors provide the potential of monitoring air quality with finer granularity at a low-cost. It is worth pointing out that, in this study, the proposed approach aims at providing a preprocessing engine capable to add value to the missing or inconsistent data collected by air quality IoT sensors. In fact, to the best of our knowledge, such aspect has not been considered in depth within the related literature. To position our approach among the most related ones, we focus on several comparison dimensions, namely *(i)* the capability of handling missing and/or corrupted data, *(ii)* the capability of handling major changes in temporal data, *(iii)* the presence of a data preprocessing phase, *(iv)* the exploitation of additional knowledge, such as weather information, and *(v)* whether the considered approach employs a mixed architecture, such as cloud- and edge-based one, or neither Table 3.1 summarizes the comparison along the described dimensions.

Table 3.1. Overview of most related approach along different properties and comparison with our proposed approach

	Capability of handling missing and/or corrupted data	Capability of handling major changes in data	Presence of a data preprocessing phase	Exploitation of additional knowledge	Usage of a mixed architecture
Our approach	✓	✓	✓	✓	✓
[30]	-	-	-	✓	-
[28]	-	✓	-	✓	-
[16]	-	-	-	✓	-
[114]	-	-	✓	-	-
[24]	-	-	✓	-	-
[25]	-	-	-	✓	-
[116]	✓	-	✓	-	-
[14]	-	-	-	✓	-
[121]	✓	-	✓	-	✓

3.4 Methodology

The final goal of this study, as depicted in Figure 3.1, is to predict the PM2.5 concentration in the atmosphere by applying deep learning on air quality IoT sensors. In this study, the most challenging problem is dealing with inconsistent data provided by IoT sensors. The term "inconsistent" refers to the fact that the actual time series from IoT sensors are generally incomplete, thus performing predictions on low-quality data will provide poor quality results. Hence, data preprocessing comprising smoothing noisy data and filling up missing data should be of equal importance and sometimes more important than providing a prediction model. The upcoming section discusses fundamental preprocessing techniques to deal with noisy and missing data and the characteristics of the prediction model. After this stage, the preprocessed data will be combined with meteorological data and feed into the LSTM technique representing the multivariate deep learning-based prediction model.

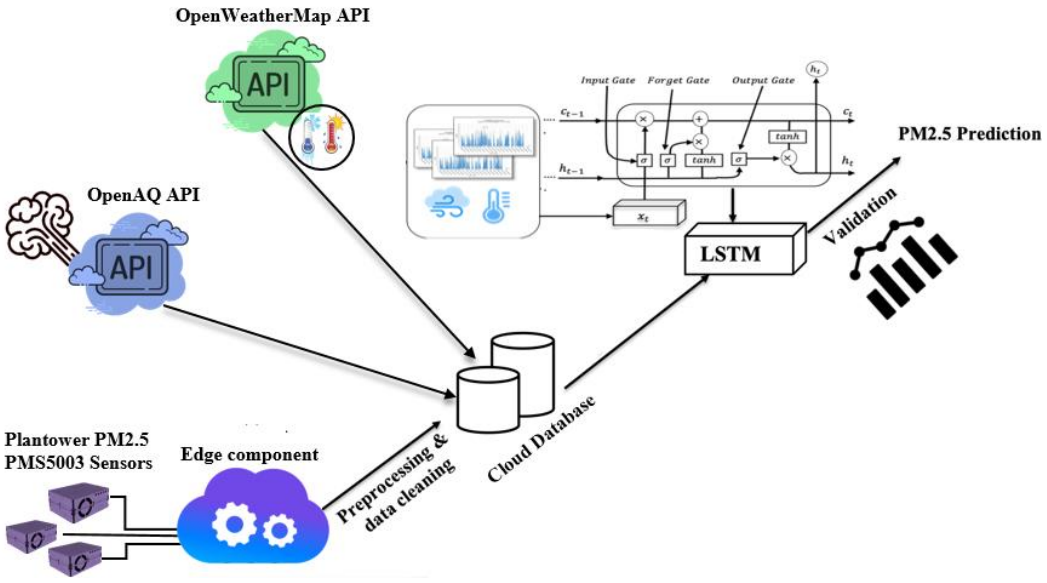


Figure 3.1. The proposed architecture for deep learning based PM2.5 prediction based on air quality IoT sensors

The overall architecture of the proposed model is as follows. Sensors are physically distributed on the monitored area. Most of the sensors are fixed on static monitoring stations; however some of them can be placed on mobile devices. Each static sensor is permanently associated with an edge node; mobile sensors are dynamically associated with an edge node based on spatial proximity. Each sensor is associated with exactly one edge node; there are no edge nodes without fixed sensors associated with them. Sensors send raw data to the edge node which is in charge of both preprocessing (aggregating and cleaning) the raw data and to deal with missing data. In order to fill missing data both temporal and spatial information are taken into account. Corrected and compacted data are then sent to the cloud in order to apply the deep learning-based prediction tasks.

In the following subsections we describe the behavior of the main components of the proposed framework; in order to help the reader, the symbols mentioned in this paper and their meanings are summarized in Table 3.2.

Table 3.2. Notation and symbols used for the description of the methodology

Symbol	Meaning
S_i	i-th PM2.5 sensor in the platform
$v_{h,j}$	j-th PM2.5 reading from a sensor within hour h
$\bar{p}(h)$	aggregated value for PM2.5 readings for hour h
T_i	Time series of aggregated values from sensor S_i
T_l	a time instant (an hour) relative to a data point in a time series
$T_i[tl]$	the real data point at t_l of the time series T_i
$T_i^*[tl]$	the estimated data point at tl for the time series T_i
$DT\ W_{ij}$	Dynamic Time Warping distance between T_i and T_j
m_k	Medoid of the cluster k
$MedAVG_{i,l}$	Temporal-based estimation for $T_i[t_l]$
$kNNED_i$	k-Nearest Neighbor set for Ti
$kEDAVG_{i,l}$	Spatial-based estimation for $T_i[t_l]$

3.4.1 Edge-Based Computation

The general architecture of the tasks carried out on each edge node is shown inFigure 3.2. Each sensor sends its raw data to its edge component. These are first elaborated by the preprocessing engine. In fact, every sensor measures the value of PM2.5 with a frequency of almost five minutes, which is a too detailed information. This module, then, is in charge of generating one data point for each hour. This is coherent with the frequency of data available for meteorological parameters [122, 123] (such as wind direction and speed, air temperature, air pressure, and air

humidity) we want to combine with sensed data for the prediction activity. The behavior of this module is detailed in Section 3.4.1.1.

After this step, two scenarios are possible: *(i)* the data from the sensor in the last hour is available, *(ii)* the data from the sensor in the last hour is missing. If the data is available, it is directly sent to the cloud, without further elaboration; however, as it will be clarified next, available data is anyway exploited to improve the accuracy of the prediction module for missing data.

On the other hand, if the data is missing, a prediction step to fill in the missing value is carried out. Incompleteness can be considered as the commonplace property of extensive real-world data set provided by IoT sensors. There are many possible reasons for incomplete data collected by air quality sensors, such as faulty IoT sensors and disturbance in data exchange infrastructure. As an example, consider the situation in which there is no air quality measurement for a time interval of five hours. The non-linear nature of air quality data makes it difficult to use a simple measure of central tendencies, such as mean or median, to fill in the missing data. Using data collected by similar sensors will be a more realistic and meaningful estimation to fill in missing data as an alternative [16, 17].

Moreover, it has been widely approved that air quality is a spatiotemporal parameter [15-17, 27]. As a result, time series provided by different air quality sensors will be temporally similar, and air quality IoT sensors will also be spatially similar to each other. As a consequence, we consider both temporal and spatial information to fill in missing values. The temporal-based estimation process is described in Section 3.4.1.2, whereas the spatial-based estimation is described in Section 3.4.1.3. Both estimations are finally combined with a weighted average (see Section 3.4.1.4); weights are dynamically set on the basis of the latest available data from the sensor. In fact, when the data from the sensor is available, a prediction for the value is carried out

anyway and the accuracy of the prediction model is assessed, in order to determine the reliability of the temporal- and spatial-based estimation. In particular, the Temporal-prediction accuracy (resp., the Spatial-prediction accuracy) is computed as the absolute difference D_t (resp., D_s) between the actual and the predicted value. The formula used to update the weighs is detailed in Section 3.4.1.5. The finally estimated value is then sent to the cloud for further elaborations.

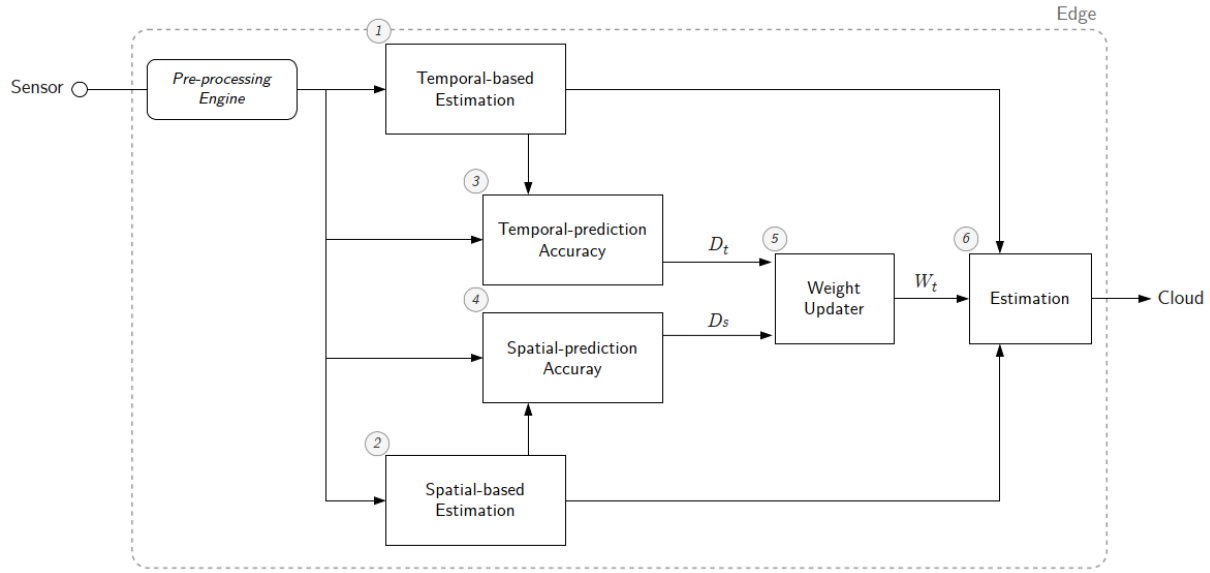


Figure 3.2. The proposed architecture for the edge component

3.4.1.1 Preprocessing Engine

Air quality prediction can be categorized as those applications in which false-negative prediction is much more critical than false-positive prediction. To support this opinion, consider a situation in which the air quality is predicted to be safe in a specific region and a patient with asthma plans to go there based on the provided air quality prediction. The importance of predicting air quality as a dangerous status is much higher than predicting it as a safe status. As a consequence, in this study, three different widely accepted techniques, namely average, maximum, and weighted

average, can be exploited to aggregate the data, depending on the level of safety one wants to apply.

The average technique uses Equation 1 to calculate the average value as an aggregation of PM2.5 each hour h.

$$\bar{p}(h) = \frac{1}{n}\sum_{j=1}^{n} v_{h,j} \quad (1)$$

Here, n is the number of PM2.5 sensor readings in hour h, and $v_{h,j}$ is the j-th reading from the sensor in hour h. Observe that, if $n = 0$ no readings are available, and $\bar{p}(h)$ is undefined. Instead, if the maximum value of PM2.5 is considered as the aggregated value for each hour, then $\bar{p}(h)$ can be defined as in Equation 2:

$$\bar{p}(h) = \underset{j=1\ldots n}{max}\{v_{h,j}\} \quad (2)$$

Finally, the weighted average allows to smooth the impact of noisy values and consider more essential values. The weighted average technique uses Equation 3 to calculate the weighted average value as an aggregation of PM2.5 each hour.

$$\bar{p}(h) = \frac{\sum_{j=1}^{n} v_{h,j}^2}{\sum_{j=1}^{n} v_{h,j}} \quad (3)$$

Using the maximum value of PM2.5 in each hour can be the safest choice for dangerous situations; however, it can also amplify the impact of noisy data. Although there are some techniques to detect noisy values, such as clustering methods, applying such methods to detect noisy values each hour significantly increases the preprocessing engine's computing cost. Since the weighted average allows to smooth the impact of noisy values with low computational resources, in the following we will assume to exploit this method to aggregate the data.

Each hour h is then considered as a time instant t_h for the subsequent computations. In particular, given a sensor S_i, the time series T_i for S_i is built upon the aggregated values for each t_h such that $T_i[t_h] = \bar{p}(h)$. If $\bar{p}(h)$ is undefined, also $Ti[th]$ will be undefined.

3.4.1.2 Temporal-Based Estimation for Missing Values

The proposed approach first computes the distances between all the pairs of time series built from the sensors associated with the edge node. Computed distances form the basis for clustering most similar groups of sensors. Based on this clustering step, and on the identification of a medoid for the cluster, the temporal-based estimation for a missing value is obtained as an average mean of the values available from the time series of similar sensors; the weight, in this case, is based on the distance of each data point from the medoid.

It is worth pointing out that the computation of distances and clusters is carried out at the edge node, and not for each sensor; a periodic update of clusters, and related distances, is carried out once a day. More specifically, the edge node records the time series of all assigned sensors for the previous day. Note that each time series represents 24 aggregated values of PM2.5 for 24 hours of the day. These data are used to compute the distance and, consequently, cluster time series for the following day. In other words, a 24-hour time window is selected to cluster similar sensors.

The distance between two time series T_1 and T_2 can be measured by different techniques, including lock-step measures (*e.g.*, Euclidean distance and other L_p norms), elastic measures (*e.g.*, Dynamic Type Warping (DTW), Longest Common Subsequences (LCSS)), threshold- and pattern-based measure [124]. DTW-based similarity measure is considered one of the mostly applied similarity measures for time series [124-127].

