

UNIVERSITY OF MODENA AND REGGIO EMILIA
ENGINEERING DEPARTMENT “ENZO FERRARI”

INTERNATIONAL DOCTORATE SCHOOL IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXXIV CYCLE

Ph.D. DISSERTATION

Towards sustainability and safety solutions in a Smart City

Candidate: **Dr. Federica Rollo**

Advisor: Prof. Laura Po

PhD Program Coordinator: Prof. Sonia Bergamaschi

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA
DIPARTIMENTO DI INGEGNERIA “ENZO FERRARI”

DOTTORATO DI RICERCA IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

CICLO XXXIV

TESI DI DOTTORATO

Verso soluzioni di sostenibilità e sicurezza per una Città Intelligente

Candidato: **Dott.ssa Federica Rollo**

Tutor: Prof.ssa Laura Po

Coordinatore del Corso di Dottorato: Prof.ssa Sonia Bergamaschi

Ai miei genitori

Abstract

A smart city is a place where technology is exploited to help public administrations make decisions. The technology can contribute to the management of multiple aspects of everyday life, offering more reliable services to citizens and improving the quality of life. However, technology alone is not enough to make a smart city; suitable methods are needed to analyze the data collected by technology and manage them in such a way as to generate useful information. Some examples of smart services are the apps that allow to reach a destination through the least busy road route or to find the nearest parking slot, or the apps that suggest better paths for a walk based on air quality.

This thesis focuses on two aspects of smart cities: sustainability and safety. The first aspect concerns studying the impact of vehicular traffic on air quality through the development of a network of traffic and air quality sensors, and the implementation of a chain of simulation models. This work is part of the TRAF AIR project, co-financed by the European Union, which is the first project with the scope of monitoring in real-time and predicting air quality on an urban scale in 6 European cities, including Modena. The project required the management of a large amount of heterogeneous data and their integration on a complex and scalable data platform shared by all the partners of the project. The data platform is a PostgreSQL database, suitable for dealing with spatio-temporal data, and contains more than 60 tables and 435 GB of data (only for Modena). All the processes of the TRAF AIR pipeline, the dashboards and the mobile apps exploit the database to get the input data and, eventually, store the output, generating big data streams. The simulation models, executed on HPC resources, use the sensor data and provide results in real-time (as soon as the sensor data are stored in the database). Therefore, the anomaly detection techniques applied to sensor data need to perform in real-time in a short time. After a careful study of the distribution of the sensor data and the correlation among the measurements, several anomaly detection techniques have been implemented and applied to sensor data. A novel approach for traffic data that employs a flow-speed correlation filter, STL decomposition and IQR analysis has been developed. In addition, an innovative framework that implements 3 algorithms for anomaly detection in air quality sensor data has been created. The results of the experiments have been compared to the ones of the LSTM autoencoder, and the performances have been

evaluated after the calibration process. The safety aspect in the smart city is related to a crime analysis project, the analytical processes directed at providing timely and pertinent information to assist the police in crime reduction, prevention, and evaluation. Due to the lack of official data to produce the analysis, this project exploits the news articles published in online newspapers. The goal is to categorize the news articles based on the crime category, geolocate the crime events, detect the date of the event, and identify some relevant features (e.g., what has been stolen during the theft). A Java application has been developed for the analysis of news articles, the extraction of semantic information through the use of NLP techniques, and the connection of entities to Linked Data. The emerging technology of Word Embeddings has been employed for the text categorization. The news articles referring to the same event have been identified through the application of cosine similarity to the shingles of the news articles' text. Finally, a dashboard has been developed to show the geolocalized events and provide some statistics and annual reports. This is the only project in Italy that starting from news articles tries to provide analyses on crimes and makes them available through a visualization tool.

Sommario

Una città intelligente è un luogo in cui la tecnologia viene sfruttata per aiutare le amministrazioni pubbliche a prendere decisioni. La tecnologia può contribuire alla gestione di numerosi aspetti della vita quotidiana, offrendo ai cittadini servizi più affidabili e migliorando la qualità della vita. Tuttavia, la tecnologia da sola non è sufficiente per rendere una città intelligente; sono necessari metodi adeguati per analizzare i dati raccolti e gestirli in modo da generare informazioni utili. Alcuni esempi di servizi intelligenti sono le app che permettono di raggiungere una destinazione attraverso il percorso più breve oppure di trovare il parcheggio libero più vicino, o le app che suggeriscono i percorsi migliori per una passeggiata in base alla qualità dell'aria. Questa tesi si concentra su due aspetti delle smart city: sostenibilità e sicurezza. Il primo aspetto riguarda lo studio dell'impatto del traffico sulla qualità dell'aria attraverso lo sviluppo di una rete di sensori di traffico e qualità dell'aria e l'implementazione di una catena di modelli di simulazione. Questo lavoro fa parte del progetto TRAF AIR, cofinanziato dall'Unione Europea, il primo progetto che monitora la qualità dell'aria in tempo reale e fa previsioni su scala urbana in 6 città europee, tra cui Modena. Il progetto ha richiesto la gestione di una grande quantità di dati eterogenei e la loro integrazione su una piattaforma dati complessa e scalabile condivisa da tutti i partner del progetto. La piattaforma è un database PostgreSQL, adatto a gestire dati spazio-temporali, che contiene più di 60 tabelle e 435 GB di dati (solo per Modena). Tutti i processi della pipeline di TRAF AIR, le dashboard e le app sfruttano il database per ottenere i dati di input ed eventualmente memorizzare l'output, generando grandi flussi di dati. I modelli di simulazione, eseguiti su risorse di HPC, utilizzano i dati dei sensori e devono fornire risultati in tempo reale. Pertanto le tecniche di identificazione delle anomalie applicate ai dati dei sensori devono eseguire in tempo reale e in breve tempo. Dopo un attento studio della distribuzione dei dati dei sensori e della correlazione tra le misure, sono state implementate e applicate alcune tecniche di identificazione delle anomalie. Per i dati di traffico è stato sviluppato un nuovo approccio che utilizza un filtro di correlazione flusso-velocità, la decomposizione STL e l'analisi IQR. Per i dati di qualità dell'aria è stato creato un framework innovativo che implementa 3 algoritmi. I risultati degli esperimenti sono stati confrontati con quelli dell'Autoencoder LSTM. L'aspetto relativo alla sicurezza nella città intelligente è legato a un progetto di analisi dei crimini, i

processi analitici volti a fornire informazioni tempestive e pertinenti per aiutare la polizia nella riduzione, prevenzione e valutazione del crimine. A causa della mancanza di dati ufficiali, questo progetto sfrutta le notizie pubblicate sui giornali online. L'obiettivo è quello di classificare le notizie in base alla categoria di crimine, geolocalizzare i crimini, identificare la data dell'evento, e individuare alcune caratteristiche. È stata sviluppata un'applicazione per l'analisi delle notizie, l'estrazione di informazioni semantiche attraverso l'uso di tecniche di NLP e la connessione delle entità a risorse Linked Data. La tecnologia dei Word Embedding è stata utilizzata per la categorizzazione del testo. Le notizie che si riferiscono allo stesso evento sono state identificate attraverso la cosine similarity sul testo delle notizie. Infine, è stata implementata un'interfaccia per mostrare su mappa i crimini geolocalizzati e fornire statistiche e rapporti annuali. Questo è l'unico progetto presente in Italia che partendo da notizie online cerca di fornire un'analisi sui crimini e la mette a disposizione attraverso uno strumento di visualizzazione.

Publications by the author

- Rollo F., Bonisoli G., Po L. (2022) Supervised and Unsupervised Categorization of an Imbalanced Italian Crime News Dataset. In: Ziemba E., Chmielarz W. Information Technology for Management: Business and Social Issues. Lecture Notes in Business Information Processing, vol 442. Springer (to appear)
- Chiara Bachechi, Federica Rollo, Laura Po, Big Data Analytics and Visualization in Traffic Monitoring, Big Data Research, Volume 27, 2022, 100292, ISSN 2214-5796, <https://doi.org/10.1016/j.bdr.2021.100292>
- Bachechi, C., Rollo, F. & Po, L. Detection and classification of sensor anomalies for simulating urban traffic scenarios. Cluster Comput (2021). <https://doi.org/10.1007/s10586-021-03445-7>
- Sudharsan, Bharath, Sheth, Dhruv, Lin Kavya Kopparapu, Eric, Arya, Shailesh, Rollo, Federica, Yadav, Piyush, Patel, Pankesh, Breslin, John, Intizar Ali, Muhammad. “ElasticCL: Elastic Parameters Quantization for Communication Efficient Collaborative Learning in IoT.” SenSys 2021: The 19th ACM Conference on Embedded Networked Sensor Systems, Coimbra, Portugal, November 15-17, 2021
- Rollo, Federica, Sudharsan, Bharath, Po, Laura and Breslin, John, Air Quality Sensor Network Data Acquisition, Cleaning, Visualization, and Analytics: A Real-world IoT Use Case. Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers. 2021
- C. Bachechi, F. Rollo, L. Po and F. Quattrini, “Anomaly Detection in Multivariate Spatial Time Series: A Ready-to-Use implementation,” 2021 17th International Conference on Web Information Systems and Technologies (WEBIST), 2021
- G. Bonisoli, F. Rollo and L. Po, Using Word Embeddings for Italian Crime News Categorization, 2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS), 2021, pp. 461-470, doi: 10.15439/2021F118
- Rollo, Federica, and Po, Laura. SenseBoard: Sensor monitoring for air quality experts. 2021 Workshops of the EDBT/ICDT Joint Conference, EDBT/ICDT-WS 2021. Vol. 2841. CEUR-WS, 2021
- Desimoni, F.; Ilarri, S.; Po, L.; Rollo, F.; Trillo-Lado, R. Semantic Traffic Sensor Data: The TRAF AIR Experience. Appl. Sci. 2020, 10, 5882

- Rollo, Federica, and Po, Laura. (2020) Crime Event Localization and Deduplication. In: Pan J.Z. et al. (eds) *The Semantic Web – ISWC 2020*. ISWC 2020. Lecture Notes in Computer Science, vol 12507. Springer, Cham. https://doi.org/10.1007/978-3-030-62466-8_23
- Bachechi, Chiara, Federica Rollo, and Laura Po. Real-Time Data Cleaning in Traffic Sensor Networks. 2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA). IEEE Computer Society, 2020
- Bachechi C., Rollo F., Desimoni F., Po L. (2020) Using Real Sensors Data to Calibrate a Traffic Model for the City of Modena. In: Ahram T., Karwowski W., Vergnano A., Leali F., Taiar R. (eds) *Intelligent Human Systems Integration 2020*. IHSI 2020. Advances in Intelligent Systems and Computing, vol 1131. Springer, Cham. https://doi.org/10.1007/978-3-030-39512-4_73
- L. Po, F. Rollo, C. Bachechi and A. Corni, From Sensors Data to Urban Traffic Flow Analysis, 2019 IEEE International Smart Cities Conference (ISC2), 2019, pp. 478-485, doi: 10.1109/ISC246665.2019.9071639
- Laura Po, Federica Rollo, Josè Ramòn Rios Viqueira, Raquel Trillo Lado, Alessandro Bigi, Javier Cacheiro Lòpez, Michela Paolucci, Paolo Nesi TRAFAIR: Understanding Traffic Flow to Improve Air Quality 2019 IEEE International Smart Cities Conference (ISC2), 2019, pp. 36-43, doi: 10.1109/ISC246665.2019.9071661
- L. Po and F. Rollo, Building an Urban Theft Map by Analyzing Newspaper Crime Reports, 2018 13th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP), 2018, pp. 13-18, doi: 10.1109/SMAP.2018.8501866

Table of contents

List of figures	xiii
List of tables	xvii
1 Introduction	1
I Sustainability Solution	5
2 TRAFAIR project	7
2.1 Goal and pipeline of the project	7
2.2 Software and data tiers	11
2.3 Traffic model	12
2.4 Air Quality model	15
3 A Smart City data platform	19
3.1 Requirements	19
3.2 Architectural choices	21
3.2.1 Timescale performance evaluation	22
3.3 Database conceptualization	23
3.3.1 Road network	24
3.3.2 Traffic sensor	27
3.3.3 Air quality sensor	28
3.3.4 Traffic flow analysis and modulation	29
3.4 Database analytics	31
3.5 Database and process monitoring dashboard	32
4 Traffic and air quality monitoring in Modena	35
4.1 Traffic sensors	35

4.1.1	Sensor network	35
4.1.2	Sensor data collection	37
4.1.3	Exploratory data analysis	39
4.1.4	Statistical overview of traffic sensors observations	41
4.1.5	Stationary study	42
4.2	Air quality sensors	44
4.2.1	Sensor network	45
4.2.2	Sensor data collection	47
4.2.3	Exploratory data analysis	48
4.2.4	SenseBoard: data visualization tool	49
5	Anomaly detection on sensor data	63
5.1	Related work	65
5.2	Anomaly detection on traffic sensors	67
5.2.1	Flow-speed correlation filter	68
5.2.2	Seasonal Trend Decomposition using Loess	70
5.2.3	FFIDCAD and ARIMA model	83
5.2.4	ST-BOF and ST-BDBCAN	93
5.3	Anomaly detection on air quality sensors	100
5.3.1	Majority Voting	100
5.3.2	Real-time anomaly detection on edge sensors	108
6	Semantic sensor data publication	117
6.1	Related work	118
6.1.1	Analysis of traffic-related ontologies	119
6.2	Data annotation and publishing	122
6.2.1	Identification of relevant concepts and properties	123
6.2.2	Data integration	126
6.2.3	Data publication and exploitation	129
6.2.4	Technological choices	133
6.3	Experimental evaluation	136
II	Safety Solution	145
7	Urban crime analysis from news streams	147
7.1	Related work	149
7.1.1	Crime categorization	150

7.1.2	Document similarity	151
7.2	The proposed approach	152
7.3	Crime data platform	155
8	The Italian crime analysis framework	157
8.1	Web scraping	157
8.2	News categorization	159
8.2.1	Text representation extraction	159
8.2.2	Supervised and unsupervised categorization	161
8.2.3	Active learning	162
8.3	Entity extraction and mapping	163
8.3.1	NER	164
8.3.2	Time extraction and normalization	164
8.3.3	Linked Open Data mapping	165
8.3.4	Geo-localization	166
8.4	Duplicate detection	166
8.4.1	Blocking technique	168
8.4.2	News articles comparison	168
8.4.3	Information merging	170
9	Use case in Modena	173
9.1	Configuration of the framework	174
9.2	Experimental results	176
9.2.1	News categorization	176
9.2.2	NER effectiveness	187
9.2.3	Time extraction and normalization effectiveness	189
9.2.4	Duplicate detection	189
9.3	Impact and scalability of this research	190
9.4	Event visualization	192
10	Conclusions	195
	References	199

List of figures

2.1	Overview of the data flow in the TRAFAIR project.	9
2.2	Overview of the software tiers in the TRAFAIR project.	12
2.3	Overview of the traffic model implementation.	15
2.4	Modena map with roads of different colours based on the SUMO output at the 6 th January 2022.	16
2.5	Modena map with the output of the GRAL model at the 6 th January 2022.	17
3.1	E/R model of the TRAFAIR database to store the road network.	25
3.2	E/R model of the TRAFAIR database to store information about the traffic sensors and their measurements.	27
3.3	E/R model of the TRAFAIR database to store information related to the air quality sensors, the measurements, the calibration and the identification of anomalies.	28
3.4	E/R model of the TRAFAIR database to store the daily average traffic flow.	30
3.5	E/R model of the TRAFAIR database to store the average traffic flow modulation.	30
3.6	E/R model of the TRAFAIR database to store the seasonal average traffic flow.	31
3.7	E/R model of the TRAFAIR database to store the seasonal average traffic flow modulation.	31
3.8	Interaction of the TRAFAIR processes and applications with the data platform.	32
3.9	Visualization of the database and process monitoring dashboard.	33
4.1	https://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/red-light-camera2.htm	36
4.2	Modena map with regional (purple spots) and urban (blue spots) traffic sensors. Map data: Google, 2020.	37
4.3	Traffic sensor data ingestion.	38

4.4	Flow and speed measurements of sensor R030_S2 in one week of September 2021.	40
4.5	Flow and speed measurements of all traffic sensors in Modena in September 2021.	41
4.6	IQR, median and standard deviation of 318 trustworthy traffic sensors. . . .	43
4.7	Points of interest for air quality monitoring (blue dots) and positions of the AQM stations (red dots). Map data: Google, 2020.	45
4.8	An air quality device (on the left) and its content inside (on the right): 4 cells/sensors for measuring the level of 4 air pollutants (NO, NO ₂ , CO, and O ₃ in this case).	46
4.9	A Libelium Smart Environment PRO device (on the left) with Particle Matter sensor (on the right).	46
4.10	Sensor data acquisition from the low-cost air quality sensor network.	47
4.11	Plots with the values of the working and auxiliary channels of the 4 gases: NO, NO ₂ , CO, and O ₃	49
4.12	Raw observations of one air quality device made in two different locations.	50
4.13	SenseBoard architecture.	51
4.14	“Sensor status and position” view.	54
4.15	Position of the air quality devices in Modena on January 4 th , 2021.	54
4.16	An anomalous behavior of a device detected on January 13 th , 2021.	56
4.17	An anomalous behavior of a device detected on January 15 th , 2021, due to a drastic reduction in the battery level.	57
4.18	A visualization of the “gas observations” view which shows the measurements of NO channels.	58
4.19	Anomalies of sensor 4006 for the last 24 hours (A), the last week (B), and the last month (C) available on January 4 th , 2021 at 11 a.m..	59
4.20	Calibrated NO observations by 5 devices located in the same place (“Parco Ferrari”) visualized on January 4 th , 2021 at 4 p.m..	61
5.1	Road A-B with five vehicles (red rectangles).	68
5.2	Flow - speed scatter plots representing the observations of traffic sensors in April 2019 with filtered observations in red.	69
5.3	Workflow of the proposed methodology.	71
5.4	The two versions of anomaly detection process: ADP1 and ADP2.	73
5.5	The STL decomposition of the time-series related to the observations of the sensor R124_S2 from April 8th, 2019 to April 14th, 2019 and anomalies detected by ADP1.	77

5.6	Time distribution of anomalies in April 2019.	78
5.7	Percentage of anomalies above the total number of observations for every day of April 2019.	78
5.8	Comparison between observed time-series (orange) and simulated flows (blue) in three locations for the 1st of April. Standard traffic simulation (STD SIM - on the top) is compared to the traffic simulation that takes advantage of the data cleaning process with ADP1 (ADP1 SIM - on the bottom).	80
5.9	Anomalies found by ADP2 on sensor R124_S2 observations from 8 th April 2019 to 14 th April 2019.	81
5.10	Overview of the data cleaning process configuration.	85
5.11	ARIMA model applied to one minute measurements of one sensor in a day of April 2019.	88
5.12	Correlation between flow and speed for sensor R009_S1 on the 8th November 2018 (orange dots are anomalies).	90
5.13	Time-series of sensor R009_S1 on the 8th November 2018 (orange dots are anomalies).	91
5.14	Anomalies found by the flow-speed correlation filter and the FFIDCAD model in 8th November 2018.	92
5.15	Sensor R009_SM19 time-series on the 8th November 2018 (blue line) and ARIMA predictions (green line).	93
5.16	The two types of anomalies generated in the measurements of the working channel of NO (NO _{we}) of December 2020. Anomalous measurements are highlighted in red.	115
6.1	Overview of the mappings and tools used for data annotation and publishing. The credit of some images used has to be accredited to the creators of the software mentioned and used here for illustration purposes, others are freely available images extracted from Pixabay. The logo of Virtuoso has been extracted from https://commons.wikimedia.org/wiki/File:Virtuoso-logo-sm.png , contributed by Deirdre Gerhardt.	123
6.2	Definition of the “isLocatedInOSMWay” and the “hasNearestOSMNode” properties.	124
6.3	Data Model described though Shape Expressions.	127
6.4	Triples generated for a sample traffic observation in Turtle syntax.	128
6.5	Karma model implemented to transform data from the “sensor_traffic” table (Figure 3.2) into Linked Data.	128
6.6	Karma model implemented to transform data from the “sensor_traffic_observation” table (Figure 3.2) into Linked Data.	128

6.7	LodView representation of the station information of the traffic sensor R001_SM3.	131
6.8	LodView representation of one observation made by the traffic sensor R001_SM3.	132
6.9	SPARQL query showing information related to sensor R001_SM3.	132
6.10	GeoSPARQL query showing the vehicle count of some sensors located in the town square of Modena on January 8th, 2019.	134
6.11	SPARQL query showing the number of sensors in Modena and in Zaragoza.	139
6.12	SPARQL query showing the number of vehicles counted by each sensor on January 8th, 2019 (only a fragment of the answer is shown).	140
6.13	SPARQL query showing the list of sensors located on the street named “Ronda Hispanidad” and the number of vehicles counted by these sensors on January 8th, 2019.	141
6.14	GeoSPARQL query showing the number of sensors in Modena located inside the area delimited by Modena’s ring road.	142
7.1	The method implemented to extract, store, analyze the news articles (the numbers in the circles represent the phases described in Section 7.2).	152
7.2	Structure of the data platform used to store data extracted from the news articles.	154
8.1	Process of document vector extraction from news articles.	160
8.2	Active learning process.	163
9.1	Crimes reported to the authorities in the province of Modena from 2016 to 2020 [SOURCE: ISTAT, data of the Italian Ministry of the Interior].	174
9.2	Histograms of the cluster distribution obtained with K-means ($n = 13$) applied to the embeddings of model M3, simple average and second pre-processing type.	184
9.3	Plot of the silhouette coefficient in the clusters of K-means ($n = 13$) on the embeddings of model M3, simple average (A1) and pre-processing with lemmatization (P2).	185
9.4	Distribution of crime reports from 2014 to 2018 in different neighborhoods of Modena.	191
9.5	Overview of the Modena Crime Dashboard.	193

List of tables

3.1	Comparison of time performance with PostgreSQL and Timescale.	23
4.1	Analysis of the reliability of traffic sensors measurements from October 2018 till May 2020.	42
5.1	Comparison of traffic model evaluation metrics between standard simulations (STD SIM) and simulations after removing anomalies of ADP1 (ADP1 SIM) in April 2019.	79
5.2	Comparison of average metrics on April 2019 for STD SIM, ADP1 SIM and ADP2+CLASS SIM.	82
5.3	Comparison of traffic model evaluation metrics between traffic simulation including anomalies (STD SIM) and traffic simulation excluding sensors faults (ADP2+CLASS SIM).	83
5.4	Parameters' configuration for the application of ST_BOF and ST_BDBCAN.	97
5.5	Results of the application of ST_BOF and ST_BDBCAN.	97
5.6	Anomalies detected by SWAD, FFIDCAD, THAD.	105
5.7	Evaluation of the Majority Voting anomaly detection on real-world air quality sensor data.	108
5.8	IoT boards chosen for on-board evaluation of TRAFAIR datasets trained anomaly detection models.	111
5.9	Evaluation results of the anomaly detection models trained on the NO ₂ measurements of the air quality TRAFAIR dataset AQ1.	116
6.1	Sensor data performance evaluation: loading time.	137
6.2	Observation data performance evaluation: loading time of hourly observations from January to December 2019	137
6.3	Loading process performance for 1-year data under different granularity and window length conditions.	139
6.4	SPARQL queries response time.	139

8.1	SPARQL queries used to retrieve the type of annotated entities.	167
8.2	SPARQL queries used to retrieve the GPS coordinates of annotated entities.	167
8.3	Validation test of the <i>3-days slot</i> duplicate detection.	170
8.4	Validation test of the <i>5-days slot</i> duplicate detection.	170
9.1	The number of news articles in the training and test sets for each category in supervised categorization.	179
9.2	Precision (P) and recall (R) of the application of different categorization algorithms on the embeddings derived from the three selected models.	180
9.3	Precision, Recall and F1-score for each crime category obtained using the embeddings of model M3 with pre-processing P1 and average A1, and Linear SVC.	181
9.4	Precision (P) and recall (R) of the application of the best six algorithms on the embeddings of M2 and M3 on “ModenaToday” news articles.	182
9.5	The number of news articles from “ModenaToday” newspaper for each category.	183
9.6	Evaluation of unsupervised categorization using the document embeddings obtained by model M3 with the simple average and the three different pre-processing phases.	183
9.7	Results of unsupervised text categorization obtained by Spectral Clustering ($n=7$) applied to the document embeddings of model M3, simple average and the third pre-processing type.	186
9.8	The number of news articles in the training and test sets for each category.	186
9.9	Final results of active learning with the first configuration.	187
9.10	Final results of active learning with the second configuration.	187
9.11	Evaluation of the coverage in detecting location reference through the application of NER and the integration of Linked Geo Data	188
9.12	Duplicate detection algorithm on different <i>day slots</i>	190

Chapter 1

Introduction

The disruptive technological development of recent years has laid the foundations for the data-driven era. An increasing number of devices to monitor plenty of different aspects of everyday life is available. Measuring the concentration of pollutants in the air we breathe, monitoring the soil composition, checking people health through biometric data, monitoring the availability of parking slots are just some examples of the most common purposes. Thanks to the effort of technology, sensors become smaller and more reliable. Small devices ensure greater portability and can be potentially deployed at fixed locations as well as on moving vehicles, offering the benefit of collecting data punctual and widespread. The large amount of data generated by the sensors may improve the quality of life of people.

However, technology alone is not enough to implement improvements. The collected data require appropriate tools for the Big Data analysis and extraction of actionable insights. The main challenges refer to understanding how to obtain more accurate and reliable measurements, managing the collection of such a huge amount of data, understanding how the collected data can be used in a perspective of continuous improvement, transmitting the information through complete and intuitive visualizations and, above all, understanding which are the best actions to perform based on the collected data. In the data-driven era, data, “the black gold of the twenty-first century”, become the guide for decision-making in both the private and public fields.

Devices do not represent the only source of useful data, also social media can contribute to the collection of worthwhile data for the extraction of insights. Police reports, as well as newspapers, may be exploited for the analysis of events in the city, such as road accidents, crimes. This could allow monitoring roads with a high number of accidents and consequently changing the road traffic rules, if needed. Besides, feedback from citizens is an invaluable source for understanding the wellness in the city.

Cities are playing a central role as drivers of innovation. Public Administrations are facing the challenges of getting accurate, comprehensive and constantly up-to-date information to ensure socio-economic development and reliable services. Besides, the aim is not only to improve the quality of life of the inhabitants but also to predict and tackle future challenges. A city that exploits technology and data-driven approaches for decision making is referred to as *Smart City*. The deployment of a wide network of sensors becomes of interest to the Public Administrations to allow continuous monitoring in several fields, such as transportation, infrastructure, sustainability, safety. In the end, the use of technology complemented by data analysis may allow to better evaluate the performance of city services, the degree of satisfaction of citizens and businesses in the work of Public Administration, the wellness of citizens. Further evolution of the concept of Smart City is identified through the concept of *Safe City*, which represents a city adopting a data & technology-driven approach to improve the safety of citizens. The use of video surveillance systems is the most popular example of a data-driven solution. Such solutions would allow identifying illegal activities in a short time and speeding up the intervention of authorities. This may result in a decrease in the number of criminal activities.

Within the definition of Smart City, citizens and local businesses are active promoters of the technological evolution of the city. In this perspective, they should have access to data collected by Public Administrations. However, Open Data are still lacking or even missing at all. For this reason, the European Union has issued the “Directive (EU) 2019/1024 on Open Data and the reuse of public-sector information”¹ on the 20th June 2019. The document revises the “Directive 2003/98/EC on the reuse of public-sector information” and imposes the publication of data related to public sectors of the cities as Open Data. As a result, citizens and businesses have a wide range of data at their disposal. Public Administration as well as citizens and businesses are the real driving force of the transition, first, from city to Smart City and, later, towards the concept of Safe City. Open Data increase transparency and stimulate citizens involvement; on the other hand, they also stimulate the economy by encouraging companies that use Open Data in their business activities. The availability of open government data can improve public services and spur inclusive economic development. For example, greater access to traffic data can be used to tackle sustainable mobility needs.

This thesis contributes to the design of data-driven solutions for urban social problems. The purpose is twofold. On the one hand, the scope is the ingestion, management and analysis of sensor data for the road traffic control and urban air quality monitoring, the evaluation of several techniques for anomaly detection on real-world sensor data, and the publication of

¹<https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32019L1024>

sensor measurements as Linked Open Data. On the other hand, the aim is to extract from newspapers semantic information related to crimes occurring in a city, derive useful analysis, and made them available for crime analysis purposes to citizens.

The work is divided into two parts.

Part I presents the sustainability solutions related to the study of the impact of road traffic on air quality through the development of a network of sensors. The work discussed refers to the TRAFAIR project, co-financed by the European Union. This is the first project with the scope of monitoring in real-time and predicting air quality on an urban scale in 6 European cities, including Modena, a medium city in the north of Italy. The final scope of the project is the release of Open Data related to the monitoring and forecast of air quality. The research objective is to build a robust and scalable Smart City data platform and provide extensive analysis and evaluation of techniques to identify anomalies, i.e., data that deviate from the normal behavior of the sensors. The latter represents a challenging research problem in order to ensure good data quality, especially when using low-cost sensors. Chapter 2 presents the goal and the pipeline of the project, focusing on the traffic simulation model and the air pollutant dispersion model. This chapter is crucial to understand the complexity of the processes. In Chapter 3 the implementation of the Smart City data platform employed in the project and shared by all the partners is discussed, providing some details on the conceptualization of the data model. Chapter 4 describes in details the traffic sensors and the air quality sensors installed in Modena, including the process of data collection and the exploratory sensor data analysis. The characteristics of the sensors are depicted, highlighting their advantage, but also limitations and criticalities. Then, in Chapter 5 some techniques to detect anomalous sensor data are investigated along with the proposal of novel approaches and extensive evaluation on the TRAFAIR traffic and air quality datasets. In the end, Chapter 6 discusses an experiment for publishing semantic traffic sensor data through the mapping on existing ontologies that have been enriched with new properties.

Part II refers to the concept of Safe City and presents a crime analysis project. Crime analysis is the set of analytical processes directed at providing timely and pertinent information to assist the police in crime reduction, prevention, and evaluation. In Italy, police reports about the crimes occurring in a city are private documents. Therefore, they are not made available for citizens. The reports of the Italian National Institute of Statistics (ISTAT)² provide a clear picture of the types of crime happen in each province during the year. However, the information provided is aggregated by time and space and become available after (at least) one year from the crime event happening. Based on these official statistics, it is not possible to perform an up-to-date analysis of the local situation in each neighborhood. Due to the lack

²<https://www.istat.it/en/>

of official data to produce the analysis, this project exploits the news articles published in online newspapers. The research objective is to provide a comprehensive framework that integrate solutions to several sub-problems of Natural Language Processing. The goal is to categorize the news articles based on the crime category, geolocate the crime events, detect the date of the event, and identify some relevant features (e.g., what has been stolen during the theft). The Italian Crime Analysis framework³ has been developed for the analysis of news articles, the extraction of semantic information through the use of NLP techniques, and the connection of entities to Linked Data. The emerging technology of Word Embeddings has been employed for the text categorization. Finally, a tool has been developed to show the geolocalized events and provide some statistics and annual reports. The proposed approach is described in Chapter II, while each step of the Italian Crime Analysis framework is detailed in Chapter 8. In the end, Chapter 9 discusses the successful application of the framework for crime analysis in the city of Modena.

After the two main parts presented above, Chapter 10 points out the conclusions. In closure of this thesis, there is the bibliography. Related publications by the author are reported before the introductory Chapter.

³<https://github.com/federicarollo/Crime-event-localization-and-deduplication>

Part I

Sustainability Solution

Chapter 2

TRAFair project

This chapter is devoted to describe the TRAFair European project and provide context to the research activities I have been involved in during the PhD and the participation in the project.

The TRAFair project started at the beginning of the first year (November 2018) and ended during the third year of my PhD (April 2021). I was involved in this project along all its duration and I led two tasks: Task 1.2 Development of tools for the automatic/semi-automatic population of a unified input dataset and Task 6.2 Web platform. After the end of the project, I continued to work on it in the following months to improve some processes and to ensure the continuity of the project. In the following, an overview of the goals of the project (Section 2.1) along with the employed software and the generated data (Section 2.2 - 2.4) is provided, while Chapters 3, 4, 5 and 6 will focus on my contributions to the project (the creation of a unified data platform, the ingestion and analysis of sensor data, the anomaly detection on sensor data, and the semantic enrichment of sensor data).

The work presented in this chapter has been published in [1].

2.1 Goal and pipeline of the project

The TRAFair project¹ aims to compensate for the lack of data and tools to estimate the level of pollution on an urban scale by increasing the sensors deployed in the cities. Nowadays, the situation about air quality is particularly critical in some member states of Europe that cannot reach the European objectives. As a step towards improving air quality, in 2013, the European Commission adopted a Clean Air Policy Package, including a Clean Air Programme for Europe² setting objectives for 2020 and 2030, and accompanying legislative measures. In

¹<https://trafair.eu>

²<https://www.eea.europa.eu/policy-documents/a-clean-air-programme-for-europe>

February 2017, European Commission warned five countries, among which Spain and Italy, of continued air pollution breaches.³ These countries fail to address repeated breaches of air pollution limits for nitrogen dioxide (NO_2). The European Commission urged these Member States to take action to ensure good air quality and safeguard public health. This effort is also required by the sustainable development goals (SDGs) defined in the 2030 Agenda for Sustainable Development [2]. Monitoring air quality is of primary importance to encourage more sustainable lifestyles and plan corrective actions. Moreover, since nearly 40% of NO_x emissions is caused by road traffic,⁴ the TRAFAIR project provides a framework for traffic monitoring, real-time urban air quality control, and air pollution forecasting at 24 or 48 hours [1, 3].

The TRAFAIR project implements two methods: (1) the use of low-cost sensors distributed on the urban area, all over around the city, and calibrated starting from the data of the legal stations for the measurement of pollutants in different urban areas, and (2) the use of simulation models to build an urban traffic model based on real-time data, in order to provide the forecast of urban air quality based on weather forecasts, traffic emissions and other emission sources.

The TRAFAIR project aims to estimate the level of pollution on an urban scale by producing the following main results:

- the provision of real-time estimates of air pollution in the city on an urban scale;
- the development of a service for forecasting urban air quality based on weather forecasts and traffic flows;
- the publication of Open Data to describe maps of urban air quality and forecast maps in 6 European cities: Zaragoza (600,000 inhabitants), Florence (382,000), Modena (185,000), Livorno (160,000), Santiago de Compostela (95,000) and Pisa (90,000).

Ten partners were involved in the project: 4 Universities, 3 Public Administrations, 1 HPC center, 1 company, and 1 research center. More than 70 persons worked in the projects with different expertise and backgrounds: they are mainly computer scientists, environmental experts, HPC experts and public administration managers.

Figure 2.1 shows the way information flows within the project: which input data are collected and by which models they are used, and what kind of outputs are made available. As can be seen in the middle of the Figure, the main models implemented in the project are the traffic model and the air pollution dispersion model that are used in each city.

³http://europa.eu/rapid/press-release_IP-17-238_en.htm

⁴Air quality in Europe - 2020 report: <https://www.eea.europa.eu/publications/air-quality-in-europe-2020-report>

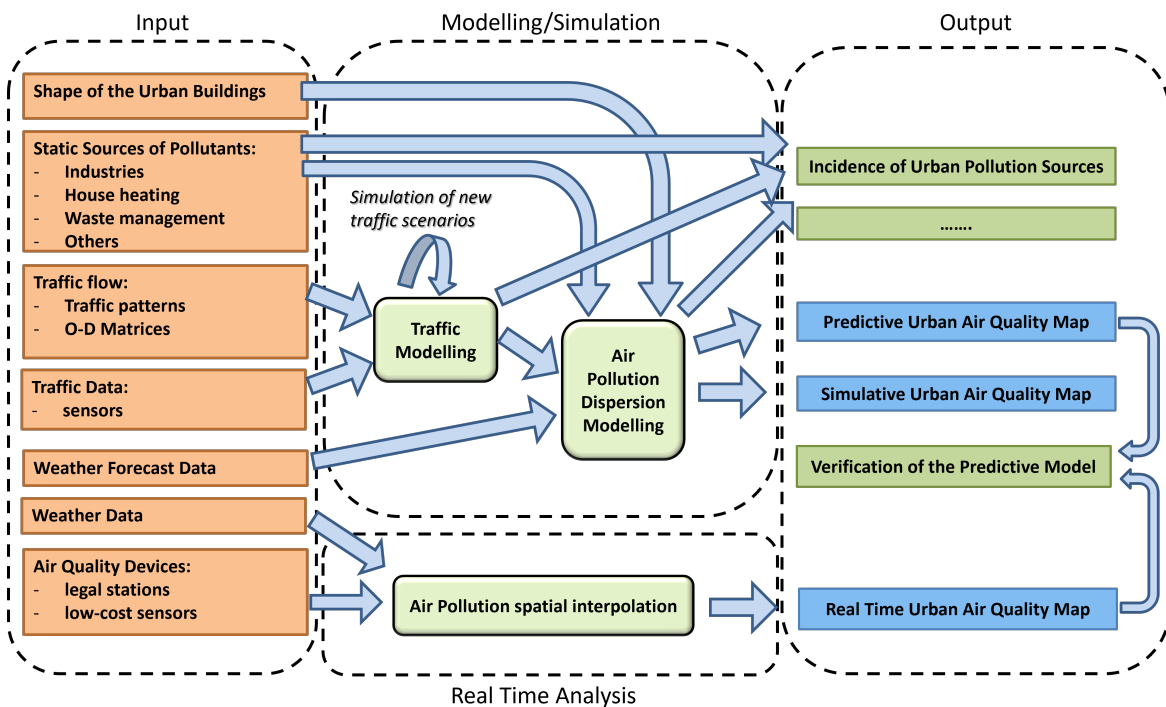


Fig. 2.1 Overview of the data flow in the TRAFAIR project.

Mainly, I contributed to the conceptualization of the data model for the input and the output, and the management of the sensor data including the data cleaning (anomaly detection) processes.

The traffic model takes in input the vehicle counts generated by the traffic sensor and aims at simulating the traffic flow in each road of a city, even where no traffic sensor is available. Starting from the simulated traffic flow, it is possible to calculate the vehicular emission. For this scope, the vehicular fleet composition of the city of interest needs to be known along with the emission factors. The latter are the real-world vehicle emissions and depend on a large number of factors, such as engine size, emission standard, fuel, class, speed, driving style, vehicle load, engine oil type, engine oil age, engine maintenance, engine age, tyre pressure, road surface, road slope, air conditioning. Therefore, any vehicular emission estimate is a “model”, which combines all the above-mentioned factors to provide estimation of the emissions. The most common procedure is the application of the COPERT methodology, i.e. emissions based upon the COPERT database.⁵

Air pollution modelling is a mean used to describe the causal relationship between emissions, meteorology, atmospheric concentrations, deposition, and other factors. Air pollution measurements give quantitative information about ambient concentrations and

⁵<https://www.emisia.com/utilities/copert/>

deposition, but they can only describe air quality at specific locations and times, without giving clear guidance on the identification of the causes of the air quality problem. Air pollution modelling, instead, can give a complete deterministic description of the air quality problem in an urban context, including an analysis of factors and causes (emission sources, meteorological processes, and physical and chemical changes), an evaluation of future scenarios, and some guidance on the implementation of mitigation measures. This makes air pollution models indispensable in regulatory, research, and forensic applications. The concentrations of substances in the atmosphere are determined by 1) emissions, 2) transport, 3) dispersion and diffusion, 4) physical and chemical transformation, and 5) deposition.

The TRAF AIR project provides both the results of modelling and measuring activities: a daily forecast of the pollution map and a semi real-time air quality information within the urban area.

As shown on the left side of Figure 2.1, the primary input is the data provided by the sensor networks: traffic flow data, weather forecast, atmospheric pollution condition. Moreover, the atmospheric emission inventory, the vehicular fleet composition, a 3D model of the urban area (shape of the buildings) are collected for each city. These inputs are used by the traffic model to produce vehicular traffic flow maps, by the air pollution dispersion model to provide a daily forecast of the air quality within the urban area and by an interpolation function to produce real-time urban air quality maps. Each city involved in the project collects the data available for that city and implemented a network of traffic and air quality sensors.

Static source of pollutants can be found on Open Data repositories at a European, national or regional level. At European level, the European Environment Agency (EEA) maintains the European Pollutant Release and Transfer Register (E-PRTR); in Italy, the reference national emission inventory is distributed by the Italian National Institute for Environmental Protection and Research (ISPRA), however the municipality of Modena has its own bottom-up emission inventory; in Spain, the national emission inventory (Registro Estatal de Emisiones y Fuentes Contaminantes) is maintained by the Ministry with competences in environmental monitoring and control (Ministerio de Agricultura y Pesca Alimentación y Medio Ambiente). The inventories are also maintained at a regional level in Spain. Thus, the region of Aragon maintains its PRTR and the region of Galicia maintains also an inventory of emissions (Registro Galego de Emisións -REGADE-).

Weather forecast simulations for each city are provided by either national or regional meteorological agencies, in order to have models tailored to the local domain, with a spatial resolution up to 1 km or less: the air pollution dispersion model provides the vertical profile (20 levels) within the urban domain temperature, humidity, wind speed and wind direction,

with an hourly time step. In addition, some main turbulence parameters are also provided by the model (e.g. friction velocity, Monin-Obukhov length and convective velocity scale) as well as the depth of the mixing layer and the local stability class, which are needed by the air pollution dispersion model used within the project. This latter model uses as input the outcomes by the weather forecast model, the traffic simulation model, and, eventually, other local atmospheric emission sources and it provides an hourly dispersion map for the urban area, with a spatial resolution of 2-4 meters for the following day.

The final goal is to create a short-range predictive modelling chain (24/48 hours) to simulate the atmospheric dispersion of NO_x emissions for the 6 cities. The TRAFair outputs provide a possible, complete, reproducible representation of the urban air quality, easily adaptable at the international level since multi-language, and compliant with the best practices and standards defined by FAIRMODE, W3C, OGC, and INSPIRE. The project, deployed on 6 European cities of different size, provides a flexible and adaptable service to show the current and predictive air quality on an urban spatial scale. Other cities can implement the same services after collecting the necessary input data and generate the expected outcomes.

2.2 Software and data tiers

The software architecture of the TRAFair framework is organized in four different tiers, as shown in Figure 2.2. I worked on the Sensor Data Acquisition tier and the Data Management tier.

At the bottom, a Sensor Data Acquisition Tier (the first layer from the bottom in Figure 2.2) provides functionality to sample the required data from the deployed traffic and air quality sensor networks to populate the TRAFair data platform (see Chapter 3) in the Data Management Tier (the second layer from the bottom in Figure 2.2). The Data Management Tier is completed with a Geospatial Data Warehouse and a Scientific Array Data Warehouse, that record respectively city geospatial data (including streets and buildings) and all environmental models collected and generated in the project. Well-known standards from the OGC and W3C are used to provide open access to the Data Management Tier. The three models implemented in the project are contained in a Data Processing Tier (the third layer from the bottom in Figure 2.2). Advanced HPC hardware and technologies are used to implement this tier. Finally, the User Interaction Tier (the top layer in Figure 2.2) provides multichannel GUIs for end-user applications, including web and mobile applications.⁶ Besides, geographic

⁶TRAFair dashboards: <https://trafair.eu/dashboards/>

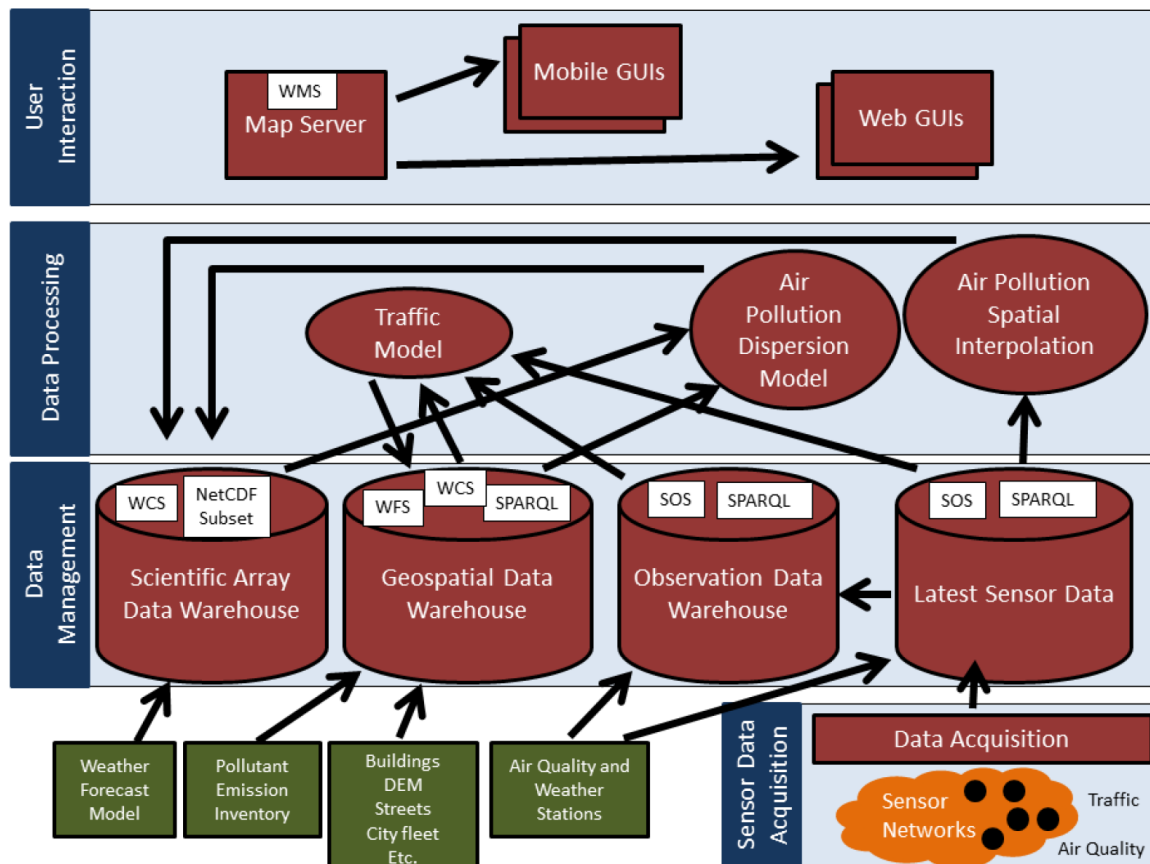


Fig. 2.2 Overview of the software tiers in the TRAF AIR project.

maps generated from the result model data are published through the well-known OGC Web Map Service interface, following INSPIRE guidance.

The implementation of the different data management and processing components of the above architecture requires High Performance Computing (HPC) technologies. In particular, the Air Pollution Dispersion Model and the Traffic Model demand hours of elaboration and large-scale data processing technologies.

2.3 Traffic model

The large number of vehicles moving in a city complicates urban planning and often leads to traffic congestion and areas of increased air pollution due to vehicle emissions. Traffic management solutions typically require the use of simulators able to capture in detail all the particular characteristics and dependencies associated with real-life traffic. Traffic models simulate the appearance of a traffic system integrating mathematical models, as described in [4].

In 2017, the European Commission (EC) established the latest Guidelines for developing and implementing a Sustainable Urban Mobility Plan (ELTIS guidelines). Through these guidelines, the EC asked the European countries to define a Sustainable Urban Mobility Plan. The scope of the plan is to satisfy the mobility needs of people and businesses in cities and their surroundings for a better quality of life. In particular, the strategic plan aims at improving road safety, reducing air and noise pollution, decreasing greenhouse gas emissions and energy consumption. Therefore, Public Administrations of the cities over 100,000 inhabitants need to make analysis of the road traffic to improve mobility and avoid traffic congestion. For this reason, traffic models are employed to understand how vehicles move in the cities.

In Italy, the acronym used for the Sustainable Urban Mobility Plan is PUMS. In Modena, in the latest released PUMS⁷ the traffic of an average working day of the year was evaluated. The data used for the evaluation have been obtained by the black boxes placed inside the cars. In this case, the traffic model employed did not aim to describe the variation of traffic in each day, but it provides an approximation over the year of the traffic flows with an error around the 8-9%. To better evaluate the impact of road traffic on air quality that is one of the TRAFAIR scopes and to obtain a more realistic traffic analysis, it is important to have detailed information about traffic, to know how the traffic changes during the day (daily trend) and the week (weekly trend), when and where congestions occur every day, how traffic changes based on the season. For these scopes, the integration of real-time traffic data in the traffic model is needed. Traffic data can be earned from several types of traffic sensors, usually traffic cameras or induction loops sensors positioned around the city or alternatively, source of traffic flow can be produced by users, such as the above-mentioned insurance black boxes placed inside the cars, or GPS tracks stored and used in applications such as Google traffic, INRIX traffic, Open Traffic, or Bluetooth receivers installed in the traffic lights [5, 6]. Each city involved in TRAFAIR as well as Modena is equipped with a network of traffic sensors that are used by the traffic model.

In Modena, the traffic model employed is the Simulation of Urban MObility (SUMO) tool⁸ that is a microscopic, collision-free, space-continuous and time-discrete simulator [7, 8]. It is open-source, highly configurable and designed to handle with large road networks. SUMO has been continuously developed for more than 15 years and has been extensively successfully applied in different projects related to urban traffic management, traffic emission and other traffic issues [9]. As a micro-traffic model, every vehicle that moves within the simulated network is modelled individually; while the macro-traffic models simulate vehicles

⁷PUMS 2030 of the city of Modena: <https://www.comune.modena.it/servizi/mobilita-e-trasporti/pums/documenti-pums/pums-2030>

⁸<https://sumo.dlr.de/index.html>

as groups or flows, moving all in the same way and with the same route as a fluid in a tube. The speed and the acceleration of the vehicle are updated depending on the traffic state of the road network. As described in [10], also the street restrictions, such as maximum velocity and right of way rules, are taken into account. SUMO supports a lot of different inputs: routes derived from sensor data, routes derived from an origin-destination matrix, GPS routes and also random routes. All of them can help to enrich the simulation and adapt it to the case scenario. In the case of Modena, the data collected by more than 400 traffic sensors have been used. More details about the type of sensors and their measurements are provided in Chapter 4.

In order to set up the model, it is important to define some SUMO objects: calibrators and virtual traffic detectors. Calibrators are used to redirect vehicle routes [11]. Their position corresponds to the position of the real traffic sensors. Each calibrator monitors the number of vehicles passing through it, removing vehicles or inserting new vehicles if the flow value measured by the real traffic sensor in that position is different from vehicle counted in the simulation, and modifying the speed of vehicles if their detected speed is different from the average speed measured in reality. The virtual traffic detectors, instead, are used to count the simulated traffic in several points.

An overview of the traffic model implementation is depicted in Figure 2.3. In [7], a detailed description can be found.

The input of the model consists of the traffic sensor position, the data generated by the sensors and the road network of Open Street Map. The main output produced by SUMO is an XML file that includes information about vehicle count, lane density and average speed for every road in the map with a predefined frequency, e.g. every minute. The generated output provides a possible, complete representation of the traffic in the urban area of Modena. If virtual induction loop detectors are inserted in the simulation, another output file is produced, containing the exact counts and speed of vehicles for every minute of simulation and every virtual induction loop detector, thus potentially in every road of Modena. Besides, other types of output can be generated, such as video, graphs and images, managing the XML file. The model output can be used to analyze the real-time traffic situation in the city of Modena, to highlight the more congested zones, to identify trends, and to calculate the vehicular emissions.

Figure 2.4 shows the traffic flow simulated by SUMO in the roads of Modena. The orange colour represents roads with a high number of vehicles, while green roads are the ones where there is low traffic.

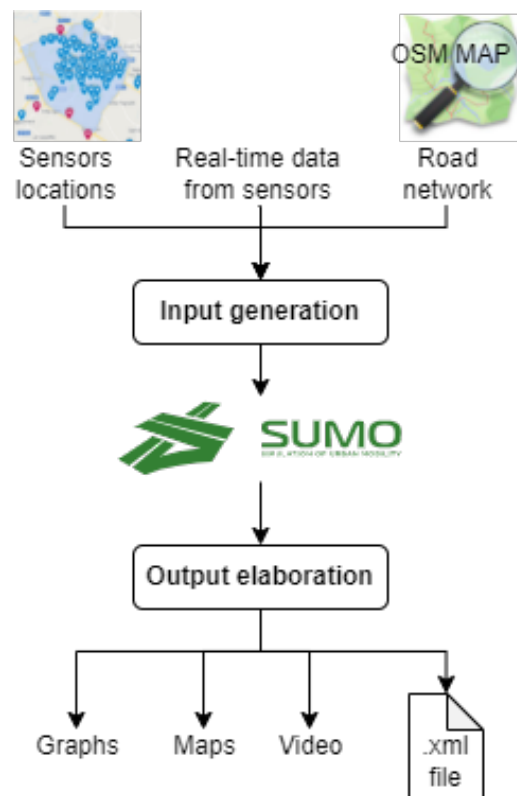


Fig. 2.3 Overview of the traffic model implementation.

2.4 Air Quality model

In Italy and in Spain, local or regional environmental protection agencies provide weather and air quality analysis referred to the respective areas observed and monitored, along with the forecasts for the following days.

In the Emilia-Romagna region, ARPAE publishes daily average of PM_{10} and $PM_{2.5}$, hourly concentration of NO_2 , NO , O_3 , CO , BTX and other regulatory pollutants, and summarize them in an Air Quality Index (AQI). ARPAE also makes use of the NINFA air quality predictive system, based on the combination of the chemical transport model Chimere [12] and the meteorological model COSMO [13]. Air quality analysis at a regional scale by the PESCO (Postprocessing and Evaluation with Statistical techniques of the Chimere Output) tool, combining NINFA products with ground-based observations [14]. Other national and European forecast modelling system are also available, an example is the QualeAria⁹ project developed by ARIANET (consulting firm operating in the environmental field) and ENEA (Italian Agency for New Technologies, Energy and Sustainable Economic Development), which main purpose is to forecast regional scale air pollution over the Italian peninsula.

⁹<http://www.aria-net.it/qualearia/it/>

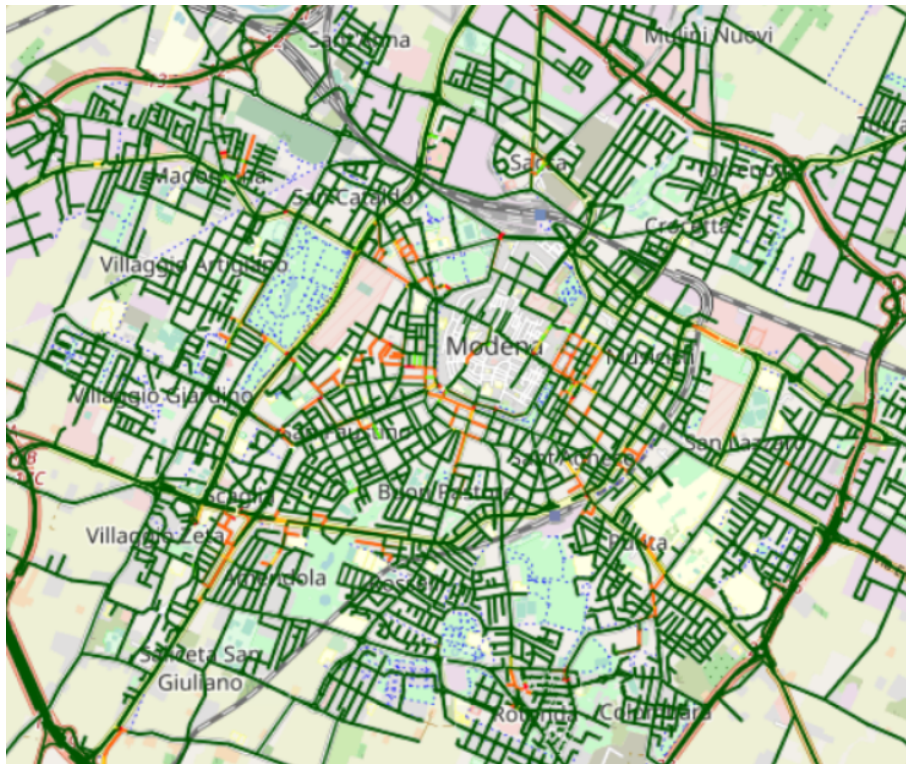


Fig. 2.4 Modena map with roads of different colours based on the SUMO output at the 6th January 2022.

It is based on FARM (Flexible Air quality Regional Model) [15], a 3D Eulerian model simulating dispersion and chemical reactions of atmospheric pollutants, managed by F-Air (ARIANET Integrated Forecast System Manager), a software environment where the data sources and the modelling components are connected. QualeAria provides for the Italian and European countries hourly average, daily and daily average concentrations for regulatory interest pollutants: SO_2 , NO_2 , CO , PM_{10} , O_3 and benzene.

Anthropogenic emissions considered by the above-mentioned systems are based upon national (ISPRA) and European (e.g. TNO-MACC) emission inventories, and they estimate seasonal and daily source variability through ancillary data, leading to an approximated description of time-dependent sources, e.g. traffic. Moreover these models are performed on large scale domains (about 300-1000 km) with a horizontal resolution of few kilometers, while the focus of TRAFAIR is to provide analysis and prediction on an urban scale. Therefore, TRAFAIR analyzed a comprehensive set of traffic flow simulations based on real-time data to generate emission estimates tailored to the city and the time of the year.

The dispersion pollutant model employed in TRAFAIR is the Graz Lagrangian Model GRAL [16] which can simulate the dispersion of chemically non-reactive pollutants, dry

deposition and sedimentation over the full range of wind speed without any lower threshold and for all atmospheric stability conditions. The model allows different type of case studies, with domains down to a region of few square kilometers with a horizontal resolution of few meters. GRAL is able to account for the presence of buildings inside the computational domain and can simulate emissions from point, area and line sources, in continuous or discontinuous way.

In Modena, an average traffic situation is calculated for weekdays, Saturdays and Sundays of each month based on the traffic flows simulated by SUMO. These averages are used by a customized version of the open source model Vehicular Emissions Inventory (VEIN) [17] to calculate the emission generated in each road of the city for each hour of the day. Finally, using the weather 48 hours forecast produced daily by ARPAE¹⁰, the GRAL air pollution model is executed to create predictions for the next 48 hours. A similar approach was followed in the other cities involved in the project.

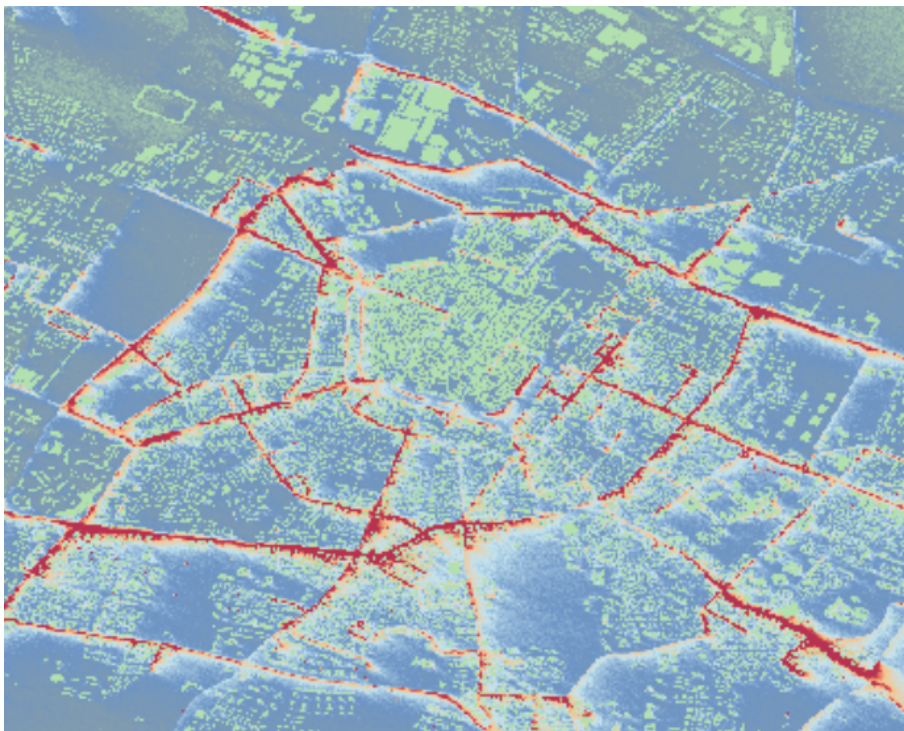


Fig. 2.5 Modena map with the output of the GRAL model at the 6th January 2022.

Figure 2.5 shows the map of Modena with areas of different colours based on the output of the air pollution dispersion model. Red colour indicates the areas with the higher concentration of pollutants. Since the input of the model mainly consists of the traffic data, red areas are often in correspondence of more congested roads. The map describes the impact

¹⁰<https://www.arpae.it/>

of road traffic on air quality in Modena. These results can raise the awareness of the citizens that work, live or transit in the city or in a specific area.

The results of the TRAF AIR project enriched the regional,¹¹ national¹² and European Data Portal¹³ adding a new kind of air quality datasets. Along with the time-series of urban air quality forecast and the maps obtained by these data (like the one in Figure 2.5), the project produced the time-series of atmospheric pollutants concentrations at specific sites generated by the low-cost devices and the interpolation maps obtained by these data for all the 6 TRAF AIR cities. A set of videos showing the project results are also available on the TRAF AIR website.¹⁴

¹¹Emilia-Romagna Open Data portal: <https://dati.emilia-romagna.it/>

¹²Italian Open Data portal: <https://www.dati.gov.it/>

¹³European Open Data portal: <https://data.europa.eu/>

¹⁴<http://trafair.eu/models/>

Chapter 3

A Smart City data platform

The pipeline and the data flow of TRAF AIR described in the previous Chapter make clear the need of a robust and flexible data platform able to collect all the input data of the employed models and their generated outputs. The large amounts of data generated by the project are complex both in structure and also in semantics. Moreover, the database needs to deal with heterogeneous data, e.g., time-series, spatial data, shape files. The scientific contribution of this work relies on the integration of heterogeneous data in a unified and scalable data platform.

This chapter focuses on the requirements and architectural choices for the design of the data platform and provides some analytics on the data storage. Besides, I explain the mechanism set up to monitor the correct interconnection of the processes implemented for the scopes of TRAF AIR and the proper functioning of the database.

Part of the work presented in this chapter has been published in [18].

3.1 Requirements

A comprehensive set of state-of-the-art applications in the field of smart city has been studied to define a list of key-points. Based on this analysis, some requirements on the data to store emerged. The data platform needs to support the storage of:

- geometry of a city, i.e., buildings, roads,
- position and measurements of in-situ devices (e.g., air quality stations, traffic sensors, cameras) which collect data in a specific point,

- position and measurements of in-situ removable devices (e.g., low-cost air quality sensor) which can be moved in different locations and provide measurements in a static position,
- results of static models that describe a situation at a specific time, such as the interpolated air quality maps that are generated by interpolating sensor measurements related to a specific time interval,
- results of dynamic models, such as the atmospheric dispersion model and the traffic simulation model, which describe a phenomenon changing over time.

The data generated by the sensors are geolocated time-series because they have a reference in time and space, other data like the traffic flow simulated by the traffic model are time-series because they describes an evolution over time. Therefore, the data to store in the data platform are mainly spatial data and time-series. The amount of data to store will increase over time since sensors produce new data every day, the traffic flow is calculated every day based on new sensor data as well as the air pollution forecast, and the historical data will be maintained for at least three years. A time-series is a chronological sequence of observations on a particular variable. Time-series data have particular features; they are large in data size, have high dimensionality, and update continuously. Various mining tasks can be performed on time-series data: clustering, anomaly detection, visual representation, short-term and long-term prediction, and others. Geospatial (or spatial geo-referenced) data describe features on the Earth's surface. Data that have both spatial and temporal dimensions are called Spatio-Temporal data (ST), this is also the case of sensor data. ST data have auto-correlation in both space and time: instances are related to each other and their properties vary in different spatial regions and time periods. Managing ST data means handling issues related to both temporal and spatial data.

After the analysis of the TRAF AIR project goals reported in Chapter 2, other requirements have been defined: short response time and scalability. Due to the strong interconnection of the TRAF AIR pipeline and the high number of processes that communicate with the data platform, a database robust to a significant number of simultaneous connections is necessary. Another requirement is to write new data very quickly because the sensor data are stored in real-time in the data platform and then they are used by the simulation models. To obtain the simulated traffic flow and the interpolation maps of the atmospheric pollutants concentrations in real-time, the time between the generation of the sensor measurements and the availability of them in the data platform need to be short. Since the TRAF AIR cities have different size, scalability in the data platform is needed because also the amount of data to store changes.

Indeed, the scope was to define a unified data platform that is shared by all the TRAFair cities and could be used by other cities interested in the goals of the project.

3.2 Architectural choices

With the growing diffusion of IoT devices, the ST data availability has also increased and a plenty of time-series databases has been developed. A time-series database (TSDB) is a database optimized for storing time-stamped or time-series data and for measuring change over time. Some examples are TimescaleDB, InfluxDB, Prometheus, Graphite, OpenTSDB, DolphinDB.¹ Choosing the best database could be difficult since the selection depends on a lot of factors and comparing the performance of the available databases is a time-consuming task.

After some analysis and comparison, the choice was to use PostgreSQL [19] exploiting the Timescale extension to perform better with very large tables containing time-series and the PostGIS extension for handling geospatial data. Besides, PostgreSQL provides the command “copy” to copy data from an external source to the database, or vice versa, saving a lot of time in write operations.

PostGIS [20] supports the use of specific data type to store geospatial data. The data types include geometry, geography, and raster. In particular, PostGIS allows the storage of points, lines, and polygons. In the TRAFair database, PostGIS was used to store the positions of the sensors that are indicated as points (couples of GPS coordinates), the roads of the city that are represented as lines, and the buildings that are polygons. Besides, several functionalities are exploited to perform spatial queries like obtaining the roads within a neighborhood or identifying in which road a sensor is located.

Timescale² is crucial to make SQL scalable for time-series data and for providing fast analytics, and scalability on PostgreSQL DB. This extension can be applied on each table with a temporal reference, i.e. a column with a date/time. When applied, the table is transformed into a hypertable and its content is split into several chunks based on the value of the timestamp and the time interval chosen by the database administrator. Each chunk contains the data related to the specified time interval. The value of the time interval depends on the type of the data stored in the table, the amount of data, and the query user intends to make over the data. Sample values can be one day, one week, one month, or a customized time interval (2 days, 15 hours, and so on). The partitioning is completely transparent

¹An independent website, DB-Engines, ranks database popularity based on search engine, social media mentions, job postings, and technical discussion volume. Ranking of time-series databases is available at <https://db-engines.com/en/ranking/time+series+dbms>.

²<https://www.timescale.com/>

to the user. Timescale automatically creates a b-tree index on the column containing the timestamp. This approach allows querying time-dependent tables very fast. In previous works [21, 22], Timescale was compared with other time-series databases, demonstrating competitive performances on all the traditional queries. Working on millions of rows, Timescale offers up to 20× higher ingest rates, at the same time supporting time-based queries to be even 14,000× faster. [23]

In the TRAFAIR database, Timescale was exploited in the tables containing the sensor data, the output of the traffic model, and the anomalies on sensor data since these tables store a big amount of data, and their size grows quickly over time. In the following, an evaluation of the advantage of using Timescale is explained.

3.2.1 Timescale performance evaluation

I conducted some tests on the configuration of Timescale to speed up the execution of the most frequent queries and to ensure a rapid response in all the processes of the project, and, in particular, in the dashboards developed to visualize the outcomes of TRAFAIR.³ The performances of these queries have been evaluated in terms of time required considering or excluding the use of the Timescale extension. In the Timescale configuration, the chunk time interval was set to one month. Assuming that one sensor makes a measurement every minute, the total number of observations in one day will be 1,440, and approximately 43,200 in one month. If supposing to monitor 400 sensors, in 2 years these sensors will produce more than 400 million records. Table 3.1 shows the time needed for planning and executing 3 time-dependent queries on a table containing this amount of records:

Q1: selecting the observations of one month,

Q2: selecting the observations of a time interval related to one sensor,

Q3: calculating the number of observations for each month.

As can be seen in the table, the time needed by Timescale is always lower than the one needed by PostgreSQL. This test focuses on time-dependent queries since the queries needed by the TRAFAIR dashboard often depend on the value of the temporal reference (for example, to visualize the measurements of a specific sensor in the time period, or extract the last measurement of each sensor to show the real-time traffic flow). With Timescale the time saved for each of the main queries performed goes from 50 to 99%. These experiments show the advantage of using Timescale.

³Traffic flow dashboard: <https://trafair.eu/trafficflow/>, Air quality dashboard: <https://trafair.eu/airquality/>

Table 3.1 Comparison of time performance with PostgreSQL and Timescale.

		PostgreSQL	Timescale	Time difference
Q1	<i>planning time</i>	0.00005 s	0.002 s	-90%
	<i>execution time</i>	149 s	15 s	
Q2	<i>planning time</i>	0.0001 s	0.004 s	-99%
	<i>execution time</i>	135 s	0.3 s	
Q3	<i>planning time</i>	0.00005 s	0.02 s	-52%
	<i>execution time</i>	1273 s	614 s	

3.3 Database conceptualization

Due to the lack of a common standardized data model which satisfies all the requirements listed above, a new data platform has been defined to store all the information managed in TRAFair project.

The model is based on International standards of the Open Geospatial Consortium (OGC), the International Organization for Standardization (ISO) and the World Wide Web Consortium (W3C). It stores data related to the road network, traffic and air quality sensors' features (i.e. vendor, position, measured variables, and functionalities), sensor measurements, anomalies in sensor measurements, the configuration of the anomaly detection algorithms, the statistics and trends of the traffic sensors measurements, the output and configuration of the traffic and air pollutant dispersion model, the shape of the buildings in the city. The actual structure of the database has also been affected by continuous changes to align with data management, so the database has been created and updated incrementally during the years, ending with the current structure that consists of more than 60 interconnected tables.⁴

In literature, the most popular ontology used to model sensor data is the W3C Semantic Sensor Network Ontology (SSN), which defines a vocabulary to describe sensors and their observations, including both observed values and required metadata of features of interest, observed properties. SSN may be used to annotate sensors and observations, in a way aligned completely to the OGC and ISO Observations and Measurements (O&M) standard, which provides a conceptual schema to represent the results of observation processes and the metadata of these processes, of the entities that are sampled by them (Features of Interest), and of the observation results obtained. Also, concepts and entities are based on SOSA (Sensor Observation, Sample and Actuator) ontology. Thus, the data types used to represent

⁴The database structure is depicted at <https://trafair.eu/database.html>.

the geospatial characteristics of the entities are based on those proposed by OGC standards, including feature geometric data types⁵ and temporal, spatial and spatio-temporal coverages.⁶

The initial conceptualization of the data model has been defined in collaboration of the University of Santiago de Compostela. As leader of Task 1.2 Development of tools for the automatic/semi-automatic population of a unified input dataset, my contribution was mainly related to the definition of the database structure and its implementation through the scripts shared with the other partners of the project, the proposal of further modifications to overcome the need of storing new data, the management of the database structure versioning.

In the following, the database structure for the storage of the most relevant data will be described in details.

3.3.1 Road network

The road network is a key-element in traffic monitoring. In this context, Open Street Map [24] (OSM) is a valid resource that collects geographical data of all the world providing information about the road network for most cities, and allows filtering and download the up-to-date geospatial data based on the use case. OSM relies on Volunteered Geographical Information (VGI) to offer free map data under the Open Database License. In addition, the spread of OSM in the world helps make TRAF AIR reproducible in other cities. Other geospatial services exist such as Google Maps [25], Apple Maps [26], Azure Maps, ArcGIS, HERE maps [27], and TomTom maps [28], however, they do not allow exporting raw map data, but only allow customizing maps to be included in other apps, and this was in contrast with our goal: to export information, such as the geometry, related to each road in the area of interest. Besides, those services are proprietary solutions and allow limited free use of data, requiring payment for a subscription later and imposing restrictions on the data usage and sharing.

OSM has been chosen for the following reasons:

- it is completely free and shared, in accordance with the purpose of TRAF AIR to use open-source services and to share project outputs.
- the roadmap data provided by OSM are satisfactory for the TRAF AIR purposes. According to existing studies, usually cities are expected to be well represented in OSM (e.g., see [29, 30]). According to [29], “VGI can reach very good spatial data quality”; some works have analyzed the quality of OSM (e.g., recent studies of OSM

⁵OGC Simple Feature Access: <https://www.opengeospatial.org/standards/sfa>

⁶OGC Coverage Implementation Schema: <http://docs.opengeospatial.org/is/09-146r6/09-146r6.html>

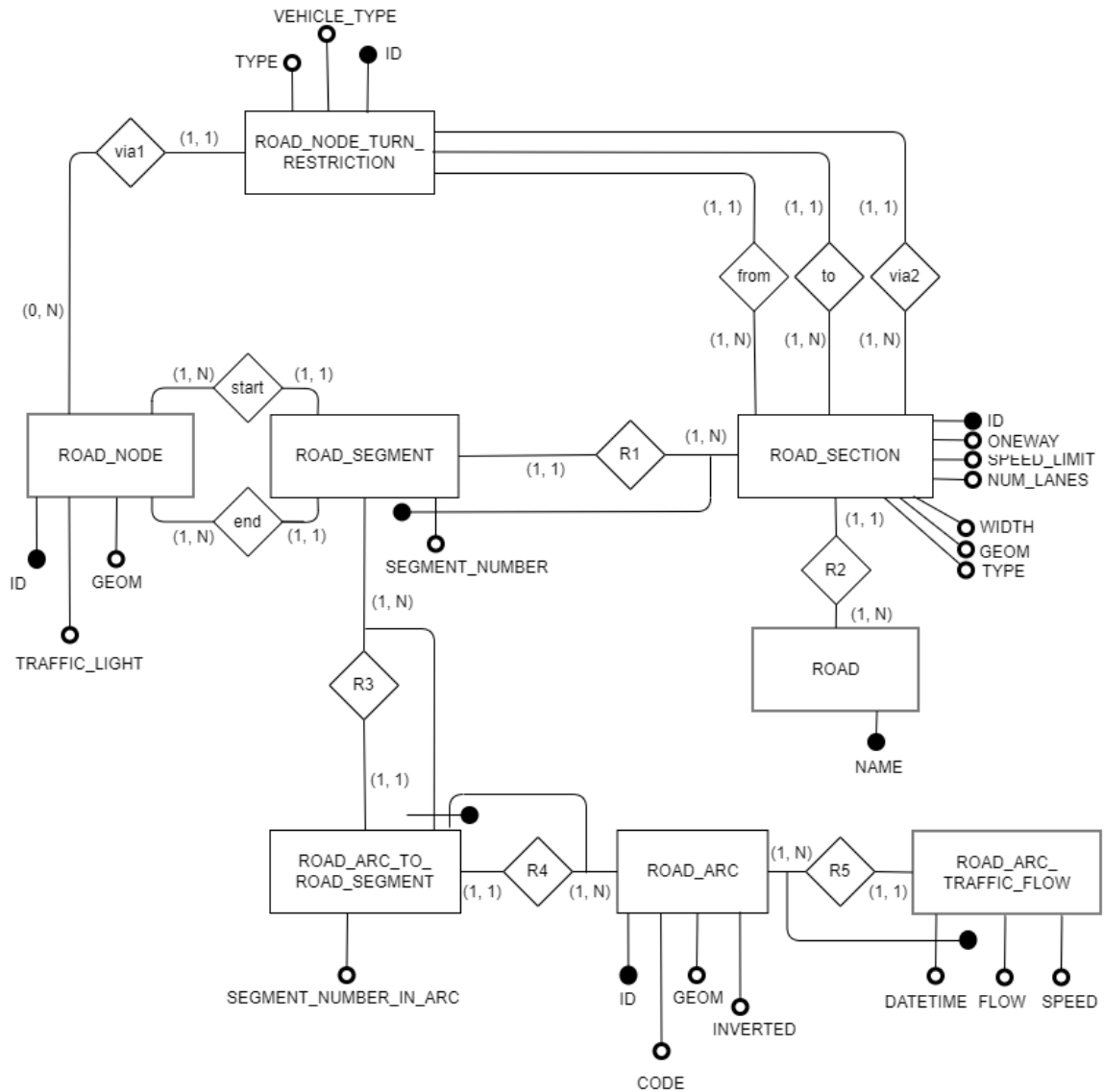


Fig. 3.1 E/R model of the TRAFAIR database to store the road network.

datasets have been presented for Spain [31] and the Lombardy region in the north of Italy [32]).

- it is kept up-to-date thanks to the contribution of volunteers and also allows easy correction of incorrect information as well as an addition of missing information. For example, for the city of Modena, the number of lanes of some roads and whether the road is one-way or not was integrated. This information is crucial to localize traffic sensors and simulate the traffic flow. This aspect also ensures better sustainability and maintenance of the project.

- OSM datasets are a great source of Open Data and can contribute to a more sustainable and transparent modelling [33].

Nevertheless, if more complete and accurate data are required in a project, other roadmap data sources can be used.

Choosing OSM also brought an advantage in the creation of the road network of SUMO, the traffic model employed in TRAFAIR, since SUMO provides ready-to-use packages to import the road network from OSM.

The information in OSM is organized in ways, nodes, and relations. Among the ways, there are the roads which can be of different types: primary, secondary, pedestrian, cycle way, and others. While the nodes represent the points over the ways, the relations are used to model geographic relationships between objects like junctions and traffic restrictions. The most important data for the scopes of TRAFAIR are the road name and the geometry, the number of lanes, the maximum speed allowed, and the roadway directions. The number of lanes and the direction are helpful to geolocate the traffic sensors.

Figure 3.1 refers to the tables implemented for the storage of the road network: the “road_segment”, “road_node”, and “road_node_turn_restriction” tables correspond to the ways, nodes, and relations, respectively. One road is split into more road sections, and one road section into more road segments. The geometries of nodes and road sections are stored into the “geom” attribute of the “road_node” and “road_section” tables, and they refer to points and lines, respectively.

OSM data has been queried by using the Overpy Python library,⁷ which allows filtering data based on the type of element (way, node, or relation) and the attributes associated with each element. The query can be limited to a specific area defined by the GPS coordinates. The obtained data are then stored in the above-mentioned data model. The traffic model exploits the OSM road network and converts it into a collection of road arcs. Consequently, the model output refers to the road arcs. Thanks to the PostGIS spatial functions, the model output has been associated to the OSM road network. A road arc can be a collection of road segments, or a portion of a road segment. To model this complex conversion, the table “road_arc_to_road_segment” has been inserted inside the TRAFAIR database. That table can manage the twofold relation between arcs and road sections. Each entry is an association between a road section identifier and a road arc identifier. The same road section can be in relation with several road arcs and the same road arc can be in relation with several road sections.