In the current research, the DTW measure is applied to measure the temporal similarity between time series. This measure finds the optimal alignment between two given time series by

allowing a comparison of one to many points [124-127]. The applied algorithm to calculate the similarity between two given time series T_1 and T_2 based on DTW measure is shown in Algorithm 1, and we now describe it more in detail. Algorithm 1 receives in input two time series T_1 and $T_2$of length n and m and indices $[1..n]$ and $[1..m]$ respectively, and the warping window parameter w indicating the number of points to include in the window. At line 1, it defines a $n + 1 \times m + 1$ matrix, referred within the algorithm as dtw, used to compute the distance of the two time series given in input; line 2, instead, adapts the value of the window to be greater than the difference of n and m. Initially, each element $dtw\ [i,j]$ is set to ∞, indicating that the distance between T_1 and T_2 up to indices i and j is not yet computed. In addition, the value $dtw\ [i,j]$ at each index i, and for each index j contained within the warping window is set to 0. Then, the algorithm iterates over each pair of indices (i,j): each iteration computes the DTW for the time series T_1 and T_2 up to the indices i and j. Formally, it is given by the sum of the distance between the points $T_1[i]$ and $T_2[j]$, defined as $d(T_1[i],T_2[j]) = |T_1[i] - T_2[j]|$, and the minimum distance obtained up to the pairs $(i-1,j),(i,j-1)$ and $(i-1,j-1)$. Lines 8–13 indicate these computations. Eventually, the DTW between T_1 and T_2 is $[n,m]$, that is the distance between T_1 and T_2 computed up to indices n and m.

Choosing the similarity measure plays an essential role in the quality of clustering algorithms. Based on the survey conducted in [128], DTW and Euclidean distance can be mentioned as the similarity measures with the highest popularity among researchers while also showing promising results in their usage in clustering algorithms. Both of these similarity measures consider the similarity within the shape of the time series to calculate the distance between them. However, the one-to-many mapping nature in the DTW similarity measure has

shown more appropriate results in comparison with the one-to-one mapping nature of Euclidean distance [129].

Algorithm 1: Dynamic Time Warping algorithm to compute the distance between two given time series

---

Input : Time series T_1 and T_2 of length n and m (with indices $[1..n]$ and $[1..m]$ resp.
Warping window w

Output: DTW distance between T_1 and T_2

$dtw \leftarrow [0..n, 0..m]$ matrix
$w \leftarrow max(w, |n - m|)$
for $i \leftarrow 0$ **to** n **do**
    for $j \leftarrow 0$ **to** m **do**
        $dtw[i,j] \leftarrow \infty$
    $dtw\,[0, 0] \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow max(1, i - w)$ **to** $min(m, i + w)$ **do**
        $dtw[i,j] \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow max(1, i - w)$ **to** $min(m, i + w)$ **do**
        $cost \leftarrow d(T1[i], T2[j])$ // $with\ d(a,b) = |a - b|$
        $dtw[i,j] \leftarrow cost + min($
            $dtw[i - 1, j]$,
            $dtw[i, j - 1]$,

16 $dtw[i - 1, j - 1]$

17)

18 **return** $dtw[n, m]$

In the current study, a k-medoids, with the Partitioning around Medoid (PAM) algorithm, will be applied as a technique of partitioning methods of clustering time series. In comparison with the k-means clustering technique, it will be worth mentioning that the prototype in this technique will be defined as the mean vector of objects in clusters [129]. When it comes to clustering time series, choosing the clustering prototype as the mean value of the time series is challenging and not trivial. The applied algorithm to clustering time series based on DTW similarity measure is shown in Algorithm 2. It depicts the common greedy search strategy on which PAM is based on. The algorithm takes as input a positive integer number k, that is the number of clusters, and an array M of pairwise DTW distances computed between the time series. The strategy is subdivided in two phases. In the first one, k time series are arbitrarily selected among all of the available time series as the medoids. The second phase consists of several steps. Let $m_1 \dots m_k$ be the medoids of each cluster respectively. In the first step, each remaining time series is associated with the cluster with the lowest DTW distance to the cluster medoid m_i. The second step consists in computing the total DTW distance S between the time series of the cluster and the clusters medoid m_i, in order to define the cost of having mi as a medoid. Then, a non medoid time series m_i^* is arbitrarily selected in each cluster, and the total DTW distance D between the time series of the cluster and the new candidate medoid $m_i^*$is computed. If $D < S$, then the medoids are swapped, and the second phase of Algorithm 2 is repeated until no such a swap occurs. Eventually, the algorithm outputs the obtained k clusters.

Now, given a time series T_i and a data point t_l, in order to estimate (on a temporal basis) $T_i[t_l]$, the following computations are carried out. Let us consider the results of k-medoids as expressed in Equation 4, in which m^k is the medoid of the k^{th} cluster, n is the number of time series in the cluster, and DTW_{ik} is the DTW distance between time series T_i and cluster medoid m_k.

Algorithm 2: A k-medoids algorithm (PAM) for partitioning time series based on medoid or central objects and using DTW similarity measure

---

Input : A number k of clusters; an array M of pairwise DTW distance between time series

Output: A set of clusters C with each cluster members

1 Arbitrarily choose k time series in M as the initial representative medoids mi

2 **repeat**

3 Assign each remaining time series to the cluster with the lowest DTW distance to the cluster medoids m_i

4 Compute the total DTW distance S from each of cluster time series to the cluster medoid mi

5 Randomly select a non medoid time series in each cluster as m_i^*

6 Compute the total DTW distance D of swapping new medoid time series m_i^*

7 **if** $D < S$ **then**

8 Swap mi with mi∗

9 **until** no swaps occurred

10 **Return** the obtained clusters

$$m_k = \{(T_1, DTW_{1k}), (T_2, DTW_{2k}), \dots, (T_i, DTW_{ik}), \dots, (T_n, DTW_{nk})\} \quad (4)$$

The weighted average technique is used as Equation 5 to calculate the weighted average $MedAVG_{i,l}$ extracted from data obtained from k-medoids clustering technique. This corresponds to the temporal-based estimation for the missing data at time t_l that is (5). Where DTW_{uk} is the u-th computed DTW in mk.

$$\text{Med}AVG_{i,l} = \frac{\sum_{u=1}^{n} \frac{1}{DTW_{uk}} T_u[t_l]}{\sum_{u=1}^{n} \frac{1}{DTW_{uk}}} \quad (5)$$

3.4.1.3 Spatial-Based Estimation for Missing Values

In addition to temporal similarity, it has been widely approved that spatial relation plays an inevitable role in analyzing air quality [28, 130-132]. We believe that spatially closer air quality IoT sensors will have more similar measurements. Based on this assumption, we applied k-Nearest Neighbor based on Euclidean Distance (kNNED) to identify the k most spatially closest air quality IoT sensors. The Euclidean distance between the location of each sensor is calculated using their geographical coordinates.

The result of kNNED for the sensor S_i is shown in Equation 6, in which ED_{ji} is the Euclidean distance between sensors S_i and S_j in the kNNED set and each T_j is the time series produced by sensor S_j; m is the number of sensors in the kNNED set.

$$kNNED_i = \{(T_1, ED_{1i}), (T_2, ED_{2i}), \dots, (T_j, ED_{ji}), \dots, (T_m, ED_{mi})\} \quad (6)$$

Now, given a time series T_i, the value of the data point $T_i[t_l]$ for T_i at the time instant t_l is estimated on a spatial basis as (7) where ED_{wi} is the w-th computed ED in $kNNED_i$.

$$kEDAVG_{i,l} = \frac{\sum_{w=1}^{m} \frac{1}{ED_{wi}} T_w[t_l]}{\sum_{w=1}^{m} \frac{1}{ED_{wt}}} \quad (7)$$

3.4.1.4 Weight Updater

As it will be formalized in the next section, when the data point $T_i[t_l]$ of the time series T_i is missing, its estimation is defined as a weighted mean of the temporal- and spatial-based estimations for $T_i[t_l]$. In order to improve the quality of predictions, each time series T_i is associated with its specific weight wT_i. wT_i is firstly initialized to 0.5 for all time series T_i. Then, the value of wT_i is dynamically determined by the edge component each time a data point $T_i[t_l]$ of the time series T_i is available; as a consequence, the Weight Updater module is activated only when $T_i[t_l]$ is actually available, otherwise the last available value for wT_i is transferred to the Estimation module. Recall that, when $T_i[t_l]$ is available, the Estimation module is not activated and $T_i[t_l]$ is directly sent to the cloud as the value available for the sensor S_i at the time instant t_l.

Observe that, when $T_i[t_l]$ is available, both the actual value for $T_i[t_l]$ and its temporal-based and spatial-based estimations are available. As a consequence, it is possible to precisely evaluate how well both estimation modules work; the weight is then set in such a way to give more importance to the best performing module. Formally, given D_{t_l} as the absolute difference between $T_i[t_l]$ and the value estimated by the temporal-based estimation module, and given D_{s_l} as the absolute difference between $T_i[t_l]$ and the value estimated by the spatial- based estimation module, the weight wT_i for the time series T_i is computed as follows:

$$w_{T_i} = \begin{cases} \frac{1}{2} - \frac{D_{t_l} - D_{s_2}}{2D_{t_l}} & \text{if } D_{t_l} < D_{s_l} \\ \frac{1}{2} + \frac{D_{s_l} - D_{t_l}}{2D_{s_l}} & \text{if } D_{s_l} > D_{t_l} \end{cases} \quad (8)$$

The ratio underlying this formula is as follows. Considering wT_i as the weight (a value in the range [0..1]) to be assigned to the temporal-based estimation, the weight for the spatial-based estimation will be $1 - wT_i$. If $D_{t_l} = D_{s_l}$, which means either that D_{t_l} and $D_{s_l}$are 0 (the prediction is perfect) or that the error in the prediction is equal, both temporal- and spatial-based estimations will have the same weight equal to $\frac{1}{2}$. If D_{t_l} is greater than D_{s_l}, then the spatial-based estimation is more reliable and, consequently it will be assigned a greater weight. On the contrary, if $D_{s_l} > D_{t_l}$ the temporal-based estimation is more reliable and, consequently, its weight will be higher than the one for the spatial-based estimation.

3.4.1.5 Final Estimation

The final estimation $T_i^*[t_l]$ for the data point t_l of the time series T_i is finally given by the weighted average between the temporal- and the spatial- based estimations. This is formalized as follows:

$$T_i^*[t_l] = w_{T_i} * \text{Med}AVG_{i,l} + \left(1 - w_{T_i}\right) * kEDAVG_{i,l} \quad (9)$$

$T_i^*[t_l]$ is then sent to the cloud as the value available for the sensor i at the time instant t_l. Recall that, if the actual sensed data $T_i[t_l]$ is available for sensor S_i, $T_i[t_l]$ is directly sent to the cloud as the value available for S_i at the time instant t_l.

3.4.2 Deep Learning-Based Air Quality Estimation

Air quality is temporal, thus an efficient predicting method should consider both long- and short-term dependencies as historical information [30]. The vanishing problem can be considered

as the most significant disadvantage of RNNs to capture long-term historical information. In comparison, LSTM as a type of RNNs can correctly address this problem by defining a stable model to memorize the important long-term historical information. The air quality prediction can be calculated from the previous status of recurrent neurons based on continuous time series of air quality in order to model the long-range dependencies.

The LSTM neuron structure is typically modelled by three different gates responsible for deciding about the importance of short- and long-term dependencies between continuous time series of air quality. As seen in Equation 10, the input gate is defined to specify how much of the input information should be entered into the neuron and how much to forget. The forget gate is responsible for the portion of necessary historical information that came from the previous neurons to be entered into the current neuron. Finally, the output gate plays a role in deciding the necessary information that can be useful for the next neuron.

$$\begin{aligned} i_t &= \sigma(x_t\omega_{xi} + h_{t-1}\omega_{hi} + c_{t-1}\omega_{ci} + b_i) \\ f_t &= \sigma\left(x_t\omega_{xf} + h_{t-1}\omega_{hf} + c_{t-1}\omega_{cf} + b_f\right) \\ o_t &= \sigma(x_t\omega_{xo} + h_{t-1}\omega_{ho} + c_{t-1}\omega_{co} + b_o) \\ z_t &= \tanh(x_t\omega_{xz} + h_{t-1}\omega_{hz} + b_z) \end{aligned} \tag{10}$$

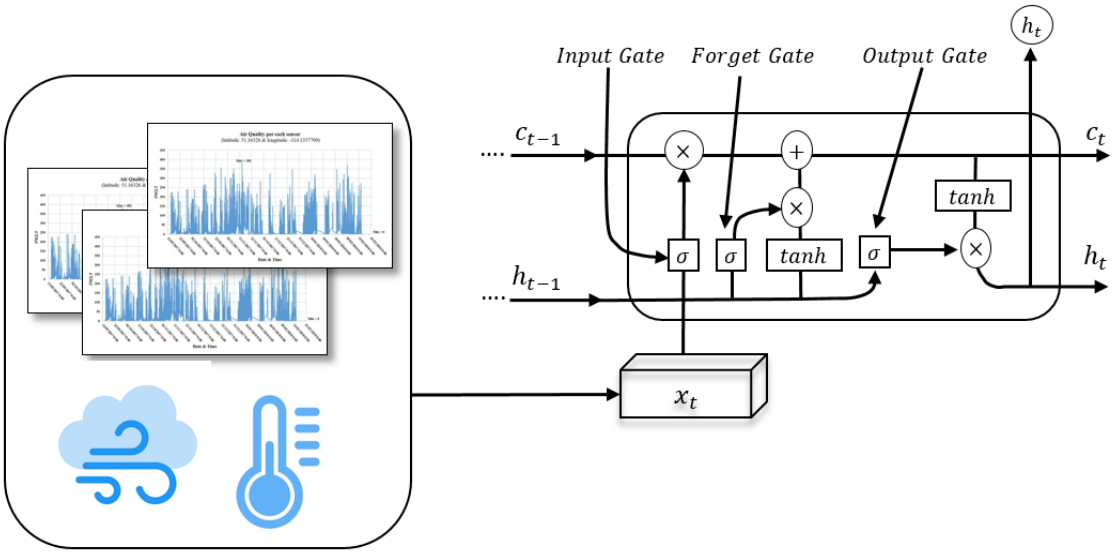


Figure 3.3. Overview of the structure of LSTM for air quality prediction

Where i_t, f_t, o_t and z_t are input gate, forget gate, output gate, and candid update respectively. ω_i, ω_f , ω_o and ω_z are the weights matrices connecting the input vector x_t or output vector h_{t-1} to the three gates and block input. b_i, b_f , b_o and b_z are the bias vectors. σ is the logistic sigmoid function which normalizes the input value in the range of $[0, 1]$ and similarly, $tan\ h$ represents hyperbolic tangent function to normalize input value in the range of $[-1, +1]$. Based on defined gates, the LSTM block can be defined as follows:

$$\begin{aligned} y_n &= \mathrm{LSTM}(x_{1:n}) \\ s_j &= R_{LSTM}(s_{j-1}, x_j) : [c_j, h_j] \\ c_j &= f \odot c_{j-1} + i \odot z \\ h_j &= o_j \odot \tanh(c_j) \\ y_j &= o_{LSTM}(s_j) \end{aligned} \tag{11}$$

Where c_j and h_j are the outputs of recurrent functions and s_j is the status of neuron in the LSTM block. It is noticeable that the dot product is shown by the $\odot$ symbol. The structure of the LSTM neuron to predict air quality is shown in Figure 3.3.