⁷<https://pypi.org/project/overpy/>

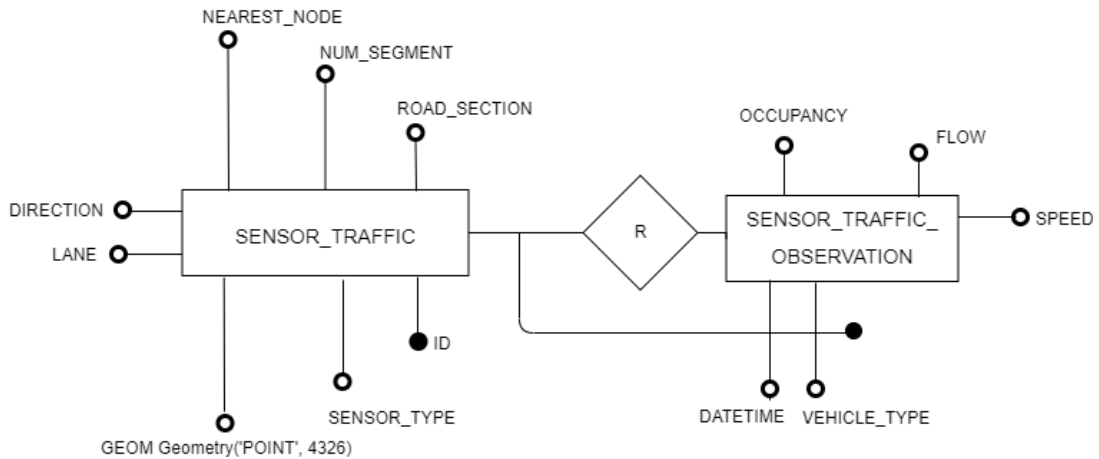


Fig. 3.2 E/R model of the TRAFAIR database to store information about the traffic sensors and their measurements.

3.3.2 Traffic sensor

Regardless of the type of sensors employed, the main data that traffic sensors are able to provide are the number of vehicles in the time interval (traffic flow) and their average speed. In some cases, traffic sensors are also able to distinguish between vehicle types.

Sensor static information is stored on the PostgreSQL database: the identifier, the type (i.e. induction loop or camera, and so on), its position as a point data type of PostGIS, and the position within the OSM road network calculated through the spatial functions of PostGIS. The data model was implemented following the standard “ISO 19156:2011 Geographic information - Observations and Measurements” which defines a conceptual schema for observations and for the features involved during the sampling of the observations [34].

The two entity types used to model the information related to traffic sensors are illustrated in Figure 3.2. The corresponding “sensor_traffic” table stores the identifier of the sensor (id), its type (sensor_type), its position as a point data type of PostGIS (geom), the identifier of the street in OpenStreetMap (OSM) in which the sensor is located (road_section), the sequential number of the specific piece of a street (segment) with the sensor (num_segment), the OSM node which is the closest one to the sensor (nearest_node), the direction of the vehicles counted by the sensor (direction, which is true if it is the same specified by order of the nodes mapped on the street in OSM and false otherwise), and the sequential number of the lane in which the sensor is located (lane, where the value zero indicates the rightmost lane in that direction). The measurements of the sensors are stored in the “sensor_traffic_observation”

table. In particular, the identifier of the sensor (id), the beginning of the sampling interval of the observation (datetime), and the type of vehicles the measurements are related to (vehicle_type), all compose the primary key of the table. For the sensors that are not able to categorize the type of vehicle, the value of vehicle_type is “unknown”. The other attributes are the number of vehicles counted by the sensor (flow), the average speed (speed), and an optional occupancy rate attribute (occupancy, which is an estimation of the time a vehicle is above the sensor). The observation rates can have different values, according to the model of the sensors and their configuration.

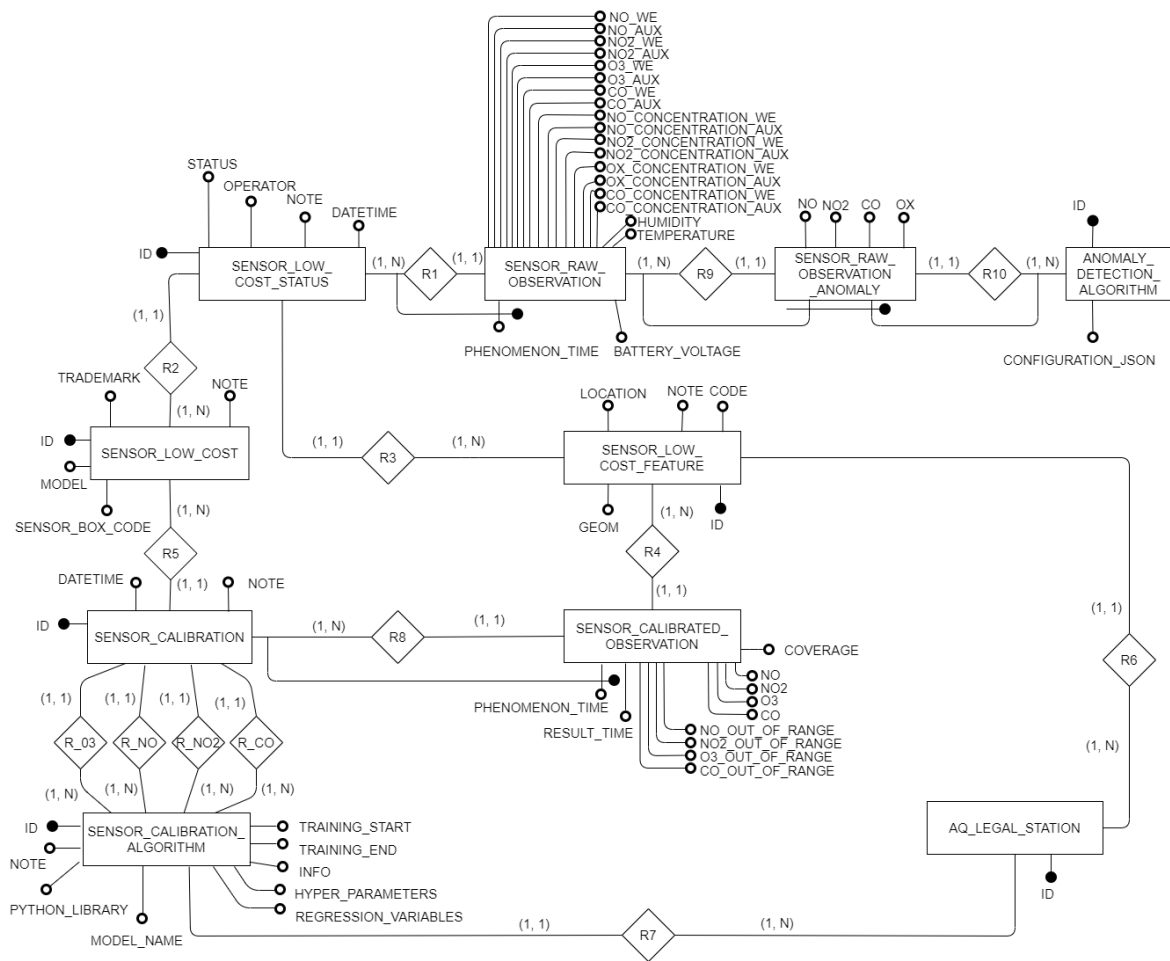


Fig. 3.3 E/R model of the TRAFair database to store information related to the air quality sensors, the measurements, the calibration and the identification of anomalies.

3.3.3 Air quality sensor

The structure of the TRAFair database for storing information related to the air quality sensors, the measurements, the results of calibration and the identification of anomalies

is shown in Figure 3.3. Again, the data model was implemented following the standard “ISO 19156:2011 Geographic information - Observations and Measurements” [34]. The tables store the technical characteristics of each device, its position, its status (running, calibration, offline, broken, warm-up), the raw observations, the concentrations obtained by both the original factory calibration and the calibration algorithm developed in TRAF AIR, and the anomalies identified by some anomaly detection algorithms applied to both raw and calibrated observations.

In each moment, every device is described by a status and is located in a point of interest. Table “sensor_low_cost_feature” collects the name and the GPS coordinates of all possible locations, while in table “sensor_low_cost” the characteristics of each device are stored. Table “sensor_low_cost_status” gives information about where each device is located in each moment, which is its status, the timestamp of the location change, and the name of the operator.

The measurements are stored continuously in the “sensor_raw_observation” table. Each record includes 19/21 measurements: air temperature, humidity, battery voltage, 8 raw measurements (2 measurements per 4 gases: NO , NO_2 , CO and O_x), 8/10 concentrations of the original factory calibration (2 measurements per each gas and one measure for $PM_{2.5}$ and PM_{10}). Each record is associated with a timestamp.

Each raw measurement can be calibrated by multiple calibration algorithms. Thus, calibrated data are identified, not only by the date of the measurement and the sensor that has provided it, but also by the algorithm that was used. One calibration model needs to be defined for each gas. In the end, several anomaly detection algorithms are applied to both raw and calibrated data. The results are stored in the tables “sensor_raw_observation_anomaly” and “anomaly_detection_algorithm” using boolean values to indicate if they are anomalous or not.

3.3.4 Traffic flow analysis and modulation

The traffic flow simulated by the traffic model is analyzed and some aggregation is performed to better understand how vehicles move in the city, identify the trend and generate the input for the calculation of the vehicular emissions, as described in Section 2.1. The modulation of traffic flow is computed in different ways for different scopes, for example, to calculate the average of each day of the week and month (e.g. average Monday in September 2019, average Wednesday in April 2020, and others) or the typical weekday in spring. The “average_traffic_flow” table shown in Figure 3.4 stores the average number of vehicles, their average speed and the flow (average hourly number of vehicles) for every portion of street (piece of road section between two road nodes) in a time interval (between `datetime_start`

and datetime_end) of each day (from Sunday - represented with 0 - to Saturday - 6 -) of every month in a certain year.

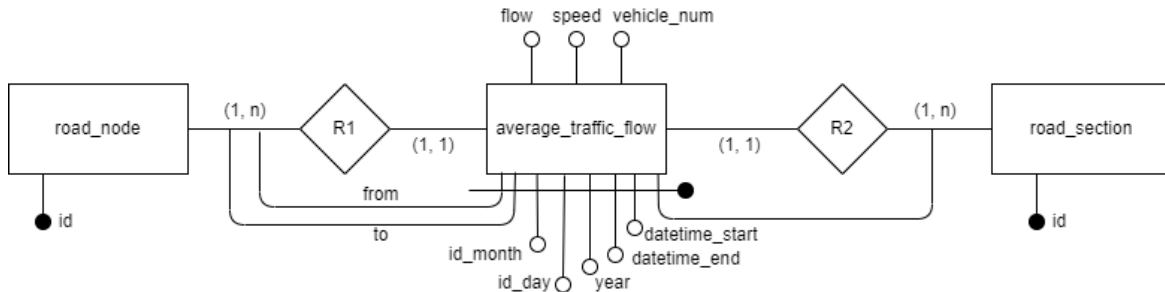


Fig. 3.4 E/R model of the TRAFAIR database to store the daily average traffic flow.

The “average_traffic_flow_modulation” and “average_traffic_flow_modulation_values” tables in Figure 3.5 store the percentages of flow in each hour (flow_percentage is an array of 24 elements, one for each hour) with respect to a reference hour. The percentages refer to a day of the week of a certain month and year and are related to a certain portion of street (piece of road section between two road nodes).

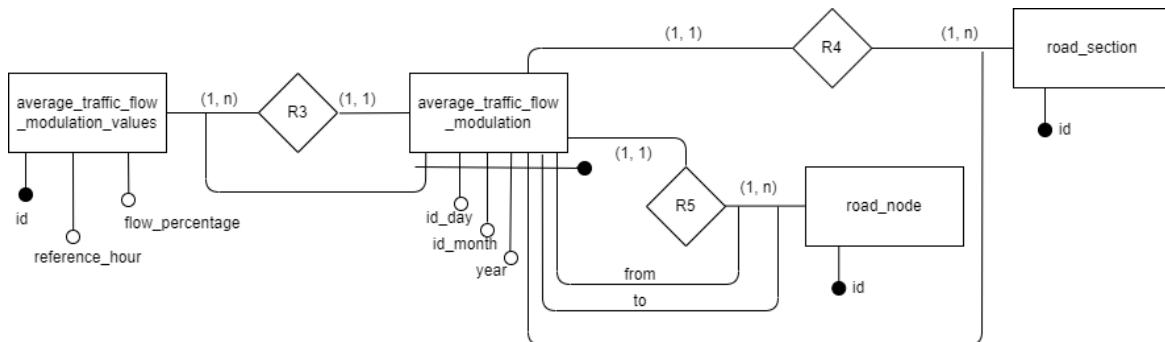


Fig. 3.5 E/R model of the TRAFAIR database to store the average traffic flow modulation.

In the “average_seasonal_traffic_flow” table of Figure 3.6 the number of vehicles, their speed and the flow of traffic are stored for each season (spring, summer, autumn and winter) and type of the day (holiday or weekday) in every portion of street (piece of road section between two road nodes).

The last type of traffic flow aggregation is shown in Figure 3.7. The “average_traffic_flow_modulation_values” and “average_seasonal_traffic_flow_modulation” tables are used to store the percentages of flow in each hour (flow_percentage is an array of 24 elements, one for each hour) with respect to a reference hour in a certain season (spring, summer, autumn and winter) and type of the day (holiday or weekday) in every portion of street (piece of road section between two road nodes).

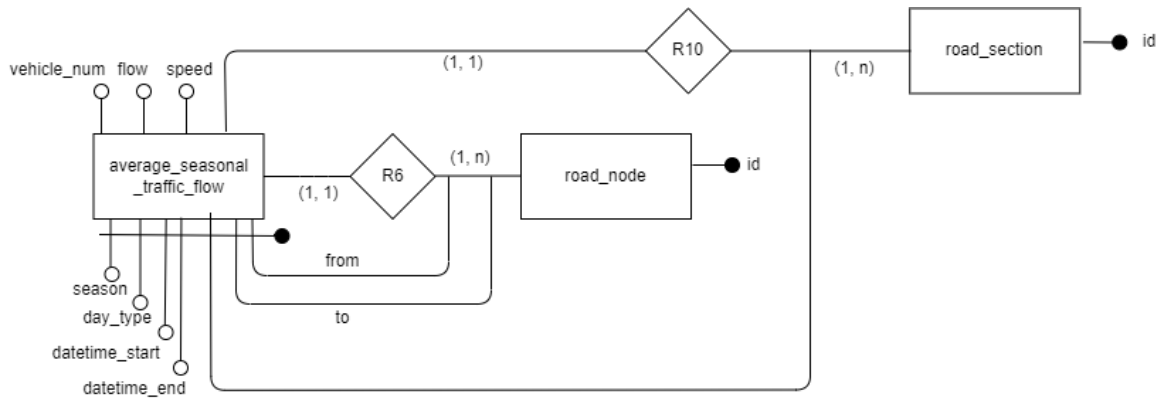


Fig. 3.6 E/R model of the TRAFAIR database to store the seasonal average traffic flow.

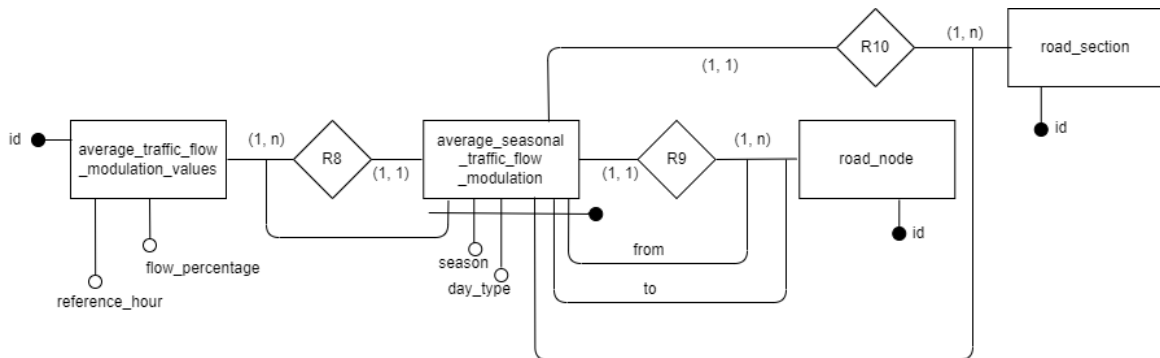


Fig. 3.7 E/R model of the TRAFAIR database to store the seasonal average traffic flow modulation.

3.4 Database analytics

The TRAFAIR database has been designed in order to be flexible and scalable in order to store an ever-increasing amount of data. To give an idea of the amount of data stored in the TRAFAIR data platform, in this Section I provide statistical information about the number of rows and the GB of data stored. The road network of an area of around 200 km^2 has been stored. From September 2018 to December 2021, the data platform collected more than 527 millions of traffic sensor observations (351 GB split into 42 monthly Timescale chunks). From the installation of the air quality sensors in August 2019 to December 2021, 5.2 millions of air quality raw measurements (2.6 GB split into 31 monthly Timescale chunks) have been produced. 340 different Machine Learning calibration algorithms and 19 anomaly detection algorithms have been stored. The latter produced around 400.000 anomalies in the traffic sensor data and more than 600.000 anomalies in the air quality sensor data. From September 2018 to December 2021, the data platform was able to collect 25,965 hours of

simulated traffic flow in 1,452 roads of Modena, 23,412 days of weather condition forecast, and 577 days of NO_x emission forecast. The total size of the TRAFAIR database is 435 GB.

3.5 Database and process monitoring dashboard

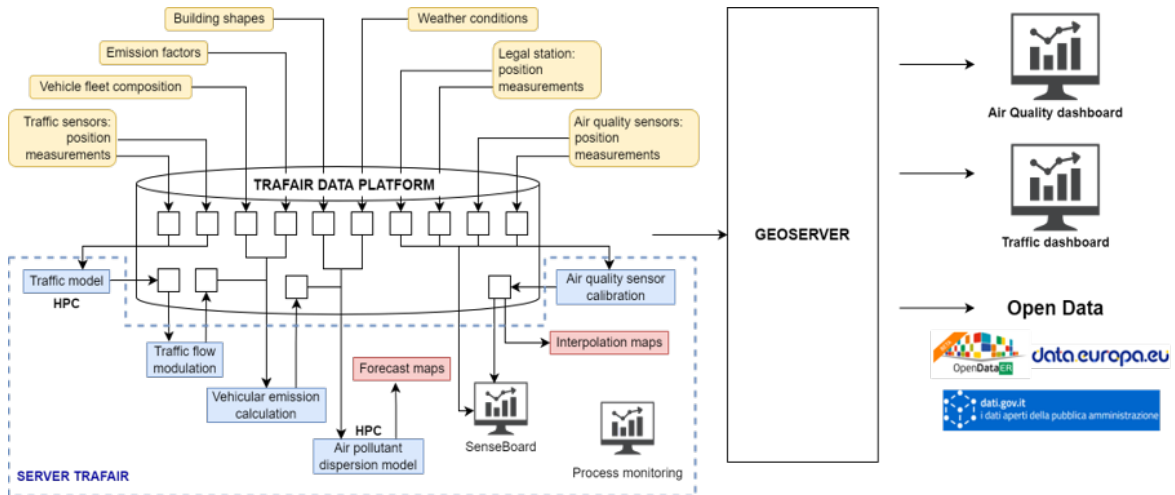


Fig. 3.8 Interaction of the TRAFAIR processes and applications with the data platform.

The TRAFAIR data platform represents a central data store, that is queried by several processes for different scopes, such as the sensor data import, the input generation for the traffic simulation model and air pollution dispersion model, the weather condition data import, and others. The schedule of these processes is managed by the cron command-line utility, the most frequent ones execute every 5 minutes. Moreover, geoserver connects to the TRAFAIR data platform to publish Open Data on specific portals and visualize data on two dashboards. Figure 3.8 illustrates the interaction between the TRAFAIR data platform and the processes implemented for the scopes of the project. The yellow rectangles represent the ingested data, while the blue rectangles are the processes related to the models, the execution of more complex algorithms, like the calibration of air quality measurements. In the end, red rectangles refers to some outputs that are not stored on the data platform, i.e., the forecast maps and the interpolation maps.

Since each of these operations is essential for the scopes of TRAFAIR and malfunction or data unavailability can cause blockages to processes, a tool for monitoring the status of each process involved in the project is necessary.

I worked on the development of the process monitoring dashboard, also known as *control room*. This tool gives an overview of the process status for the city of Modena and is available

on a webpage⁸ protected by login and available only to the persons in charge involved in the project.

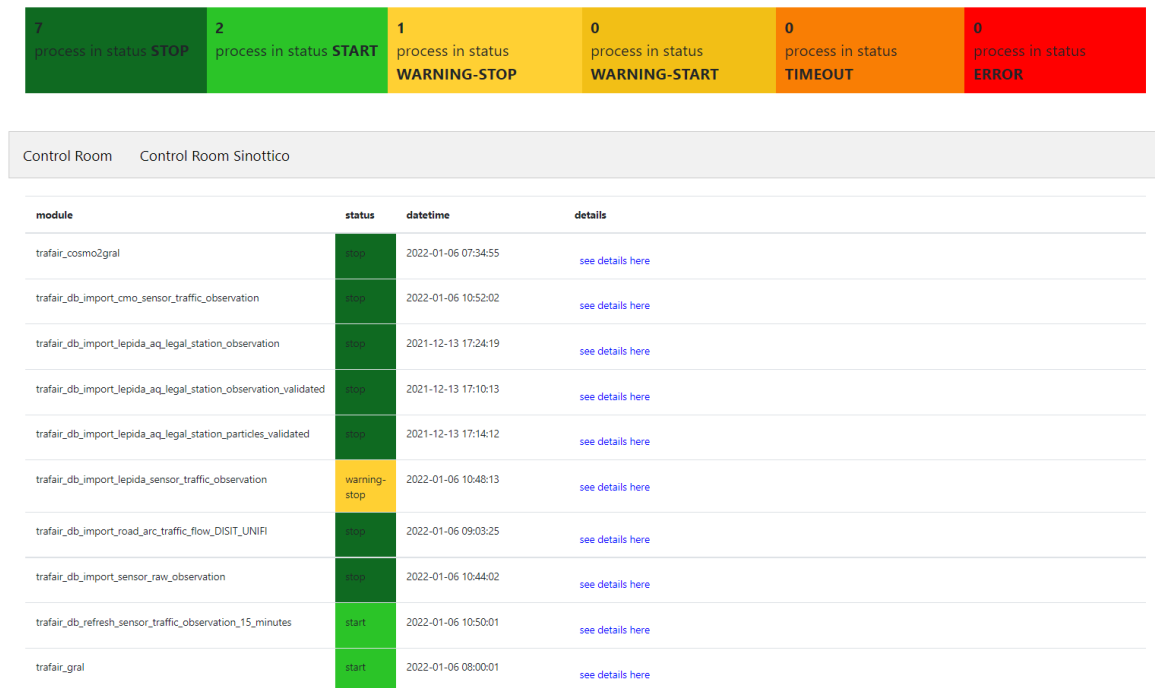


Fig. 3.9 Visualization of the database and process monitoring dashboard.

The exemplar webpage is shown in Figure 3.9 and consists of a list of process names with the corresponding status, the timestamp of the last update and some additional details. The possible statuses are:

stop: the process has finished successfully (i.e. without errors),

start: the process has started and it is yet not finished,

warning-stop: the process has finished with a warning, such as the timestamp of the latest sensor data imported is old (i.e. the time difference between the current timestamp and the one of the more recent sensor observation is higher than a pre-defined value),

warning-start: the process has started but some warnings have been found,

timeout: some request in the process was taking longer than expected to finish and it has been interrupted/terminated,

⁸Database and process monitoring dashboard: <https://trafair-srv.ing.unimo.it/aqsensors?request=ControlRoom>

error: an error has occurred in the execution code of the process.

The field “details” in Figure 3.9 is a customized message and can contain any information of interest. For example, if the process refers to the storage in the database of some data, the number of imported records is indicated in the additional details.

The functioning of the monitoring dashboard is managed by the MQTT (Message Queue Telemetry Transport) protocol, that is a publish/subscribe messaging transport. This protocol distinguishes clients in two categories: MQTT publishers, that write messages, and MQTT subscribers, that receive messages. Each TRAF AIR process is an MQTT publisher; then, an MQTT subscriber has been implemented, this is a daemon that waits for messages of a specific topic. Indeed, each message refers to a specified topic. The Broker is responsible for receiving all messages, filtering them according to the topic, and forwarding each message to the proper subscribed clients. The broker has been implemented in Python through the paho-mqtt library.⁹ The MQTT connection is established between one client and the broker. Clients never connect to each other directly. To initiate a connection, the client sends a “connect” message to the broker. Then, the broker is responsible to keep the connection open until the client sends a disconnect command or the connection breaks. Every time a TRAF AIR process starts, it publishes a message with the status “start” under the topic containing the name of the process. The daemon, that is subscribed to the topics of all the TRAF AIR processes, is notified that the process starts. Then, if the process ends without errors it will publish a new message with status “stop”.

In addition, a weekly check has been implemented within the daemon MQTT subscriber for the processes in charge of importing the sensor data. Once a week, an email is sent to the data managers with a summary reporting the latest timestamp of the data of each sensor. This allows to highlight malfunction in the sensors or errors in the process.

⁹<https://pypi.org/project/paho-mqtt/>

Chapter 4

Traffic and air quality monitoring in Modena

This Chapter is devoted to describe the traffic sensors (Section 4.1) and the air quality devices (Section 4.2) used in the city of Modena, in Italy. Details on the data collection process are given, as well as analysis and statistical information on the sensor measurements.

The work presented in this chapter has been published in [35–39].

4.1 Traffic sensors

Technologies used to measure traffic can be split into two categories: intrusive methods and non-intrusive methods. On the one hand, intrusive methods usually consist of a data recorder and a sensor placed on the road like pneumatic road tubes or induction loops. On the other hand, non-intrusive techniques are based on remote observations, such as manual counting, microwave radars, or video image detection. These techniques allow the detection of different types of data, such as the volume of traffic (counts of the numbers of vehicles on different road segments in a city), travel speeds, and in some cases even the specific types of vehicles (cars, motorbikes, buses, vans, pickup trucks, trailer trucks, large trucks, articulated lorries, etc.), occupancy rates, and other measures.

4.1.1 Sensor network

In Modena, around 400 traffic sensors are spread in different locations, usually near traffic lights. They are induction loops and collect the number of vehicles and their average speed with a certain frequency.

Induction loops are the most commonly used detectors present in a lot of cities. They can be placed on one or more lanes (see Figure 4.1) or on a service lane reserved for buses and/or taxi. An induction loop is an electromagnetic communication system that uses a moving magnet to induce an electrical current in a nearby wire [40]. When a vehicle passes over the electrically conducting loop (sensor) or is stopped within the loop, some of the vehicle's ferrous body material increases the loop's inductance. On the other hand, the peripheral metal of the vehicle decreases the inductance due to eddy currents that are produced [41]. The final effect is an overall reduction in the inductance of the wire loop that tends to decrease the electrical impedance of the wire to alternating current. This actuates the electronics unit output relay or solid-state optically isolated output, which sends a pulse to the traffic signal controller signifying the passage or presence of a vehicle.

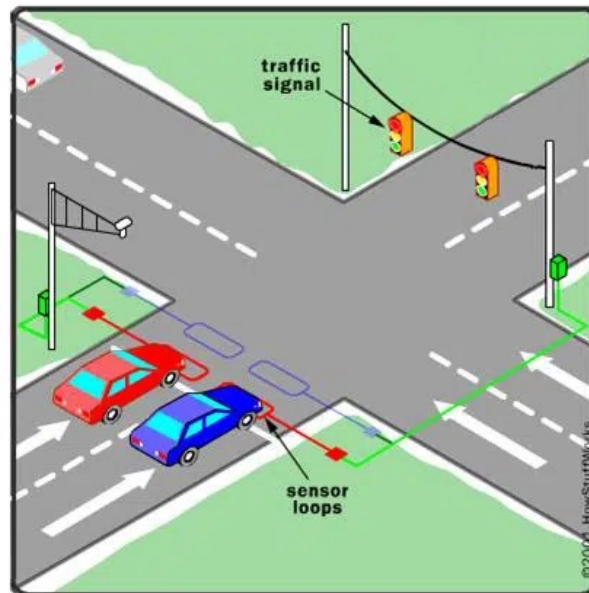


Fig. 4.1 Exemplar installation of inductive loop traffic sensors [taken from HowStuffWorks¹].

Figure 4.2 displays the fixed location of each sensor,² that is placed in a single lane of the main roads. As can be seen, sensors are in the central areas of Modena (blue are in Figure 4.2) as well as in suburban areas, in particular:

- the urban traffic sensors (blue spots in Figure 4.2) are placed on municipality roads within the ring road near the traffic lights; the provider of their data is the Municipality of Modena,³

¹<https://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/red-light-camera2.htm>

²A detailed sensor map of Modena is available at <https://trafair.eu/modenasensormap/>

³<https://www.comune.modena.it/>

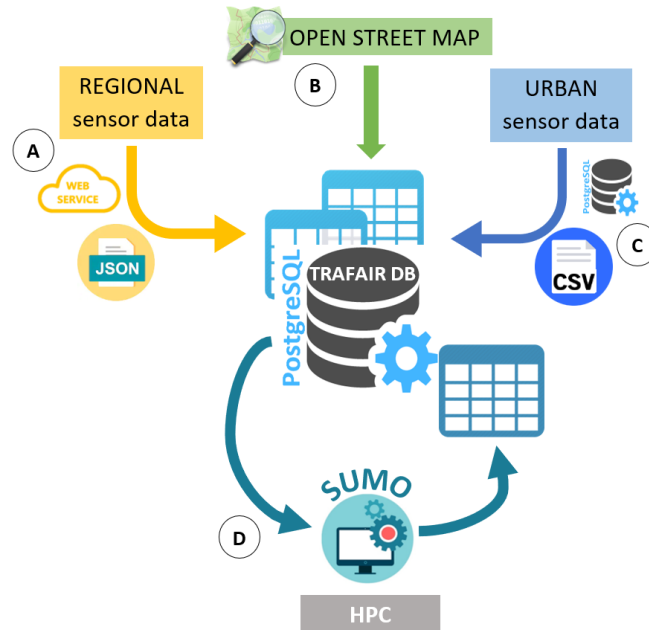


Fig. 4.3 Traffic sensor data ingestion.

All the real-time data of the traffic sensors coming from different providers are stored in the TRAFAIR database. As shown in Figure 4.3, the real-time data of the traffic sensors are collected using two different procedures according to the providers of the data.

The regional sensor data are queried through a web service, while the urban sensor data are first stored in a local PostgreSQL database and then copied in the TRAFAIR database. The regional web service is queried by using a link with the following structure:

http://195.62.186.135/sensornet/getAllHourData/measure_ID/YYYY-MM-DD%20HH:MM:SS

where *measure_ID* is the identifier of the traffic sensor, and *YYYY-MM-DD HH:MM:SS* is the timestamp indicating the hour of the measurement. The web service approximates the timestamp to the hour removing any minutes and seconds and provides a JSON file⁵ with four observations (one observation for every 15 minutes in the interval). Each observation contains its timestamp, the total number of the vehicles passing through the traffic sensor in the following 15 minutes starting from the timestamp, the number of each type of vehicle and the average speed of the vehicles. To obtain the real-time data, a script in Python sends one request to the web service every 15 minutes for each regional traffic sensor. The script uses the Python module `request`⁶ of the `urllib` library to open the URL of the web service

⁵An example of a JSON file is available at <https://trafair.eu/lepida-json/>

⁶<https://docs.python.org/3.0/library/urllib.request.html>

and the library `json`⁷ to decode the output of the web service. Before sending the request, the script selects the timestamp of the last observation of the specified regional traffic sensor. Indeed, a table in the database is devoted to store the timestamp of the latest observation of each regional traffic sensor every time a new observation is inserted. After the request, the script parses the information of the JSON file and stores it in the TRAF AIR database. Then, the script updates the last timestamp of that traffic sensor. The 54 regional traffic sensors produce an average of 48900 observations per day.

The urban sensor data are made available through the ssh connection to the local PostgreSQL database implemented by the Municipality of Modena, that is the owner of the urban traffic sensors. Each urban traffic sensor provides one observation each minute. Each observation contains the timestamp, the total number of the vehicles in one minute starting from the timestamp and the average speed. To obtain the real-time data, the query is sent to the local PostgreSQL database to select the observations starting from the latest timestamp for the urban traffic sensors in the TRAF AIR database. Then, the result of the query is copied on the TRAF AIR database. The 345 urban traffic sensors produce around 379204 observations every day.

4.1.3 Exploratory data analysis

Each observation from the traffic sensors consists of a record containing the identifier of the sensor, the flow measured by the sensor, the vehicles speed, the type of the vehicles (only for provincial and regional sensors), and the time slot of the measurement. The observations of each sensor are geolocated time-series, since they have a reference in time and space. Furthermore, seasonalities and trends can be identified easily. Figure 4.4 shows the observations of sensor R030_S2 in one week of September 2021. As can be noticed, in the working days there are higher values of flow than the ones in the weekend. It is easy to identify also the daily trend since flow is lower in the night hours than in the daylight hours. In addition, traffic sensor measurements are multivariate time-series since they provide information about two variables that are correlated: the traffic flow and the average speed of vehicles. Indeed, when there are a lot of vehicles (high flow), the vehicles are forced to move slowly (low speed). In Figure 4.5 the correlation between the flow and speed values of all the available sensors in September 2021 is illustrated in a scatter plot. First, there are some very high values of speed, most of them are related to low values of flow, as expected. The highest speed values are most probably anomalous data, since in Italy the maximum speed limit is 130 *Km/h* on the motorway. Then, it can be noticed that there are very high values

⁷<https://docs.python.org/3/library/json.html>

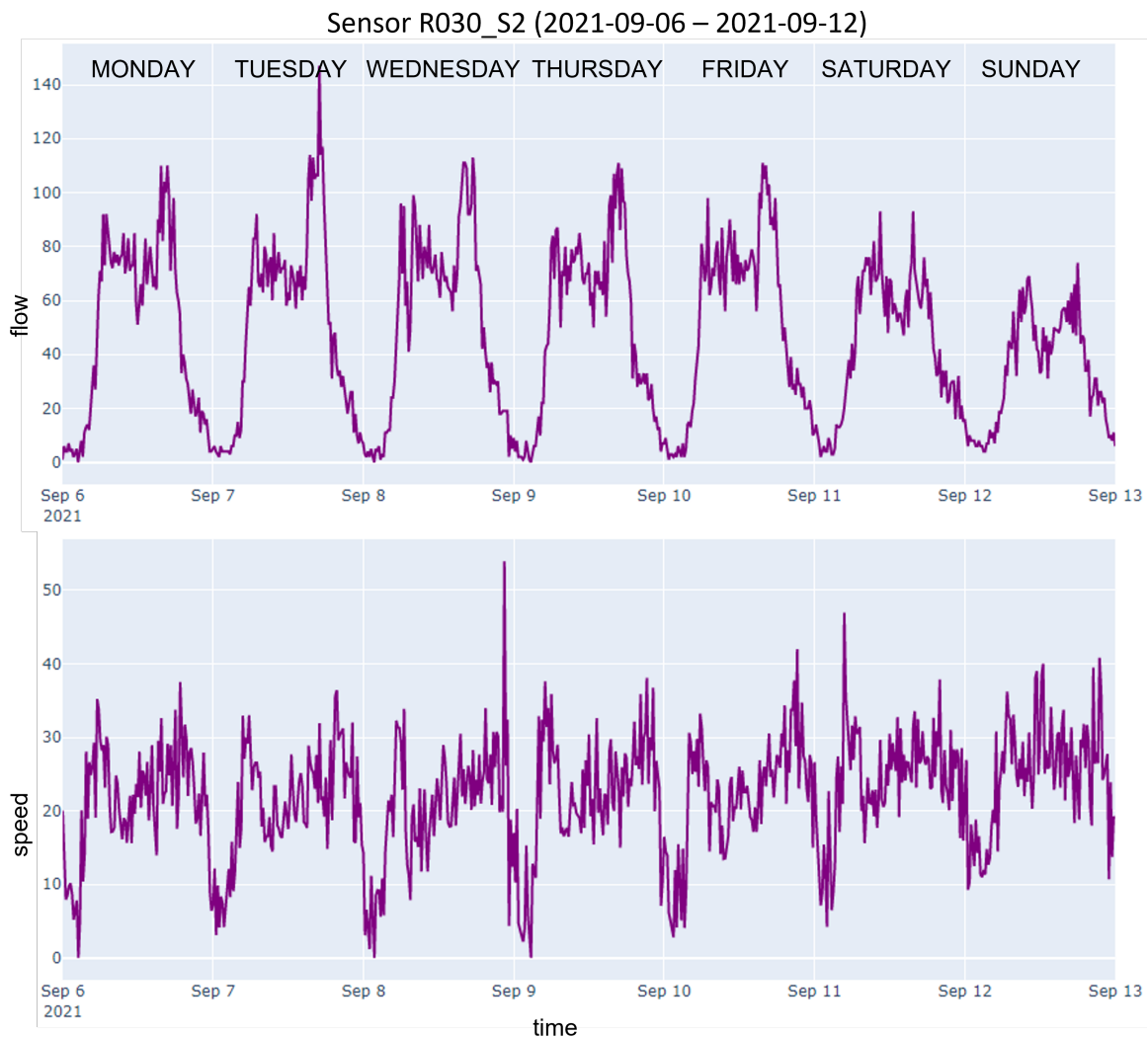


Fig. 4.4 Flow and speed measurements of sensor R030_S2 in one week of September 2021.

of flow corresponding to 150 as speed value. Yet, these data could be anomalous because it should not be feasible that more than 150 vehicles pass over the sensor in just 1 minute.

In addition, since the traffic sensors available in Modena are located near the traffic lights, their measurements are affected by the traffic lights logic and the presence of the red light. When the traffic light is red, vehicles are stopped; in this case, the sensor counts one vehicle (if the vehicle stops over the sensor) or zero (if the vehicle stops before the sensor). This is the reason why traffic sensor data have been aggregated every 15 minutes to generate the traffic model input. This allows to reduce the effect of traffic lights logic.

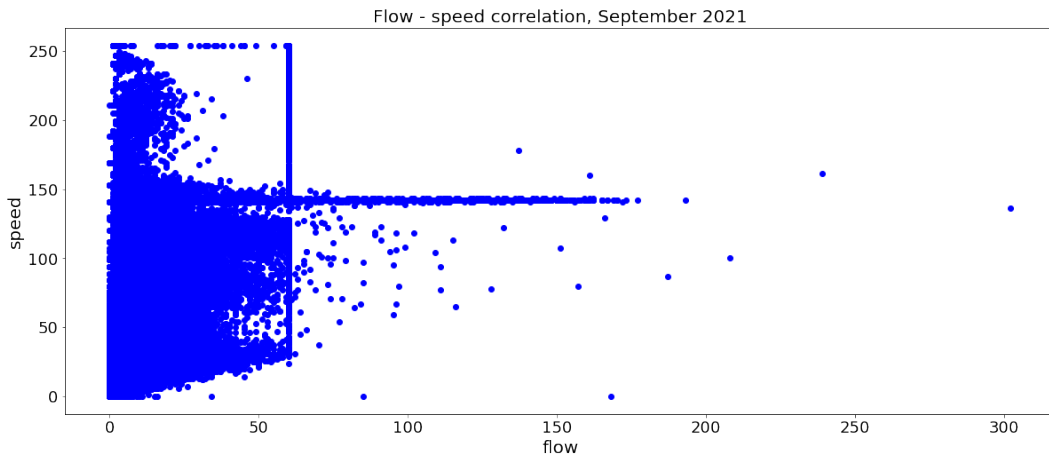


Fig. 4.5 Flow and speed measurements of all traffic sensors in Modena in September 2021.

4.1.4 Statistical overview of traffic sensors observations

In order to better understand the available sensor data, a statistical analysis of the data distribution has been conducted. Considering all the measurements from October 2018 till May 2020, 33 out of 400 sensors did not provide any measurement. Then, the observations provided by the remaining sensors were aggregated every 15 minutes and median, mean, standard deviation, and interquartile range (IQR, i.e. the difference between the third quartile and the first quartile) were evaluated for each sensor. As a result of this evaluation, 35 sensors were considering malfunctioning since their mean, median, and IQR of traffic flow were zeros. Observing the value of IQR of traffic flow, some sensors have an IQR equal to zero. This means that there is no difference between first and third percentile, and all the measurements have a very similar value. This is not the regular behavior of a traffic sensor, where measurements are supposed to vary during the day. Thus, the 6 sensors with IQR equal to zero were excluded. Moreover, for 8 sensors the rate of measurements with a flow value equal to zero was above the 90%; thus, they cannot be considered reliable. To summarize, the total number of untrustworthy traffic sensors is 49 in addition to 33 not working sensors, as reported in Table 4.1. The total number of reliable sensors is 318, these sensors are used as input to the traffic model.

The Table in Figure 4.6 shows some statistical evaluation for the distribution of IQR, median, and standard deviation considering only the 318 reliable traffic sensors. The median of the traffic flow values has an average value of 30 vehicles in 15 minutes; however, there are sensors with a median equal to zero and sensors with a median equal to 132. Observing the graph showing the distribution of the median, the majority of sensors have a traffic flow median value between 0 and 25. This happens because the time-series of measurements

Table 4.1 Analysis of the reliability of traffic sensors measurements from October 2018 till May 2020.

Traffic sensors installed in Modena	400
Not working sensors (no measurements)	33
Sensors with zero mean, median, and IQR	35
Sensors with zero IQR	6
Untrustworthy sensors (90% rate of zero values)	8
Reliable sensors	318

of traffic flow also includes the night period where the traffic flow values are all near zero and the standard deviation and the IQR are significantly reduced. Besides, the graphs show that the distribution of median, IQR, and standard deviation values is right-skewed: mean is higher than mode. Thus, there are few sensors with high values and more sensors with lower values.

4.1.5 Stationary study

In a stationary time-series, each time-series measurement reflects a system in a steady state. A series X_t is called “stationary” if, loosely speaking, its statistical properties (mean, variance and covariance) do not change over time [42]. There are three types of stationary: (1) a *strict stationary* series satisfies the definition as mentioned above of stationary; (2) a *trend stationary* series exhibits a trend, that, if removed, makes the resulting series strict stationary; in the end, (3) a *difference stationary* series can be transformed into a strict stationary series by differencing. Before applying any prediction model to a non-stationary time-series, the time-series has to be converted into a strict stationary series. For this reason, a stationary analysis has been conducted on the traffic sensor data. The easiest way to do this is by differencing, which means to compute the difference of consecutive terms in the series, following the formula $Y_i = X_i - X_{i-1}$, where i is the time instant and X_i is the value of the time-series at instant t .

The most popular tests to check if a time-series is stationary or non-stationary are:

- Augmented Dickey Fuller test (ADF);
- Kwiatkowski Phillips Schmidt Shin test (KPSS).

ADF test is a statistical test and, in particular, a unit root test that aims at determining how strongly a time-series is defined by a trend. A unit root is a characteristic element of a

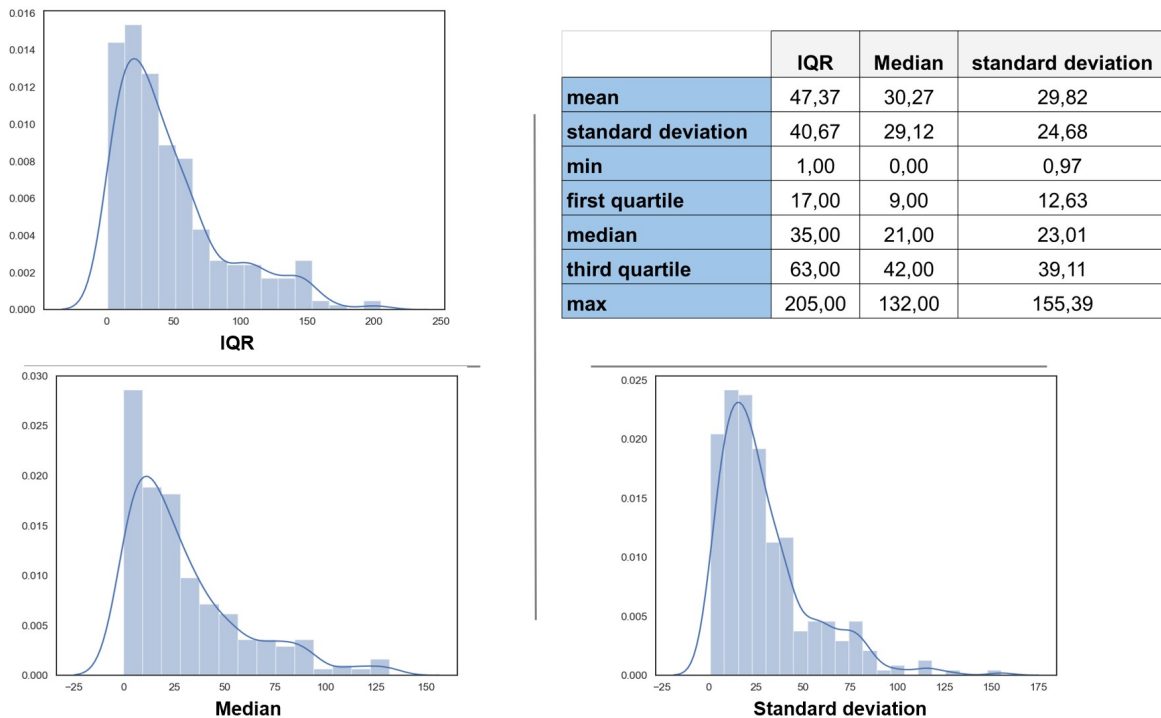


Fig. 4.6 IQR, median and standard deviation of 318 trustworthy traffic sensors.

time-series that makes it non-stationary; Besides, the number of unit roots in a time-series corresponds to the number of differencing operations required to make the series stationary. Since the ADF test is a statistical test, there are two hypotheses to be tested: a null hypothesis (H_0) and an alternate hypothesis (H_1). The null hypothesis is that the time-series is not stationary and has some time-dependent structure that can be represented by a unit root. The alternate hypothesis instead is that the time-series is stationary. The p-value obtained by the ADF test is exploited to interpret the result of the test, i.e., whether to reject the null hypothesis or not. In statistical, the p-value is a probability score that establishes the significance of an observed effect. In other words, it is the probability of seeing the effect E when the null hypothesis is true ($p - value = P(E | H_0)$). If the p-value is lower than a threshold, the time-series is stationary (H_0 rejected); otherwise, it is non-stationary (H_0 not rejected). Typically, the threshold is set to 0.05.

Also, the KPSS test is a unit root test to study the stationary of the time-series around a deterministic trend. A time-series exhibits a deterministic trend if the slope of the trend does not change permanently; even if the series goes through a shock, it tends to regain its original path. The difference from the ADF test is that the null hypothesis of the KPSS test is the stationary of the series. Therefore, the p-value must be interpreted oppositely: if the p-value is lower than the threshold, the series is non-stationary.

Both tests have been applied to the daily measurements of April 2019 aggregated every 15 minutes to be sure of the stationarity of the time-series by using the statsmodels package in Python. In some cases, the results of the two tests could conflict with each other. In particular, if the result of KPSS test is “stationary” and the one of ADF test is “non-stationary”, then the series will be trend stationary; on the other hand, if the result of KPSS test is “non-stationary” and the one of ADF test is “stationary”, the series will be difference stationary. Then, the p-values of each daily time-series have been compared to the commonly used significance threshold of 0.05. All the p-values are far less than the threshold. Therefore the daily time-series are classified as stationary, as expected.

4.2 Air quality sensors

The air quality devices employed in Modena as well as in the other TRAFAIR cities are low-cost, cheaper and less reliable than the AQM stations managed by the Environmental Agencies. However, these devices can provide reliable measurements if they are co-located for a certain period of time close to the AQM stations where they “learn” how to measure air quality. A device is a box where different sensors are placed. Each sensor, also called cell, is devoted to the measurement of a specific pollutant. Employing low-cost devices is the only way to perform hyper-local air pollution monitoring. Indeed, AQM stations are expensive and usually there are few of them even in big cities. Data coming from a group of low-cost devices spread around a city might generate widespread hyper-local insights into air pollution. The low-cost air quality sensors are complex and sensitive, they require specific environmental skills. Data they generate need to be converted into relevant and crucial insights to allow the monitoring of air quality by politicians and to enable the achievement of the sustainability goals.

The approach used in TRAFAIR is to install the devices for a certain period close to the AQM stations. During this period (calibration period) a Machine Learning algorithm is trained on the measurements provided in millivolts by the low-cost devices (raw observations) and the AQM station measurements. Then, when environmental experts evaluate the device is “ready” (usually after 3 weeks of co-location), it can be moved to different location and start providing air quality measurements. When the device is “ready”, thanks to the Machine Learning algorithm previously trained, calibrated data (concentrations) are generated from the raw observations. Usually, periodically every 6 months, the devices are once again co-located near the AQM stations for re-calibrating, thus maintaining a good quality of measurements.

Low-cost devices provide “raw” measures, i.e. a datum in millivolts; to convert this datum into a reliable concentration of pollutant it is necessary to carry out a calibration period during which some Machine Learning algorithms are trained in order to generate, from the raw measurements, pollutant concentrations in line with those estimated by the AQM stations.

4.2.1 Sensor network

In Modena 13 air quality devices (52 low-cost sensors) have been installed in different locations. There have been identified 2 locations for calibration (the red dots in Figure 4.7), close to the AQM stations, and 10 locations of interest (the blue dots in Figure 4.7), that are placed in areas of different kinds, such as residential, industrial, or green areas.

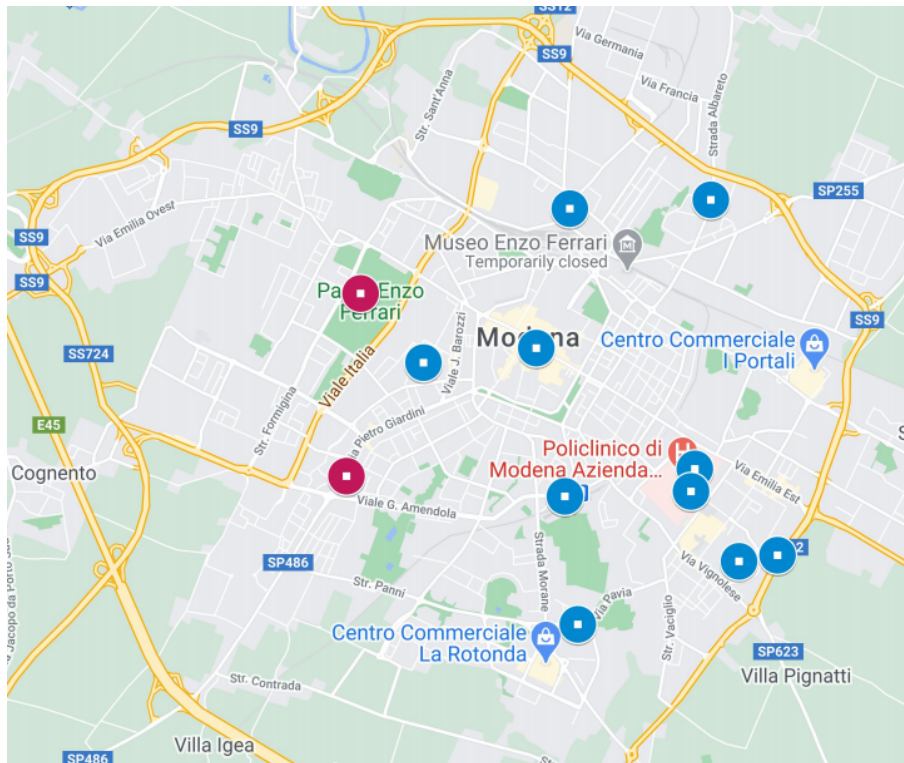


Fig. 4.7 Points of interest for air quality monitoring (blue dots) and positions of the AQM stations (red dots). Map data: Google, 2020.

Two types of low-cost devices have been exploited: 12 Decentlab Air Cubes⁸ (see Figure 4.8) and 1 Libelium Smart Environment PRO⁹ (see Figure 4.9). All the devices are equipped with 4 cells (sensors), one for each gas (NO , NO_2 , CO and O_x). Each cell measures the

⁸<https://www.decentlab.com/products/air-quality-station-no2-no-co-ox-for-lorawan>

⁹<https://www.libelium.com/iot-products/plug-sense/>

gas concentration through 2 channels (the auxiliary and the working channels). In addition, the Libellium device measures the level of $PM_{2.5}$ and PM_{10} . For each channel, the raw observations are provided in mV , moreover, a basic concentration based on the original factory calibration¹⁰ is provided in $\mu g/m^3$. Besides, these devices are able to measure the air temperature and humidity, and provide the battery voltage. Therefore, the total number of measurements provided by one sensor is 19 for Decentlab cubes and 21 for Libellium sensor.

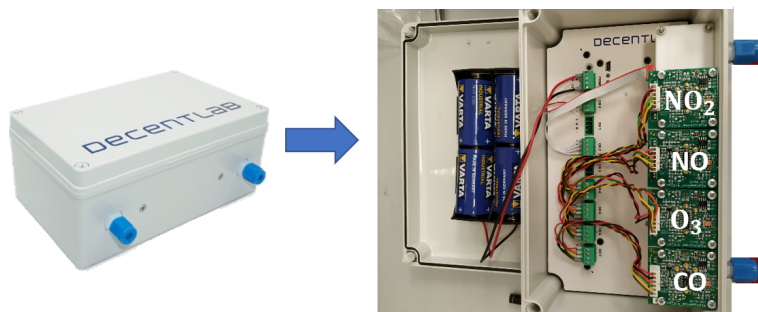


Fig. 4.8 An air quality device (on the left) and its content inside (on the right): 4 cells/sensors for measuring the level of 4 air pollutants (NO, NO₂, CO, and O₃ in this case).



Fig. 4.9 A Libellium Smart Environment PRO device (on the left) with Particle Matter sensor (on the right).

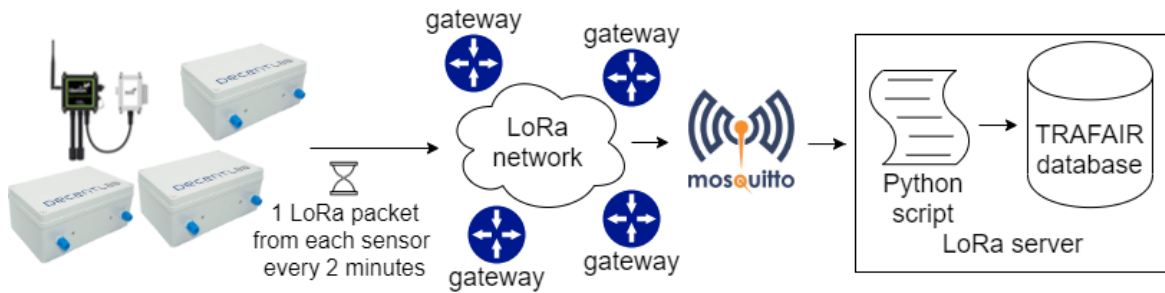


Fig. 4.10 Sensor data acquisition from the low-cost air quality sensor network.

4.2.2 Sensor data collection

The sensor data acquisition is managed by the Long Range Wide Area Network (LoRaWAN) implemented in the city of Modena. LoRaWAN [43] is a media access control (MAC) protocol widely used in Smart City applications thanks to its easy installation and cost-effectiveness. It employs some gateways i.e. antennas that receive broadcast messages from the enabled devices (the air quality devices, in this use case) and forward them to the server. The message from one device can be received by more gateways at the same time, the server will deal with duplicates. The LoRaWAN network exploits low radio frequencies and provides for long-range communications (up to five kilometers in urban areas, and up to 15 kilometers or more in rural areas). The network coverage depends a lot on the geographic landscape.

The air quality devices have been registered to the LoRaWAN network of Modena through their identifier (DevEUI) and following the Over-the-Air Activation (OTAA) process. The data rate has been set up to 125 kHz, and the spreading factor to 7, to allow devices transmitting data every 2 minutes.

Figure 4.10 shows the data acquisition process. The devices encapsulate the acquired measurements into LoRa packets and send them to the LoRaWAN server through the gateways. Every LoRa packet contains 19 (or 21) measurements (both mVs and basic concentrations for each gas and channel, including air temperature, humidity and battery voltage) and in one day each device produces 13,680 ($19 * 720$) measurements. 4 gateways have been installed in Modena, mainly on the roofs of the highest buildings, to cover the whole urban area of the city and ensure the coverage of the LoRaWAN network in the points of interest. Moreover, the gateways have been registered on the LoRa server. When the gateways receive a message, they send it to the LoRa server where the MQTT (Message Queue Telemetry

¹⁰This is obtained by applying to the raw observations a formula provided by the manufacturing company with the calibration parameters that are different for each device.

Transport) Broker Mosquitto [44] is running. The paho-mqtt Python library¹¹ has been used to implement the open source message broker in a Python script. The script is always running and exploits the client class to enable the connection to the MQTT broker, publish messages, subscribe to topics and receive messages. Then, messages are decoded and measurements stored in real-time into the TRAFAIR database. When a device is moved from one location to another, it automatically connects to the nearest gateway and restart sending messages. Since the messages received in the LoRa Server are described by the identifier of the device, the change of gateway to which the device connects is completely transparent. The LoRa server keeps storing the measurements of each device no matter how they are moving in the urban context.

4.2.3 Exploratory data analysis

The values measured by the working channel and the auxiliary channel of each air quality sensor are highly correlated. Considering the data in year 2020 of the channels of one sensor, the Spearman correlation coefficient returns 0.5 for NO , 0.65 for O_x , 0.81 for CO , and 0.83 for NO_2 . This means that the two time-series for each gas are positively related: when the value of the working channel increases, the one of the auxiliary channel grows accordingly. Figure 4.11 shows one plot for each measured gas with the values in millivolts provided by the two channels. In each plot the positive correlation can be noticed.

The accuracy of raw measurements of the air quality devices is affected by events, such as low battery voltage, weather conditions, air humidity, physical disturbances, and others (see Figure 4.16). In this cases, all sensors of the device should be affected by the situation producing anomalous data. Since the air quality devices are continuously moved in the city from one predefined location to another, it is possible that this mechanism affects the sensor behavior producing anomalous data. When the devices are switched off and then switched on in a new location, they experienced a “warm-up” for some minutes or even hours. The warm-up is a period of variable duration when the device tries to achieve a thermo-mechanical balance in the measuring system as well as an optimal operating temperature of the electronic components.

Figure 4.12 shows the raw measurements collected from one device. As it can be seen, at 9 a.m. approximately the device has been switched off. One hour later, the device has been switched on in a new location. The zoomed area of the graph shows the first two measurements for each gas made in the new location. The values drop off sharply due to the

¹¹<https://pypi.org/project/paho-mqtt/>

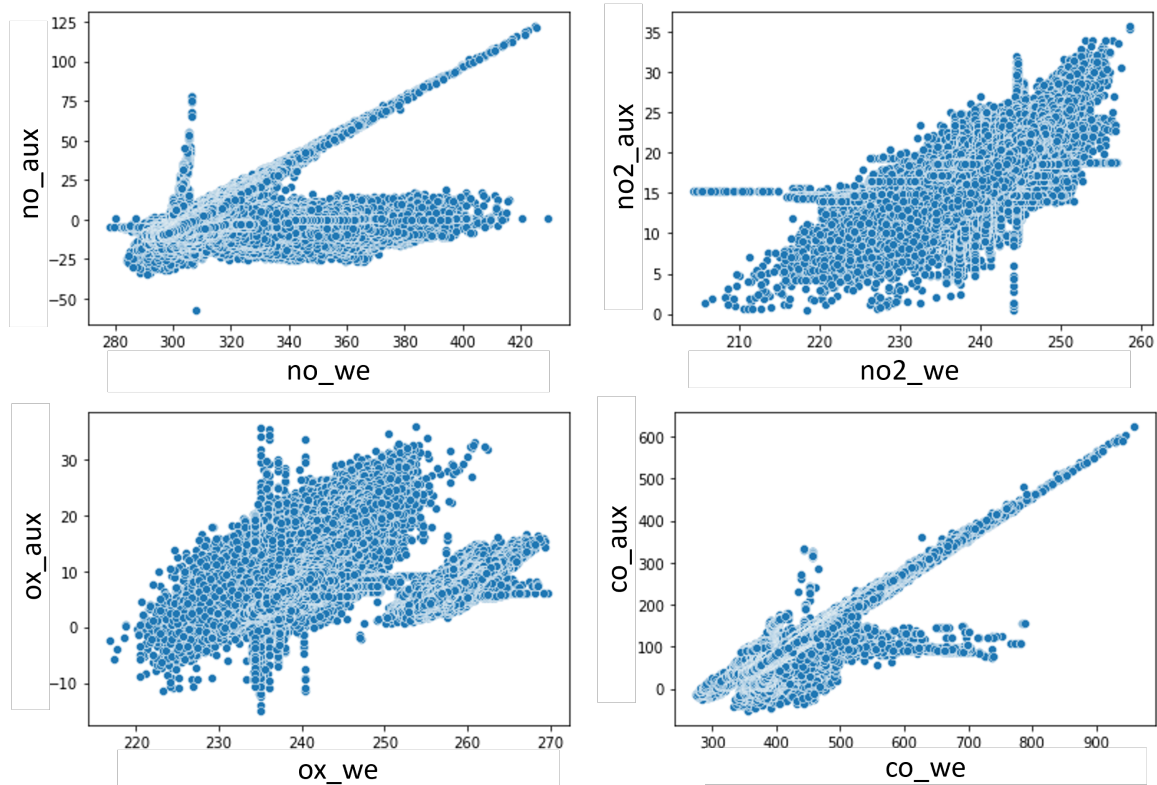


Fig. 4.11 Plots with the values of the working and auxiliary channels of the 4 gases: NO, NO₂, CO, and O₃.

warm-up. Consequently, these values have to be discarded from the reliable observations and flagged by the environmental experts as “not reliable”.

Sensor behavior is continuously monitored by environmental experts to detect when the warm-up period is over thanks to the TRAFair monitoring dashboard for air quality devices (it is described in Section 4.2.4). Environmental experts can interact with the dashboard’s rich UI to obtain detailed visualization of live or historical (day, week, month) air quality data patterns.

4.2.4 SenseBoard: data visualization tool

Environmental experts hunger for a control platform to perform sensor data calibration and anomaly detection. The availability of effective visualizations to process and interpret collected data is essential. Besides, a tool is needed for managing the change of location or status of the devices. Also, it is important to compare anytime the calibrated data with the measurements of the AQM stations (no matter in which location the sensor is), to determine

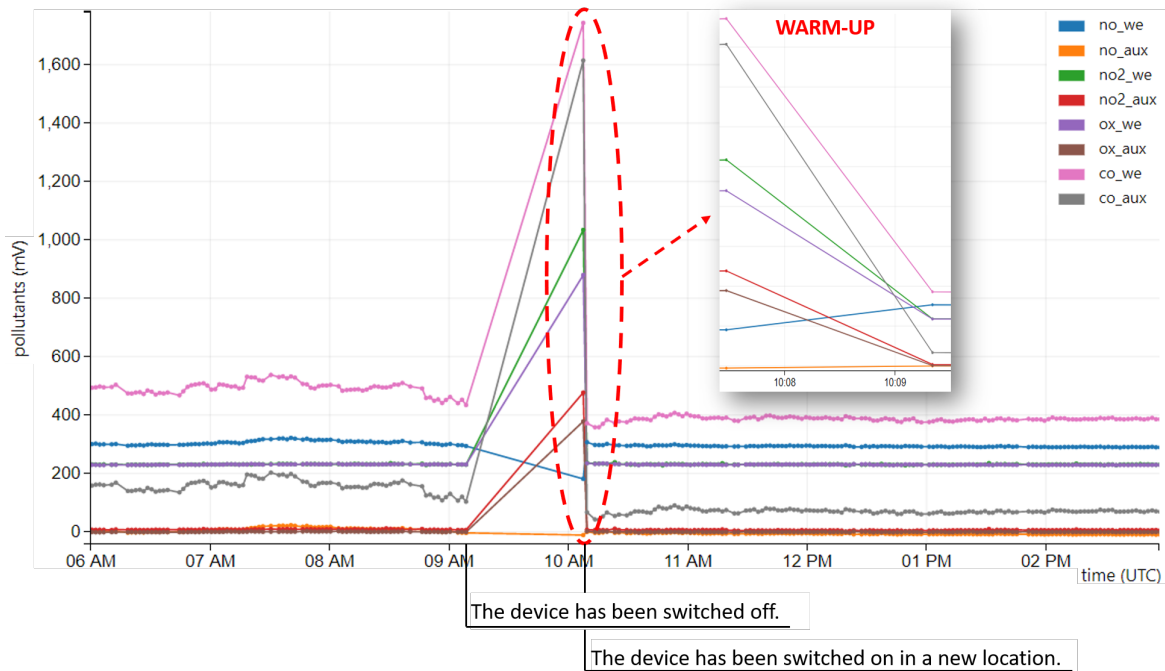


Fig. 4.12 Raw observations of one air quality device made in two different locations.

when the device needs to be re-calibrated (usually when the calibrated data and AQM measurements differ a lot).

The monitoring activity can be made through the dashboard SenseBoard, an interactive tool addressed to environmental experts that brings together heterogeneous and dynamic data for real-time analysis and management of air quality network. SenseBoard is devoted to support environmental experts in the monitoring and control of the air quality sensor network, in the supervision of the calibration process and in the detection of anomalous values. Accordingly, the environmental experts need to constantly visualize the data produced by the device and monitor the behavior of each device.

The need for a monitoring tool comes from the environmental experts. For this reason, they have been directly involved in the definition of the requirements to be satisfied through the dashboard. After some discussions, 6 requirements have been outlined:

- R1** providing an overview of the current position and status of each sensor,
- R2** recording the location or status change for a certain sensor without hand-writing the SQL query to store the modification on the database,
- R3** visualizing sensor observations without data aggregation to detect anomalous values and compare them each other, and with data aggregation to better understand the trend,

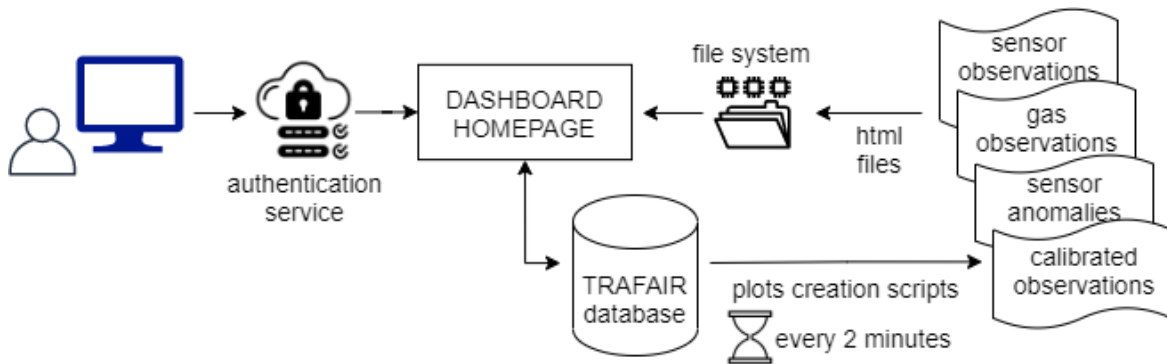


Fig. 4.13 SenseBoard architecture.

R4 comparing observations of co-located sensors in the same place,

R5 showing the concentrations produced by the calibration algorithm and comparing them with the certified values of the AQM legal stations, to understand if the algorithm works appropriately or if it is necessary to extend the co-location period of the device,

R6 displaying the anomalies identified by the anomaly detection algorithm to control the efficiency of the automated algorithm.

After the development of SenseBoard, during the usage, additional feedback from environmental experts has been continuously collected to improve the functionalities of the dashboard.

SenseBoard¹² is a general and flexible dashboard that can be adapted for the monitoring of any air quality sensor network. The scalability of the dashboard allows replicability in cities of different size with a variable number of sensors. The dashboard is not affected by the type of employed sensors and it can be easily modified to visualize other parameters measured by the sensors.

The following Section describes the design of the dashboard and some views (data visualization) related to the use case of Modena.

SenseBoard design

SenseBoard¹³ is a Python web application which exploits Tornado¹⁴ as web framework. It runs on a Debian 9 machine with 32 Intel(R) Xeon(R) Silver 4108 CPU at 1.80GHz

¹²SenseBoard system demo:
https://www.youtube.com/watch?v=cfE7IJ0ZvNc&ab_channel=BigVisWorkshop

¹³<https://trafair-srv.ing.unimo.it/aqsensors>

¹⁴<https://www.tornadoweb.org/>

processors and 256 GB RAM. Figure 4.13 shows the architecture of the dashboard. Firstly, users need to login to access the dashboard. The authentication phase is performed through the Lightweight Directory Access Protocol (LDAP). The list of people allowed to access is currently limited to the environmental experts working in TRAF AIR.

After the authentication, the user is able to visualize the current status of each device and send other requests through the navigation bar at the top: he/she can ask for observations (raw measurements), anomalies, calibration (calibrated measurements), and AQM station (measurements from the AQM stations). For each request, the dashboard queries the TRAF AIR database to obtain the appropriate data and creates plots of the time-series data by using the matplotlib Python library.¹⁵ More complex plots are periodically generated by ad-hoc Python scripts¹⁶ which query the database and save plots in the file system as html files through the `save_html` function of the `mpld3` library.¹⁷ This library is also exploited for the `InteractiveLegendPlugin`,¹⁸ which allows to connect the plot to an interactive legend. This legend is very useful in these plots since it allows customizing the visualization by adding or removing some lines in the plot. The user can click on the rectangle generated in the legend near the labels. If the rectangle is colored, the corresponding data is shown on the plot; if the rectangle is white, these data are removed from the plot. The html files are, then, included in the html page of the corresponding request. The expression “more complex plots” indicated the plots which require an elaboration of the data stored in the database and manage a big amount of data (i.e. the raw observations of each sensor related to one month). This choice was made to save time in the visualization of the plots. Indeed, this solution decreases the server response time of 35 seconds for the most time-consuming request.

Some examples of visualizations (views) are described in the Section “Views”. The views are static to allow users to navigate and explore all the plots in the view without any interference. However, the user can click on the “update” button to see the updated views.

Users and scope

The scope of SenseBoard is the monitoring and control of the air quality sensor network and the supervise of the calibration and anomaly detection processes.

Regarding the monitoring of the network, SenseBoard allows to identify and update the status of the sensors, change their location when they are moved in different position and perform any maintenance, if necessary.

¹⁵<https://matplotlib.org/>

¹⁶The scripts run every 2 minutes and generate the plots in 4-27 seconds.

¹⁷<https://mpld3.github.io/>

¹⁸https://mpld3.github.io/examples/interactive_legend.html

Considering the supervise of the calibration process, SenseBoard lets to compare raw measurements of co-located sensors, raw and calibrated measurements of the sensors, and, in particular, the calibrated observations generated during the calibration period with the legal observations from the AQM stations. This last operation is the crucial one in the calibration process because it allows experts to understand if the training period of the Machine Learning algorithm is sufficient, i.e. if the concentrations, elaborated by the Machine Learning algorithm, are in line with those of the AQM station.

Other tasks are the detection of issues in the network communication, the discovery of disruptions or failures in the sensor's behaviour, the identification of anomalous gas concentrations, the comparison of co-located sensors measurements, the correlation study of the pollution level in the area of sensor installation.

The primary users of this visual analytic dashboard are the environmental experts in charge of installation, maintenance and calibration of air quality sensors.

Views

In SenseBoard, 6 views have been developed to allow environmental experts to have complete control of the air quality sensor network status and the operations that are performed on the sensor data. Each view is described in detail in the following sub-sections.

Sensor status and position The first view, i.e. the homepage of the dashboard after the login, aims at satisfying requirements R1 and R2. Here, users are able to visualize a table with a summary of the main information related to the air quality devices. For each device, in the table, there are listed its identifier, the name of the location where the device is currently installed, the timestamp of the installation, the name of the person in charge of the installation, the sensor status and any possible notes.

Besides, as shown in Figure 4.14, for each device, two buttons are available: the “edit” button allows to update the location and/or the status of the corresponding device. After clicking on the button, the user has to specify the timestamp representing the instant of the update (of the location or status), the location (one of the points of interest in Figure 4.7), the status, and, optionally, its name and notes. The “save” button stores the information in the TRAF AIR database. The status update is exploited in different situations. For example, if the device is moved from a point of interest to the AQM station, its status changes from “running” to “calibration”. In addition, if the environmental experts notices an abnormal behavior of the device, he/she can modify the status in “broken” indicating as timestamp the date of the first abnormal measurement. Then, he/she needs to add the “running” status from the first regular measurement.

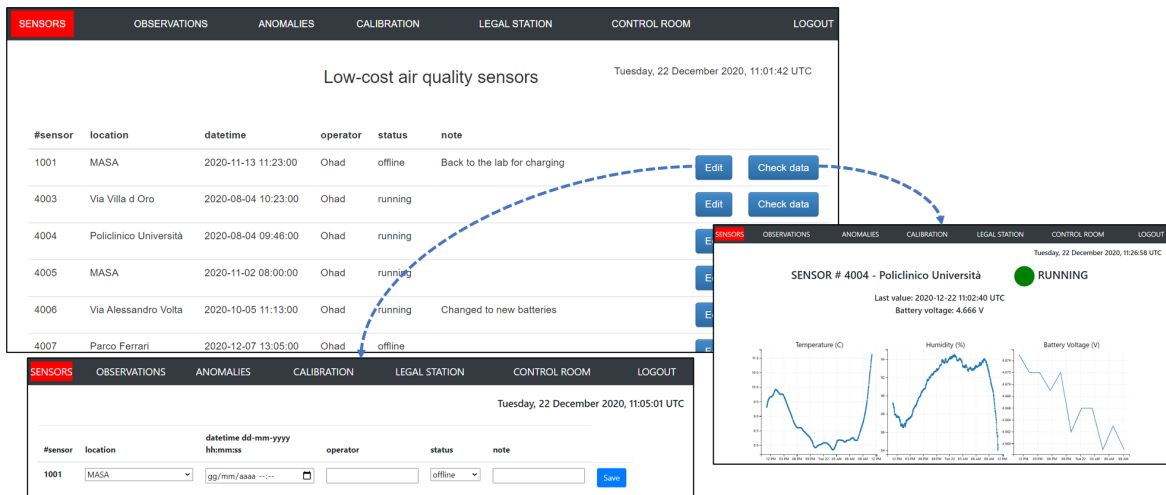


Fig. 4.14 “Sensor status and position” view.

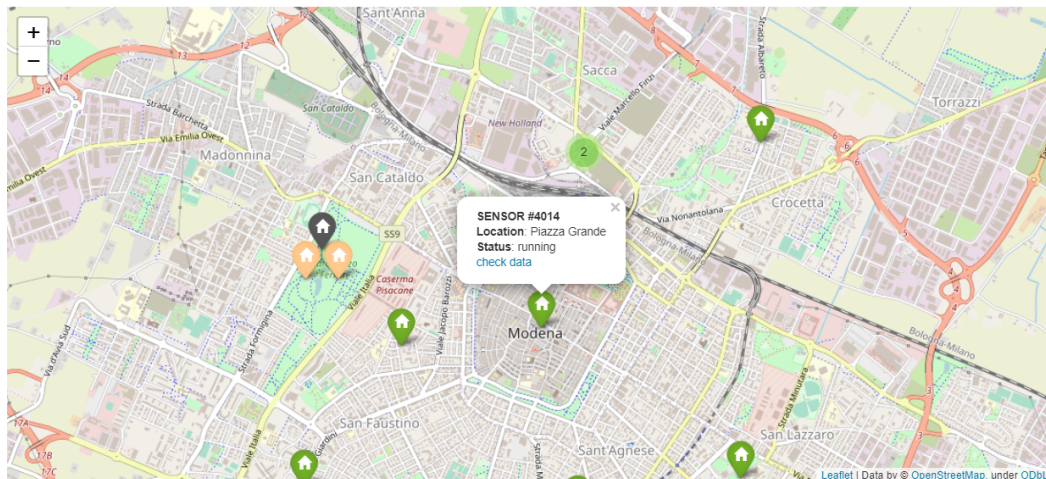


Fig. 4.15 Position of the air quality devices in Modena on January 4th, 2021.

In addition to the “edit” button, the “check data” button connects to the “sensor observations” view.

Besides, in this view, the user can interact with a map (Figure 4.15), where the current position of each device is visualized with an icon of different colors according to the status of the device. If more devices are in the same location, a bigger icon is displayed on the map with the number of devices in that position. By clicking on this icon, an icon for each device is visualized. If you click on the icon of a device, you can see its name, its status, the name of the location, and the link to the “sensor observations” view of that specific sensor. Folium¹⁹ is the Python library used to create the map.

¹⁹<https://pypi.org/project/folium/0.1.5/>

Sensor observations The “sensor observations” view satisfies requirement R3 and includes 6 plots with the observations of one device. At the top of the page, the name of the device, its status and location, and the timestamp of the last observation with the level of battery voltage are reported. This allows the managers of the sensor network to check immediately if the sensor is not sending data or if the batteries need to be changed.

The 6 plots show the measurements of:

1. the relative humidity in percentage (%),
2. the temperature in Celsius degree,
3. the battery voltage in Volt (V),
4. the raw observations of the 4 gases for auxiliary and working channels in mV,
5. the observations of the 4 gases for auxiliary and working channels calibrated through the original factory calibration in $\mu g/m^3$,
6. the observations calibrated through the TRAFair calibration algorithms in $\mu g/m^3$.

Only for the Libelium device another plot is provided, which shows the level of PM_{2.5} and PM₁₀.

Each plot can visualize data for different time interval (last 24 hours, week, or month) and data aggregation (2 minutes - which means no aggregation, 5 minutes, and 15 minutes), generating 9 different combinations for each plot. The visualization changes according to the option selected by the user.