3.5 Implementation and Testing

In this section we address the implementation, deployment and testing of the proposed approach on a real use case. In particular, in Section 3.5.1 we describe the real use case and the dataset exploited in our experimental design; Section 3.5.2 focuses on the data preprocessing steps, whereas Section 3.5.3 introduces the details for the prediction model tailored to the use case. Finally, the evaluation of the prediction model on both unprocessed and preprocessed datasets, along with a discussion on obtained results is provided in Section 3.5.4.

3.5.1 Data

The data used in this study has been collected by 40 air quality stations distributed in Calgary, Canada from September 2017 to March 2018. In each station, a Plantower PM2.5 PMS5003 sensor is used to measure the concentration of PM2.5 with declared accuracy of ± 10 micro gram per quibic meters ($\mu g/m^3$) [133] every five minutes. In total, 1,048,576 PM2.5 data records are collected from all 40 air quality IoT sensors. The spatial distribution of air quality IoT sensors is represented in Figure 3.4. The Open Geospatial Consortium (OGC) developed SensorThings Application Programming Interface (STA API)[1] to provide an interoperable framework for the interconnection of IoT devices, data, and applications on the Web [134, 135]. This paper utilizes the OGC STA data model to provide interoperability among air quality sensors. The OGC STA data model for a single air quality sensor used in this research is illustrated in Figure 3.5.

[1] https://developers.sensorup.com/docs/#introduction

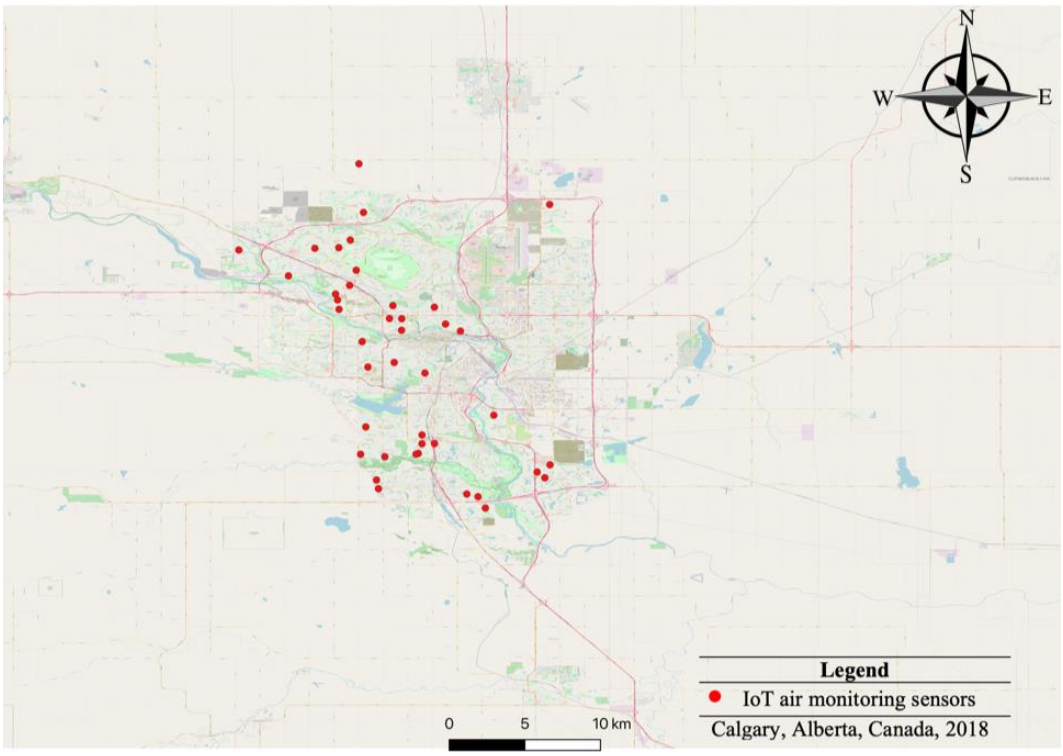


Figure 3.4.The spatial distribution of IoT air monitoring sensors in Calgary, Canada

For the experimentation purposes, edge nodes have been simulated with a Raspberry Pi 3 emulator, which is sufficiently lightweight but provides sufficient computational power for the preprocessing tasks included in our approach. As seen in

Table *3.3*, we considered four different scenarios using different numbers of edge nodes. In the first scenario, we considered a single edge node and assigned all 40 IoT sensors to this node. In this case, k is assumed to be 40 and five for spatial-based and temporal-based estimation approaches, respectively. Considering this setting, all 40 IoT sensors contributed to the spatial-based estimation approach. Also, the sensors clustered into five clusters using DTW similarity measure and k-medoid clustering in the proposed temporal-based estimation approach. In the remaining three scenarios, the number of edge nodes increased and, consequently, the number of assigned sensors per each edge node and the value of k for spatial-based and temporal-based

estimation approaches decreased; values for k have been empirically set. In the first scenario, the execution time to estimate a missing value was 154 milliseconds. The time taken for the second, third, and fourth scenarios were respectively 98, 38, and 17 milliseconds. So, it is clear that increasing the number of edge nodes from a single edge node to five edge nodes could decrease the execution time for data preprocessing. Recall that the calculation of DTW similarities between time series and k-medoid clustering technique is run only once per day, and all that it takes is about 350 milliseconds for the first scenario.

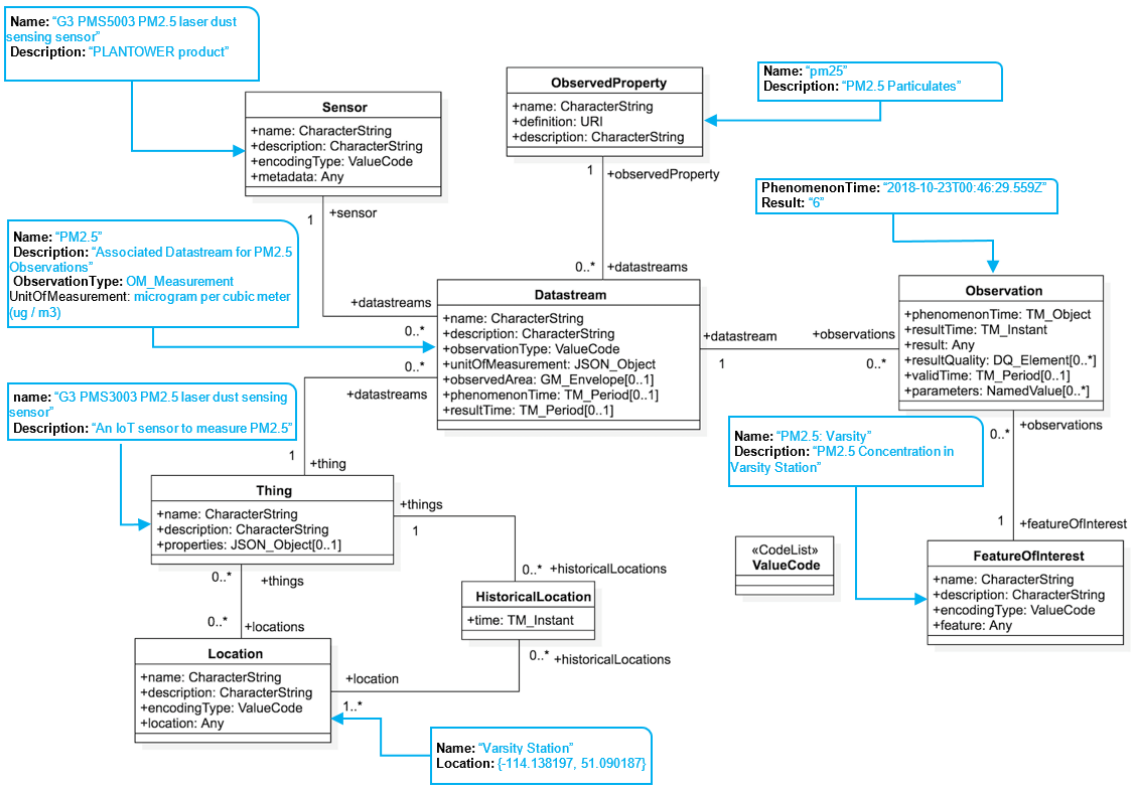


Figure 3.5. The UML diagram of OGC SensorThings API for a single air quality IoT monitoring station

Table 3.3. Network setting considering four different number of edge nodes

Number of edge nodes	Number of sensors (per node)	k value in kNNED	k value in k-medoid
1	40	40	5
2	20	20	3

4	10	10	2
5	8	8	1

In this research, OpenAQ platform API[1] and OpenWeatherMap API[2] are used to provide air quality and weather data respectively. In more detail, 13 dimensions of data including PM2.5, PM10, SO_2, NO_2, O_3, CO, temperature, pressure, humidity, wind speed, wind direction, time, and location are used overall as the input layer of the LSTM prediction model. Among these dimensions, PM2.5, time, and location are provided by IoT air quality sensors; PM10, SO_2, NO_2, O_3, and CO are provided by OpenAQ and the remaining dimensions are provided by OpenWeatherMap API. Table 3.4 summarizes the data fields that have been used in this research.

Table 3.4.Dimensions used for air quality prediction and corresponding providers

Data	Data provider	Type of measurement/data
PM2.5	IoT sensor	$\mu g/m^3$
Date Time	IoT sensor	timestamp
Location	IoT sensor	[latitude, longitude]
PM10, NO_2, SO_2, O_3, CO	OpenAQ API	$\mu g/m^3$
Temperature	OpenWeatherMap API	℃
Pressure	OpenWeatherMap API	kPa
Humidity	OpenWeatherMap API	g/m^3
Wind Speed	OpenWeatherMap API	m/s
Wind Direction	OpenWeatherMap API	deg

[1] https://openaq.org/
[2] https://openweathermap.org/api

3.5.2 Data Preparation

In our experiment, 1,048,576 measurements were transmitted from all 40 air quality IoT sensors. Figure 3.6 shows the temporal PM2.5 concentration in one of the IoT air quality stations. As explained in Section 3.4.1, the edge component is responsible for data preprocessing (aggregation and prediction of missing values). The PM2.5 values collected by air quality IoT sensors are aggregated for each hour to be consistent with air quality and meteorological data provided on an hourly basis from OpenAQ and OpenWeatherMap APIs. As an example of data aggregation, all of three mentioned aggregation techniques (as described in Section 3.4.1.1) are applied on raw PM2.5 measurements of an air quality IoT sensor, and the obtained results are depicted in Figure 3.7. To investigate the impact of predicting missing values on the performance of LSTM models, both unprocessed (*i.e.*, containing missing values) and preprocessed datasets are considered in our experimental design. Although both datasets consist of all 13 dimensions, there are 137,383 and 163,200 (170 days × 24 hours × 40 sensors) data per each dimension in un-processed and preprocessed datasets, respectively. As a consequence, 25,817 PM2.5 missing values have been estimated with the approach presented in Section 3.4.1.5. Figure 3.8 indicates the percentage of missing values per air quality sensor in our real-world experiment.