There are altogether 711 plots (13 devices * 6 plots * 3 time interval * 3 data aggregation + 1 PM plot * 3 time interval * 3 data aggregation). Since the creation of a plot took on average 12 seconds, these plots are generated asynchronously through one Python scripts. This means that the plots are generated independently by the user choice, and when the user selects an option (for time interval and data aggregation), he/she enables the visualization of a ready-made plot. This time-saving design choice is also motivated by the user behavior. After three months from the first release of SenseBoard, it was noticed that users are interested in visualizing several plots, exploring different gases with different aggregations or for a different time interval. If the plots are created synchronously with the user’s choice, jumping from one plot to another requires waiting for the generation of the relative plot each time. In agreement with environmental experts, the plots are generated asynchronously and reloaded every 2 minutes.

Figures 4.16 and 4.17 are two examples of visualization available in the “sensor observations” view. In Figure 4.16 the measurements of the 4 gases for the auxiliary and working

channels related to one device are plotted in a lines chart. An anomalous behavior of the device has been highlighted in red: the values of the measurements in that time interval are very different from the previous ones. SenseBoard allows the detection of the wrong data. After the maintenance work by the environmental experts, the device reaches the stability and the measurements proceed with the expected values. Through the “edit” button of the “sensor status and position” view, the time period related to the red area is flagged with the “broken” status. Figure 4.17 highlights an abnormal behavior of another device. In this case, the anomalous measurements are due to a drastic reduction in the battery level. At 2 a.m., approximately, the battery died and the device stopped sending data. After changing the battery, at 10 a.m., the device restarts providing reliable measurements.

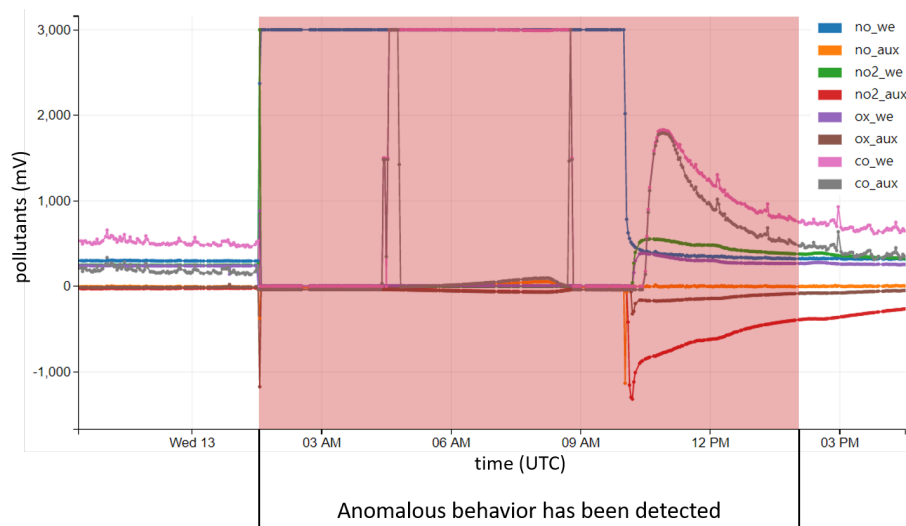


Fig. 4.16 An anomalous behavior of a device detected on January 13th, 2021.

Gas observations In the “gas observations” view, a plot for each gas and channel is generated, as shown in Figure 4.18 for NO. This view meets requirements R3 and R4. The user can choose to visualize the data of the last 24 hours, week or month. The visualization could seem confused, however the user is able to hide one or more lines in the plot thanks to the interactive legend, and zoom in a specific area of the plot. In the web page, the plots related to the two channels of the same gas are placed next to each other (as in Figure 4.18) to facilitate the comparison of these measurements and detect the correlation between the two channels. Thanks to this view, the behavior of the cells can be regularly checked and the maintenance planned.

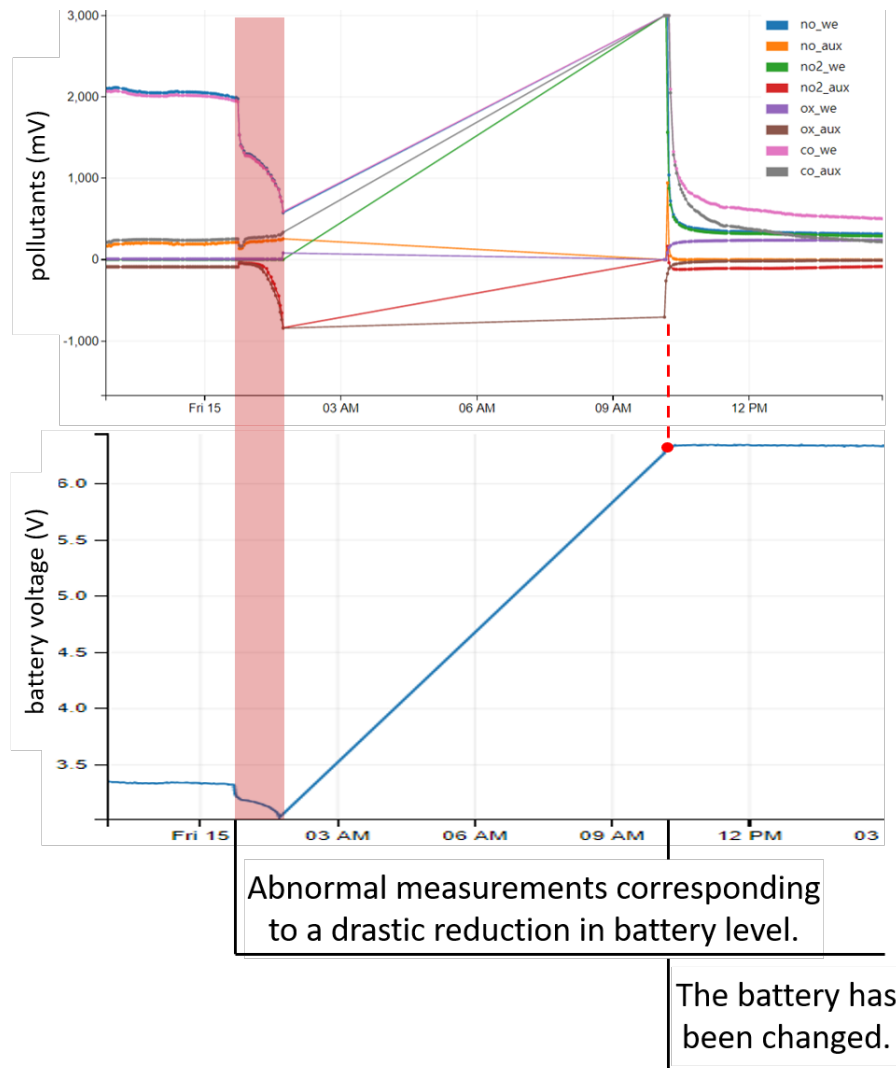


Fig. 4.17 An anomalous behavior of a device detected on January 15th, 2021, due to a drastic reduction in the battery level.

Sensor anomalies The accuracy of the raw measurements can be influenced by multiple factors, i.e. the low level of battery voltage, the weather conditions, the air humidity. Distinguishing not correct data allows for providing more reliable data and could improve the results of the calibration task. The anomalies visualized in this view are the results of the Majority Voting described in Section 5.3.1.

The “sensor anomalies” view consists of one plot for each sensor with the raw observations and the anomalies identified by the majority voting system (requirement R6). Also in this case, the user can choose for the observations of the last 24 hours, week, or month.

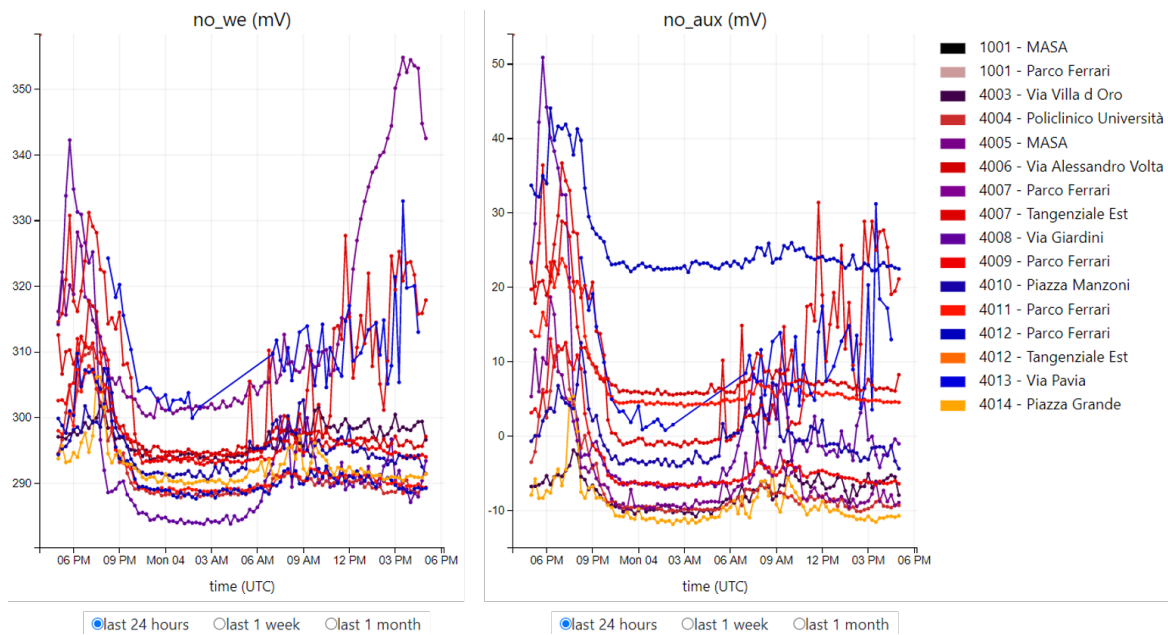


Fig. 4.18 A visualization of the “gas observations” view which shows the measurements of NO channels.

Figure 4.19 is an example of anomalies visualization for sensor 4006. Anomalies are identified by a point. As can be seen in the figure, in most cases anomalies are detected in the upper peaks of the time-series.

Calibrated observations The results of the calibration process (continuously re-trained Random Forest algorithm) consist of the concentrations of the 4 measured gases. Starting from 2 values for each gas (one value for each of the two channels) in millivolts, the calibration provides one value in $\mu\text{g}/\text{m}^3$.

The “calibrated observation” view shows the result of the last calibration algorithm, that is the most recent and accurate algorithm available for the visualized data. This view meets requirement R5. The calibrated observations are organized in 4 plots, one for each gas, and the user can distinguish the measurements of each device through the integration of the interactive legend.

The calibrated values can be directly compared with the measurements of the AQM stations since they are in the same unit of measure. To validate the calibrated data one local warning threshold for each gas has been defined based on the measurements of the AQM stations. Each threshold has been calculated as $1.25 * M$, where M is the maximum value measured by the AQM stations for the specific gas in the year preceding the date of the observation to be compared. If the concentration of the gas is higher than the corresponding

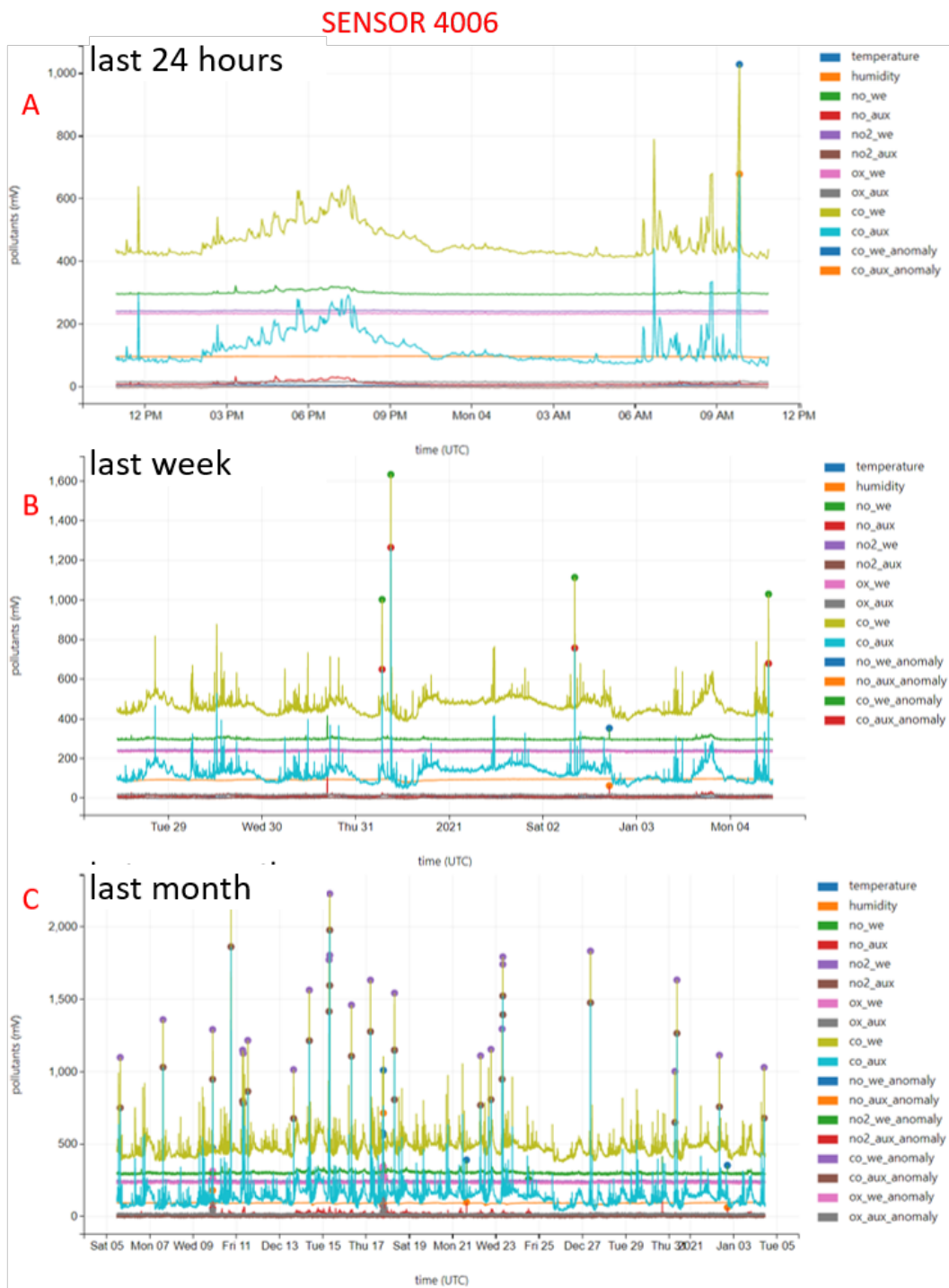


Fig. 4.19 Anomalies of sensor 4006 for the last 24 hours (A), the last week (B), and the last month (C) available on January 4th, 2021 at 11 a.m..

threshold, it is automatically flagged as “anomalous” in the TRAFair database by a Python process running in real-time. The warning threshold is valid only in the area of Modena

since it is provided by the certified values of the AQM stations of Modena and it changes every year. This threshold allows to exclude very high values that are most likely due to malfunction of the sensor. It is not to be confused with the alert thresholds of the European Commission²⁰ or the reference levels of the European Environment Agency²¹, which defines the values to assess the level of pollution in the area.

In the plots of the “calibrated observations” view, a line in correspondence of the threshold value is plotted only if at least one measurement exceeds the threshold. The plots in Figure 4.20 show the measurements of *NO* made by 5 different devices installed in the same location named “Parco Ferrari” (this is also the location of an AQM station). Only the devices in the same location have been selected through the interactive legend. The concentrations measured by the devices are very similar, as expected. In the “last month” plot the blue line indicates the above-mentioned local warning threshold and only one value is higher than this threshold.

Certified AQM station measurements The sixth view of SenseBoard is devoted to the visualization of AQM station observations. They are hourly certified data related to the concentrations of *NO*, *NO*₂, *NO*_x, and *O*₃ measured by the two AQM stations installed in Modena (red points in Figure 4.7).

Expert evaluation

SenseBoard has been regularly used by 4 environmental experts from January 2020 till the end of TRAF AIR and it is still active. It has allowed:

1. the recording of 291 location/status updates,
2. the identification of network malfunctions in real-time (which occurred twice in the last year and caused the loss of 1-2 days of data),
3. the detection of sensor faults in semi-real time and anomalous cell behaviour (which occurred 4 times and brought to the cell replacement),
4. the identification of low battery level which caused anomalous observations (33 times in around 14 months),
5. the daily comparison of concentrations from low-cost sensors and certified measurements from AQM stations to evaluate the calibration algorithm,

²⁰https://ec.europa.eu/environment/legal/law/5/e_learning/module_2_18.htm

²¹<https://www.eea.europa.eu/themes/air/air-quality/resources/air-quality-map-thresholds>

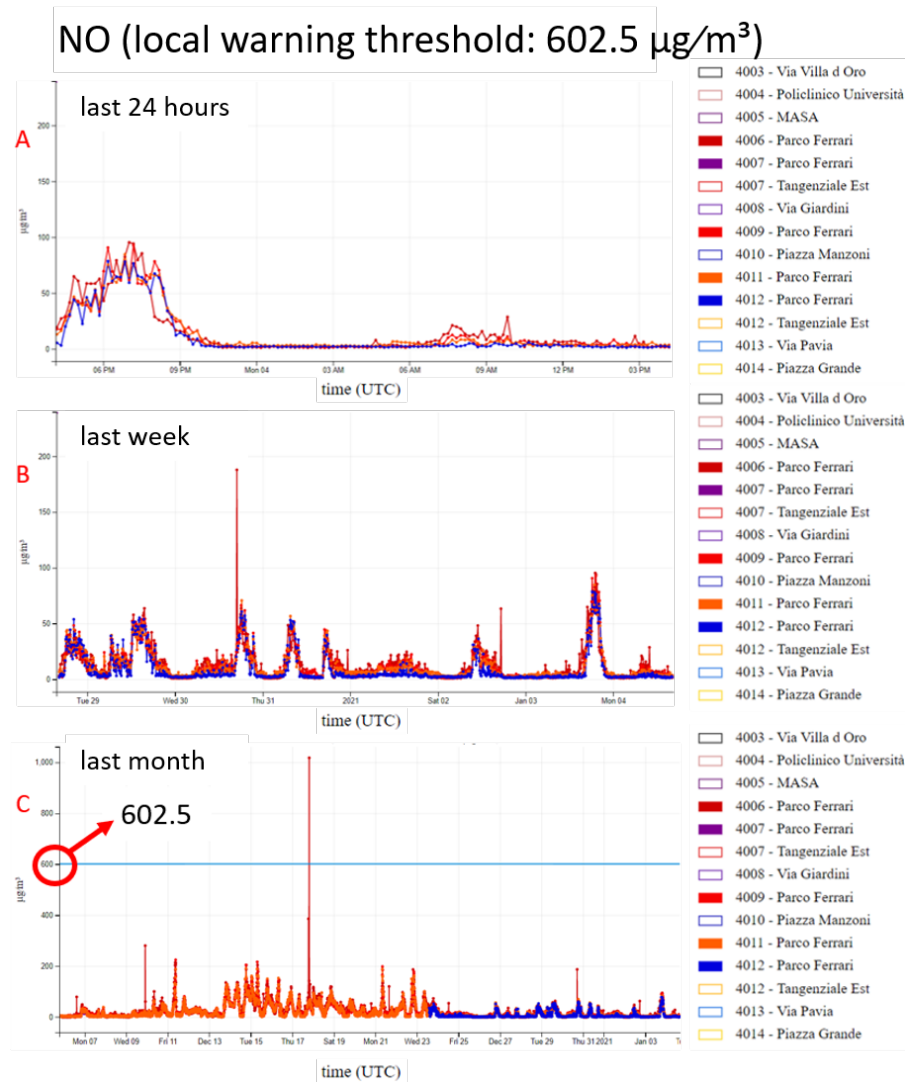


Fig. 4.20 Calibrated NO observations by 5 devices located in the same place (“Parco Ferrari”) visualized on January 4th, 2021 at 4 p.m..

6. the detection of strange behaviour in the anomaly detection process which allowed to retrain the algorithm and restart it.

The effectiveness of SenseBoard was widely appreciated by environmental engineers who would not have had the opportunity to compare sensor measurements and calibrations and to carry out such sudden checks and maintenance.

Chapter 5

Anomaly detection on sensor data

Anomaly detection is a research area of Data Science and Machine Learning that has received much attention and has been the topic of several surveys and review articles [45–51]. It is one of the major challenges in computer science and concerns several application domains, such as wireless sensor network, privacy and cybersecurity, communication networks, and social media.

In recent years, with sensors pervading our everyday life, an exponential increase in the availability of streaming time-series data has been registered. Although the sensor technology has evolved greatly allowing for an improving in the sensor performance and an increase in the variety of sensors, these devices are prone to malfunctions. The environment where IoT devices are developed makes them vulnerable to failure and malfunction, leading to the generation of unusual and erroneous data [52–55]. Therefore, data cleaning processes are needed to detect outliers and remove unreliable data before further analysis devoted to derive actionable insights.

Anomalies are characterized by two aspects: they differ from the normal instances and they are rare. Probably, the first definition of anomaly was given by Grubbs in 1969 [56]: “An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs”. Anomaly detection (or outlier detection) is the task of identifying these abnormal instances.

In the field of sensor networks, anomalous data can be generated by several factors. Based on the description of traffic and air quality sensor data provided in Chapter 4, performing anomaly detection on sensor data means performing anomaly detection on time-series (if only one sensor at a time is considered) or spatial time-series (if more than one sensor and their relative position are considered). Spatio-temporal outlier detection is the identification of instances that exhibit abnormal behavior either spatially, and/or temporally. In some cases, the comparison of data coming from close sensors (neighborhood information) can

be exploited to improve the identification of anomalies. Exploiting spatial information for anomaly detection makes the approach more robust because it allows not only to compare one sensor's measurements with respect to its past measurements but also to the measurements of close sensors. On the other hand, managing the spatial features is very challenging.

Anomalies in traffic sensor observations can heavily affect the results of the subsequent analysis, such as traffic flow simulation, traffic trend analysis, traffic monitoring, and prediction. On the other hand, anomalies on air quality sensors have an impact on the study of the air pollutant interpolation and the information released on air quality at specific sites. Also, it is important to understand the causes of anomalies. Outliers can have two different meanings: they can be observations related to noise, erroneous, or unwanted data, or they can be unusual but real phenomena. In traffic sensor observations, anomalies can derive from malfunctioning or non-conventional traffic conditions, such as accidents, slowdowns, or street closures, representing a regular change in the environment. Road traffic congestion can happen as consequence of traffic accidents or road works. These situations can be named "unusual traffic conditions" and shall not be discarded from the dataset. In air quality sensors, anomalies can be due to low battery voltage, adverse weather conditions, air humidity, physical disturbances, or damage of the measuring cell. In some cases, the human intervention is required to solve the problem.

Removing anomalies in time-series data increases the number of missing values. For some scopes this should represent a problem. Thus, anomaly repairing techniques aim at replacing anomalous data with reliable measurements.

There are several challenges regarding anomaly detection, including the unavailability of labeled data. This makes the evaluation of the anomaly detection methods very difficult. In literature, different techniques to perform data cleaning on sensor data have been discussed, classified, and compared. In the following, a review of the main techniques is presented along with the application and evaluation of some anomaly detection algorithms on the TRAFAIR traffic and air quality datasets. The experiments have been executed on a High-Performance Computing (HPC) Debian machine with 32 Intel(R) Xeon(R) Silver 4108 CPUs @ 1.80GHz and 256 GB RAM. In the end, experimental execution of anomaly detection Machine Learning algorithms on edge devices (Section 5.3.2) is provided. This work is done with the collaboration of the National University of Ireland Galway (NUIG), in Ireland, where I spent three months from September 2021 to December 2021.

Part of the research presented in this chapter has been published in [37, 57, 58].

5.1 Related work

Anomaly detection techniques can be classified based on several aspects. In the literature, there are mainly three types of anomaly detection techniques:

- *supervised algorithms* [59] need the knowledge of both normal and anomaly class, the training dataset contains labeled instances identified as normal or anomaly and is used to build a predictive model able to identify the type of data,
- *unsupervised algorithms* [60] do not require training, they detect anomalies based on the assumption that anomalies occur rarely,
- *semi-supervised algorithms* require a training dataset that does not contain anomalies, thus, the built model is able to detect anomalies as they deviate from the normal behavior learnt by the model.

Also, anomalies can be classified in three classes: *point anomalies* (or global outliers) are single data points that are not compliant with the trend of the examined variable; *contextual anomalies* (or local outliers) are data points that are unexpected in a specific context, e.g., a high number of vehicles in the night hours; *collective anomalies* are consecutive data points that are anomalous with respect to the entire trend or dataset [61]. Point anomalies can be detected by statistical-based algorithms or, most in general, by algorithms able to understand the patterns and trends in the data. Instead, contextual anomalies require knowledge on other features in addition to the analyzed data. Finally, identifying collective anomalies requires extensive and cleaned training sets.

Since in TRAFIR labeled datasets are not available, the analysis of related works has been focused on unsupervised techniques, with a particular attention to methods more suitable for univariate or multivariate time-series. Unsupervised anomaly detection [50] can be performed through statistical approaches, Machine Learning algorithms, deep Learning using autoencoders [48, 62], and neural networks [63–65]. A classification of these approaches includes nearest-neighbor based algorithms, clustering or distance-based techniques [66, 67], subspace based algorithms, classifier, prediction [68–70], and angle based algorithms [71].

In [72], 20 different univariate anomaly detection methods of the above-mentioned categories are compared and evaluated on publicly available datasets. Also, Seasonal-Trend decomposition using Loess (STL) [73] is exploited for anomaly detection combined with other methods, such as the Interquartile Range (IQR) analysis. STL decomposition is a widely used method to perform outlier identification on time-series data. In [73], a mathematical explanation of the decomposition of time-series in seasonal, trend, and residual components is provided. The residual component can be analyzed to identify data points that deviate

from normal behavior. The approach described in [74] combines STL decomposition with extended isolation forest.

In [75], some time-series anomaly detection algorithms are proposed: ARIMA (Autoregressive Integrated Moving Average), LSTM (Long Short-Term Memory), DBSCAN (Density-Based Spatial Clustering), and GANs (Generative Adversarial Networks). ARIMA is a general-purpose technique effective at detecting anomalies in data with regular daily or weekly patterns. Extensions of ARIMA enable the automatic determination of seasonality. ARIMA has also been applied in the context of traffic anomaly detection that considers imbalanced, non-stationary properties of the traffic sensor network [76, 77], and it showed remarkable detection precision and real-time performance. In [78], a two-layer outlier detection approach is proposed: firstly, the non-stationarity and periodic variation of the time-series are studied, then observable variables in the environment are used to explain any additional signal variation. The authors of [79] combine STL decomposition with the SARIMA model, an extension of ARIMA, to detect anomalies in non-periodic time-series.

Focusing on spatial time-series, in literature, there are not many algorithms that exploit both spatial and temporal features. Some examples of algorithms based on temporal components are presented in [80, 81], while the Local Outlier Factor (LOF) [82], DBSCAN [83] and LDBSCAN [84] are based on the spatial component. For detecting spatio-temporal outliers using both the spatial and temporal features, a promising approach has been recently published, combining the Spatio-Temporal Behavioral Outlier Factor (ST-BOF) in cascade with the Spatio-Temporal Behavioral Density-based Clustering of Applications with Noise (ST-BDBCAN) [85] algorithm. In this case, correlation among sensors is taken into account.

Correlated anomaly detection is a cutting-edge topic. Especially from streaming data, it is an essential task in several real-time data mining applications. In [86], a framework is introduced for better detecting correlated anomalies from large streaming data of various correlation strengths. The experiment refers to the U.S. stock daily price dataset and shows balanced recall and estimated accuracy. In the traffic context, the correlation among traffic sensors to detect anomalies has been analyzed in [87]. The authors exploited a statistical baseline method, along with a sensor correlation analysis. They evaluated the approach by comparing the detected anomalies against traffic alerts, which are emitted by Traffic Agents on Twitter. In [88], faulty readings from traffic sensors are identified by examining the correlations among them and by taking advantage of the ubiquitous citizens through crowd-sourced data. The authors evaluate cross-correlation between sensors using, firstly, the Pearson metric, and then employing a multivariate ARIMA model to detect anomalies considering correlated sensors. The authors of [89] present an isolation-based distributed

outlier detection framework that exploits the spatial correlation among sensors and employs the Local Outlier Factor (LOF) together with the nearest neighbor algorithm.

To distinguish anomalies due to sensor faults from unusual but real phenomena correlation between sensor measurements and social events or accidents can be analyzed. In [90], an accurate and transferable accident detection approach is described. The detection methodology is based on the relationship between traffic variables and observed traffic accidents. Authors employ a deep learning-based method calibrated using part of the collected traffic variables and the pre-assigned traffic accidents.

In environmental data, the detection of anomalies on raw measurements cannot consider correlation between close sensors. Even if sensors are located in the same position, it is not possible to compare their raw data since the chemical cells inside each sensor are unique and can measure very different millivolt values for the same values of pollutants concentration in the air. For this reason, every single cell needs a specific calibration model and the anomaly detection procedure cannot be based on neighboring sensors. Several techniques have been explored in order to detect outliers in gases or particle observations through functional analysis [91], probability finite state automata-based algorithm [92], statistical methods [93, 94], or combined methods [95]. In the air quality monitoring context, anomaly detection, data cleaning, and repairing methodologies are usually applied on the calibrated observations, as post-processing techniques [96], e.g., concentrations obtained from raw measurements after applying a trained calibration model (details on the calibration process are explained in Section 4.2). In [97], the author proposes the implementation of two iterative algorithms and their application on both synthetic data and real-time data. The scope is to understand how gas emissions of printer jobs impact the concentrations of pollutants (CO , CO_2 , NO_2 , and O_3) in the indoor environment of an office. The authors use the Forgetting Factor Iterative Data Capture Anomaly Detection (FFIDCAD). In [70], a real-time univariate anomaly detection method has been developed. The method requires no pre-classification of the anomalies and is applied to a heterogeneous sensor network.

5.2 Anomaly detection on traffic sensors

Since traffic sensors are installed under the surface of the street, their maintenance cannot be continuously granted, and sensors can be faulty. The goal of anomaly detection in the TRAFair pipeline is to identify anomalies among the data coming from the traffic sensors and exclude them from the input of the traffic simulation model. Since the traffic model takes in input sensor data aggregated every 15 minutes, in most cases anomaly detection is



Fig. 5.1 Road A-B with five vehicles (red rectangles).

performed on this aggregation. The flow is summed up and the average of speed weighted on the flow is calculated.

Firstly, a filter based on the correlation among flow and speed measured by the sensors has been defined to exclude measurements that are anomalous based on the meaning of the measures, e.g., very high values of speed and/or flow that are unfeasible in real life. Then, additional approaches to detect anomalies have been evaluated on the TRAFAIR traffic datasets, including STL decomposition and IQR analysis, ARIMA model, FFIDCAD, and the combination of ST_BOF and ST_BDBCAN.

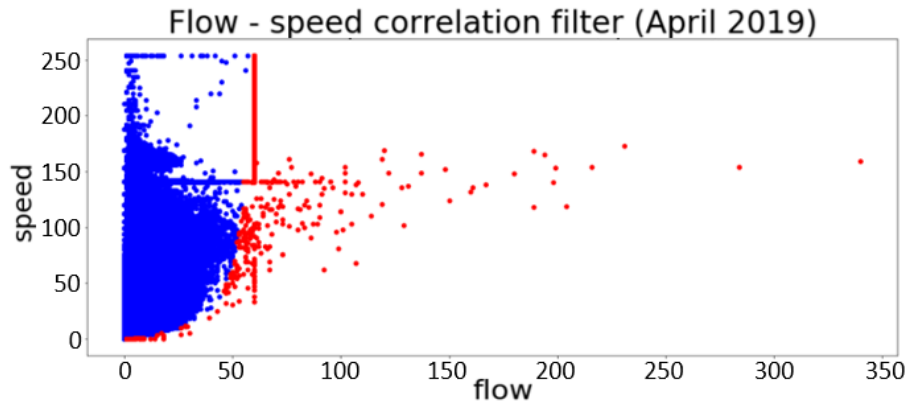
5.2.1 Flow-speed correlation filter

Flow and speed are in relation to each other, e.g., when there are a lot of vehicles (high flow), the vehicles are forced to move slowly (low speed). In a fixed time interval, there is a maximum number of vehicles that can pass on a road at a certain speed. Figure 5.1 tries to clarify the proposed approach. The road is the segment from point A to point B. Each red rectangle is a vehicle whose average length is 4 meters. Each vehicle has a safe distance (white space between vehicles) from the previous vehicle that depends on the speed and can be calculated as:

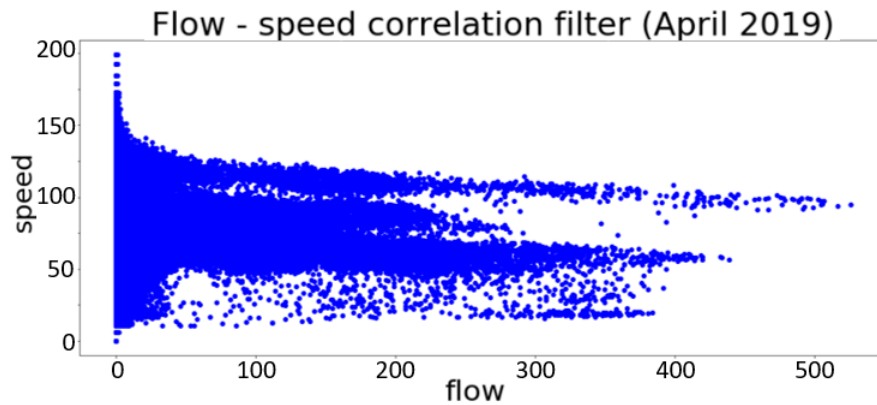
$$safe_distance = \frac{speed}{3.6}$$

where speed is the speed measured by the sensor. The scope is to calculate how many vehicles can pass over point B with a certain average speed $average_speed$ in Km/h during a time interval of one hour. This is equivalent to find how many vehicles can stay in a road of length $average_speed$. The following example demonstrates this assumption: if a road A-B of 50 Km length is considered, the first vehicle near point A is 50 Km far point B and takes one hour to reach point B if it passes with a 50 Km/h speed. All the vehicles between A and B will reach point B during this hour. Therefore, if we figure out how many vehicles can fill the road A-B, this is also the maximum number of vehicles that can pass over B in one hour with a speed of 50 Km/h. This number is calculated by the formula:

$$num_vehicles = \frac{speed(Km/h) * 1000}{vehicle_length + safe_distance}$$



(a) Traffic sensor observations in the urban area (1 minute time interval).



(b) Traffic sensor observations in provincial and regional roads (15 minutes time interval).

Fig. 5.2 Flow - speed scatter plots representing the observations of traffic sensors in April 2019 with filtered observations in red.

All the measurements with flow higher than *num_vehicles* computed considering the speed of the measurement are anomalies, thus, they can be discarded.

Experiments and results

In April 2019, the number of observations coming from the traffic sensors was 13 millions, and they were produced by 338 sensors.

Using the flow-speed correlation filter, 450845 observations are filtered out (3% of the total number of observations). These filtered observations are related to 259 sensors. The scatter plots in Figure 5.2 show the values of flow and speed of the observations of urban sensors (Figure 5.2a) and sensors outside the urban area, in provincial or regional roads (Figure 5.2b) in April 2019; the red points are the filtered observations. As can be seen, no filtered observations are found among data coming from provincial and regional sensors. In

Figure 5.2a, it can be noticed that very high values of speed are considered “non-anomalous” for low values of flow. Indeed, if there is no traffic (low flow), it could be possible that vehicles move at high speed, especially at night. However, for higher values of flow, an observation with a very high speed is considered “anomalous” and filtered.

The filter has been implemented as a trigger in the TRAFAIR data platform (Chapter 3) every time a new traffic measurement is stored, and filtered measurements are labeled in the database. Therefore, the filter is applied to non-aggregated measurements.

5.2.2 Seasonal Trend Decomposition using Loess

The method described in this section integrates the use of STL decomposition with the above-explained filter. The filter identifies anomalous values that would not be recognized as anomalies by the STL decomposition technique. Detected anomalies are then classified to identify sensor faults. This method does not assume labeled historical data and can be applied in real-time, i.e., just after storing sensor data on the data platform. To evaluate the performance of such an approach, the improvement provided by the anomaly detection has been evaluated by comparing the traffic model outputs, considering or excluding sensor faults.

As represented in Figure 5.3, the methodology consists of several steps:

1. Filtering observations with the flow - speed correlation filter as described in Section 5.2.1.
2. Repairing anomalies by replacing “filtered” observations with average values. Flow values are summed up to evaluate the 15-minutes aggregated data replacing the flow of filtered observations with the average of the reliable (non-filtered) flows measured by the sensor in the aggregated time interval (i.e., the 15-minutes interval the filtered observation belongs to). For the speed, it is assumed that filtered observations have speed equal to the weighted averaged speed evaluated considering only the non-filtered measurements in the 15 minutes time interval. This anomaly repairing technique is not applied when less than 2 reliable observations are available for a sensor in the 15 minutes time interval. In this case, the aggregated flow and speed of that 15 minutes time interval are classified as anomalies, and they are not used in the following steps.
3. Detecting anomalies through STL decomposition following two different approaches. Anomalies are classified as unusual traffic conditions when they occur contemporary in correlated sensors, otherwise, they are labeled as sensor faults.

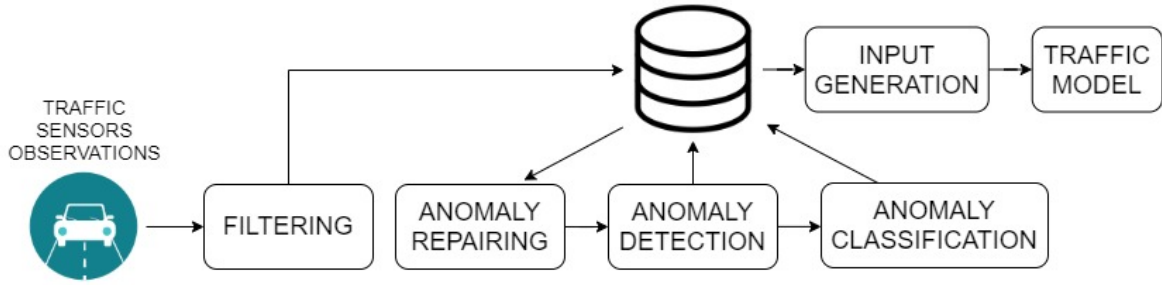


Fig. 5.3 Workflow of the proposed methodology.

4. Classifying anomalies in “sensor faults” and “unusual traffic conditions” by using the Detrended Cross-Correlation Analysis (DCCA) method and identifying sensors with similar trend of observations. Usually, sensors placed in the same junctions are often correlated or sensors on the same road and lane are correlated sensors.
5. Generating the input for the traffic simulation model by removing sensor faults.
6. Running the traffic model.

The focus of this thesis is on phase 3, i.e., the use of STL decomposition for detecting anomalies, the detailed explanation of the other phases can be found in [57].

The Seasonal-Trend Decomposition using Loess (STL) is a filtering procedure for decomposing a time-series into three components: the trend, the seasonal, and the remainder (also called residual) [98]. Splitting the time-series into components can help in identifying anomalies. The additive decomposition considers the time-series model described by the following formula:

$$y_t = \tau_t + s_t + r_t, \quad t = 1, 2, \dots, N$$

where y_t represents the observation at time t , τ_t is the trend in time-series, s_t is the seasonal signal with period T , and r_t is the remainder component.

The trend component consists of the low-frequency variations in the data with non-stationary, long-term changes, i. e. continuous increase or decrease. The seasonality instead is composed of the variations (periodic patterns) in the data near a baseline. Typically, trend changes faster than seasonal. The remaining variations are included in the residual.

The decomposition of the time-series is obtained by applying the locally-weighted regression (Loess smoother) several times, iteratively. The result of these applications is a curve representing a smoothing of the original time-series, computed taking into account the value of a variable number of observations in the neighborhood.

A variant of STL is the RobustSTL [73], defined as a robust and generic seasonal-trend decomposition method able to extract seasonality from data with a long seasonality period and high noises. The method applied by the RobustSTL consists of four steps:

1. noise removal by applying bilateral filtering and using neighbors in a window of $2H + 1$ observations with similar values to smooth the time-series;
2. trend extraction by using the least absolute deviations (LAD) loss with sparse regularization;
3. seasonality extraction through non-local seasonal filtering which takes into consideration K seasonal neighborhoods of $2H + 1$ observations with different weights according to their distance in the time dimension and their seasonality values;
4. final adjustment by calculating the mean of seasonality, which is added to trend and removed from seasonality.

Several configuration parameters must be provided to the algorithm: the period T , that is the number of observations in each cycle of seasonality, the hyper-parameters of the bilateral filter of step 1 ($dn1$ and $dn2$), the number of the neighborhood (H), the regularization parameters for trend extraction ($reg1$ and $reg2$), the number of past season samples (K), and the hyper-parameters of the bilateral filter in seasonality extraction step ($ds1$ and $ds2$).

After each step, the input signal is updated by removing the component extracted in that specific step. After step 4, the steps are repeated until you get convergence between the remainder of the current iteration and the one of the previous iteration. The convergence is computed by the following formula:

$$convergence = \sqrt{(r_i - r_{i-1})^2}, \quad i = 1, 2, \dots, N$$

where r_i is the remainder at iteration i (current iteration) and r_{i-1} is the remainder at iteration $i - 1$ (previous iteration). If the convergence is higher than a threshold, the process will continue with another iteration, otherwise, the decomposition obtained is considered definitive. In the latter case, the results are 3 time-series representing trend, seasonal, and residual.

Once the decomposition is concluded, the curve of residual can be analyzed to detect anomalies by using different methods. A solution could be the application of the interquartile range (IQR), which allows the calculation of the two fences to define reliable values. More details are provided in the following.

Two versions of the anomaly detection process have been evaluated: ADP1 and ADP2. The operations of each version are explained in Figure 5.4. ADP1 applies the logarithm

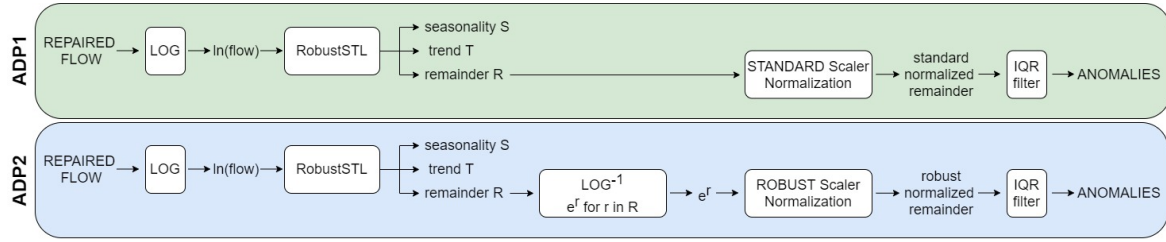


Fig. 5.4 The two versions of anomaly detection process: ADP1 and ADP2.

to the flow values obtained after anomaly repairing and provides it to RobustSTL. Then, remainder values are normalized by using mean and standard deviation, as follows:

$$r_normalized_t = \frac{r_t - mean}{standard\ deviation}$$

where *mean* and *standarddeviation* are computed on the remainder. Thus, the first ($Q1$) and third ($Q3$) quartiles are calculated on the normalized residual to find lower and upper fences with the following formulas:

$$lower\ fence = Q1 - k * IQR$$

$$upper\ fence = Q3 + k * IQR$$

where *IQR* is the difference between third and first quartiles, and *k* is a multiplier. The observations with a residual value lower than *lowerfence* or higher than *upperfence* are outliers. The higher the multiplier value, the fewer outliers are found. The value of *k* depends on which type of outliers to detect; high values of *k* are used for extreme outliers.

In ADP2, after the application of RobustSTL, the inverse function of the logarithm is applied to the residual values. The obtained values are normalized using the Robust Scaler, as follows:

$$r_normalized_t = \frac{r_t - median}{IQR}$$

where *median* and *IQR* are evaluated on the remainder after the inverse logarithm function. Then, the lower fence and upper fence are calculated on the normalized remainder values and the IQR filter is applied to them. Again, the observations with residual value out of the range between the lower fence and the upper fence are outliers.

n processes have been set up to apply STL decomposition to sensors data in real-time, where n is the number of sensors in the dataset. Indeed the decomposition of the time-series

of one sensor is independent of the others. An implementation of RobustSTL available online¹ has been exploited.

In both anomaly detection process versions, the decomposition is performed every 15 minutes. The time-series, used as input, is generated grouping by 15 minutes the logarithm of one-week flow measurements of a specific sensor. After the application of IQR, only the anomalies of the last 15 minutes are taken into account since the anomalies on the previous time slots have already been extracted by the previous applications of the decomposition. The choice of using one week observations is a trade-off between the need to provide context for decomposition and the requirement of having results in a short time. Indeed, the time required for one month decomposition (on average 12 minutes) is far greater than the one for week decomposition (on average 15 seconds). Therefore, the period T was set up to the number of observations in the week divided by 7. After several experiments, the choice was to set the values of the other configuration parameters in this way: $dn1=1$, $dn2=1$, $H=3$, $reg1=10$, $reg2=0.5$, $K=1$, $ds1=50$, $ds2=1$. In the end, the value of k in the IQR method was initialized to 3 since the scope is to avoid that real unusual traffic conditions are labeled as anomalies. Using a lower value for k , a lot of observations were considered anomalies and, checking the values of flow and speed, they did not seem like real anomalies. Probably this is due to the fact that the sensors are located near the traffic lights, therefore, the presence of peaks is possible and could be related to the turned green of the traffic light.

Experiments and results

To demonstrate the effectiveness of the proposed methodology, the traffic model performances without a data cleaning procedure (standard simulation) and excluding anomalies (cleaned simulation) have been compared for each day of April 2019. First, the results of ADP1 are discussed. Then, the performances of the traffic simulations obtained excluding anomalies detected by ADP2 and classified as sensor faults are presented.

Evaluation of the traffic simulation model The main goal of a traffic model is to simulate a traffic flow close enough to real values observed in urban streets. The model provides as output the number of vehicles counted in the exact position where the traffic sensor is located. The vehicles number simulated in a point where a traffic sensor is located can be compared with the real vehicle counts. Even if traffic sensor data are used to feed the model, calibrators are not always able to insert the required number of vehicles; thus, the vehicles inserted in the simulation and the vehicles counted by real sensors can be different. This mainly happens when a sensor measures a very high flow and its calibrator inserts many vehicles causing a

¹<https://github.com/LeeDoYup/RobustSTL>

jam in the simulation and avoiding other calibrators, placed nearby, to eventually add other vehicles. This very high flow could be caused by a sensor fault that affects the performance of the entire simulation.

The evaluation method of the presented traffic model is described in [36]. For each sensor, whose measurements are employed as input for the model, the distance between the two time-series, i.e., real flows observed by the sensors and simulated flow, is calculated with three different metrics: the fast Dynamic Time Warping (DTW), the PointWise Distance (PWD), and the Count Time slots Distance (CTD).

The DTW is evaluated using FastDTW, a less complex version of DTW described in [99]. This metric allows sequences to be stretched along the time axis, is able to find corresponding regions in time-series, and can tolerate noise, time shifts, and scaling in the y axis [100]. The PWD is evaluated by summing all the differences between the measured flow and the simulated flow in all the time steps of the simulation. Distances are summed considering their sign since a subsequent time slot can compensate for the difference observed in a previous one. The CTD is the number of time slots in which the absolute difference between the measured and simulated flow is higher than 2 vehicles per minute.

Calibrators have been classified considering the presented metrics as “aligned” and “not aligned”. A calibrator is considered “not aligned” if the DTW distance between real measurements and simulated flow is higher than 1200, the PWD is higher than 30, and the CTD is higher than half of the total time steps of the simulation. A calibrator classified as “not aligned” is not able to insert correctly the expected number of vehicles as required by the given input.

The evaluation of the simulation performances needs metrics able to summarize with a unique value the distances observed in each sensor position. Five metrics have been defined to compare simulations performed considering or excluding anomalies referring to the whole simulation: percentage of aligned calibrators, mean Root Mean Squared Error (RMSE), mean DTW reduction, mean PWD reduction, and mean CTD reduction. For each simulation, the number of calibrators classified as aligned has been divided by the total number of calibrators to obtain the percentage of aligned calibrators. A value of the percentage of aligned calibrators is calculated for the simulation performed including anomalies and the simulation of the same period excluding anomalies from the input of the model. The RMSE between the real measurements and the simulated flows is evaluated for each sensor and averaged. For each simulation including and excluding anomalies, a value of mean RMSE is calculated. A higher mean RMSE stands for a bigger error and thus a decrease in the performance of the model. Mean DTW reduction, mean PWD reduction, and mean CTD reduction are obtained comparing the performances of two simulations: the simulations with

all the available measurements (standard simulation) and the simulation excluding anomalies (cleaned simulation). If the value of the reduction is positive, the distance has been reduced, thus the cleaned simulation performed better; otherwise, the standard simulation has better performances.

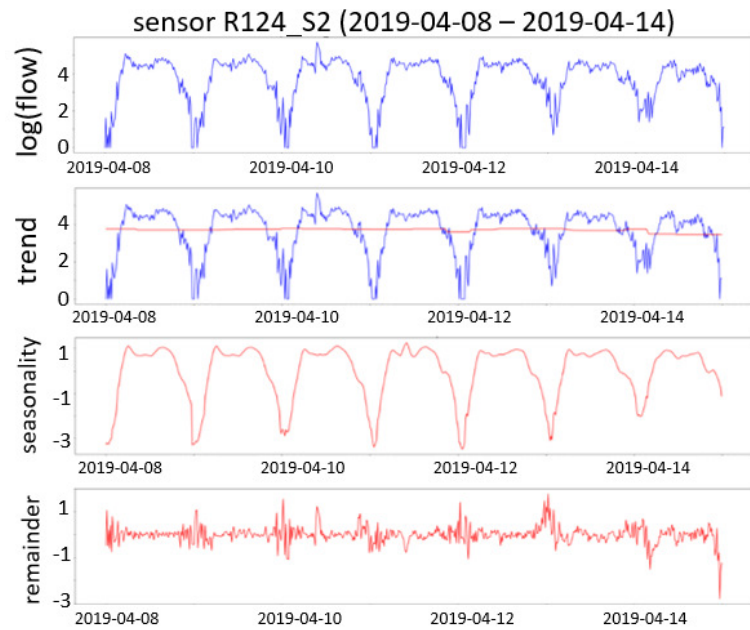
Improvement in traffic simulation with anomaly detection (ADP1) After the anomaly repairing phase is applied to one minute filtered data, all sensor observations are aggregated every 15 minutes. The STL decomposition, applied to aggregated traffic data by using the IQR method, detects 13932 anomalies (less than 0.1% of the observations) related to 310 sensors. Figure 5.5a shows an example of decomposition which refers to the observations of one sensor from April 8th to April 14th, 2019. time-series decomposition involves thinking of a series as a combination of trend, seasonality, and remainder (also called noise or residual) components. Figure 5.5b draws anomalies, highlighted in orange, on the time-series of all observations; while Figure 5.5c shows anomalies on the remainder component of the time-series. This last Figure highlights that the anomalies are detected in positive and negative peaks on the remainder component of the time-series.

Analyzing the time distribution of the anomalies, most of them are detected at night, as can be seen in Figure 5.6. Figure 5.7 shows the percentage of anomalies for every day of April 2019: in one day (April 6th, 2019) the percentage exceeds 2%.

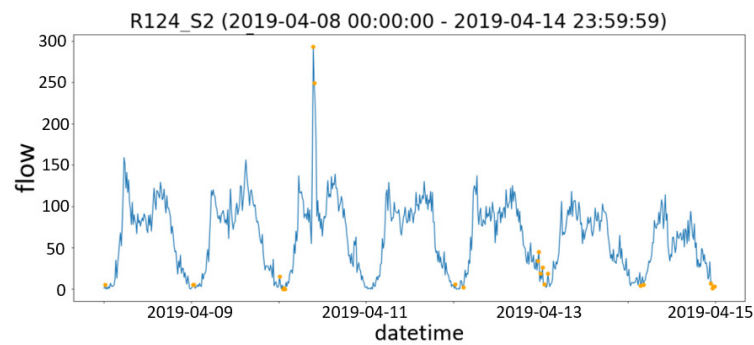
Once the anomalies have been detected by ADP1, they are stored in the database. Each anomaly is linked to the 15-minutes time interval of the aggregated observations and the sensor it belongs to.

For each day of April 2019, two simulations have been performed. The STD SIM uses as input all the available sensors observations (no data cleaning is performed on them). The cleaned simulation ADP1 SIM, instead, is obtained applying ADP1 to traffic data and removing anomalous observations from the input. The anomalous observations are removed from the input of the traffic model and calibrators simulate vehicles considering previous and next observations. Classification of anomalies is not applied in this case. The evaluation of both the standard simulation (STD SIM) and the ADP1 SIM are performed considering only the non-anomalous points. This is due to the assumption that real measurements labeled as anomalous are not reliable and cannot be used to estimate the error.

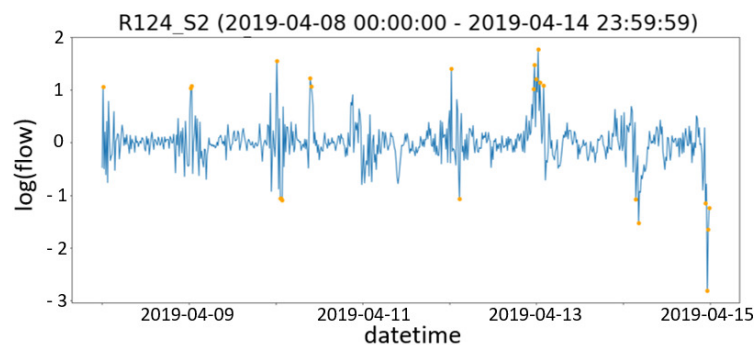
In Table 5.1, all the evaluated metrics are displayed for each day of April 2019. Comparison metrics are evaluated, considering as the first simulation the standard simulation (STD SIM), and as the second simulation the simulation without anomalies detected by ADP1 (ADP1 SIM). In the 77% of daily simulations, the percentage of aligned calibrators increased excluding anomalies (as can be seen in the third column of Table 5.1), only on 3



(a) Decomposition of the time-series.



(b) time-series with anomalies (i.e. the orange dots).



(c) Remainder with anomalies (i.e. the orange dots).

Fig. 5.5 The STL decomposition of the time-series related to the observations of the sensor R124_S2 from April 8th, 2019 to April 14th, 2019 and anomalies detected by ADP1.

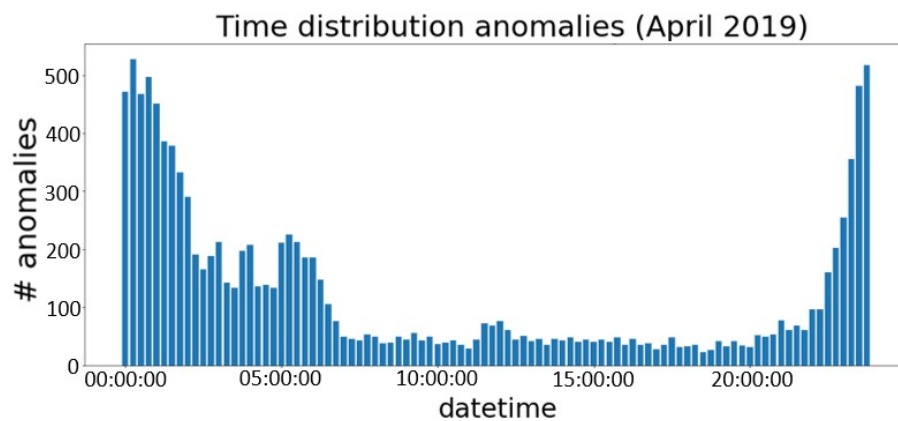


Fig. 5.6 Time distribution of anomalies in April 2019.

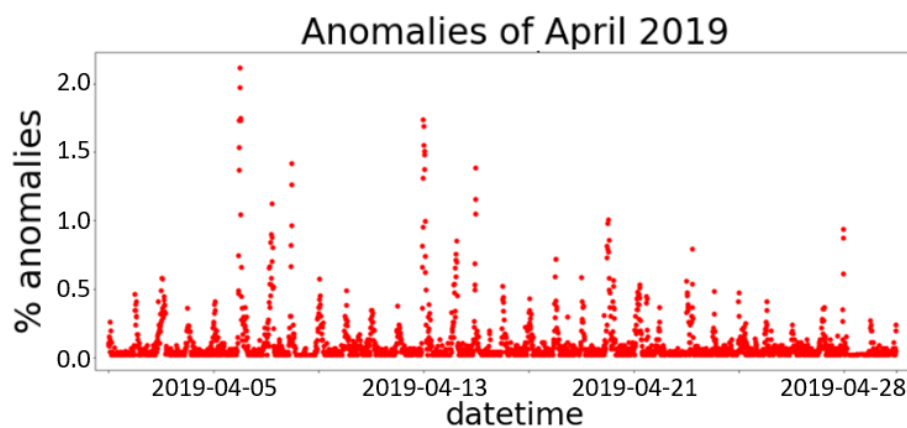


Fig. 5.7 Percentage of anomalies above the total number of observations for every day of April 2019.

days the number of aligned calibrators decreases. In the 73% of cases, the mean RMSE error decreases (as can be seen in the fifth column of Table 5.1). If mean DTW reduction has a positive value, the exclusion of anomalies reduces mean DTW distance. In only the 60% of daily simulations, the mean DTW reduction is positive (as can be seen in the sixth column of Table 5.1). Moreover, the mean PWD reduction is positive in the 60% of daily simulations (as can be seen in the seventh column of Table 5.1), this means that in the majority of days, the mean PWD was reduced through the data cleaning process. Finally, the mean CTD reduction is positive in the 90% of daily simulations (as can be seen in the eighth column of Table 5.1), thus, the mean number of time slots in which the distance between the simulated flow and the real measurements was higher than 2 (*veh/minute*) is significantly reduced. Overall, the 83% of days shows an improvement on at least 3 out of 5 metrics in the ADP1 SIM. For each day, the number of filtered one-minute data and the number of anomalies detected using STL

Table 5.1 Comparison of traffic model evaluation metrics between standard simulations (STD SIM) and simulations after removing anomalies of ADP1 (ADP1 SIM) in April 2019.

day	% aligned		mean RMSE		DTW reduction	PWD reduction	CTD reduction	filtered	anomalies
	STD SIM	ADP1 SIM	STD SIM	ADP1 SIM					
1	0.86	0.88	28.79	27.50	3.59	1.30	2	9821	404
2	0.85	0.85	28.96	27.28	88.03	2.41	2	9512	589
3	0.83	0.86	30.86	29.37	142.05	0.46	2	9979	603
4	0.84	0.82	29.19	30.15	-131.43	-0.98	4	9931	624
5	0.79	0.81	33.07	30.84	114.21	1.90	1	10188	1026
6	0.82	0.83	27.26	27.08	10.05	2.05	2	11185	681
7	0.91	0.94	19.41	19.12	-14.50	0.85	3	10914	784
8	0.83	0.85	27.83	30.38	6.26	-1.21	5	9678	116
9	0.83	0.81	28.35	28.56	-59.62	-0.39	2	9770	254
10	0.83	0.85	29.28	29.21	66.02	0.84	3	9908	309
11	0.81	0.84	30.04	29.76	73.66	-0.02	3	9758	212
12	0.81	0.79	29.73	32.06	-114.46	-1.19	5	10393	449
13	0.83	0.83	25.84	26.63	-16.12	0.13	3	11092	677
14	0.92	0.94	18.52	18.55	87.98	1.00	2	10664	723
15	0.82	0.84	29.21	30.05	-118.95	0.41	4	9514	114
16	0.83	0.83	30.95	30.15	33.93	1.15	3	9500	274
17	0.82	0.83	31.87	33.02	-67.18	-0.33	3	10001	225
18	0.84	0.84	30.33	28.40	-294.32	1.47	-1	9950	243
19	0.85	0.87	29.83	24.01	-29.10	-1.13	1	10364	336
20	0.84	0.89	23.76	19.87	50.86	0.22	1	10753	600
21	0.93	0.96	15.53	13.00	-42.23	-1.08	1	10262	543
22	0.93	0.96	17.69	13.88	37.32	0.37	0	9980	35
23	0.84	0.89	27.67	20.46	78.73	1.32	-2	10182	371
24	0.80	0.90	29.56	21.98	171.81	1.75	0	10099	176
25	0.91	0.94	18.11	14.35	7.75	0.21	0	10432	250
26	0.82	0.85	29.22	23.17	7.01	0.16	-1	10048	178
27	0.84	0.85	25.28	21.78	-135.00	-0.61	1	11026	139
28	0.86	0.90	29.69	27.63	19.99	-0.46	1	10587	360
29	0.81	0.85	31.49	25.35	28.37	-1.66	1	9694	42
30	0.82	0.84	32.57	25.85	-89.27	-0.50	1	9517	173

is calculated. The days with the highest number of filtered values and anomalies are the ones with better performances (see for example 5th, 6th, and 20th April in Table 5.1).

The reason of this enhancement of performances was further investigated, observing the time-series of measurements and simulation flows in the more affected locations. In Figure 5.8, the comparison between the observed and the simulated flow on the 1st April, without excluding anomalies (STD SIM) and removing anomalies detected by ADP1 (ADP1 SIM) shows that the performances in 3 sensors locations are significantly affected by the anomaly

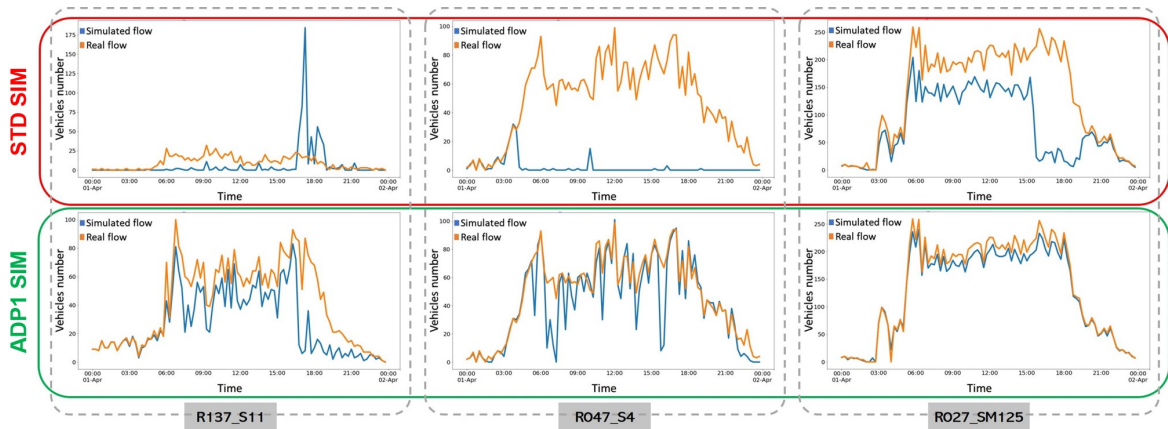
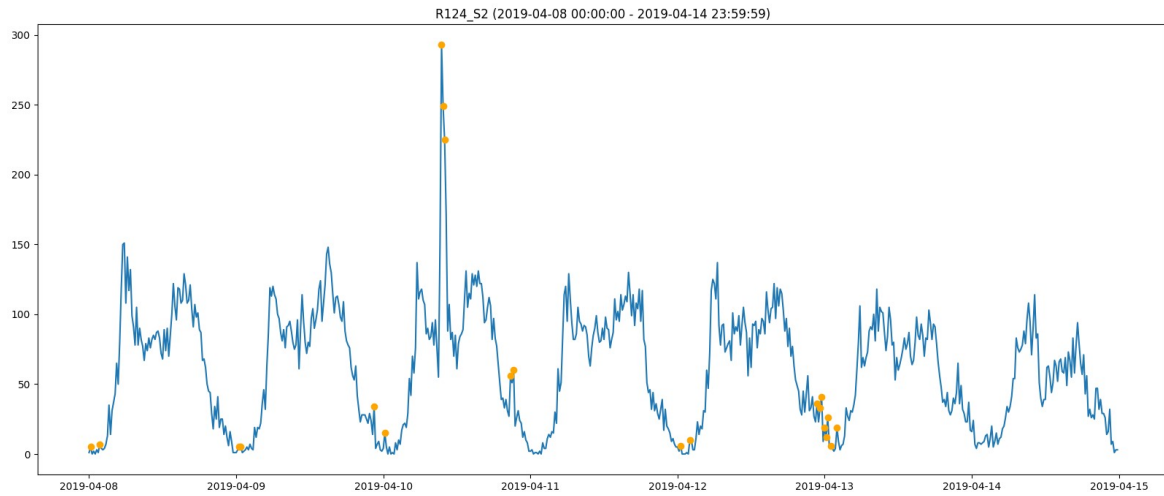


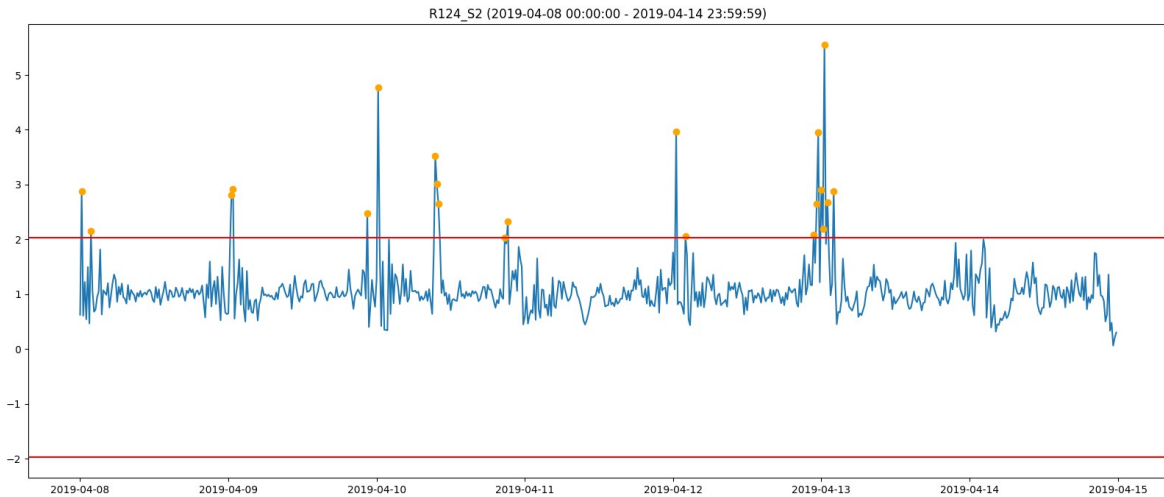
Fig. 5.8 Comparison between observed time-series (orange) and simulated flows (blue) in three locations for the 1st of April. Standard traffic simulation (STD SIM - on the top) is compared to the traffic simulation that takes advantage of the data cleaning process with ADP1 (ADP1 SIM - on the bottom).

detection process. The calibrators located in the position of sensors “R137_S11”, “R047_S4” and “R027_SM125” before the data cleaning process were not able to correctly follow real measurements: “R137_S11” had some time slot with very high flow, “R047_S4” had zero flow for the most part of the simulation, and “R027_SM125” was not able to follow the real flow in the second half of the STD SIM. For these reasons, they were all classified as not aligned calibrators. After the data cleaning process, the similarity between the observed and the simulated flow rises, and the calibrators were classified as aligned. Finally, the outputs (flow and speed in every road section) of the simulations were compared. Since some high flow values detected as anomalies are removed, the expected result was a decrease in the total number of vehicles. Yet, the number of vehicles increased globally and the total number of vehicles per day increased on average in the 57% of road sections. The daily mean speed has a difference with an absolute value higher than 10 km/h for only 11% of road sections. Calculating the sum of the values of speed increment and speed reduction due to anomaly detection, the final result is that the data cleaning process speeds up the vehicles in the simulation.

Improvement in traffic simulation with anomaly detection and classification (ADP2+CLASS) In this paragraph, the application of anomaly detection and classification has been tested in order to exclude from the input of the traffic simulation not all the anomalies but only the sensor faults.



(a) Anomalies on the time-series (i.e., the orange dots).



(b) Anomalies on the remainder of the time-series (i.e., the orange dots).

Fig. 5.9 Anomalies found by ADP2 on sensor R124_S2 observations from 8th April 2019 to 14th April 2019.

ADP2 was used to detect anomalies in April 2019. Then, the detected anomalies were classified and only the ones labeled as sensor faults were removed from the traffic model input. Thus, in this case, ADP2 was combined with the anomaly classification.

ADP2 detected 26792 anomalies, which are related to 333 sensors. Compared to the number of anomalies detected by ADP1 in the same period, ADP2 finds twice as many anomalies. Figure 5.9.a shows anomalies detected by ADP2 on the time-series observations of the sensor “R124_S2” from April 8th to April 14th, 2019, while Figure 5.9.b reports the

same anomalies on the remainder component after the application of the inverse logarithm function (the red lines represent the lower fence and the upper fence).

These figures can be compared with the ones of Figure 5.5. Obviously, the remainder values of Figure 5.9.b and Figure 5.5.c are different since they are calculated in different ways. Anomalies on the high values of April 10th are identified by both anomaly detection processes; in addition, ADP2 finds another very high value in April 10th. Not all the anomalies on low flow values are detected by ADP2; indeed, the low flow observations on the nights between 13th and 14th, and 14th and 15th are not highlighted as anomalies. Besides, at the end of April 10th, ADP2 identifies anomalies on two positive peaks.

After anomaly detection, the traffic of each day of April 2019 was simulated using the traffic model described in Section 2.3. The results of the simulation considering anomalous values (STD SIM) and the simulation excluding them (ADP2+CLASS SIM) for all days of April 2019 were compared. As displayed in Table 5.2, the average value of DTW reduction is very high: 76.71. This is a significant improvement, considering that the same value was negative (-2.82) using ADP1. Moreover, the average value of PWD reduction is 0.68 (it was 0.28 with ADP1); however, the average value of CTD reduction is 1 and is lower than the one in ADP1 (2).

Taking into account the days with the worst performances using ADP1 (red values in Table 5.3), the values of the metrics described previously are evaluated considering only non-anomalous measurements for both the standard simulation (STD SIM) and the simulation performed removing sensor faults detected with ADP2 (ADP2+CLASS SIM). Since the anomalies detected by the two versions of the anomaly detection algorithm are different and the performances are evaluated only on non-anomalous values, the performances of the STD SIMs are different from the ones in Table 5.1 even if the simulation is the same.

The results show that ADP2 combined with classification significantly improves the performances of 4th, 9th, and 17th of April; however, removing anomalies reduces the performance of April 12th, only the mean RMSE error is reduced.

Table 5.2 Comparison of average metrics on April 2019 for STD SIM, ADP1 SIM and ADP2+CLASS SIM.

	% aligned STD SIM	% aligned	mean RMSE STD SIM	mean RMSE	DTW reduction	PWD reduction	CTD reduction
ADP1	0.84	0.87	27.33	25.32	-2.82	0.28	2
ADP2+CLASS	0.86	0.86	29.83	24.99	76.71	0.68	1

Table 5.3 Comparison of traffic model evaluation metrics between traffic simulation including anomalies (STD SIM) and traffic simulation excluding sensors faults (ADP2+CLASS SIM).

day	% aligned		mean RMSE		DTW	PWD	CTD
	STD SIM	ADP2+CLASS SIM	STD SIM	ADP2+CLASS SIM	reduction	reduction	reduction
4	0.83	0.82	31.19	29.66	-2.73	0.62	1
9	0.82	0.85	31.19	29.18	138	0.22	-1
12	0.80	0.77	32.14	31.93	-104.73	-2.07	4
17	0.81	0.82	35.70	32.00	111.12	1.19	0

5.2.3 FFIDCAD and ARIMA model

The anomaly detection method described in this section is based on the idea of detecting anomalies by modeling the average trend of data and then identifying deviations from the trend.

The methodology consists of:

- anomaly detection: abnormal patterns that deviate from the vast common are discovered for each sensor;
- anomaly classification: anomalies are classified as sensor faults or unusual traffic conditions based on the correlation among traffic sensors and identifying sensors with similar trends.

Anomalies are detected by combining the Forgetting Factor Iterative Data Capture Anomaly Detection (FFIDCAD) and the ARIMA model. A thorough search of the relevant literature revealed that ARIMA has never been tested in combination with FFIDCAD in the context of traffic anomalies detection, nor in combination with the correlation among traffic sensors.

FFIDCAD

FFIDCAD [101, 102] stands for Forgetting Factor Iterative Data Capture Anomaly Detection. It is a streaming anomaly detection algorithm whose technique is based on two other models: the data capture anomaly detection (DCAD) model and the iterative DCAD (IDCAD) model.

FFIDCAD assumes the data fit the multivariate normal distribution. It exploits the correlation among different features of data and identifies the anomalies by creating hyperellipses around the sampled data distribution. The model employs a continuous learning strategy

to estimate the hyperellipsoidal boundary that covers the normal data incrementally; each iteration of the algorithm adjusts the hyperellipsoidal model based on the measurements up to the current time. Mean, standard deviation and covariance of the correlated features are calculated to build the hyperellipses. In the discussed use case, the features are the flow and the speed calculated by the sensors.

Besides, this unsupervised model is suitable to process data incrementally as they arrive in time and detect anomalies “on-the-fly” in near real-time.

ARIMA

ARIMA (Autoregressive Integrated Moving Average) model is a statistical method for time-series analysis and forecast. ARIMA is a general-purpose technique for modeling temporal data with seasonality [103]. It allows capturing a set of standard temporal structures in time-series data to forecast new data. In this scope, a model of the sample time-series is built, then the anomaly detection can be performed by comparing the forecast with the real data. As its name suggests, it combines an autoregression (AR) model and a moving average (MA) model, and it is integrated (I), which means it exploits the differencing technique to make stationary the non-stationary time-series. Indeed, as all the regression models, also the ARIMA model can be applied to non-stationary time-series only after made it stationary since trends negatively affect the model.

The autoregression model identifies the dependent relationship between an observation and a variable number of lagged observations. This dependency is explained by the following formula:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \dots + \beta_p Y_{t-p} + \varepsilon_t$$

where Y_{t-1} is the first lag of the series, β_1 is its coefficient estimated by the model and α is the intercept term.

On the other hand, the moving average model detects the dependency between an observation and the lagged forecast errors ε_t caused by the autoregressive model, following the formula:

$$Y_t = \alpha + \varepsilon_t + \omega_1 \varepsilon_{t-1} + \omega_2 \varepsilon_{t-2} + \dots + \omega_q \varepsilon_{t-p}$$

The model exploits three configuration parameters, each of them related to one of the abovementioned components. The parameters are (1) p , the lag order, i.e., the number of lag observations included in the autoregressive model, (2) d , the degree of differencing, i.e., the number of times the raw observations are differenced to achieve stationary and to remove any seasonality or trends, and (3) q , the order of moving average, i.e., the number of lagged forecast errors in the prediction formula. The challenge is to find the right combination of

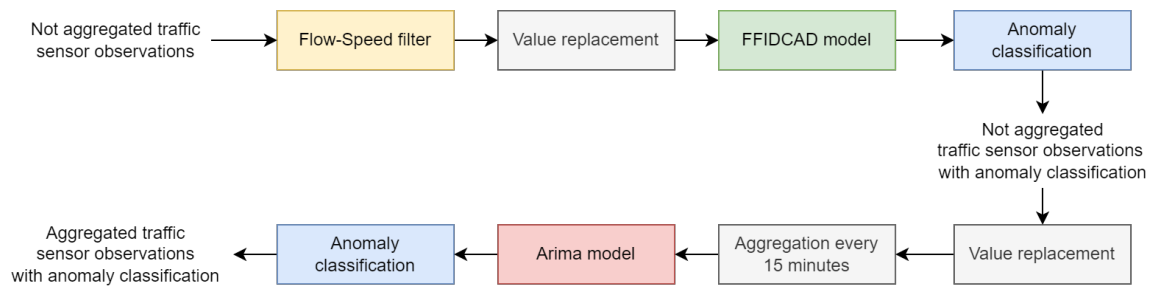


Fig. 5.10 Overview of the data cleaning process configuration.

these parameters which fit better the sampled data. This task requires an iterative process building several models with different parameters and checking the result. The model produces a prediction of the future measurements; the real measurements are compared with the prediction to find anomalies.

Data cleaning process configuration

The final data cleaning process configuration adopted is described in Figure 5.10. Firstly, not aggregated data coming from sensors are filtered through the flow-speed correlation filter removing some measurements. The remaining measurements are given as input to the FFIDCAD model; then, the model results are classified and divided into sensor faults and unusual traffic conditions considering the correlation between sensors. The obtained collection of not aggregated observations labeled with anomaly classification is then processed to remove sensor faults which are replaced with an average of proximal observations. Then, the obtained modified observations are aggregated every 15 minutes and given as input to the ARIMA model. Anomalies detected by ARIMA are then classified, and the final result is provided. The process is repeated for each traffic sensor.

The flow-speed correlation filter, FFIDCAD and ARIMA model are employed as complementary techniques to find all the possible anomalies. It was not expected to find a significant intersection between the anomalies found by the different algorithms. However, to verify this assumption, the anomalies found by each method are compared aggregating data every 15 minutes. The anomalies discovered by the flow-speed correlation filter are not detected by the FFIDCAD model. However, this model works on the correlation between flow and speed, too. This happens because the model does not know that the speed value represents the speed of a certain number of vehicles. Therefore, it is not able to detect that a vehicle cannot pass with a certain speed on the road. In conclusion, the flow-speed correlation filter cannot be replaced by FFIDCAD; it is a complementary technique. Comparing FFIDCAD with ARIMA, it was observed that only four anomalies classified as sensor faults and 449 classified as unusual

traffic conditions were detected by both methods considering 8th November 2018 data. This was the evidence of the complementing behavior of the methods.

The conclusion was that all the methods have to be applied to the measurements since there is not a significant overlap. In the next subsections, other considerations are made on the configuration of each method.