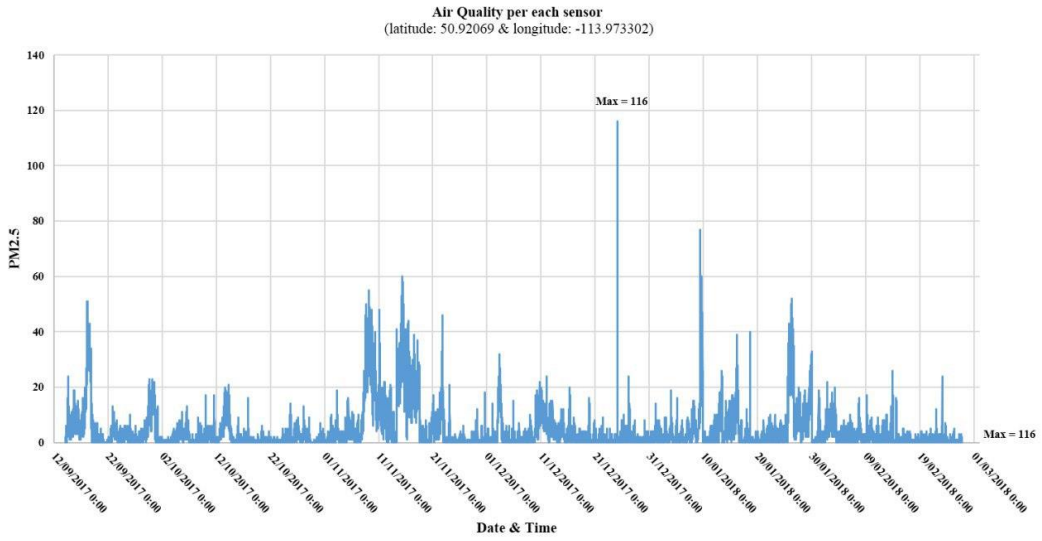


Figure 3.6. Representation of temporal PM2.5 concentration measured by an air quality IoT sensor from September 2017 to March 2018 in Calgary, Canada

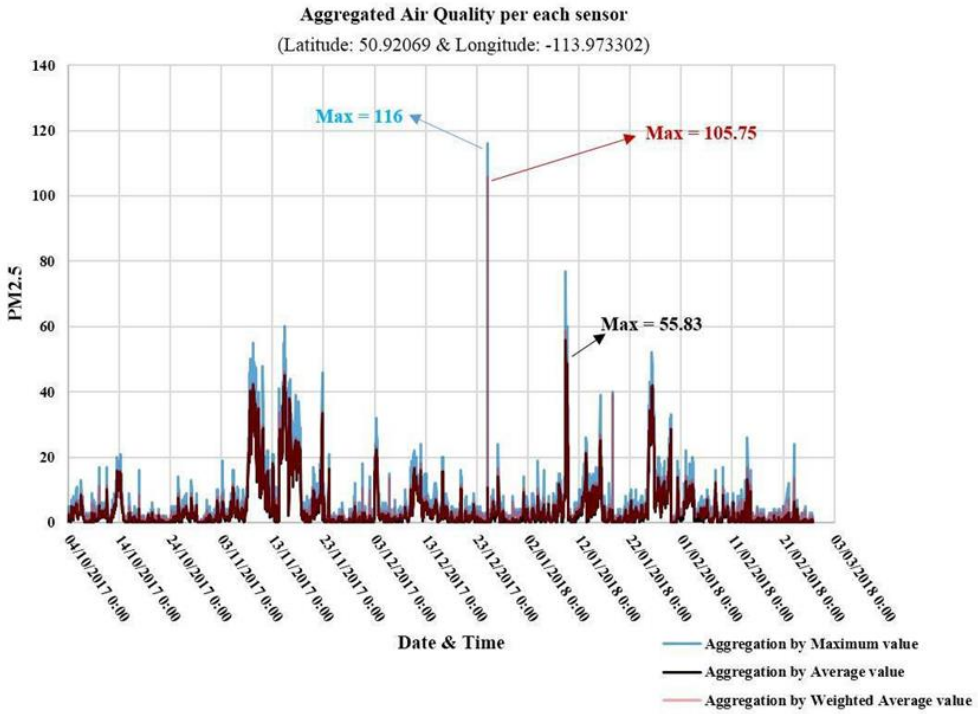


Figure 3.7. A representation of three aggregation techniques applied on raw PM2.5 measurements to provide one PM2.5 data point for each hour

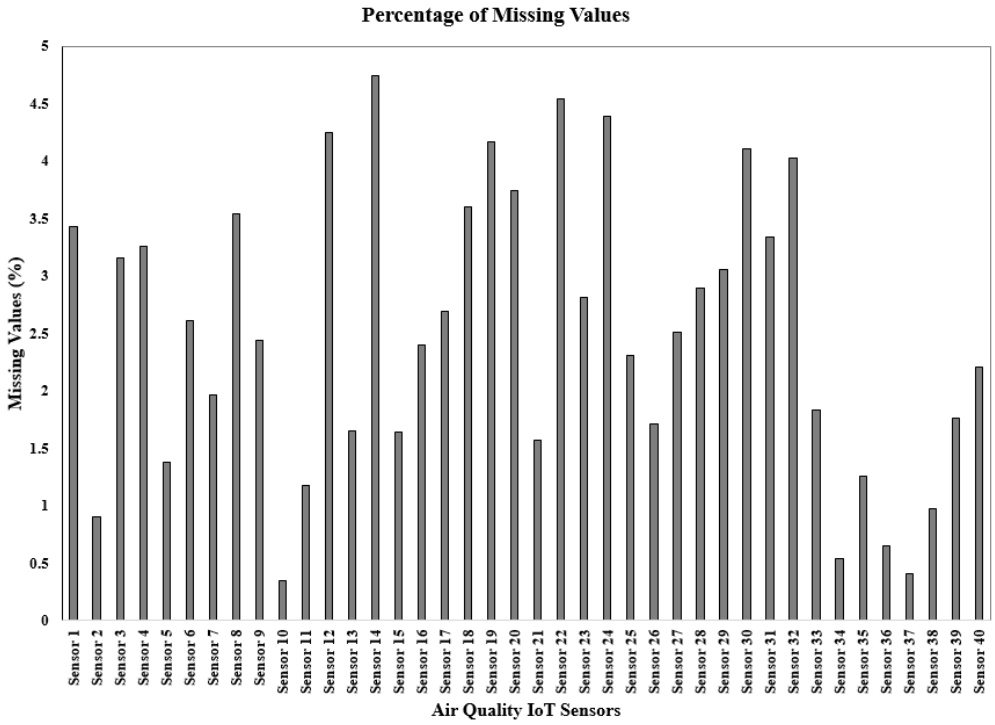


Figure 3.8. A representation of existing missing values per air quality IoT sensor

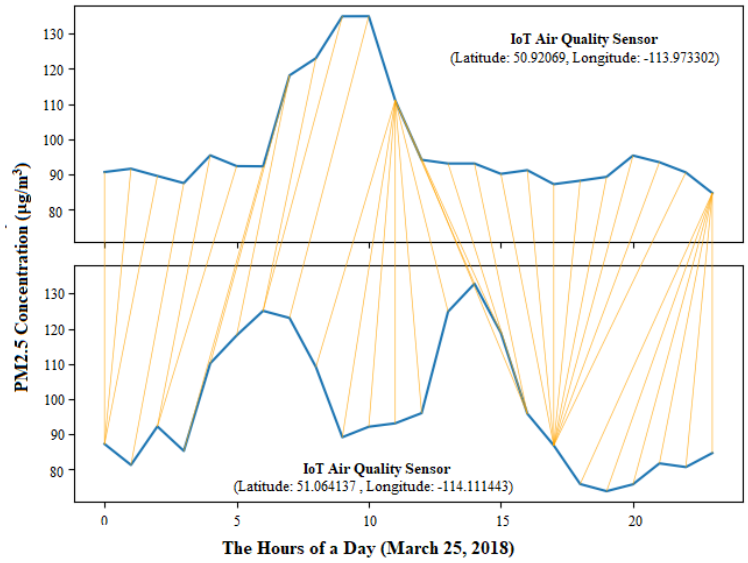
As illustrated in Section 3.4.1, the DTW technique is employed within the edge component to measure the temporal similarity between IoT sensors. For example, the optimal alignment and DTW distance between two time series of PM2.5 concentration provided by two IoT sensors are shown in Figure 3.9. In our experiment, the warping window size is assumed to be a constant value of 30% for the DTW approach. A DTW similarity matrix is derived from applying the DTW approach to measuring the temporal similarity between IoT sensors' time series. Just to show an example, in the following we present the result of using the DTW technique in an edge component with $n = 5$ IoT sensors:

$$DTW \text{ distances} = \begin{bmatrix} - & 16.78 & 43.26 & 51.86 & 60.87 \\ - & - & 59.77 & 28.82 & 34.12 \\ - & - & - & 44.69 & 29.96 \\ - & - & - & - & 53.06 \\ - & - & - & - & - \end{bmatrix}$$

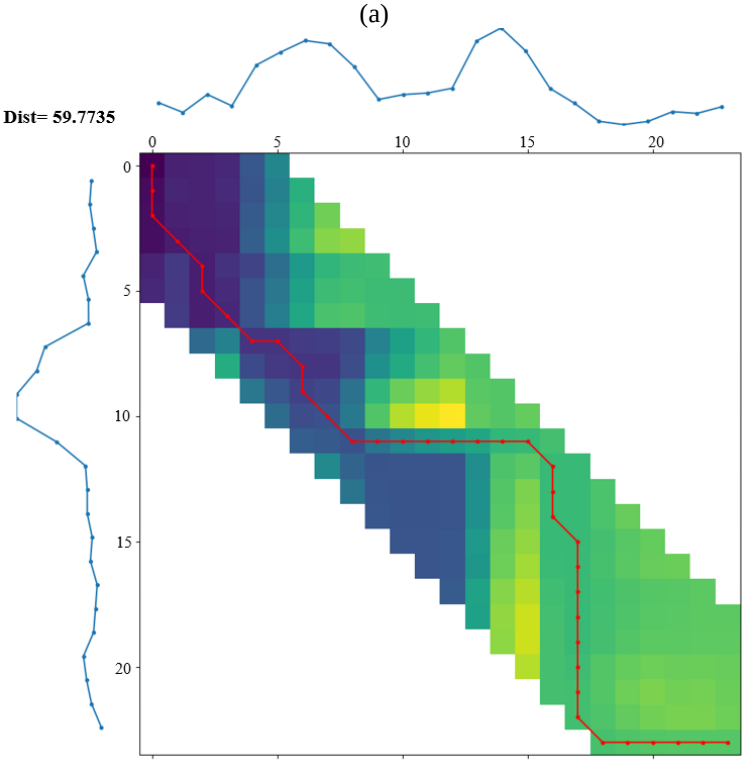
Table 3.5. Achieved results of applying 5-medoids clustering algorithm based on DTW distance on 40 time series provided by 40 IoT sensors, where dm indicates the distance between the cluster member and the medoid of the same cluster

Cluster Medoids									
Sensor4		Sensor13		Sensor33		Sensor36		Sensor39	
Cluster member	dm	Cluster member	dm	Cluster member	dm	Cluster member	dm	Cluster member	dm
Sensor2	28.81	Sensor1	41.86	Sensor24	13.96	Sensor3	23.80	Sensor9	46.57
Sensor5	10.06	Sensor6	38.55	Sensor29	14.55	Sensor8	28.64	Sensor16	46.04
Sensor11	37.05	Sensor7	32.84	Sensor32	16.57	Sensor18	21.07	Sensor31	52.75
Sensor12	63.77	Sensor10	40.13	Sensor34	10.98	Sensor20	19.70	Sensor35	62.78
Sensor14	33.69	Sensor17	38.89	Sensor38	18.92	Sensor22	24.90	Sensor37	67.14
Sensor15	49.90	Sensor19	40.52	Sensor40	11.28	Sensor23	40.05		
Sensor28	50.38	Sensor21	45.43			Sensor26	12.21		
						Sensor27	37.60		
						Sensor30	10.65		

After calculating the DTW distance matrix, a k-medoids algorithm depicted in Algorithm 2 is applied to assign time series to each cluster medoid based on the higher similarity between cluster members and the lower similarity between members of different clusters. The achieved results after applying a 5-medoids clustering algorithm on preprocessed dataset for the first scenario, based on DTW distances between the time series of the use case, are summarized in Table 3.5.



(a)



(b)

Figure 3.9. Application of DTW on time series: (a) Warping result for the investigated time series, (b) Optimal alignment between time series using a warping window size of 30%

3.5.3 Prediction Model

In our experiment, a LSTM model for multivariate PM2.5 prediction is developed with Tensorflow and Keras deep learning libraries. The architecture of the implemented LSTM is shown inFigure 3.3. The objective of the developed prediction model is to predict the PM2.5 concentration for the next hour based on past air quality and meteorological data. For both unprocessed and preprocessed datasets, an hourly 13-dimension data is entered (as X_t) to the hidden layer at each time point.

As seen in Figure 3.3, the hidden layer consists of three gates, including forget, input, and output gates, whose input and output need to be determined separately. Starting from the forget gate, the input $[X_t]_{13\times1}$, as the output of preprocessing engine, is multiplied by the weight matrix $\left[W_f\right]_{1\times13}$ of the forget gate. The output of the developed LSTM model is $[PM2.5_t]_{1\times1}$. It can be inferred from Equation 10 that the hidden state matrix should be defined as $[h_t]_{1\times1}$ and then multiplied by the weight matrix $\left[U_f\right]_{1\times1}$ to get the forget gate $[f_t]_{1\times1}$. In the next step, the Sigmoid function (σ) will be applied on the forget gate to convert the output as a value between zero to one. The closer value to zero means the more forget the state of the neuron, while the closer value to one means the more about the neuron state should be remembered to be used in the following neurons. To calculate the input gate $[i_t]_{1\times1}$ and output gate $[o_t]_{1\times1}$ similar calculations need to be applied. As seen in Figure 3.3, Equation 10, and Equation 11, the vanishing gradient problem is addressed by using a *tanh* function, which determines candidate values to be added to the internal state.

3.5.4 Evaluation

To evaluate the performance of the developed LSTM model, 80% of data in each unprocessed and preprocessed datasets (*i.e.,* 109,906 and 130,560 data records in unprocessed and

preprocessed datasets, respectively) is utilized to train the model, and the rest 20% of data in each datasets is considered to test the model. We employed k-fold cross validation (with $k = 5$) to evaluate the performance of the prediction model and minimize the bias at the training phase. To this end, the dataset is shuffled and partitioned into 5 equal-sized datasets each including 20% of data records (*i.e.*, 24,475 and 32,640 data records in unprocessed and preprocessed datasets, respectively). In each round of the 5-fold cross validation, one sub-datasets is used for testing the performance and the remaining four ones are considered to train the model.

In the network setting, we considered the maximum number of training epochs as 50, and the batch size is set to 72. The network is also trained and optimized by Adam [136] with a learning rate of 0.001 and Mean Absolute Error (MAE) as the loss function. The experimental results showed that for the epoch number larger than 25, the training and test set loss tend to be stable. Therefore, the epoch number is set at 25 in our experiment.

In this research, four statistical evaluation measures including Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and coefficient of determination (R^2) are utilized to evaluate the performance of the prediction model on both unprocessed and preprocessed datasets. As shown in Equation 12, MAE represents the average absolute distance between the predicted PM2.5 value (P_i) and the actual PM2.5 value (O_i). The standard deviation of the prediction errors is what RMSE shows in Equation 13. MAPE considers relative errors to reflect the specificity of the average predicted value using Equation 14. Finally, as shown in Equation 15, R^2 indicates how well the predicted values match the actual observations.

$$MAE = \frac{\sum_{j=0}^{n-1} |P_i - O_i|}{n} \tag{12}$$

$$RMSE = \sqrt{\frac{\sum_{j=0}^{n-1}(P_i - O_i)^2}{n}} \tag{13}$$

$$MAPE = \frac{1}{n}\sum_{j=0}^{n-1}\frac{|P_i - O_i|}{O_i} \tag{14}$$

$$R^2 = 1 - \frac{\sum_{j=0}^{n-1} v_{h,j}\,(P_i - O_i)^2}{\sum_{j=0}^{n-1} v_{h,j}\,(P_i - \bar{P})^2} \tag{15}$$

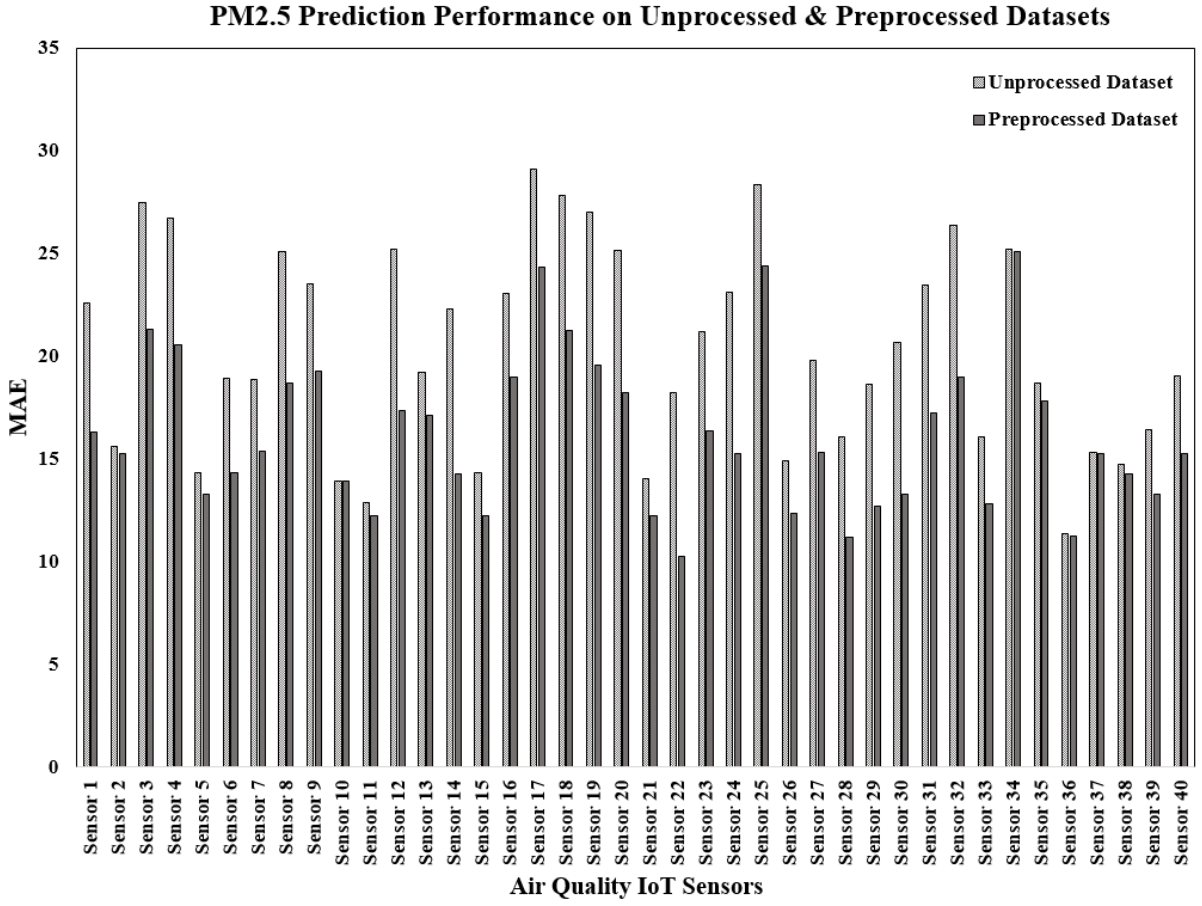


Figure 3.10. The performance evaluation for 40 IoT air quality sensors

Figure 3.10 shows the MAE of the LSTM model applied on both datasets (*i.e.*, unprocessed and preprocessed data) to predict PM2.5 for the next hour for all 40 IoT sensors. From the analysis of this Figure, we observe how the LSTM model produces better results on the preprocessed dataset. For some sensors (*e.g.*, Sensor_{10}, Sensor_{36}, and Sensor_{37}), the difference between the MAE of the prediction model on the two datasets is negligible. The investigation on sensor

measurements (*i.e.*, Figure 3.8) shows that less than 1% of all missing values (*i.e.*, 25,817 missing values in total) exist on data provided by these sensors. However, the most considerable difference in MAE between the two datasets can be attributed to more missing values in the PM2.5 measurements provided by some other sensors (*e.g.*, $\text{Sensor}_1, \text{Sensor}_{12}, \text{Sensor}_{32}$).

The detailed results of the performance of the LSTM model (*i.e.*, MAE and RMSE) on the two datasets are provided inTable 3.3. As shown in

Table *3.3*, four different scenarios are investigated in our experiment using edge nodes. Table 3.7 summarizes the average results of applying the LSTM model to both unprocessed and preprocessed datasets for all 40 IoT sensors in each scenario. Analyzing this Table shows how the LSTM model can obtain lower MAE, RMSE, MAPE, and higher R^2 when applied to preprocessed data compared to unprocessed data. As seen in Figure 3.11, it is worth mentioning that the best prediction results in our experiment ware achieved when four edge nodes were taken into account in the preprocessed dataset. The average result achieved by applying the LSTM model to the preprocessed data set obtained by four scenarios shows that the model can predict PM2.5 values 15.45% closer to the actual observations than it did with the unprocessed data. Taking the MAPE results, it is also possible to conclude that the accuracy of PM2.5 prediction is improved by 40.18% in average using the proposed preprocessing technique.

Table 3.6. Evaluation performance on the prediction of PM2.5 values

	MAE ($\mu g/m^3$)		RMSE ($\mu g/m^3$)	
	Unprocessed Dataset	Preprocessed Dataset	Unprocessed Dataset	Preprocessed Dataset
Sensor1	22.5818	16.2908	25.8466	19.6708
Sensor2	15.5743	15.2454	17.9692	18.0686
Sensor3	27.4518	21.3256	28.454	23.1903
Sensor4	26.6982	20.5489	29.7212	24.3707
Sensor5	14.3282	13.2545	15.7849	15.6021

	MAE ($\mu g/m^3$)		RMSE ($\mu g/m^3$)	
	Unprocessed Dataset	Preprocessed Dataset	Unprocessed Dataset	Preprocessed Dataset
Sensor6	18.9172	14.3125	19.2484	15.9776
Sensor7	18.8562	15.3489	22.5337	19.1283
Sensor8	25.0881	18.6589	27.0633	21.1399
Sensor9	23.5198	19.2354	26.3922	22.729
Sensor10	13.9179	13.8915	15.1362	16.0262
Sensor11	12.854	12.2374	14.8555	15.0353
Sensor12	25.1841	17.3652	26.5206	20.3228
Sensor13	19.2084	17.1327	21.7176	20.0987
Sensor14	22.2618	14.2662	23.0522	16.1355
Sensor15	14.3027	12.2315	16.555	14.7461
Sensor16	23.0597	18.9564	25.5	21.8988
Sensor17	29.0892	24.3254	29.6617	25.9423
Sensor18	27.8434	21.2491	30.2362	24.7677
Sensor19	27.0268	19.5362	28.5593	22.0014
Sensor20	25.1327	18.2341	25.0471	19.6607
Sensor21	14.0466	12.2486	15.7478	15.7271
Sensor22	18.1955	10.2357	20.1181	13.283
Sensor23	21.1859	16.3524	23.6016	20.055
Sensor24	23.1012	15.2418	23.0998	17.0078
Sensor25	28.3695	24.3589	29.5004	26.3884
Sensor26	14.925	12.3518	15.7016	13.7553
Sensor27	19.7711	15.2945	21.4894	18.0733
Sensor28	16.0487	11.1574	17.9663	14.8845
Sensor29	18.5973	12.6606	20.7629	16.3145
Sensor30	20.6705	13.2543	21.5224	14.604
Sensor31	23.4645	17.2544	24.7307	19.2975

	MAE ($\mu g/m^3$)		RMSE ($\mu g/m^3$)	
	Unprocessed Dataset	Preprocessed Dataset	Unprocessed Dataset	Preprocessed Dataset
Sensor32	26.3629	18.954	27.1576	21.0857
Sensor33	16.0733	12.8057	18.7818	16.0378
Sensor34	25.1832	25.0607	25.8628	26.094
Sensor35	18.7064	17.8056	20.2113	20.8062
Sensor36	11.3573	11.2341	11.7964	12.2587
Sensor37	15.2835	15.2359	16.2845	16.7037
Sensor38	14.7417	14.2548	15.6498	16.0061
Sensor39	16.4212	13.2578	17.3937	15.1122
Sensor40	19.0202	15.2486	19.6767	17.2826
Average	20.3605	16.1979	21.9228	18.6823

In another experiment, the impact of three aggregation techniques (*i.e.*, average, maximum, and weighted average) on prediction quality is investigated for the preprocessed dataset. Figure 3.12 shows the RMSE for the LSTM model using each aggregation technique. Analyzing this Figure shows that LSTM performance is not significantly changed by using different aggregation techniques. More specifically, the average RMSE of the LSTM model for all 40 IoT sensors is 18.22, 18.68, and 19.08 for average, weighted average, and maximum aggregation techniques, respectively. Analyzing this Figure, it does not appear that there is a significant difference between average, weighted average, and maximum value of PM2.5 reported by sensors on hourly intervals.

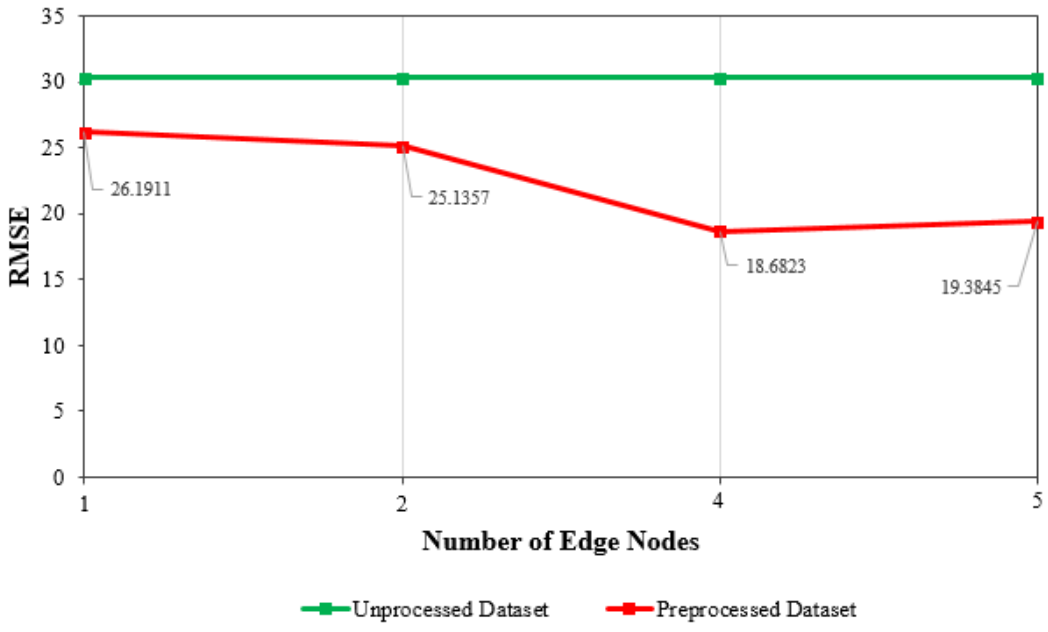
Table 3.7. Average performance of the LSTM model on both unprocessed and preprocessed datasets using four evaluation metrics: MAE, RMSE, MAPE, and R^2

Datasets	Number of Edge Nodes	MAE	RMSE	MAPE (%)	R^2 (%)
Unprocessed	1,2,4,5	26.38	30.32	31.87	64.31
Preprocessed	1	20.38	26.19	25.48	72.45

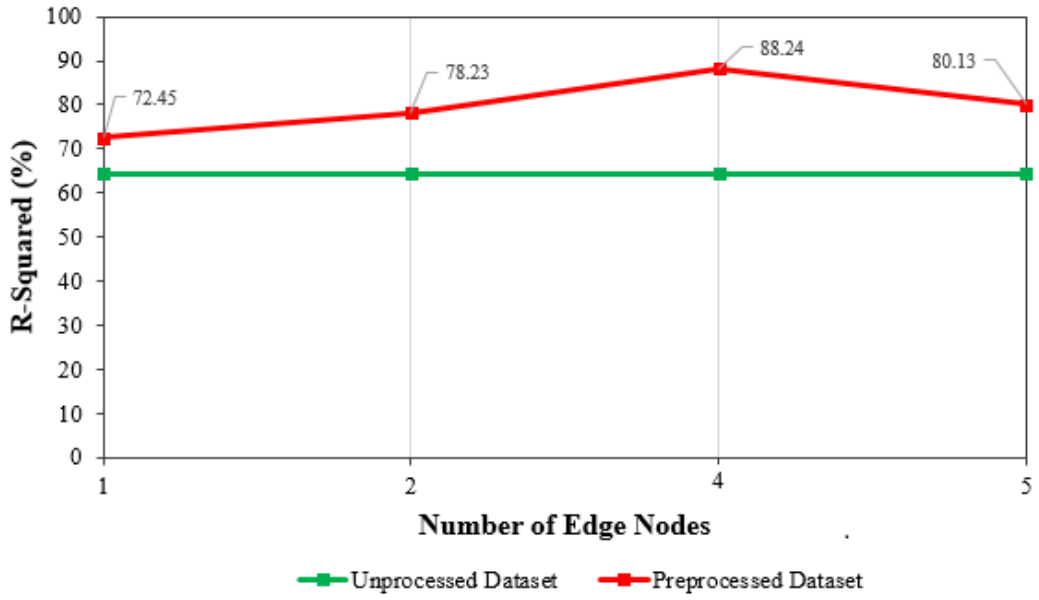
2	18.92	25.14	23.79	78.23
4	16.20	18.68	18.83	88.24
5	18.13	19.38	22.84	80.13

3.6 Conclusion

Air quality prediction can be considered as a very interesting research topic and an important task due to its importance for public health. Affordable IoT air quality sensors allow for a pervasive air quality monitoring, using both fixed and mobile devices. However, the number of involved devices imposes at least a decentralized preprocessing of the data; moreover, low-price devices provide inconsistent raw measurements with possibly missing data.