Aggregation of the input data for anomaly detection

The anomaly detection is performed on the measurements of each sensor separately. The flow-speed correlation filter has to be applied to the measurements as they are provided by the sensors, without aggregation since the application on the average speed could modify the result significantly. Indeed considering two measurements, both related to one minute time interval, the first with $flow = 1$ and $speed = 150$ and the second with $flow = 50$ and $speed = 60$. The first measurement is considered non-anomalous by the filter, while the second is an anomaly. The weighted average of the two speeds is around 62. Aggregating the two measurements and considering the weighted average, the aggregated measure is detected as anomalous by the filter. In conclusion, it is better to apply the filter to the raw observations to avoid the exclusion of some anomalies that are instead valid measurements. Also, FFIDCAD and ARIMA have been applied to raw data, i.e., data not aggregated, and to data aggregated every 15 minutes. When data is aggregated, the values of the flow are summed up for the sensors of the municipality since it represents the actual number of vehicles in the time interval of one minute; while a weighted average is evaluated to obtain a representative value of average speed in the aggregated interval, following the formula:

$$average_speed = \frac{\sum_{i=1}^n (flow_i * speed_i)}{\sum_{i=1}^n flow_i}$$

where $flow_i$ and $speed_i$ are the values of flow and speed of the $i - th$ measurement in the aggregation time interval.

The result obtained by using different aggregation intervals were compared in order to define the best choice for each anomaly detection technique. Firstly, the FFIDCAD method was tested on the data of the 8th November 2018 of 270 sensors. The total number of anomalies found by FFIDCAD in 15-minutes aggregated data is 2,286. Instead, the number of detected anomalies on the not aggregated input data is 11,358. By applying classification, the total number of sensor faults detected by FFIDCAD on aggregated data is 19, a very low number considering that the period taken into account covers 24 hours, i.e., 388,800 observations. FFIDCAD applied to not aggregated data detects 77 sensor faults on data referring to the same day. Considering another day (15th April 2019), the anomalies

detected aggregating data are 3618, and 61 of them are classified as sensor faults; without the aggregation of data, the total number of anomalies grew to 12,129, and the sensor faults are 414. FFIDCAD studies the correlation between speed and flow. When aggregating data every 15 minutes, some anomalies cannot be detected since the relationship between flow and speed changes when evaluating the sum of the flow and the weighted mean of the speed in 15 minutes. For this reason, the application of the FFIDCAD algorithm should be applied to raw input data.

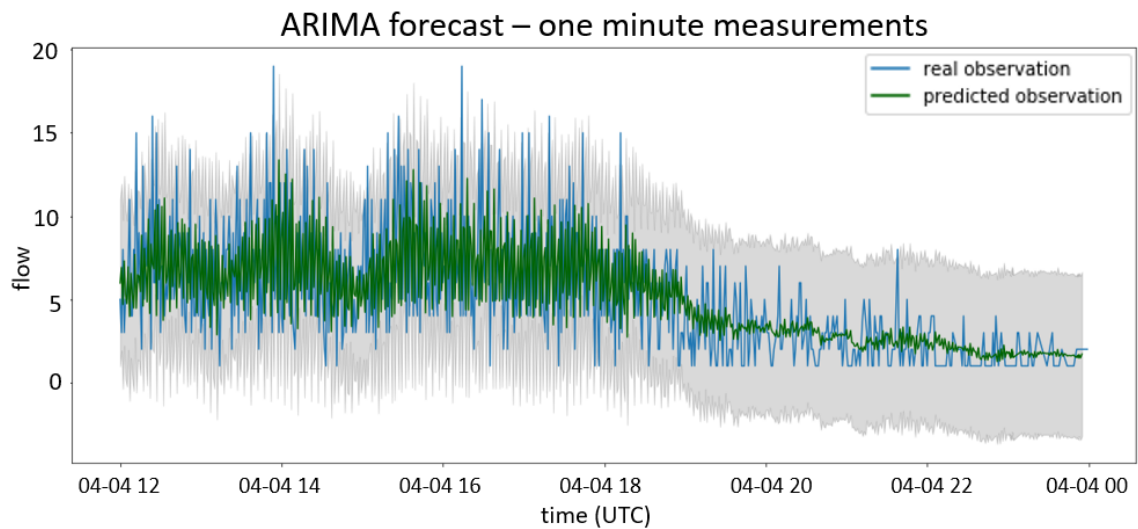
Then, the application of ARIMA was evaluated on both the raw measurements and the measurements grouped by 15 minutes. The results were compared discovering that the configuration, which considers the raw measurements, detects many anomalies. Plotting these anomalies and analyzing their values of flow and speed, they did not seem to be anomalous values.

An example is provided by Figure 5.11a which represents the measurements of one traffic sensor in a day of April 2019 (the blue line) and the prediction of ARIMA (the green line). Figure 5.11b highlights with orange points the measurements considered as anomalies by the model. As can be seen, many measures are anomalies, however they do not seem to be anomalies. Probably, this erroneous behavior happens because the sensors are installed near traffic lights; therefore, it is very common the flow value grows fast and then again takes on lower values. Studying the “unsteady” trend of the time-series, the ARIMA model is not able to predict the one-minute measurements in the right way. Therefore, the choice was to apply the ARIMA model to measurements grouped by 15 minutes with the sum of the flow and the weighted average of the speed.

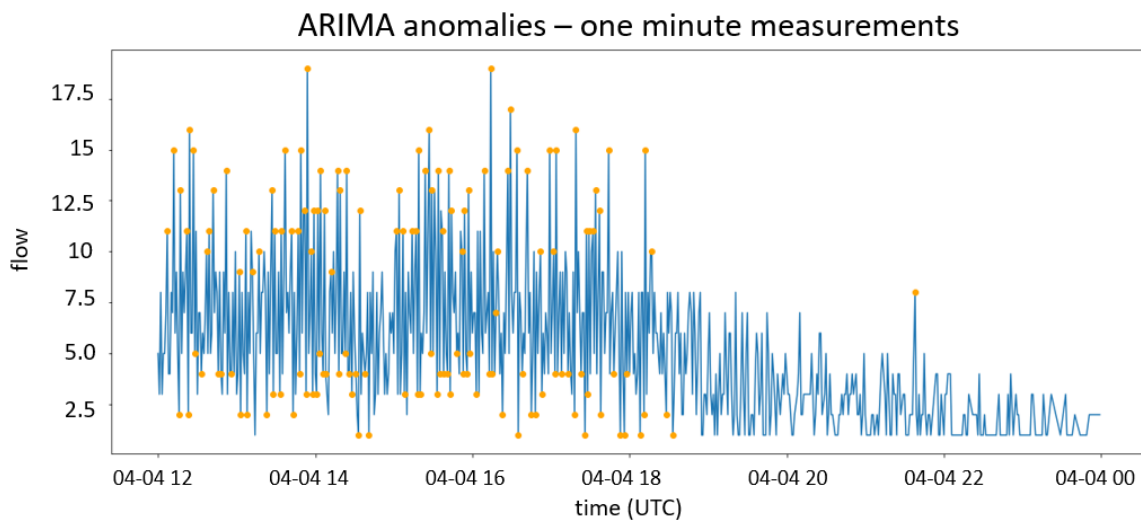
Definition of the time interval of concern

The anomaly detection models can be applied to different time intervals. For the FFIDCAD model, this means that the model can take as input the measurements related to periods of different lengths, i.e., one day, one week, one month, and so on. Mean, standard deviation, and covariance are calculated on the entire dataset provided as input. In this way, the model finds anomalies for the whole period. For the ARIMA model, the different duration of the time interval is related to the training set.

FFIDCAD algorithm was applied to the entire month of November 2018 and then on a single day: 8th November 2018. The anomalies detected by the algorithm trained on the entire month (referred as monthly version) are different from the ones detected from the same algorithm trained only on the 8th November (daily version). The total number of anomalies detected in the whole month was 7,226 and only 281 of them on the 8th November (only 26 classified as sensor faults). Instead, with a daily interval of application, the detected



(a) Time-series representing the measurements of the sensor (blue line) and the prediction of ARIMA (green line).



(b) Anomalies (orange points) detected by ARIMA.

Fig. 5.11 ARIMA model applied to one minute measurements of one sensor in a day of April 2019.

anomalies for the only 8th November were 2,286 (19 sensor faults). The reduction in the number of sensor faults is due to the fact that more anomalies are detected in the daily version w.r.t. the monthly version and some of these anomalies are simultaneous with the one previously erroneously classified as sensor faults and now classified as unusual traffic conditions. The two different intervals of application detect different anomalies, only 185 anomalies were detected by the monthly and daily versions. 21 of the 26 anomalies classified as sensor fault by the monthly version are also detected by the daily version. However, only 3 sensor faults detected by the daily version are detected by the monthly version.

The reason why some anomalies are not detected by the monthly version is that training the model on the whole month means also consider holidays and weekends that have a singular trend and normally a lower flow and can influence the detection of anomalies in regular working days. For this reason, the daily version approach of FFIDCAD was used.

In the ARIMA model, instead, the train is made on the entire month to predict one day. In this way, different trends can be included in the model, i.e., the daily trend, but also the weekly trend, the different behavior of the sensors in holidays, and so on. Thus, the ARIMA model is used to predict sensor observations related to one day, but the model is trained on the whole month.

Combination of FFIDCAD and ARIMA

After this experiment, different techniques to integrate the results obtained by these algorithms have been taken into account. Firstly, not aggregated data are given as input to the FFIDCAD algorithm since it detects anomalies, considering the correlation between speed and flow. Then, anomalies classified as sensor fault are removed and replaced by the average of proximal measurements of flow and speed. Data aggregated every 15 minutes is given as input to the ARIMA model. ARIMA detects anomalies, considering the trend of previous measurements. Thus, the detected anomalies are classified, and the ones labeled as sensor faults are removed from the input of the traffic model. The unusual traffic conditions detected by the ARIMA method at the end of this process are considered traffic events. The relevance of the traffic event is also evaluated considering the number of unusual traffic conditions anomalies detected by FFIDCAD in the one-minute data aggregated for the same sensor in the 15 minutes time interval that the ARIMA anomaly refers to. Given the large size of the data, scalability and time-efficiency are needed.

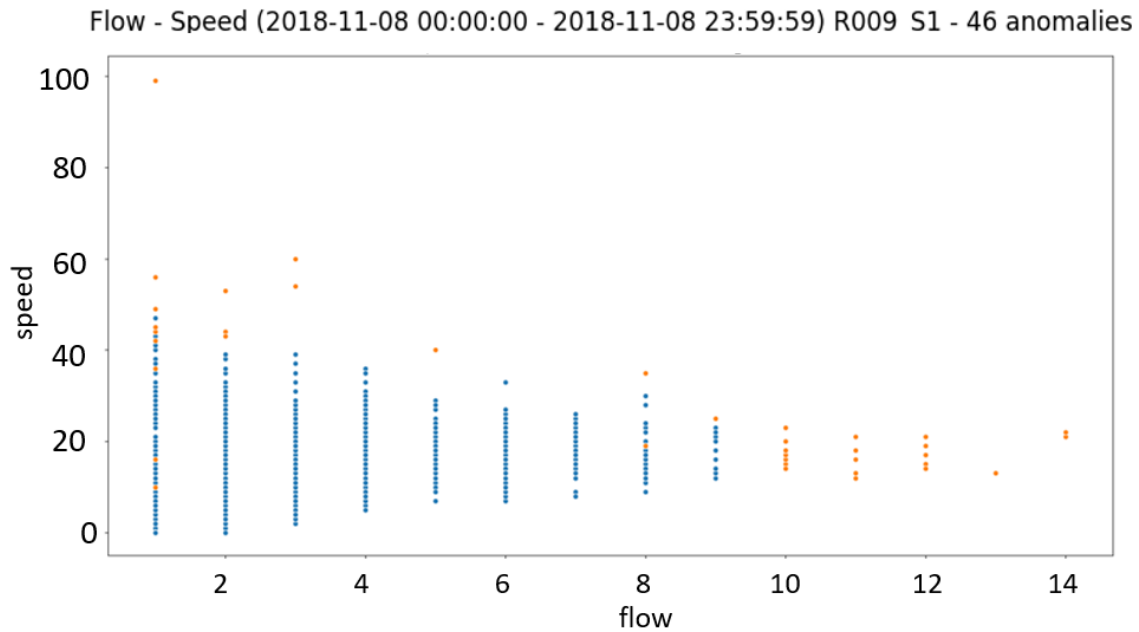


Fig. 5.12 Correlation between flow and speed for sensor R009_S1 on the 8th November 2018 (orange dots are anomalies).

Experiments and results

The proposed methodology has been applied to the measurements of two days: the 8th November 2018 and the 15th April 2019. These two days were selected since there were road accidents in streets controlled by the sensors in Modena, so it was possible to check if the proposed methodology can distinguish between sensor faults and unusual traffic conditions.

Test 1 - 8th November 2018 The total number of raw measurements on that day is 383,421, and the number of sensors with at least one measurement is 270. The filter finds 14,076 anomalies (3.7% of the total number of measurements), related to 233 sensors.

The filtered measurements were removed, and the FFIDCAD model was applied to the remaining measurements. The model was implemented in Python, providing as input the time-series of the measurements to be analyzed, the correlated features, and the value of the forgetting factor. The forgetting factor was set to 0.999, as suggested by the authors in [101]. The information returned is the list of anomalies. The model is applied to the measurements of each sensor separately. Therefore, 270 FFIDCAD models were generated, providing as input the measurements of only one sensor on the 8th November 2018. The FFIDCAD model finds 11,147 anomalies (3% of the total number of measurements) related to 255 sensors. The time required by FFIDCAD to find anomalies is less than 1 second for each sensor.

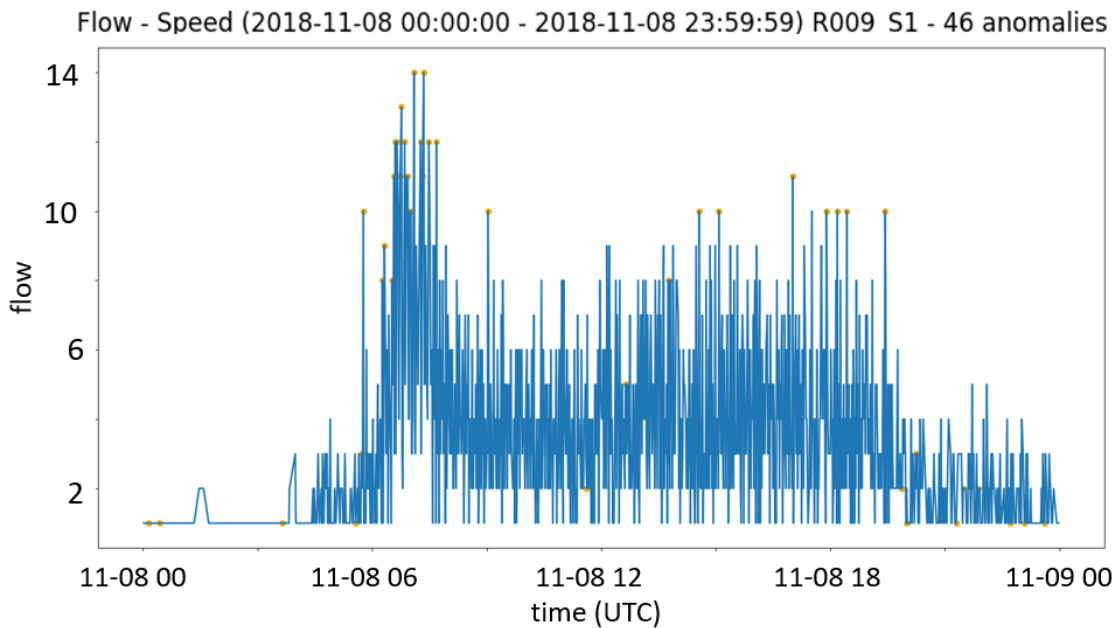


Fig. 5.13 Time-series of sensor R009_S1 on the 8th November 2018 (orange dots are anomalies).

Figure 5.12 shows the correlation between flow and speed of sensor R009_SM19 in 8th November 2018 found by the FFIDCAD model, while Figure 5.13 displays the time-series representing the same data of Figure 5.12. The points in orange are the anomalies found by the FFIDCAD model; the blue points are the real measurements identified as non-anomalous.

Figure 5.14 shows the anomalies found by the flow-speed correlation filter and the FFIDCAD model in the flow-speed scatter plotted. As can be seen, most anomalies found by the FFIDCAD model are related to low values of flow, while the anomalies of the flow-speed correlation filter are related to very high values of speed.

Anomalies detected by the FFIDCAD model were classified: the 0.02% (204) of them as sensor faults and the others (10,943) as unusual traffic conditions.

The ARIMA model was deployed by using the Python implementation provided by the statsmodels library. The `auto_arima()` function allows discovering the optimal configuration parameters for each model automatically. Therefore, it is not necessary to find these values with tests manually. The model is trained on a set of measurements; then, it is applied to predict the measurements of a time interval of variable duration. The ARIMA model was trained on the measurements of the previous 30 days aggregated every 15 minutes (from 7th October 2018 to 7th November 2018) to forecast the measurements of the next one hour. The model was retrained with the real measurements of the predicted hour to forecast the next hour and so on until all the specific day was completed. With this approach, the

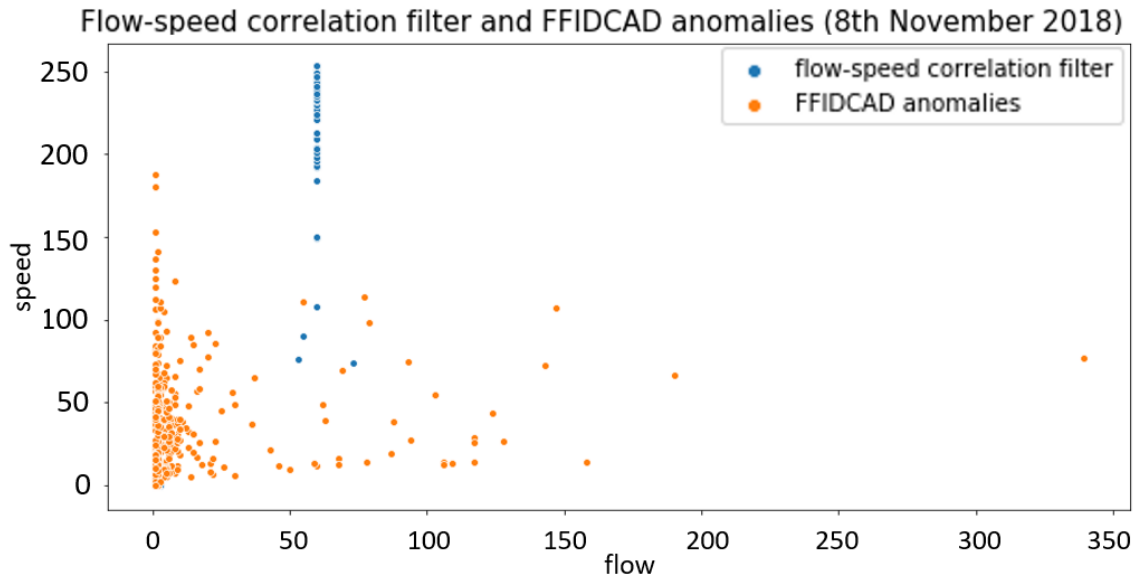


Fig. 5.14 Anomalies found by the flow-speed correlation filter and the FFIDCAD model in 8th November 2018.

model requires less than 10 minutes to predict the measurements of the whole day. Figure 5.15 displays the real measurements of the sensor R009_SM19 represented by the blue line, while the green line is the prediction supplied by the ARIMA model. The gray area is delimited between the lower and upper bounds found by the model. The measurements with a flow value that is out of this area are detected as anomalies. The measurements of the 8th November 2018 aggregated every 15 minutes are 25,943 in total. Measurements filtered by the flow/speed correlation filter and the ones identified as sensor faults by FFIDCAD were replaced by the average of proximal measurements. At this point, ARIMA finds 1,431 anomalies. The detected anomalies have been classified as discovering 96 sensor faults and 1,335 unusual traffic conditions. Anomalies represent 6% of the total measurements of that day. The anomalies classified as sensor faults are related to 46 sensors, the ones classified as unusual traffic conditions consider 189 sensors.

Analyzing the values of flow and speed of the anomalies, it can be noticed that the majority of sensor faults are detected when the speed has low values. The values with very high speed and flow could be anomalies, however, they are not detected by the proposed approach.

Test 2 - 15th April 2019 The raw observations available on the 15th of April 2019 are 442802, related to 335 sensors. The flow-speed correlation filter finds 14461 anomalies (3%).

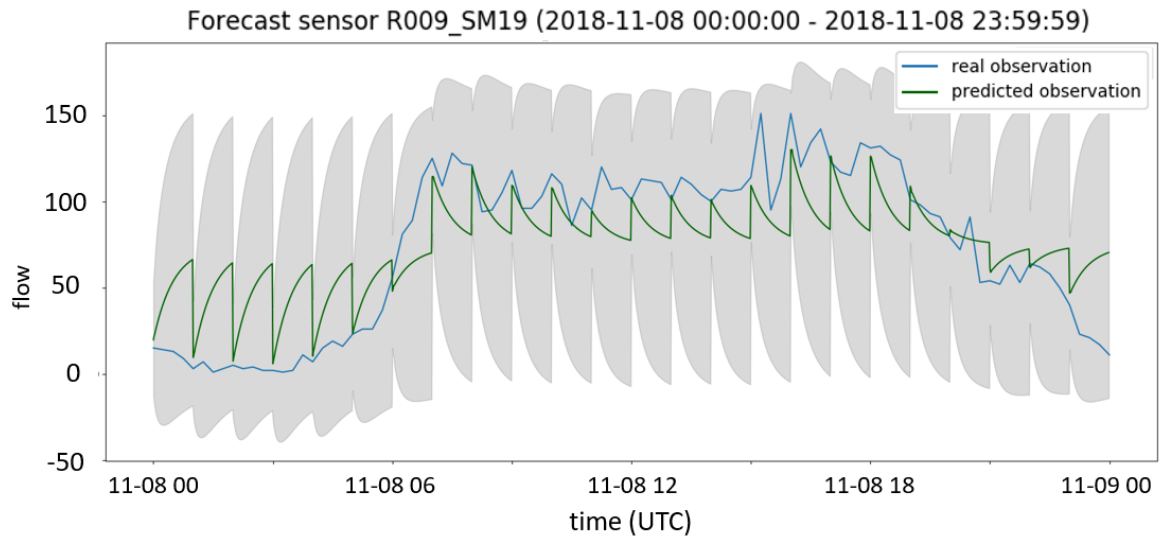


Fig. 5.15 Sensor R009_SM19 time-series on the 8th November 2018 (blue line) and ARIMA predictions (green line).

After removing the anomalies of the filter, the FFIDCAD model finds a total of 16975 anomalies (less than 4% of the total measurements), 13% of them classified as sensor faults and the rest as unusual traffic conditions. Anomalies are detected in 309 sensors, even if only 65 sensors have at least one sensor fault.

The observations of the 15th April 2019 aggregated every 15 minutes are 31613. By replacing measurements filtered or identified as sensor faults from the FFIDCAD model, the total number of anomalies detected by ARIMA is 2485 (around 8% of the total aggregated observations).

The detected anomalies have been classified discovering 263 sensor faults and 2,222 unusual traffic conditions. The anomalies classified as sensor faults are related to 72 sensors, the ones classified as unusual traffic conditions to 227 sensors. As in the previous experiment, the majority of sensor faults are detected when the speed has low values.

5.2.4 ST-BOF and ST-BDBCAN

ST-BDBCAN is based on the distinction between spatio-temporal and behavioral attributes. Spatio-temporal attributes indicate the position of the sensors or provide temporal information about the observation. Behavioral attributes instead are all the other attributes that refer to the given spatio-temporal point, i.e. measured values, environment variables. ST-BDBCAN groups objects with similar behavioral attributes as clusters and detects objects with abnormal behavioral attributes as outliers, exploiting the outlier factors evaluated by ST-BOF. Since no

code implementation of these two algorithms has been found, ST-BOF and ST-BDBCAN have been studied and implemented in Python (the code is available online² with exemplar data to execute the algorithm). The proposed implementation is realized for multivariate spatial time-series but can be easily adapted to spatio-temporal time-series (where both spatial and temporal dimensions vary for each observation). The proposed implementation is suitable for the application of the algorithm to different contexts. The algorithm has been applied to data collected by 49 of the traffic sensors installed in the city of Modena. Several experiments have been conducted with different configuration parameters of ST-BOF and ST-BDBCAN in order to investigate the difference in the types of anomalies detected.

In [85], the *Spatio-Temporal Behavioral Outlier Factor* (ST-BOF) and the *Spatio-Temporal Behavioral Density Based Clustering of Applications with Noise* (ST-BDBCAN) are combined to execute in cascade. This combination allows defining a locality-based spatio-temporal context for each instance to analyze. The instances are the input data, e.g. the observations of some IoT devices. Two types of attributes are identified for the spatio-temporal data: the contextual attributes are the spatio-temporal attributes that define the “location” of the instances and the time reference; while the behavioral attributes describe a feature of the instance.

Firstly, ST-BOF is applied to evaluate a score that represents the potential outlieriness of each instance based on its behavioral attributes w.r.t. the neighbors. Then, ST-BDBCAN, which is the clustering algorithm for spatio-temporal data, exploits the outlier factor evaluated by ST-BOF in the generation of clusters. ST-BOF takes two positive integers as parameters: *MinPts* which is the number of spatio-temporal neighbors to consider and *k* which defines the order of the neighbors to determine the behavioral reachable distance of the instances. The behavioral reachable distance of two instances is calculated by finding the maximum value between the distance of the behavioral attributes of the two instances to compare and the distance of the behavioral attributes of the second instance from its *k*th nearest spatio-temporal neighbor (*behavioralk – distance*). When evaluating that distance, different weights can be given to each available behavioral attribute. Given *MinPts* and *k*, the formula to calculate ST-BOF is the following:

$$ST-BOF(p) = \frac{1}{|ST-N(p)|} \sum_{o \in ST-N_{MinPts}(p)} \frac{ST-BRD(o)}{ST-BRD(p)}$$

where $ST-N(p)$ is the ensemble of the spatio-temporal neighborhood of object p with $MinPts$ neighbors. Then $ST-BRD$ indicates the Spatio-Temporal Behavioral Reachable Density, that is the inverse of the average behavioral reachable distance of the object p w.r.t. its $MinPts$

²https://github.com/quattrinifabio/ST-BOF_ST-BDBCAN

neighbors. This value is high if p has spatio-temporal neighbors whose behavioral attributes are similar to p . In the end, ST-BOF is the average of the ratios of the $ST-BRD$ of p 's neighbors w.r.t. the $ST-BRD$ of p . If an object p has an ST-BOF greater than 1, then its spatio-temporal attributes are very different from the spatio-temporal neighbors' attributes. On the other hand, if p has an ST-BOF less than 1, then its behavioral attributes are very similar to its neighbors' behavioral attributes. Thus, ST-BOF allows quantifying the potential outlierness of each instance by showing how much its behavioral attributes diverge from the ones of its spatio-temporal neighbors.

ST-BDBCAN detects outliers by grouping instances with similar behavioral attributes in the same cluster and identifies instances with abnormal behavioral attributes as outliers based on their spatio-temporal locality. Thus, given an instance, this algorithm exploits the spatio-temporal attributes to identify its neighboring observations. Then, the behavioral attributes of the instance and the neighbors' behavioral attributes are compared to define clusters.

Firstly, the algorithm marks every instance as unclassified and calculates ST-BOF for each of them. Then, the upper bound of ST-BOF (ST-BOFUB) is calculated considering the percentage of anomalies expected to find (AP) that is a configuration parameter. Instances with ST-BOF values above ST-BOFUB are labeled as spatio-temporal outliers. After setting ST-BOFUB, every unclassified instance with an ST-BOF lower than ST-BOFUB is selected as a candidate core instance. Then, to become a core instance, at least $MinPtsInCluster$ neighborhoods of p should have an ST-BOF lower or equal to ST-BOFUB. Moreover, at least $MinPtsInCluster$ neighborhoods (o) of p should verify the condition:

$$\frac{ST-BRD(o)}{(1+pct)} < ST-BRD(p) < ST-BRD(o) * (1+pct)$$

where pct is the percentage of variation accepted in ST-BRD. If p is a core instance, a cluster can be generated starting from that instance by finding instances with similar behavioral attributes whose ST-BOF is lower than ST-BOFUB. In this way, a spatio-temporal behavioral-based cluster containing the instance is generated. This process is repeated till none of the remaining instances can be a core instance or can be inserted in a cluster. At the end of this process, all the unclassified instances are marked as noise.

Implementation

The ST-BOF and ST-BDBCAN combined algorithm in the context of spatial time-series generated by a sensor network has been implemented in Python. This implementation can be

exploited also in different contexts where spatial time-series are available in a predefined and fixed set of locations.

The mutual distances between the locations are pre-calculated to reduce the execution time of the algorithm and its complexity. Two libraries have been created separately for ST-BOF and ST-BDBCAN to be eventually used in other applications. Moreover, a Python script that exploits and combines the two libraries were implemented. The script takes as input two “csv” files: one with the sensors’ measurements and the other with the distances between the sensors’ locations in meters. The user should also indicate the names of the behavioral attributes. Even for spatio-temporal time-series where the positions dynamically change over time (e.g. mobile sensors, trajectories of values), the implementation can work associating a unique id to each observation and pre-calculating the spatial distance for each observation. The generated output is a “csv” file with the classification of the measurements; the outliers are labeled with “clusterID” equal to -1. Furthermore, the script allows the user to define different configurations of the algorithms in order to obtain different results. The ST-BOF and ST-BDBCAN parameters can be customized and additional parameters are added to allow a better configuration based on the use case: (i) the user can give different weights to spatial, temporal, and behavioral attributes, (ii) the user can optionally specify a minimum percentage of outliers to detect, (iii) specifying the sensor identifier, the algorithm will execute considering exclusively the temporal dimension. Besides, the implementation allows detecting two different types of anomalies based on the configuration parameters: contextual point anomalies and contextual collective anomalies. In the context of a sensors network, contextual point anomalies are sensor faults and contextual collective anomalies are real, but unusual conditions detected by sensors. Changing the parameters’ configuration, the algorithm can be employed to find only sensor faults or both sensor faults and unusual conditions. Four different configurations have been tested.

Experiments and results

The traffic sensor data are aggregated every 15 minutes excluding filtered observations. Four experiments with different configuration parameters are performed on 3,423,179 observations collected in April 2019 by 49 sensors located in the city of Modena.

The first experiment (Exp.1) highlights the influence of the spatial dimension in the detection of anomalies comparing its results with the ones of Exp.2. In Exp.3, the parameter k controls the statistical fluctuation in the computation of ST-BOF; thus, while increasing k , the behavioral distance is evaluated for a more distant spatio-temporal neighbor. Moreover, increasing the value of $ST-BDBCAN_{MinPts}$ (the number of observations that potentially belong to a cluster), the number of outliers is reduced. Generally, high values of k and

$minPts$ are suitable for the detection of contextual point anomalies. Finally, in Exp.4 only the parameter AP is modified keeping the same configuration of Exp.2 for the others. The parameter AP is the percentage of anomalies to detect and influences the evaluation of ST-BOFUB. In all the experiments, some parameters had the same values: the value of $ST-BOF_{MinPts}$ was 20, pct was set to 0.2, and the minimum number of points in a cluster was 5. The other parameters vary as displayed in Table 5.4. ST-BOF is defined with generic distance functions, the Manhattan distance was used as behavioral distance function to give the same importance to flow and speed. The Manhattan distance is evaluated between two points measured along axes at right angles as the sum of the absolute values of the difference between its coordinates. The selected units of measure are meters and minutes. This configuration is more suitable for the detection of contextual collective anomalies. The implemented code allows attributing a different weight to the spatial and temporal dimensions. However, since the assumption was that none of the dimensions should be preferred in evaluating the distance in the use case, the weights were equally distributed in the last three experiments.

Table 5.4 Parameters' configuration for the application of ST_BOF and ST_BDBCAN.

	ST-BOF k	ST-BDBCAN _{MinPts}	ST-BDBCAN AP
Exp.1	4	20	1
Exp.2	4	20	1
Exp.3	9	100	1
Exp.4	4	20	3

Table 5.5 Results of the application of ST_BOF and ST_BDBCAN.

	anomalies	sensors with % anomalies			avg anomalies per sensor	simul. anomalies	exec. time (minutes)
		< 1%	1% - 5%	> 5%			
Exp.2	2688	17	25	4	55	632	76
Exp.3	1620	25	11	2	33	307	180
Exp.4	5051	15	22	9	103	1418	173

Experiment 1 The first experiment is performed without taking into account the presence of other sensors in the sensor network. The sensor $R002_S2$ data are the only ones given as input to the algorithm. ST-BOF evaluates the outlier factor considering only temporal neighbors and ST-BDBCAN determines clusters based on the temporal distance and the similarity of behavioral attributes. The algorithm splits sensor data into 61 clusters with a percentage of noise points of 25% and detects 731 anomalies.

Experiment 2 The second experiment takes as input the multivariate time-series of measurements produced by all sensors in the selected area. The ST-BDBCAN algorithm identifies 207 clusters. Table 5.5 shows the total number of detected anomalies, the number of sensors with a very low (lower than 1%), normal (from 1% to 5%) and high (higher than 5%) percentage of anomalies in one month, the mean number of anomalies for each sensor, the number of simultaneous anomalies, and the execution time in minutes required to evaluate the anomalies for the whole April month. Simultaneous anomalies are observations labeled as anomalies that refer to the same timestamp and belong to different sensors. The sensors without anomalies are 3. Exp.2 detects only 4 anomalies for sensor *R002_S2* in the whole month. Besides, sensor *R002_S5*, which is located in the same crossroad of sensor *R002_S2*, has a 13% of outliers (375 anomalies). In general, the value of flow detected by sensor *R002_S5* is more than double of the flow detected by the other sensors in the crossroad. This leads to the classification of its values as anomalous even if they are aligned with the normal trend of the sensor itself. In particular, the majority of anomalies are detected during the weekend and the holidays.

In Exp.1, sensor *R002_S2* had a high number of anomalies because its values were not compared with the neighboring sensors but only with its own trend. When integrating the space dimension and comparing its measurements with the simultaneous measurements of the sensors in the same crossroad, most observations that are classified as anomalous in Exp.1 are contextualized with the neighboring sensors observations and no more classified as anomalies. However, sensor *R002_S5* has a trend that significantly differs from the other sensors in the crossroad and its number of anomalies is higher.

Experiment 3 The third experiment is a modified version of Exp.2 that was performed to reduce the number of detected anomalies. In order to do that the values of k and $ST-BDBCAN_{MinPts}$ are increased. As shown in Table 5.5, the number of anomalies is reduced to 1620, the number of sensors without anomalies is triplicated and the number of sensors with more than 5% of observations labeled as outliers is halved compared with Exp.2. The number of clusters detected by Exp.3 is 25; increasing 5 times the $ST-BDBCAN_{MinPts}$ produced a reduction of around 8 times in the number of clusters. For sensor *R002_S2* zero anomalies are detected and for sensor *R002_S5* only one anomaly is detected. It is a real anomaly that corresponds to a very high and not realistic value of average speed.

Therefore, it seems that this solution detects sensor faults or contextual point anomalies instead of unusual traffic conditions or contextual collective anomalies. Analyzing the anomalies detected for sensor *R131_S1* (one of the two sensors with a very high percentage of outliers), it was noticed that the observations of weekends and Easter holidays that are

unusual traffic conditions are labeled as anomalies. Exp.3 tries to identify sensor faults or contextual point anomalies rather than contextual collective anomalies.

Experiment 4 The fourth experiment is a modified version of Exp.2 with a higher value of the p parameter. The percentage of anomalies that should be detected (AP) is set to 3%. As a consequence, the value of ST-BOFUB decreases from 2 in Exp.2 to 1.68, a lower upper bound generates more anomalies. Lowering the ST-BOFUB observations with a lower ST-BOF will be labeled as anomalies; thus, the number of contextual collective anomalies will increase and more unusual traffic conditions will be identified (e.g. incidents or traffic jams). Table 5.5 displays that this configuration identifies 5051 anomalies and the number of sensors with a very high percentage of anomalies is doubled compared with Exp.2; however, the number of sensors with less than 1% of anomalies is only slightly reduced. The number of clusters (218) slightly increased compared with Exp.2. For sensor $R002_S2$, only 5 anomalies are detected, however for sensor $R002_S5$ more than 1000 anomalies are identified. For sensor $R002_S5$, anomalies are detected in weekends, holidays and during the night period when traffic flow drops significantly. For sensor $R031_S2$, the configured parameters allow detecting all the point anomalies in the time-series but do not highlight unusual traffic conditions.

Discussion

Traffic sensors are particularly challenging because the traffic flow and the average speed of vehicles are not continuous phenomena in the space-time domain. The time-series of each sensor can have a very different amplitude and trend compared with the others in the same area. Moreover, in the city of Modena, traffic sensors are located near traffic lights; thus, even if data are aggregated every 15 minutes to reduce the effect of the traffic light logic, the flow and the average speed are strongly influenced by the viability of the crossroad. Thus, even if the spatio-temporal distance between two observations is small, the value of the behavioral variables can be very different. For example, sensor $R002_S2$ had very few anomalies in all experiments excluding Exp.1, the reason is that its values of flow are very low, generally behind 20 vehicles every 15 minutes; thus, compared with the other sensors that have higher variability in traffic flow the difference between its observations is reduced. Although, when the observations of sensor $R002_S2$ are considered singularly, as in Exp.1, the algorithm can recognize anomalous peaks. Exp.2 demonstrates good performances in detecting sensor faults or point anomalies in the majority of sensors, but when the percentage of anomalies is very high (above 5%) the detected anomalies also include unusual traffic conditions. Thus, when the percentage of anomalies for a sensor is low (less than 1%) a good solution to find unusual traffic conditions is to perform anomaly detection for that single sensor, as in Exp.1.

The set of parameters of Exp.3 guarantee the identification mainly of sensor faults. Indeed, in Exp.4, both sensor faults and unusual traffic conditions are identified.

5.3 Anomaly detection on air quality sensors

The need for a hyper-local sensing system able to monitor pollutants variations in the urban areas collides with the poor quality of measurements generated by low-cost air quality sensors. The presence of anomalies in raw data negatively influences the performances of data-driven calibration models. For this reason, the anomaly detection techniques presented in this thesis are performed on raw data. Data cleaning pre-processing of the raw data allows the calibration model to run on cleaned information, thus improving the precision and reliability of the air quality monitoring system. Since the measurements generated by the two channels (auxiliary *aux* and working *we*) are correlated, anomaly detection is performed on multivariate time-series and/or correlated variables. However, when one of the channel generates anomalous data, the other channel can continue to generate reliable measurements.

In Section 5.3.1, a majority voting system which combines 3 classifiers is presented and evaluated on the TRAFair air quality dataset. Thanks to low-cost sensors and AI techniques, anomalies are identified in the raw data measurements and removed, and the generated missing values are repaired.

5.3.1 Majority Voting

The majority voting (MV) system applied to the raw measurements combines 3 classifiers: (1) the Sliding Window anomaly detection which considers the consecutive measurements and the IQR to find anomalies far from the normal behavior of the system, (2) the FFIDCAD (Forgetting Factor Iterative Data Capture Anomaly Detection) which is an iterative algorithm, and (3) an algorithm based on the correlation between the values of each gas (NO, NO₂, CO and O₃) and the measurements of air temperature and humidity. Every time a new measurement is done by a sensor, just after storing the measurement into the TRAFair database, the three classifiers are applied to the measurements. The research for anomalous data is performed on both channels of each pollutant and device independently, since each device is individual and performs differently from the other devices even if they are in the same location.