(a)



(b)

Figure 3.11. Performance evaluation of the prediction model considering four different scenarios of network topology: (a) The RMSE of the prediction model using different number of edge nodes, (b) The R2 of the prediction model using different number of edge nodes

In this study we designed a mixed edge-based and cloud-based architecture for the prediction of PM2.5 hourly values, addressing these issues. The edge based component is in charge of remotely preprocessing the data by cleaning it and filling missing values; both temporal and spatial information are used to improve the quality of the process. The cloud-based component applies deep learning strategies to predict PM2.5 values exploiting both preprocessed data and additional, external, information such as weather data. Tests on a real-world use case demonstrated the effectiveness of the proposed approach and the importance of data preprocessing at the edge for the prediction purposes.

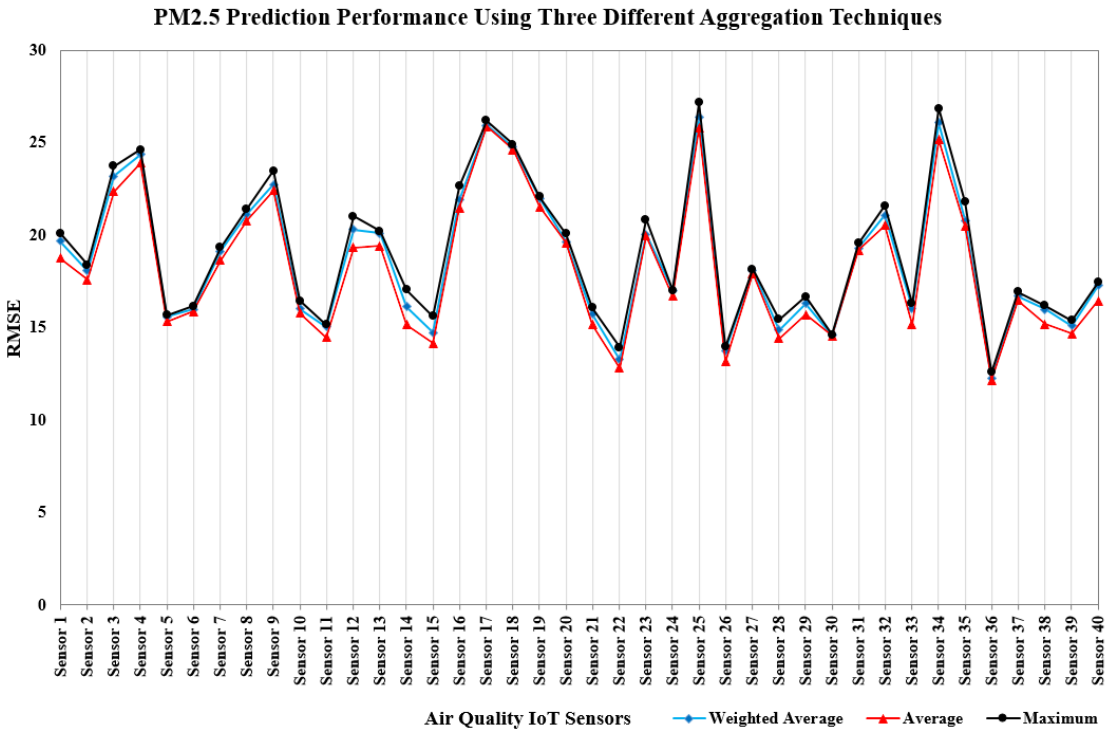


Figure 3.12. The performance evaluation for 40 IoT air quality sensors using three different aggregation techniques

As far as future work is concerned, the ideas presented in the present work can be extended in several directions. First of all, the preprocessing activity can be further improved; in fact, beside filling missing values, the proposed approach can be easily adapted to identify faulty sensors or anomalies in the data. The identification of such situations may allow to automatically repair the data and to tag sensors based on their reliability; both tasks may allow to further improve the quality of predictions. This is also particularly true in the case of the usage of edge nodes. Indeed, after receiving the prediction of PM2.5 from the cloud, the edge nodes may exploit this data in order to better assess both the reliability of each sensor and its influence in correct predictions; as a consequence, the Weight Updater module at the edge node might be further improved in order to take also this information into account. Lightweight methods for deep-learning based data predictions can also be investigated, in order to transfer the entire prediction process on the edge;

this may allow to limit the amount of information sent to the cloud only to the most relevant expected variations of PM2.5 values. In a future study, we plan also to investigate the effects of varying the size of the warping windows in the DTW algorithm on prediction performance. A further interesting research direction is the development of algorithms for the dynamic assignment of sensors to the edge nodes in presence of moving sensors; moreover, testing the impact of different hardware configurations for edge nodes on system performances is an interesting future research direction.

4 General Discussion

4.1 Overview

With an increase in incorporating various IoT devices in human lives and advances in communication technologies, the demand for processing and analyzing spatial and temporal data has increased exponentially. Low price IoT sensors can provide an enormous volume of data every second that can be utilized in different applications. For instance, the analysis of human movement, which is time series of location information across space and time, can be considered as crucial knowledge to control the spread of infectious diseases. Similarly, low price air quality sensors can provide precious data in a fine spatial granularity which can then be analyzed to predict the future status of different phenomena such as air quality in smart cities.

In order to be able to analyze spatiotemporal data provided by IoT sensors, the first challenge is defining the spatial and temporal data model. Since the quality of input data directly influences the quality of outputs from spatiotemporal analysis, the second challenge is validating and preprocessing IoT datasets. In other words, the dataset provided by IoT sensors needs to undergo a validation process in order to assure more reliable results would be expected from the spatiotemporal analysis. Finally, the designed IoT architecture needs to support scalability to meet the requirements of analyzing continuous sensor readings and interoperability among various IoT applications.

In this research, two different applications of using IoT devices for identifying safe zones for vulnerable people have been presented. In the first application, we designed an IoT architecture that is able to consider spatial, temporal, and contextual information about users' movements in

indoor environments for an enhanced contact tracing application. In this application, we proposed a hierarchical graph-based data model based on OGC IndoorGML, which is able to different parameters in an enhanced, scalable IoT architecture. In the second application, we focused on air quality prediction using spatiotemporal IoT data provided by air pollution sensors. The edge-based preprocessing technique offered in this application reduces the cost of spatiotemporal analysis on the cloud and increases the quality of data provided by IoT air quality sensors. A summary of our main findings in each application is given in the next section, supporting this research's primary objectives (explained in Section 1.3).

4.2 Main Findings

- *Evaluating the hierarchical data model for COVID-19 contact tracing application:*

  A graph-based semantic indoor trajectory data model was designed for the COVID-19 contact tracing application that can be utilized in different indoor trajectory analyses. Three spatial, temporal, and contextual hierarchical structures were considered in the proposed data model to support different granularity levels for trajectory data representation.

  In one experiment, the performance of spatiotemporal analysis in terms of required execution time for person-to-person and person-to-place scenarios is examined. In the first conclusion, we found that when we increase the number of query trajectories by 400 times, the required query execution time increases almost ten times. Also, we found that considering temporal constraints in the spatiotemporal analysis will reduce the query execution time. As the second finding, it has been observed that increasing the number of movement trajectories resulted in an increase in the standard deviation of spatiotemporal query execution time. Further analysis of the standard deviation of spatiotemporal query execution time showed that the proposed contact tracing application is more sensitive to

the size of databases than the traditional person-to-person contact tracing application. The average execution time was less than five milliseconds with an average standard deviation of less than one millisecond for both traditional and enhanced contact tracing applications. Finally, experimental results showed that the number of reported possible COVID-19 infected users decreased by 44.98 percent in our proposed advanced contact tracing application compared to the standard contact tracing application. However, the average execution time of the advanced contact tracing application increased by 58.3 percent.

In another experiment, we considered different numbers of cleaning users to show the importance of considering users' contextual information (*i.e.,* disinfecting activities) in coronavirus transmission. Evaluation results showed that considering users' contextual information reduced the number of possible COVID-19 infected users by 20.06 percent, 32.34 percent, and 48.16 percent when 5 percent, 10 percent, and 20 percent of the users are considered as cleaning users. It can be concluded that the sequential order of disinfecting activities has a considerable effect on the COVID-19 contact tracing application. So, our proposed graph data model provides the ability to incorporate users' contexts in an enhanced contact tracing application to correctly filter out some users that have been recognized as possibly infected users by traditional contact tracing applications. It can be concluded that the proposed graph-based data model could be effectively applied even for the most complicated contact tracing queries.

- *Validating semantic indoor movement trajectories:*

Taking the logical connectivity graph extracted from OGC IndoorGML into consideration, a filtering algorithm was proposed to clean up the trajectory data. In the real-world

experiment, the proposed filtering algorithm was able to remove 73.53 percent of the semantically invalid trajectory points.

- *Developing an IoT cloud-based architecture for contact tracing applications:*

Amazon Cloud Web Services was incorporated as a way to implement scalable data storage and data management within the Amazon Cloud. The real-world experiment uses a smartphone app to collect movement trajectories for twenty users. Our experiment involved the deployment of 41 BLE beacons at the University of Calgary campus, assuming that at least one beacon should be deployed per room. In order to assess the scalability of the IoT architecture, three simulated movement trajectories with 200, 2,000, and 20,000 users were created. The maximum execution time for the advanced contact tracing application with the highest number of users was less than 10 seconds.

- *Designing an edge-based preprocessing technique to improve the quality of IoT data:*

We proposed the edge-based component responsible for remotely preprocessing the data by cleaning it and filling missing values. In our real-world experiment, there were 25,817 missing data measurements in total. Using edge-based preprocessing, we could estimate missing air quality values by considering both temporal and spatial information. We designed four different experiments with four different numbers of edge nodes (*i.e.*, 1, 2, 4, and 5 edge nodes). We have used three different data aggregation techniques (*i.e.*, minimum, maximum, and weighted average) to reduce data granularity from minutes to an hour. Experimental results revealed no significant difference in prediction precision between using average, weighted average, and maximum value of PM2.5 reported by

sensors at hourly intervals. In the second finding, it has been observed that the precision of the prediction algorithm for those sensors that have reported fewer missing values is higher than those with a higher number of missing values.

- *Predicting air quality using historical IoT data:*

We utilized a cloud-based prediction algorithms (*i.e.,* LSTM technique) to predict the PM2.5 value for the next hour using collected meteorological and air quality data. The performance of the prediction technique has been evaluated on both preprocessed and unprocessed datasets. There was a negligible performance difference for some sensors responsible for less than 1% of missing values. However, the performance difference was considerable for other sensors with a higher percentage of missing values. In other words, the most considerable difference in MAE between the two datasets can be attributed to more missing values in the PM2.5 measurements provided by some other sensors. The average result achieved by applying the LSTM model to the preprocessed data set showed that the prediction model could predict PM2.5 values 15.45% closer to the reality than the unprocessed data. Taking the MAPE results, it is also possible to conclude that the accuracy of PM2.5 prediction is improved by 40.18% on average using the proposed preprocessing technique. It can be concluded that data preprocessing is required at the edge for prediction purposes.

4.3 Future Directions

The ideas presented in this thesis work can be further extended in several directions. In our first application, the execution time for trajectory data analysis for 20,000 simulated users was analyzed. However, further investigation of the execution time required for trajectory data

analytics with many users (*e.g.*, millions of users) is required in future studies. In this application, users are asked to manually select their contextual information from a list of predefined contextual information. However, considering recent advances in IoT devices, user contexts can be automatically extracted in future studies. For example, using the information provided by smartwatches, smartphones, and smart speakers, various user contexts, such as their activity types, can be automatically extracted without user intervention. Finally, the proposed graph-based data model has been tested for the COVID-19 contact tracing application in this application. However, we have designed a data model to provide high flexibility to be utilized in other indoor location-based applications. As an interesting direction for future studies, the similarity between users taking their indoor movement trajectories can be calculated for different applications.

In our second application, we have used emulators to test the functionality of the edge node for data preprocessing. However, testing the impact of different hardware configurations for edge nodes on system performance is an interesting future research direction. We intend to investigate the effects of varying the size of the warping windows in the DTW algorithm on prediction performance in a future study. Another exciting research area is developing algorithms for the dynamic assignment of sensors to edge nodes when moving sensors are present.

5 Bibliography

1. Ojagh, S., S. Saeedi, and S.H. Liang, *A Person-to-Person and Person-to-Place COVID-19 Contact Tracing System Based on OGC IndoorGML.* ISPRS International Journal of Geo-Information, 2021. **10**(1): p. 2.
2. Zamanifar, A., E. Nazemi, and M. Vahidi-Asl, *DMP-IOT: A distributed movement prediction scheme for IOT health-care applications.* Computers & Electrical Engineering, 2017. **58**: p. 310-326.
3. Hu, Z., Z. Bai, Y. Yang, Z. Zheng, K. Bian, and L. Song, *UAV aided aerial-ground IoT for air quality sensing in smart city: Architecture, technologies, and implementation.* IEEE Network, 2019. **33**(2): p. 14-22.
4. Duangsuwan, S., A. Takarn, R. Nujankaew, and P. Jamjareegulgarn. *A study of air pollution smart sensors lpwan via nb-iot for thailand smart cities 4.0*. in *2018 10th International Conference on Knowledge and Smart Technology (KST)*. 2018. IEEE.
5. Ojagh, S., M.R. Malek, S. Saeedi, and S. Liang, *A location-based orientation-aware recommender system using IoT smart devices and Social Networks.* Future Generation Computer Systems, 2020. **108**: p. 97-118.
6. Maayan, G.D. *The IoT Rundown For 2020: Stats, Risks, and Solutions*. 2020 [cited 2020 November 25, 2020]; Available from: https://securitytoday.com/articles/2020/01/13/the-iot-rundown-for-2020.aspx#:~:text=Every%20second%E2%80%94127%20new%20IoT,devices%20will%20be%20installed%20worldwide.
7. Liang, S.H., S. Saeedi, S. Ojagh, S. Honarparvar, S. Kiaei, M.M. Jahromi, and J. Squires, *An Interoperable Architecture for the Internet of COVID-19 Things (IoCT) Using Open Geospatial Standards—Case Study: Workplace Reopening.* Sensors, 2021. **21**(1): p. 50.
8. Mohammadi, M. and A. Al-Fuqaha, *Exploiting the Spatio-Temporal Patterns in IoT Data to Establish a Dynamic Ensemble of Distributed Learners.* IEEE Access, 2018. **6**: p. 63316-63328.
9. Li, W., S. Wang, X. Zhang, Q. Jia, and Y. Tian, *Understanding intra-urban human mobility through an exploratory spatiotemporal analysis of bike-sharing trajectories.* International Journal of Geographical Information Science, 2020. **34**(12): p. 2451-2474.
10. Yang, C., K. Clarke, S. Shekhar, and C.V. Tao, *Big Spatiotemporal Data Analytics: A research and innovation frontier*. 2020, Taylor & Francis.
11. TeamSense. *Empower the deskless members of your team*. 2020 [cited 2020 December 14]; Available from: https://www.teamsense.com/features.
12. Canada, G.o. *Download COVID Alert today*. 2020 [cited 2020 December 14]; Available from: https://www.canada.ca/en/public-health/services/diseases/coronavirus-disease-covid-19/covid-alert.html.
13. Alberta, G.o. *ABTraceTogether*. 2020 [cited 2020 December 14]; Available from: https://www.alberta.ca/ab-trace-together.aspx.
14. Du, S., T. Li, Y. Yang, and S.-J. Horng, *Deep air quality forecasting using hybrid deep learning framework.* IEEE Transactions on Knowledge and Data Engineering, 2019.
15. Prybutok, V.R., J. Yi, and D. Mitchell, *Comparison of neural network models with ARIMA and regression models for prediction of Houston's daily maximum ozone concentrations.* European Journal of Operational Research, 2000. **122**(1): p. 31-40.
16. Donnelly, A., B. Misstear, and B. Broderick, *Real time air quality forecasting using integrated parametric and non-parametric regression techniques.* Atmospheric Environment, 2015. **103**: p. 53-65.
17. Jensen, S.S., *Background concentrations for use in the Operational Street Pollution Model (OSPM)*. 1998: National Environmental Research Inst.´.
18. Kontarinis, A., K. Zeitouni, C. Marinica, D. Vodislav, and D. Kotzinos. *Towards a semantic indoor trajectory model*. 2019.
19. Alattas, A., P. van Oosterom, S. Zlatanova, D. Hoeneveld, and E. Verbree, *LADM-IndoorGML for exploring user movements in evacuation exercise.* Land Use Policy, 2020. **98**: p. 104219.
20. Ramadhan, H., Y. Yustiawan, and J. Kwon, *Applying Movement Constraints to BLE RSSI-Based Indoor Positioning for Extracting Valid Semantic Trajectories.* Sensors, 2020. **20**(2): p. 527.
21. Berke, A., M. Bakker, P. Vepakomma, R. Raskar, K. Larson, and A. Pentland, *Assessing disease exposure risk with location data; A proposal for cryptographic preservation of privacy.* arXiv preprint arXiv:2003.14412, 2020.
22. Kumar, U. and V. Jain, *ARIMA forecasting of ambient air pollutants (O 3, NO, NO 2 and CO).* Stochastic Environmental Research and Risk Assessment, 2010. **24**(5): p. 751-760.
23. Biancofiore, F., M. Busilacchio, M. Verdecchia, B. Tomassetti, E. Aruffo, S. Bianco, et al., *Recursive neural network model for analysis and forecast of PM10 and PM2. 5.* Atmospheric Pollution Research, 2017. **8**(4): p. 652-659.
24. Liu, B.-C., A. Binaykia, P.-C. Chang, M.K. Tiwari, and C.-C. Tsao, *Urban air quality forecasting based on multi-dimensional collaborative support vector regression (svr): A case study of beijing-tianjin-shijiazhuang.* PloS one, 2017. **12**(7): p. e0179763.

25. Rao, K.S., G.L. Devi, and N. Ramesh, *Air quality prediction in Visakhapatnam with LSTM based recurrent neural networks.* International Journal of Intelligent Systems and Applications, 2019. **11**(02): p. 2019.
26. Qi, Z., T. Wang, G. Song, W. Hu, X. Li, and Z. Zhang, *Deep air learning: Interpolation, prediction, and feature analysis of fine-grained air quality.* IEEE Transactions on Knowledge and Data Engineering, 2018. **30**(12): p. 2285-2297.
27. Li, X., L. Peng, Y. Hu, J. Shao, and T. Chi, *Deep learning architecture for air quality predictions.* Environmental Science and Pollution Research, 2016. **23**(22): p. 22408-22417.
28. Soh, P.-W., J.-W. Chang, and J.-W. Huang, *Adaptive deep learning-based air quality prediction model using the most relevant spatial-temporal relations.* Ieee Access, 2018. **6**: p. 38186-38199.
29. Zhang, J., Y. Zheng, D. Qi, R. Li, and X. Yi. *DNN-based prediction model for spatio-temporal data*. in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2016.
30. Wang, J. and G. Song, *A deep spatial-temporal ensemble model for air quality prediction.* Neurocomputing, 2018. **314**: p. 198-206.
31. Cauteruccio, F., G. Fortino, A. Guerrieri, A. Liotta, D.C. Mocanu, C. Perra, et al., *Short-long term anomaly detection in wireless sensor networks based on machine learning and multi-parameterized edit distance.* Information Fusion, 2019. **52**: p. 13-30.
32. Whitelaw, S., M.A. Mamas, E. Topol, and H.G. Van Spall, *Applications of digital technology in COVID-19 pandemic planning and response.* The Lancet Digital Health, 2020.
33. Guinchard, A., *Our digital footprint under Covid-19: should we fear the UK digital contact tracing app?* International Review of Law, Computers & Technology, 2020: p. 1-14.
34. Kampf, G., S. Lemmen, and M. Suchomel, *Ct values and infectivity of SARS-CoV-2 on surfaces.* The Lancet Infectious Diseases, 2020.
35. Mondelli, M.U., M. Colaneri, E.M. Seminari, F. Baldanti, and R. Bruno, *Low risk of SARS-CoV-2 transmission by fomites in real-life conditions.* The Lancet Infectious Diseases, 2020.
36. Medicine, T.L.R., *COVID-19 transmission—up in the air.* The Lancet. Respiratory Medicine, 2020.
37. Organization, W.H. *Coronavirus disease (COVID-19): How is it transmitted?* 2020 [cited 2020 December 12]; Available from: https://www.who.int/news-room/q-a-detail/coronavirus-disease-covid-19-how-is-it-transmitted.
38. Morawska, L. and J. Cao, *Airborne transmission of SARS-CoV-2: The world should face the reality.* Environment International, 2020: p. 105730.
39. Van Doremalen, N., T. Bushmaker, D.H. Morris, M.G. Holbrook, A. Gamble, B.N. Williamson, et al., *Aerosol and surface stability of SARS-CoV-2 as compared with SARS-CoV-1.* New England Journal of Medicine, 2020. **382**(16): p. 1564-1567.
40. He, H., R. Li, R. Wang, J. Bao, Y. Zheng, and T. Li, *Efficient Suspected Infected Crowds Detection Based on Spatio-Temporal Trajectories.* arXiv preprint arXiv:2004.06653, 2020.
41. Riemer, K., R. Ciriello, S. Peter, and D. Schlagwein, *Digital contact-tracing adoption in the COVID-19 pandemic: IT governance for collective action at the societal level.* European Journal of Information Systems, 2020: p. 1-15.
42. Ahmed, N., R.A. Michelin, W. Xue, S. Ruj, R. Malaney, S.S. Kanhere, et al., *A survey of covid-19 contact tracing apps.* IEEE Access, 2020. **8**: p. 134577-134601.
43. Braithwaite, I., T. Callender, M. Bullock, and R.W. Aldridge, *Automated and partly automated contact tracing: a systematic review to inform the control of COVID-19.* The Lancet Digital Health, 2020.
44. Kumar, K., N. Kumar, and R. Shah, *Role of IoT to avoid spreading of COVID-19.* International Journal of Intelligent Networks, 2020. **1**: p. 32-35.
45. OpenTrace. *OpenTrace*. [cited 2020 Oct. 03, 2020]; Available from: https://github.com/opentrace-community.
46. CovidSafe. *CovidSafe*. [cited 2020 Oct. 03, 2020]; Available from: https://github.com/AU-COVIDSafe.
47. PACT. *East Coast*. [cited 2020 Oct. 03, 2020]; Available from: https://pact.mit.edu/.
48. Morawska, L. and D.K. Milton, *It is time to address airborne transmission of COVID-19.* Clin Infect Dis, 2020. **6**: p. ciaa939.
49. Jensen, C.S., H. Lu, and B. Yang. *Graph model based indoor tracking*. in *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. 2009. IEEE.
50. Alattas, A., P. van Oosterom, S. Zlatanova, D. Hoeneveld, and E. Verbree, *LADM-IndoorGML for exploring user movements in evacuation exercise.* Land Use Policy, 2020: p. 104219.
51. Gu, F., S. Valaee, K. Khoshelham, J. Shang, and R. Zhang, *Landmark Graph-based Indoor Localization.* IEEE Internet of Things Journal, 2020.
52. Li, K.-J., G. Conti, E. Konstantinidis, S. Zlatanova, and P. Bamidis, *OGC IndoorGML: A standard approach for indoor maps*, in *Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation*. 2019, Elsevier. p. 187-207.
53. Control, C.f.D. and Prevention, *Cleaning and disinfection for households interim recommendations for US households with suspected or confirmed coronavirus disease 2019 (COVID-19)[Accessed April 13, 2020]. https://www. cdc. gov/coronavirus/2019-ncov/prevent-getting-sick/cleaning-disinfection. html.* Updated March, 2020. **28**.

54. Zheng, Y., *Trajectory data mining: an overview.* ACM Transactions on Intelligent Systems and Technology (TIST), 2015. **6**(3): p. 1-41.
55. Parent, C., S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, et al., *Semantic trajectories modeling and analysis.* ACM Computing Surveys (CSUR), 2013. **45**(4): p. 1-32.
56. Gómez, L.I., B. Kuijpers, and A.A. Vaisman, *Analytical queries on semantic trajectories using graph databases.* Transactions in GIS, 2019. **23**(5): p. 1078-1101.
57. Spanier, E.H., *Algebraic topology*. 1989: Springer Science & Business Media.
58. Montjoye, Y.-A., L. Radaelli, V. Singh, and A. Pentland, *Unique in the shopping mall: On the reidentifiability of credit card metadata.* Science (New York, N.Y.), 2015. **347**: p. 536-9.
59. Zhuang, Y., J. Yang, Y. Li, L. Qi, and N. El-Sheimy, *Smartphone-based indoor localization with bluetooth low energy beacons.* Sensors, 2016. **16**(5): p. 596.
60. Andrushchak, V., T. Maksymyuk, M. Klymash, and D. Ageyev. *Development of the iBeacon's Positioning Algorithm for Indoor Scenarios*. in *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*. 2018. IEEE.
61. Simko, L., J.L. Chang, M. Jiang, R. Calo, F. Roesner, and T. Kohno, *COVID-19 Contact Tracing and Privacy: A Longitudinal Study of Public Opinion.* arXiv preprint arXiv:2012.01553, 2020.
62. Bianconi, A., A. Marcelli, G. Campi, and A. Perali, *Efficiency of COVID-19 mobile contact tracing containment by measuring time-dependent doubling time.* Physical biology, 2020. **17**(6): p. 065006.
63. Ferretti, L., C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, et al., *Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing.* Science, 2020. **368**(6491).
64. Hellewell, J., S. Abbott, A. Gimma, N.I. Bosse, C.I. Jarvis, T.W. Russell, et al., *Feasibility of controlling COVID-19 outbreaks by isolation of cases and contacts.* The Lancet Global Health, 2020.
65. estimote. *Workplace safety with wearables*. 2020 [cited 2020 December 14]; Available from: https://estimote.com/wearable/?gclid=CjwKCAiAiML-BRAAEiwAuWVggkJsbosByEODh11g7RBWrxIJ-XSIP5oGOfKhZ3z0F01_ONi9rKAUsBoCyF8QAvD_BwE.
66. Stevens, H. and M.B. Haines, *TraceTogether: Pandemic Response, Democracy, and Technology.* East Asian Science, Technology and Society: An International Journal, 2020. **14**(3): p. 523-532.
67. SensorTower. *TraceTogether*. 2020 [cited 2020 December 12]; Available from: https://sensortower.com/ios/SG/government-technology-agency/app/tracetogether/1498276074/overview.
68. Worldometer. *Countries in the world by population (2020)*. 2020 [cited 2020 December 12]; Available from: https://www.worldometers.info/world-population/population-by-country/.
69. Singapore, G.o. *TraceTogether, safer together*. 2020 [cited 2020 December 12]; Available from: https://www.tracetogether.gov.sg/.
70. Corona-Warn-App. *Corona-Warn-App Open Source Project*. 2020 [cited 2020 December 12]; Available from: https://www.coronawarn.app/en/.
71. SensorTower. *Corona warning app*. 2020 [cited 2020 December 12]; Available from: https://sensortower.com/ios/de/robert-koch-institut/app/corona-warn-app/1512595757/overview.
72. Shubina, V., S. Holcer, M. Gould, and E.S. Lohan, *Survey of Decentralized Solutions with Mobile Devices for User Location Tracking, Proximity Detection, and Contact Tracing in the COVID-19 Era.* Data, 2020. **5**(4): p. 87.
73. Lewis, D.M., *Coepi: Community epidemiology in action*. 2020.
74. Fenwick, R., M. Hittle, M. Ingle, O. Nash, V. Nguyen, J. Petrie, et al., *Sydney Von Arx, and Tina White. Covid watch*. 2020.
75. Rong, C., C. Lin, Y.N. Silva, J. Wang, W. Lu, and X. Du. *Fast and scalable distributed set similarity joins for big data analytics*. in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. 2017. IEEE.
76. disease, C.f.m.m.o.i. *Estimates for Singapore*. 2020 [cited 2020 December 12]; Available from: https://epiforecasts.io/covid/posts/national/singapore/.
77. disease, C.f.m.m.o.i. *National and Subnational estimates for Germany*. 2020 [cited 2020 December 12]; Available from: https://epiforecasts.io/covid/posts/national/germany/.
78. Krumm, J. and E. Horvitz. *Predestination: Inferring destinations from partial trajectories*. in *International Conference on Ubiquitous Computing*. 2006. Springer.
79. Lee, J.-G., J. Han, and K.-Y. Whang. *Trajectory clustering: a partition-and-group framework*. in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 2007.
80. Arslan, M., C. Cruz, and D. Ginhac, *Semantic enrichment of spatio-temporal trajectories for worker safety on construction sites.* Personal and Ubiquitous Computing, 2019. **23**(5-6): p. 749-764.
81. Liu, C. and C. Guo, *STCCD: Semantic Trajectory Clustering based on Community Detection in Networks.* Expert Systems with Applications, 2020: p. 113689.
82. Sun, Y., T. Gu, C. Bin, L. Chang, H. Kuang, Z. Huang, and L. Sun. *A multi-latent semantics representation model for mining tourist trajectory*. in *Pacific Rim International Conference on Artificial Intelligence*. 2018. Springer.
83. Cai, G., K. Lee, and I. Lee, *Itinerary recommender system with semantic trajectory pattern mining from geo-tagged photos.* Expert Systems with Applications, 2018. **94**: p. 32-40.

84. Nardini, F.M., S. Orlando, R. Perego, A. Raffaetà, C. Renso, and C. Silvestri, *Analysing trajectories of mobile users: from data warehouses to recommender systems*, in *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*. 2018, Springer. p. 407-421.
85. Yuan, N.J., Y. Zheng, L. Zhang, and X. Xie, *T-finder: A recommender system for finding passengers and vacant taxis.* IEEE Transactions on knowledge and data engineering, 2012. **25**(10): p. 2390-2403.
86. Werner, M., L. Schauer, and A. Scharf. *Reliable trajectory classification using Wi-Fi signal strength in indoor scenarios*. in *2014 IEEE/ION Position, Location and Navigation Symposium-PLANS 2014*. 2014. IEEE.
87. Guo, S., H. Xiong, and X. Zheng, *A novel semantic matching method for indoor trajectory tracking.* ISPRS international journal of geo-information, 2017. **6**(7): p. 197.
88. Wang, R., R. Shroff, Y. Zha, S. Seshan, and M. Veloso. *Indoor trajectory identification: Snapping with uncertainty*. in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015. IEEE.
89. Chen, Y., P. Yuan, M. Qiu, and D. Pi, *An indoor trajectory frequent pattern mining algorithm based on vague grid sequence.* Expert Systems with Applications, 2019. **118**: p. 614-624.
90. Li, H., H. Lu, X. Chen, G. Chen, K. Chen, and L. Shou, *Vita: A versatile toolkit for generating indoor mobility data for real-world buildings.* Proceedings of the VLDB Endowment, 2016. **9**(13): p. 1453-1456.
91. Alattas, A., S. Zlatanova, P. Van Oosterom, E. Chatzinikolaou, C. Lemmen, and K.-J. Li, *Supporting indoor navigation using access rights to spaces based on combined use of IndoorGML and LADM models.* ISPRS international journal of geo-information, 2017. **6**(12): p. 384.
92. Ojagh, S., M.R. Malek, S. Saeedi, and S. Liang, *A location-based orientation-aware recommender system using IoT smart devices and Social Networks.* Future Generation Computer Systems, 2020.
93. Ojagh, S., M.R. Malek, and S. Saeedi, *A Social–Aware Recommender System Based on User's Personal Smart Devices.* ISPRS International Journal of Geo-Information, 2020. **9**(9): p. 519.
94. Zheng, V.W., Y. Zheng, X. Xie, and Q. Yang, *Towards mobile intelligence: Learning from GPS history data for collaborative recommendation.* Artificial Intelligence, 2012. **184**: p. 17-37.
95. Cauteruccio, F., L. Cinelli, E. Corradini, G. Terracina, D. Ursino, L. Virgili, et al., *A framework for anomaly detection and classification in Multiple IoT scenarios.* Future Generation Computer Systems, 2020. **114**: p. 322-335.
96. Cauteruccio, F., L. Cinelli, G. Fortino, C. Savaglio, G. Terracina, D. Ursino, and L. Virgili, *An approach to compute the scope of a social object in a Multi-IoT scenario.* Pervasive and Mobile Computing, 2020. **67**: p. 101223.
97. Hu, F., Z. Li, C. Yang, and Y. Jiang, *A graph-based approach to detecting tourist movement patterns using social media data.* Cartography and Geographic Information Science, 2019. **46**(4): p. 368-382.
98. Niu, X., T. Chen, C.Q. Wu, J. Niu, and Y. Li, *Label-based trajectory clustering in complex road networks.* IEEE Transactions on Intelligent Transportation Systems, 2019.
99. Sabarish, B., R. Karthi, and T.G. Kumar, *Graph Similarity-based Hierarchical Clustering of Trajectory Data.* Procedia Computer Science, 2020. **171**: p. 32-41.
100. Kang, H.-K. and K.-J. Li, *A standard indoor spatial data model—OGC IndoorGML and implementation approaches.* ISPRS International Journal of Geo-Information, 2017. **6**(4): p. 116.
101. Lee, J., K.-J. Li, S. Zlatanova, T.H. Kolbe, C. Nagel, and T. Becker, *Ogc indoorgml.* Open Geospatial Consortium standard, 2014.
102. Kassir, R., *Risk of COVID-19 for patients with obesity.* Obesity Reviews, 2020. **21**(6).
103. England, P.H. *Guidance to assist professionals in advising the general public*. 2020 [cited 2020; Available from: https://www.gov.uk/government/publications/novel-coronavirus-2019-ncov-guidance-to-assist-professionals-in-advising-the-general-public/guidance-to-assist-professionals-in-advising-the-general-public.
104. de Silva, F., *Execution Time Analysis of Electrical Network Tracing in Relational and Graph Databases*. 2019.
105. Dar, A.B., A.H. Lone, S. Zahoor, A.A. Khan, and R. Naaz, *Applicability of mobile contact tracing in fighting pandemic (COVID-19): Issues, challenges and solutions.* Computer Science Review, 2020: p. 100307.
106. Ojagh, S., M.R. Malek, S. Saeedi, and S. Liang. *An Internet of Things (IoT) Approach for Automatic Context Detection*. in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 2018. IEEE.
107. (CDC), C.f.D.C.a.P. *Cleaning and Disinfecting Your Facility*. 2020 [cited 2020 December 13]; Available from: https://www.cdc.gov/coronavirus/2019-ncov/community/disinfecting-building-facility.html.
108. (WHO), W.H.O. *Coronavirus disease (COVID-19): Cleaning and disinfecting surfaces in non-health care settings*. 2020 [cited 2020 December 13].
109. Çaliş, G., B. Becerik-Gerber, A.B. Göktepe, L. Shuai, and L. Nan, *Analysis of the variability of RSSI values for active RFID-based indoor applications.* Turkish Journal of Engineering & Environmental Sciences, 2013. **37**(2).
110. Ravì, D., C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang, *Deep learning for health informatics.* IEEE journal of biomedical and health informatics, 2016. **21**(1): p. 4-21.

111. Ong, B.T., K. Sugiura, and K. Zettsu, *Dynamically pre-trained deep recurrent neural networks using environmental monitoring data for predicting PM 2.5.* Neural Computing and Applications, 2016. **27**(6): p. 1553-1566.
112. Schmidhuber, J., *Deep learning in neural networks: An overview.* Neural networks, 2015. **61**: p. 85-117.
113. Kadilar, G.Ö. and C. Kadilar. *Assessing air quality in Aksaray with time series analysis*. 2017. AIP Publishing LLC.
114. Singh, K.P., S. Gupta, A. Kumar, and S.P. Shukla, *Linear and nonlinear modeling approaches for urban air quality prediction.* Science of the Total Environment, 2012. **426**: p. 244-255 %@ 0048-9697.
115. Kök, I., M.U. Şimşek, and S. Özdemir. *A deep learning model for air quality prediction in smart cities*. 2017. IEEE.
116. Fan, J., Q. Li, J. Hou, X. Feng, H. Karimian, and S. Lin, *A spatiotemporal prediction framework for air pollution based on deep RNN.* ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2017. **4**: p. 15 %@ 2194-9042.
117. Sánchez, A.B., C. Ordóñez, F.S. Lasheras, F.J. de Cos Juez, and J. Roca-Pardiñas. *Forecasting SO2 pollution incidents by means of Elman artificial neural networks and ARIMA models*. 2013. Hindawi.
118. Mahajan, S., C.-H. Luo, D.-Y. Wu, and L.-J. Chen, *From Do-It-Yourself (DIY) to Do-It-Together (DIT): Reflections on designing a citizen-driven air quality monitoring framework in Taiwan.* Sustainable Cities and Society, 2021. **66**: p. 102628 %@ 2210-6707.
119. Ojagh, S., S. Saeedi, and S.H.L. Liang, *A Person-to-Person and Person-to-Place COVID-19 Contact Tracing System Based on OGC IndoorGML.* ISPRS International Journal of Geo-Information, 2021. **10**(1): p. 2.
120. Ho, C.-C., L.-J. Chen, and J.-S. Hwang, *Estimating ground-level PM2. 5 levels in Taiwan using data from air quality monitoring stations and high coverage of microsensors.* Environmental Pollution, 2020. **264**: p. 114810 %@ 0269-7491.
121. Ahmed, M., R. Mumtaz, S.M.H. Zaidi, M. Hafeez, S.A.R. Zaidi, and M. Ahmad, *Distributed Fog Computing for Internet of Things (IoT) Based Ambient Data Processing and Analysis.* Electronics, 2020. **9**(11): p. 1756.
122. Sayegh, A., J.E. Tate, and K. Ropkins, *Understanding how roadside concentrations of NOx are influenced by the background levels, traffic density, and meteorological conditions using Boosted Regression Trees.* Atmospheric Environment, 2016. **127**: p. 163-175 %@ 1352-2310.
123. Dai, Z., D. Liu, K. Yu, L. Cao, and Y. Jiang, *Meteorological variables and synoptic patterns associated with air pollutions in eastern China during 2013–2018.* International journal of environmental research and public health, 2020. **17**(7): p. 2528.
124. Wang, X., A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, *Experimental comparison of representation methods and distance measures for time series data.* Data Mining and Knowledge Discovery, 2013. **26**(2): p. 275-309 %@ 1573-756X.
125. Dilmi, D., L. Barthès, C. Mallet, and A. Chazottes. *Modified DTW for a quantitative estimation of the similarity between rainfall time series*. 2017.
126. Guan, X., C. Huang, G. Liu, X. Meng, and Q. Liu, *Mapping rice cropping systems in Vietnam using an NDVI-based time-series similarity measurement based on DTW distance.* Remote Sensing, 2016. **8**(1): p. 19.
127. Cauteruccio, F., C. Stamile, G. Terracina, D. Ursino, and D. Sappey-Mariniery. *An automated string-based approach to white matter fiber-bundles clustering*. 2015. IEEE.
128. Aghabozorgi, S., A.S. Shirkhorshidi, and T.Y. Wah, *Time-series clustering–a decade review.* Information Systems, 2015. **53**: p. 16-38 %@ 0306-4379.
129. Gupta, L., D.L. Molfese, R. Tammana, and P.G. Simos, *Nonlinear alignment and averaging for estimating the evoked potential.* IEEE transactions on biomedical engineering, 1996. **43**(4): p. 348-356 %@ 0018-9294.
130. Hwa-Lung, Y. and W. Chih-Hsin, *Retrospective prediction of intraurban spatiotemporal distribution of PM2. 5 in Taipei.* Atmospheric Environment, 2010. **44**(25): p. 3053-3065 %@ 1352-2310.
131. Beckerman, B.S., M. Jerrett, M. Serre, R.V. Martin, S.-J. Lee, A. Van Donkelaar, et al., *A hybrid approach to estimating national scale spatiotemporal variability of PM2. 5 in the contiguous United States.* Environmental science & technology, 2013. **47**(13): p. 7233-7241 %@ 0013-936X.
132. Chu, H.-J., H.-L. Yu, and Y.-M. Kuo, *Identifying spatial mixture distributions of PM2. 5 and PM10 in Taiwan during and after a dust storm.* Atmospheric environment, 2012. **54**: p. 728-737 %@ 1352-2310.
133. Bulot, F.M.J., S.J. Johnston, P.J. Basford, N.H.C. Easton, M. Apetroaie-Cristea, G.L. Foster, et al., *Long-term field comparison of multiple low-cost particulate matter sensors in an outdoor urban environment.* Scientific reports, 2019. **9**(1): p. 1-13 %@ 2045-2322.
134. Liang, S., C.-Y. Huang, and T. Khalafbeigi, *OGC SensorThings API Part 1: Sensing, Version 1.0.* 2016.
135. Liang, S.H.L., S. Saeedi, S. Ojagh, S. Honarparvar, S. Kiaei, M.M. Jahromi, and J. Squires, *An Interoperable Architecture for the Internet of COVID-19 Things (IoCT) Using Open Geospatial Standards—Case Study: Workplace Reopening.* Sensors, 2021. **21**(1): p. 50.
136. Kingma, D.P. and J. Ba, *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.