For each gas, the algorithms consider the two channels of raw measurements (e.g., NO_{aux} and NO_{we}) separately, if at least one of them is anomalous, then the gas measurement is an anomaly. Each algorithm has the same weight in the ensemble method and detects anomalies

on each pollutant individually; thus, if at least two out of three algorithms classify the pollutant measurement as anomalous, that measurement will be considered an outlier. The three selected algorithms exploit different strategies and are based on diverse correlations among data; therefore, their combination allows identifying anomalies with higher confidence than applying only one algorithm.

This approach has been implemented in the AIRSense framework that is a comprehensive solution to deal with low-cost sensor data, from the collection of raw observations to the generation of reliable pollutants concentrations. The framework is available as open source software.³ Thanks to low-cost sensors and AI techniques, anomalies have been identified in the raw data measured by the sensors and repaired. This data cleaning pre-processing of the raw data allows the calibration model to run on cleaned information, thus improving the precision and reliability of the Air Quality (AQ) monitoring system.

Sliding Window Anomaly Detection: SWAD

The Sliding Window Anomaly Detection algorithm (SWAD) is a combination of the Differences Based algorithm and the Interquartile Range (IQR). This algorithm is an example of univariate anomaly detection.

The Differences Based algorithm evaluates the difference between two consecutive values. The “difference threshold” corresponds to the maximum variation allowed in a fixed period. Given the threshold ε , if $|x_t - x_{t-1}| > \varepsilon$, then x_t is classified as anomalous. Intuitively, the variation of observed values in a few minutes is expected to be small since pollutant measurements do not change rapidly. Defining the optimal value of the threshold is tricky, it might be chosen by a domain expert or exploiting statistical analyses. The weak aspect of this method is the incapacity of finding consecutive anomalies since the difference between two anomalies is assumed to be small. For this reason, this algorithm is combined with the study of the Interquartile Range (IQR).

The Interquartile Range (IQR) analyzes the distribution of the raw measurements and identifies the first quartile ($q1$) and the third quartile ($q3$). By using these values, the upper bound and the lower bound of the acceptable measurements range is found, as follows:

$$\text{upper bound} = q3 + q * IQR$$

$$\text{lower bound} = q1 - q * IQR$$

³https://drive.google.com/open?id=1LqZSVXA_2A1Hk_7fk9UwDOYEda-J6qvG

The parameter q defines the width of the range and can be set to a custom value according to the type of outliers to detect; high values of q are used for extreme outliers. The measurements out of the range are labeled as outliers.

SWAD takes in input the time-series with the observations of each gas channel and performs anomaly detection on a sliding window of a predefined size (k measurements). Anomaly detection is made on each channel separately. During the initialization phase of this algorithm, given an array of k measurements $\{x_0, \dots, x_{k-1}\}$, mean, standard deviation, lower bound and upper bound are calculated. When a new observation x_{t+1} is received, the difference between x_{t+1} and x_t is calculated. If the variation is higher than the “difference threshold”, then x_{t+1} is classified as anomalous, otherwise the instance gets normalized using the mean and standard deviation previously calculated, as follows:

$$z_{t+1} = \frac{x_{t+1} - \text{mean}}{\text{standard deviation}}$$

If z_{t+1} is higher than the upper bound or the IQR range or smaller than the lower bound, then x_{t+1} is considered an outlier. Thereafter, the window is updated removing the oldest observation x_{t-k} , and adding x_{t+1} . Finally, the parameters (mean, standard deviation, lower bound and upper bound) are updated. The algorithm proceeds until new data arrive.

SWAD is able to find anomalies on the measurement of each channel and gas, and the temperature and humidity data.

Forgetting Factor Iterative Data Capture Anomaly Detection: FFIDCAD

The Forgetting Factor Iterative Data Capture Anomaly Detection (FFIDCAD) [97] allows implementing a multivariate anomaly detection algorithm studying the correlation of two or more correlated features. The algorithm takes as input the values of the correlated features and defines an ellipsoidal boundary around these. The observations out of bound are classified as anomalous. Five algorithms have been run: one algorithm for each pollutant and one for temperature and humidity. The correlated features for each gas are the two channel measurements, while for the last algorithm the correlated features are the measurements of temperature and humidity. The first 4 algorithms provide the anomalies on the pollutant measurements, while the last algorithm detects anomalies on temperature and humidity.

The hyper-ellipsoid on the correlated features is defined as follows:

$$\text{ell}_k(m_k, S_k^{-1}, t) = \{x \text{ in } R^d \mid (x - m_k)^T S_k^{-1} (x - m_k) \leq t^2\}$$

where m_k is the array containing the mean of the features, x is the data point, t^2 is the confidence space of the data distribution, and S_k^{-1} is the inverse of the covariance, that can be also defined as the precision matrix.

During the initialization phase of FFIDCAD, the mean of the first two data points is calculated and the precision matrix S is initialized to an identity matrix I with size $n \times n$, where n is the number of the analyzed features. Then, S is a 2×2 matrix and contains the values of the working and the auxiliary channels (or temperature and humidity) of the first two consecutive observations of the time-series.

The diagonal elements of precision matrix measure how the variables are clustered around the mean. The off-diagonal elements measure independence and their values are equal to 0 if features are independent. Thus, the higher the diagonal elements are, the more aggregated the values are to the mean.

When a new observation x_k is available, the precision matrix gets updated as following:

$$S_{k+1}^{-1} = \frac{kS_k^{-1}}{k-1} \left[I - \frac{(x_{k+1} - m_k)(x_{k+1} - m_k)^T S_k^{-1}}{\frac{k^2-1}{k} + (x_{k+1} - m_k)^T S_k^{-1} (x_{k+1} - m_k)} \right]$$

Also, the mean is updated incrementally: $m_k = \lambda m_{k-1} + (1 - \lambda)x$.

The new instance x_{k+1} is considered an anomaly if:

$$(x_{k+1} - m_k)^T S_k^{-1} (x_{k+1} - m_k) > bound$$

The *bound* value is calculated through the percent point function, taking the parameter p as the percentage. p is the p-value as defined in statistics and identifies the confidence space: the range of values considered non-anomalous. For example, if $p = 0.98$, then the ellipsoid will cover 98% of the data. By assigning different values to p , different confidence spaces were obtained. The more p is close to 1, the fewer anomalies will be found. p can be set through a heuristic computation, as follows: $p = 1 - 10^{-i}$. Increasing the value of the exponent i , the value of p approaches 1, and the number of detected anomalies decreases. At the end of each iteration, the mean and precision matrix is updated considering the value of the new observation.

FFIDCAD exploits a forget factor $\lambda \in (0, 1)$ to update its parameters; it is introduced since after several matrix elements have been processed, the difference between m_{k+1} and m_k approaches zero, not allowing to correctly update the mean.

Temperature and Humidity Based Anomaly Detection: THAD

The Temperature and Humidity Based Anomaly Detection algorithm (THAD) is based on the dependencies among all the values measured by the sensor.

Before the implementation of this algorithm, an analysis has been conducted on the dependency between the pollutants measurements and other features, that are the status of the sensor, the season of the year, and the values of humidity and temperature. The aim is to find the feature that the working and the auxiliary channels are most dependent on. Analyzing the data, it was noticed that the working and auxiliary channels values of NO and NO₂ are more dependent on the temperature value, while O₃ depends more on the values of humidity; therefore, the presence of anomalies is studied in relation to the value of temperature or humidity.

During the training phase of THAD, the measurements for each channel and pollutant are grouped into different ranges according to the values of temperature or humidity. The raw data collected by working and auxiliary channels of NO and NO₂ are grouped by six temperature ranges; while voltages measurements of O₃ are grouped by five humidity ranges. For each group, mean and standard deviation are calculated separately and the lower and upper bounds are evaluated by using the IQR algorithm. There is a significant difference between this algorithm and the previous ones. Indeed, in this case, mean, standard deviation, and the values of lower and upper bounds are calculated only once (during the training phase) and they are not updated after the analysis of a new observation. Therefore, the anomalies of this algorithm really depend on the dataset provided for the initialization of the algorithm. Thus, the training dataset should have a comprehensive range of temperature and humidity values, including all the seasons (one year data).

During the detection phase, the measurement of each pollutant and channel is assigned to a group according to its value of temperature or humidity. Then, it is normalized by using the mean and standard deviation of that group. After that, the normalized value is compared to the lower bound and the upper bound of the group. If the normalized measure is out of the range, it is classified as anomalous. In this algorithm, the value of each channel is analyzed individually: if at least one channel's value is classified as anomalous, both the measurements of that pollutant, in the corresponding observation, are considered anomalous. Anomaly detection on temperature and humidity is performed by using the IQR algorithm applied to the whole dataset. If the value of temperature or humidity is anomalous, all the measurements of that observation are classified as anomalies. However, the temperature and humidity sensors are more reliable than the other sensors (NO, NO₂, and O₃) and anomalies are very rare.

Table 5.6 Anomalies detected by SWAD, FFIDCAD, THAD.

	NO	NO ₂	O ₃	Temp.	Hum.
SWAD	16,720	35,436	38,986	1,176	1,176
FFIDCAD	6,577	4,705	5,364	562	562
THAD	9,050	14,606	26,978	0	0
MV	10,591	15,365	17,365	9	9
(% of tot.)	(0.26%)	(0.37%)	(0.42%)	(~0%)	(~0%)

Configuration and application of the algorithms

The anomaly detection algorithms described in the previous sections have been applied to the raw measurements collected by 12 low-cost sensors from August 2019 to April 2021 (21 months). Each algorithm takes in input the same dataset, which consists of 4,122,541 observations. The configuration of each algorithm has been set up after some experiments [104].

SWAD has been configured to use a window of 2,000 observations; thus, the algorithm detects anomalies based on the data distribution of the previous 66 hours, approximately. The “difference threshold” has been set to 2,000 to detect very different consecutive observations. After some experiments, assuming that anomalies rarely occur in the available data, the parameter q of the IQR algorithm has been set to 6 since this value showed the expected rate of anomalies (0.1%).

The only configuration parameter used by THAD is q , set to 6.

Some experiments have been conducted to choose the exponent i for the parameter p in FFIDCAD. The value of i which detects the expected rate of anomalies was 16.

Table 5.6 shows the number of anomalies for each gas detected by each algorithm and the ones selected by the MV. As can be seen, there are few anomalies among the values of temperature and humidity. This confirms the assumption made in the implementation of THAD. The higher number of anomalies has been detected by the SWAD on NO₂ and O₃; while FFIDCAD is the one that finds fewer anomalies among the pollutants measurements. MV classifies as anomalies on average 0.3% of the total number of measurements. Analyzing these anomalies, it was noticed that SWAD is the most accurate algorithm since its anomalies are also detected by at least one of the other two algorithms. In particular, the percentages of anomalies detected by SWAD compared to the anomalies included in MV are: 99% for NO, 100% for NO₂, and 96% for O₃.

Experiments and results

Evaluating the performance of anomaly detection is challenging, especially when annotations made by experts are not available. Experiments on real-world data have been evaluated in

two ways: supervised evaluation with the help of environmental experts and comparison of the calibration performance including or excluding anomalies. In these experiments, CO was excluded since its concentration is very low in the city of Modena and the environmental agency ARPAE⁴ decided to stop the monitoring of CO. Therefore, a comparable value was not available.

Supervised evaluation considering experts annotations. Environmental experts regularly monitor the behavior of the sensors (thanks to the dashboard shown in Section 4.2.4) and change the status of the sensors when their behavior is not reliable; thus, to validate the results of the MV, the status of the sensors in the timestamp of the detected anomalies was checked. In 70% of cases, anomalies are related to “broken” status. This means that the environmental experts labeled them as an unexpected behavior of the sensor and are likely to be real anomalies. Since probably environmental experts did not recognize all the anomalies of the sensors, other proofs are needed to guarantee the validity of the proposed methodology. For this reason, the results obtained by MV have been compared with the ones obtained by the LSTM (Long Short-Term Memory) [105, 106] autoencoder.

Calibration performance evaluation. To evaluate the effects of anomaly detection on the calibration models that estimate the pollutant concentration values, two different experiments were conducted on the raw data collected from the low-cost sensor network of the city of Modena for NO, NO₂, and O₃. The experiments are reproducible since all datasets are available as Open Data. In the first experiment (Exp1), the 10-minutes aggregated raw data were directly used to train the calibration model; in the same way, in the test phase, raw data were directly aggregated without excluding anomalies and then given as input to the calibration model. In the second experiment (Exp2), the MV anomaly detection methodology is employed to recognize anomalies in raw data of training and test set. Only the reliable raw measurements are used to evaluate the 10-minutes average and generate the aggregated time-series used as input by the calibration model.

The training datasets of the two experiments contain data from August 2019 to March 2020, while the test datasets include data from the 15th of June to the end of September 2020. To train and test the models, only the period in which the sensors were located near the legal station (“calibration” mode) is considered. In order to evaluate the performances of the calibration models, the concentration of pollutants generated by applying the calibration models to the test dataset is compared to the measurements of the legal station.

The metrics used to evaluate the results are Root Mean Square Error (RMSE) and accuracy. RMSE is the root of the mean of the squared differences between calibrated values and corresponding values observed by the legal station in the same 10-minutes time

⁴<https://www.arpae.it/it>

interval. Since the differences are squared before they are averaged, the RMSE penalizes large errors. RMSE can range from 0 to ∞ . RMSE values give an idea of the absolute value of the error on the calibrated values. In some applications that use a range of values to drive information about air quality, these errors are non-material. This is the case of air quality dashboards, where air quality information is conveyed through color scale maps. Several color scales with different ranges are available, the choice was the one provided by the European Environmental Agency (EEA)⁵ to measure the ability of the algorithms to correctly predict the right color class. Both observed values from the legal station and the calibrated values are associated with the corresponding color in the color scale. Accuracy has been calculated as the ratio between the number of correct predictions and the total number of input samples.

Table 5.7 shows the values of RMSE and accuracy obtained for each gas and each sensor in the two experiments. Besides, for each gas and experiment, an average value of RMSE and accuracy is provided. On average, the RMSE values of Exp2 are always lower than the ones of Exp1. In Exp2, the number of anomalies detected in the training set is 4,198, while in the test set is 700. The effect on the performance of the model results in a significant reduction of the RMSE. However, the exclusion of anomalies generates 38 missing values in the multivariate time-series during the test phase of Exp2; thus, for that timestamps the calibrated values are not generated.

For all the pollutants, the RMSE is significantly reduced introducing anomaly detection before calibration. A particular case is the one of sensor 4008 that reported a very high RMSE values in Exp1 for both NO and NO₂. Analyzing the difference between the calibrated data and the actual concentrations provided by the legal station, there is a very high difference in only one 10-minutes interval and this difference strongly influences the value of RMSE. Some raw measurements related to that 10-minutes interval have been classified as anomalies in Exp2, thus the aggregated value changed and the RMSE decreased.

As can be seen in the Table, the values of accuracy are always very high for NO and NO₂. O₃ has lower values of accuracy. This could have two explanations. Firstly, the size of the training set for O₃ is smaller than the one for NO and NO₂ since O₃ is measured by only one legal station. Therefore, the sensors have to be close to that legal station to collect data useful for the calibration. Secondly, the ranges of the color scale for O₃ are 20-units large, while the ones for NO and NO₂ are 50-units, 100-units, and 200-units large. Therefore, also a small error in O₃ affects the classification.

⁵<https://www.eea.europa.eu/it>

Table 5.7 Evaluation of the Majority Voting anomaly detection on real-world air quality sensor data.

gas	sensor	RMSE		ACCURACY	
		Exp1	Exp2	Exp1	Exp2
NO	4003	5.24	3.53	0.99	0.99
	4005	2.82	2.34	1	1
	4006	2.53	2.59	1	1
	4007	2.74	3.15	0.99	0.99
	4008	93.24	4.37	0.99	0.99
	4010	3.8	2.48	0.99	0.99
	4011	4.16	3.92	0.99	0.99
	4013	26.24	2.2	0.99	0.99
	4014	2.39	2.81	1	1
	M	15.91	3.04	0.99	0.99
NO ₂	4003	8.25	15.07	0.97	0.98
	4005	12.2	12.44	0.95	0.95
	4006	9.55	8.81	0.97	0.97
	4007	10.76	9.83	0.95	0.95
	4008	347.03	9.38	0.96	0.96
	4010	7.78	7.58	0.98	0.98
	4011	64.54	8.89	0.98	0.98
	4013	8.01	8.2	0.98	0.98
	4014	10.05	9.33	0.96	0.97
	M	53.13	9.95	0.97	0.97
O ₃	4003	20.03	18.51	0.53	0.56
	4005	19.9	17.7	0.65	0.75
	4006	19.41	17.67	0.61	0.65
	4007	14.62	12.23	0.8	0.82
	4008	19.79	22.49	0.61	0.57
	4013	25	22.52	0.48	0.5
	4014	222.53	86.27	0.71	0.52
M	48.75	28.20	0.63	0.62	

5.3.2 Real-time anomaly detection on edge sensors

Executing anomaly detection directly on edge sensors allows saving time in obtaining reliable sensor data without the need of post-processing tasks. After training a Machine Learning algorithm, the obtained model can be ported on IoT board where it is executed every time a new observation is performed by the sensor.

The top 12 unsupervised Machine Learning anomaly detection algorithms of the current state-of-the-art have been selected and evaluated. In the following, each of them is briefly

described. Since the algorithms are unsupervised methods it is not necessary to have a labeled dataset for training. Every algorithm takes in input a subset of the TRAFair air quality dataset, analyzes the data and outputs the clean data. Also, this operation allows to obtain the trained model that can be applied on other data. As explained in the following list, the selected algorithms are of 6 types: nearest-neighbor based algorithms, clustering based, statistical methods, subspace based, classifier, and angle based algorithm.

A. Nearest-neighbor based

K-Nearest Neighbors Detector (KNN) [107] is a distance based algorithm that considers, for each point, the sum of the distances from its k nearest neighbors. The points with the highest values are the outliers.

Local Outlier Factor (LOF) [82] uses a density based technique aimed at providing an anomaly local degree factor to each element of the dataset. The resulting degree factor resembles how each element is isolated with respect to the surrounding neighborhood. The more isolated the element is, the higher is the chance that the element is an anomaly.

Connectivity-Based Local Outlier (COF) is an evolution of the LOF algorithm that points out that “isolation” and “low density” are different concepts. Isolation can imply low density, but low density does not necessarily imply isolation. COF integrates LOF with the assumption that isolation is not the only factor that determines if an element is an anomaly but also the density of the surroundings elements is an important feature to evaluate.

Stochastic Outlier Selection (SOS) exploits the concept of “affinity”. Affinity is proportional to the similarity between two data points. The higher is the similarity, the higher is the affinity. Similarity can be computed using distance measures, such as the Euclidean distance. The outliers are identified as those points which have insufficient affinity with respect to all other points.

B. Clustering based

Clustering-Based Local Outlier (CBLOF) is an evolution of the LOF algorithm that extends the concept of anomaly from an instance level to a cluster level. It assumes that an anomaly is not necessarily an isolated element but anomalies can also be grouped into small clusters of anomalies. The different size of the clusters is the feature that distinguishes a cluster of non-anomalous elements from a cluster of anomalous instances. Smaller clusters are considered anomalous clusters, while bigger datasets are considered non-anomalous datasets.

C. Statistical

Histogram-based Outlier Detection (HBOS) [108] is an unsupervised algorithm which models univariate features densities exploiting histograms with bins of fixed or dynamic width. After normalization, each histogram is used to assign an anomaly score to each data instance. The score is based on the inverse of the estimated density.

Minimum Covariance Determinant (MCD) [109] exploits the covariance matrix as a parameter of distribution to define when data points are normal or outliers. Since the determinant of a covariance matrix measures how broad the distribution is, the algorithm tries to find the data points in a dataset whose covariance matrix has the lowest determinants. This operation allows to select the subset of the data that is most tightly distributed and to exclude anomalies, which lie further away from the rest of the data.

D. Subspace based

Principal Component Analysis (PCA) [110] is a method for linear dimensionality reduction that can be used for detecting anomalies. The scope is to project the input data to a lower dimensional space and analyze the variance in the data. This method calculates the covariance matrix of the data and decomposes it into orthogonal vectors (eigenvectors) associated with eigenvalues. The eigenvectors with high eigenvalues capture most of the variance in the data. The outlier score of each sample is obtained calculating the sum of the projected distance of that sample on all eigenvectors of the hyperplane. This method is highly sensitive to extreme value outliers.

Subspace Outlier Detection (SOD) [111] detects outliers in varying subspaces of a high dimensional feature space. The size of these subspaces is less than the one of the input feature space. Then, the algorithm calculates how well each instance of the input data fits into the axis-parallel subspace spanned by its neighbors. Any object deviating from this subspace will be regarded as an anomaly.

E. Classifier

Isolation Forest (IF) [112] is a tree-based model aimed at isolating anomalous data instead of profiling normal behaviors. It assumes that isolating a non-anomalous data requires a high number of splits of the dataset, while isolating anomalous data requires a low number of splits. Each split represents an edge of the generated tree, the shorter is the path that connects the root to the observed element, the higher is the probability that the observed data is an outlier.

One-class SVM detector (OCSVM) [113] tries to separate high-density elements from

low-density elements by calculating the smallest hypersphere that allows to maximize the distance between the two groups of elements.

F. Angle based

Angle-base Outlier Detection (ABOD) [114] assesses the spectrum of observed angles for a point to all other pairs of points. The broader is the spectrum, the higher is the number of surrounding points in all different directions. As opposite, the smaller is the spectrum, the lower is the number of surrounding points and the higher is the probability for that point to be an outlier. This approach, in contrast with distance-based techniques like the Local Outlier Factor, and mitigates the effects of the “curse of dimensionality” on mining high-dimensional data and big datasets.

Table 5.8 IoT boards chosen for on-board evaluation of TRAF AIR datasets trained anomaly detection models.

Board Name	Processor, Flash, SRAM, Clock (MHz)
B1: Teensy 4.0	ARM Cortex-M7, 2MB, 1MB, 600
B2: STM32 Nucleo H7	ARM Cortex-M7, 2MB, 1 MB, 480
B3: Arduino Portenta	ARM Cortex-M7+M4, 2MB, 1MB, 480
B4: Feather M4 Express	ARM Cortex-M4, 2MB, 192KB, 120
B5: Generic ESP32	Xtensa LX6, 4MB, 520KB, 240
B6: Arduino Nano 33	ARM Cortex-M4, 1MB, 256KB, 64
B7: Raspberry Pi Pico	ARM Cortex-M0+, 16MB, 264KB, 133

Train and port Machine Learning algorithms to plain C To execute any classifier or anomaly detection models on IoT edge sensors, the user needs to port the standard algorithms (trained Machine Learning classifier or detector) to its MCU executable plain C versions [115]. Then, the generated model needs to be stitched with the IoT use-case application, followed by efficient deployment and execution. The method for training and porting is general since it can be applied to a range of Machine Learning tasks such as regression, classification, including the unsupervised anomaly detection.

In IoT edge sensors, the flash memory (program space) is always much greater than the available SRAM (see Table 5.8). Thus, the proposed method, when realized, produces a C version of the models which does not depend on the SRAM during execution. Instead, it exploits the larger flash memory in order to enable the deployment and execution of bigger classifiers. In other words, the idea is to sacrifice flash memory in favor of the limited SRAM since it is the scarcest resource in the majority of MCUs. The proposed SRAM optimized

method hard codes the model splits in C, without storing any reference of the splits and other models related parameters/values into variables. Since the application of Machine Learning model does not allocate any variables, 0 bytes of SRAM will be consumed to execute the C version of the ported classifier to produce inference results. Thus, explained SRAM optimized porting of Machine Learning classifiers to plain C is summarised in Algorithm 1.

Algorithm 1 Pseudocode of anomaly detection algorithm training and optimized porting to plain C running on GPU. The generated plain C model is execution-ready on edge devices.

```

1: Import implementation of unsupervised anomaly detection algorithms
2:   ▷ can also use other algorithms like Perceptron, RidgeClassifier, KernelSVC, etc. for
   detection
3:   import packages: time, train_test_split, from sklearn.metrics import classifica-
   tion_report
4:   dataset ← TRAFair historical dataset           ▷ cleaned dataset
5:   train_features, test_features, train_types, test_types ← train_test_split (dataset) ▷
   splits into random train, test subsets
6:   train ▷ function for anomaly detection algorithms training using loaded dataset
7:   clf_OCSVM ← OCSVM.implementation ()           ▷ define One-class SVM
   (OCSVM) detector
8:   clf_OCSVM ← clf_OCSVM.fit (train_features, train_types)           ▷ trained
   OCSVM detector
9:   evaluate           ▷ function to evaluate trained model
10:  predictions ← clf_OCSVM.predict (test_features)
11:  inference time ← use clf_OCSVM.predict () inside time.time () ▷ pass data
   for 1 sample and 100 samples
12:  evaluation ← use module classification_report (test_types, predictions) ▷
   f1-score, support, macro avg, accuracy, etc. are reported
13:  if evaluation satisfactory then
14:    port ← m2cgen or sklearn-porter ▷ Schemes like micromlgen, emlearn can
   also be used
15:  else
16:    re-tune OCSVM parameters (e.g. kernel, gamma, degree, etc), then use
   functions train, evaluate, and port
17: Repeat Lines 5 to 15 by replacing OCSVM with other anomaly detection algorithms

```

Execute ported Machine Learning algorithms on IoT boards The IoT application executed by edge sensors receives the input data in different formats such as image frames, audio signals - sensor readings in the examined case. When there is a demand for edge level offline analytics, a high-quality Machine Learning algorithm is trained to produce inference results based on the data produced by edge sensors, then port that model to C code using the

method from Algorithm 1. In this section, the structure of the generated C code is described. Then, it will be explained how to stitch the C code with the IoT application and perform inference whenever required by the user or the IoT edge application.

Algorithm 2 Pseudocode for executing anomaly detection models on edge devices.

```

1: load ported OCSVM detector and TRAFair test sets
2:   #include OCSVM_model.h           ▷ file contains ported OCSVM in C
3:   predict (X)                     ▷ links ported OCSVM model with main IoT application
4:   #include TRAFair_test.h         ▷ file contains data (test set) to supply during onboard
   inference
5:   2D array X [test_set_size] [features_dim]   ▷ test_set_size rows containing
   features_dim features of test set
6:   Y [test_set_size]                 ▷ one row containing labels of all test set samples
7: for i in test_set_size do
8:   predictions ← predict (X[i])           ▷ data passed to predict function inside
   OCSVM_model.h
9: onboard evaluate                   ▷ function to evaluate ported classifier on MCUs
10:  inference time ← use predict () inside millis ()   ▷ pass data for 1 sample and 100
   samples
11:  accuracy ← compare predictions with Y [test_set_size]

```

To obtain prediction results on edge sensors, using presented method, no dependencies or shared libraries are required to be added in the file system along with the C code of the model. Just the .h file needs to be compiled along with the user's main IoT edge application and flashed via the device supported software. The interior of the .h model version contains the C code of the TRAFair dataset trained anomaly detection models.

The execution method of the ported C version of Machine Learning algorithm is given in Algorithm 2 that refers to OCSVM as an exemplar method, however the procedure is the same for all the available Machine Learning anomaly detection algorithms. During the programming or edge application design phase, the users have to just include the generated .h model as a header file at the beginning of their program (see Line 2). Inside any of the model files generated using porting tools, there exist a function named *predict* (different function name depending on porting method in use), to which the main program can pass the values for which it needs predictions (see Line 8). When *predict* is called, the MCU starts to execute the model using its default available C compiler (without requiring any dependencies or external libraries) as a subroutine, without disturbing the device's main routine, which is handled by the main IoT edge application.

Solution evaluation

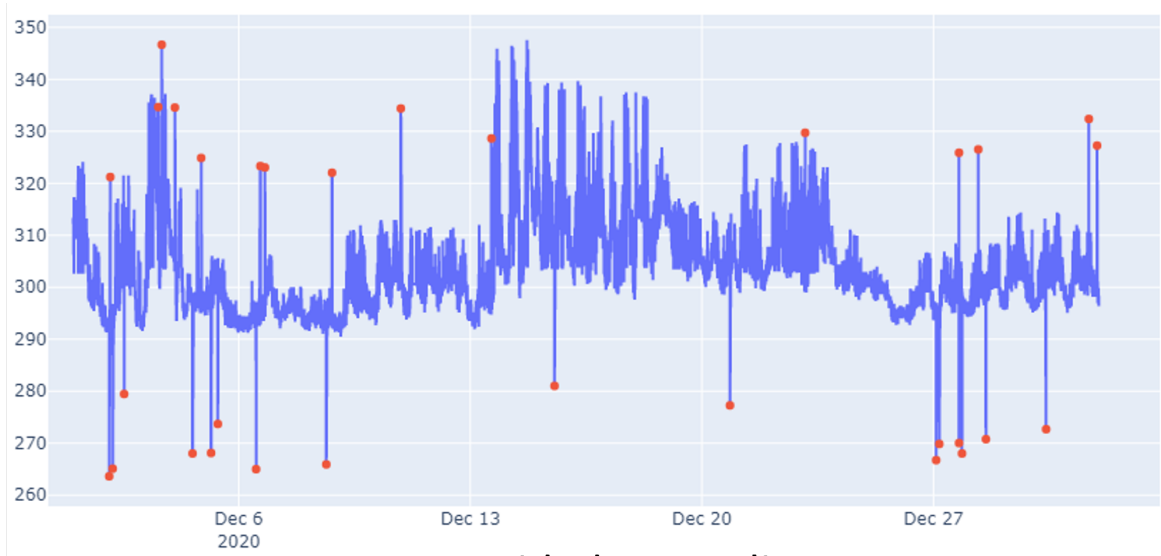
The proposed solution for detecting anomalies on edge IoT device has been evaluated on measurements of the TRAFair air quality sensors. The experiments are conducted on a portion of the whole available datasets, that is a partially synthetic time-series dataset. In the following, it is explained how the synthetic dataset has been generated. Then, the evaluation is explained in terms of the accuracy and f1-score of the detection and size of the model.

Partially synthetic dataset generation The experiments have been conducted on the measurements of NO, NO₂, CO and O_x of one exemplar sensor during the year 2020. To create the labeled synthetic dataset, firstly, the measurements identified by the environmental experts as broken/offline/warmup status have been excluded. Then, the values of each channel and gas are taken into account as a separated time-series, along with the timestamp of the observation. In this way, 8 different time-series are obtained. Even if the anomalies labeled by the environmental experts have been excluded, these time-series can contain other anomalous data. For this reason, an iterative procedure to repair the data was set up. Then, artificial anomalies are generated in the repaired data. To repair the data, the first (Q1) and third (Q3) quartiles and the interquartile range (IQR - i.e., the difference between Q1 and Q3) of the measurements are calculated considering 2 days at time. Based on these values, the lower and upper bounds are calculated by the formula:

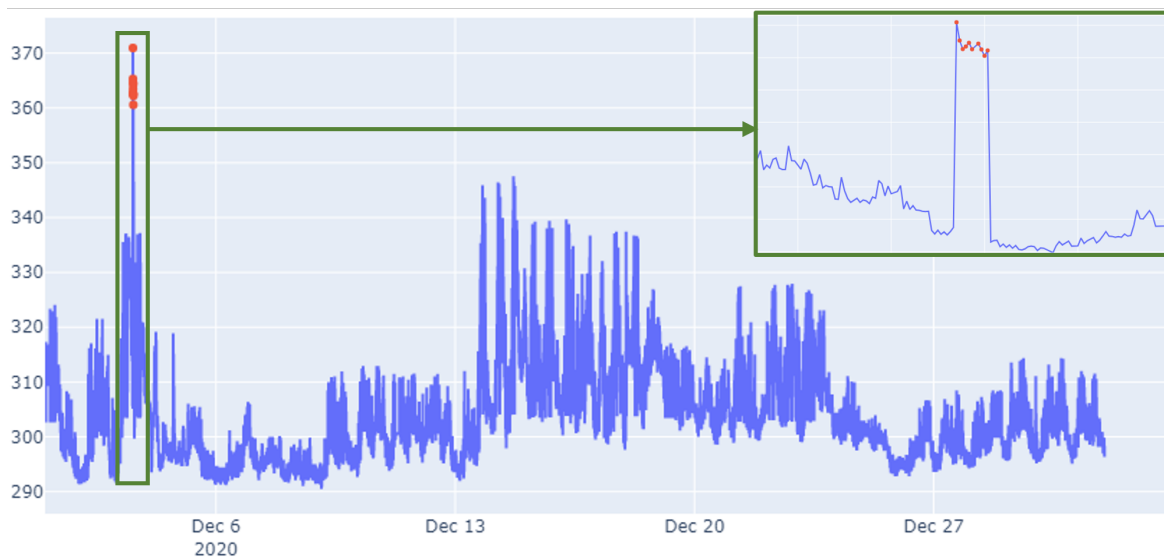
$$lower_bound = Q1 - 1.5 * IQR$$

$$upper_bound = Q3 + 1.5 * IQR$$

Measurements out of the range, i.e. higher than the upper bound or smaller than the lower bound, are repaired with the simple average of the measurements in nearest one hour, i.e. the 30 previous minutes and 30 following minutes, excluding the values out of the range. If all the surrounding measurements are out of the range, the value is not repaired and a missing value is generated. After obtained the repaired time-series, two different types of anomalies are generated: two-sided anomalies and chunked anomalies. The first are values higher or lower than the normal data in magnitude and are generated at random timestamp adding or subtracting a certain value to the real measured value, while the latter consists of multiple consecutive anomalous values. Figure 5.16 shows an example of the two types of anomalies generated. In this way, two datasets are obtained: AQ1 is the dataset of the exemplar sensor of year 2020 containing only the two-sided anomalies, while AQ2 contains only the chunked anomalies.



a. Two-sided anomalies



b. Chunked anomalies

Fig. 5.16 The two types of anomalies generated in the measurements of the working channel of NO (NO_{we}) of December 2020. Anomalous measurements are highlighted in red.

Training The same procedure described in Algorithm 1 for training and porting the selected Machine Learning models was followed for AQ1 and AQ2. The two datasets AQ1 and AQ2 are treated separately. Since each sensor in the air quality devices acts independently of the other sensors, 4 models are trained, each one related to a different gas. Each algorithm takes in input the measurements of the working and auxiliary channels of one gas. Therefore, 4 models need to be ported on the IoT devices. The measurements of January 2020 are used

for training, while the data of the other months are used for testing considering one month at time. This experiment was done to understand when the performance of the model decreases. Before training the models, the measurements are normalized by using the Robust Scaler that is more suitable when the dataset contains anomalies.

Table 5.9 Evaluation results of the anomaly detection models trained on the NO₂ measurements of the air quality TRAFair dataset AQ1.

	Feb		Mar		Apr		May		Jun		Jul		Aug		Sep		Oct		Nov		Dec	
	A	F1	A	F1	A	F1	A	F1	A	F1	A	F1	A	F1	A	F1	A	F1	A	F1	A	F1
ABOD	1	.50	1	.50	1	.50	1	.50	1	.50	1	.50	1	.50	1	.50	1	.50	1	.50	1	.50
CBLOF	.95	.54	.96	.55	.81	.46	.84	.47	.75	.44	.69	.42	.78	.45	.91	.50	.98	.57	.99	.66	1	.62
COF	.99	.50	.99	.52	1	.51	.99	.50	1	.50	1	.51	.99	.50	.99	.51	.99	.52	1	.51	1	.50
IF	1	.50	.99	.50	.98	.50	.98	.50	.97	.49	.97	.50	.98	.50	.99	.50	1	.50	.99	.51	1	.50
HBOS	1	.51	.99	.50	.97	.50	.95	.49	.95	.49	.95	.49	.95	.49	.98	.50	.98	.49	.99	.50	1	.50
KNN	.91	.51	.90	.50	.62	.39	.58	.37	.52	.35	.44	.31	.56	.36	.77	.45	.91	.49	.98	.60	1	.70
LOF	.96	.50	.95	.49	.96	.50	.98	.49	.97	.50	.98	.51	.98	.50	.96	.49	.96	.49	.96	.49	.96	.49
SVM	.92	.51	.94	.52	.76	.44	.76	.44	.68	.41	.60	.38	.71	.42	.86	.48	.96	.52	.98	.59	1	.60
PCA	.99	.61	.99	.52	.94	.51	.94	.49	.91	.49	.91	.49	.95	.50	.97	.52	.99	.50	.99	.52	1	.52
MCD	1	.72	1	.52	.98	.56	.98	.51	.97	.53	.98	.55	.98	.51	.99	.55	.99	.51	1	.52	1	.52
SOD	.92	.53	.98	.56	.99	.53	1	.62	1	.65	.96	.66	.97	.61	.90	.60	.90	.60	1	.59	.99	.55
SOS	.94	.54	.91	.53	.99	.55	.94	.53	.96	.62	.94	.67	.90	.55	.90	.61	.95	.53	.91	.67	.94	.67

Results The size of the generated models for each gas and algorithm is less than 1 MB. Table 5.9 shows the values of accuracy and f1-score for each test and algorithm performed on the NO₂ measurements. The evaluation for the other gases is quite similar to the one in the table, therefore, it is not provided. The values of accuracy in the table are often high (most frequent values are from 0.95 to 1). On the other hand, the f1-score is low in most of the cases. This is due to the high number of true negatives and the true positives (correctly detected anomalies) are few. An important consideration can be done taking into account how the values of accuracy and f1-score decrease when the test set consists of measurements made in summer. Indeed, for most models performances get worse in June, July and August. This is due to the fact that the measurements of the sensors are influenced by temperature and humidity.

The tests performed on the air quality TRAFair dataset AQ2 did not provide good results, highlighting that the selected algorithms are not suitable to detect chunked anomalies without a proper fine tuning of parameters or the combination of other techniques.

Chapter 6

Semantic sensor data publication

Publishing Open Data has become an increasingly pressing need within government bodies and public administrations. The principles of sharing public information have been defined by the International Open Data Charter [116] and are the following: (1) *Open By Default*, (2) *Timely and Comprehensive*, (3) *Accessible and Usable*, (4) *Comparable and Interoperable*, (5) *For Improved Governance & Citizen Engagement*, and (6) *For Inclusive Development and Innovation*.

Opening up data often happens in an ad-hoc manner, and in many cases thousands of datasets are published without adhering to commonly-agreed standards and without reusing common identifiers. Hence, finding, reusing, and integrating data from different sources is a real challenge. Linked Data can respond to these challenges and can lead to smarter and more efficient government services and applications. Therefore, a crucial aspect when sharing Open Data is to follow the Linked Data principles [117].

Moreover, to publish high-quality, semantically annotated Open Data, it is crucial to identify the ontologies that better describe the domain of interest [118]. Ontologies provide a formal representation of the domain of interest and constitute the component with which the LOD (Linked Open Data) consumers (both humans and software programs) interact. Then, also the mapping between ontologies and data is important, since it is used to translate the operations on the ontology in terms of concrete actions on the data.

Unfortunately, semantic modelling often leads to slow performance, higher data size and increasing time required to query data w.r.t. relational databases. This is the reason why it would not be feasible to use Linked Data instead of the PostgreSQL database described in Chapter 3. However, Linked Data allows to organize data in such a way that is sharable, extensible, and easily re-usable, and to enrich their semantic information. In addition, this research provides the models that can be used by other people to transform structured data into semantic Linked Data.

The research of this chapter has been published in [119].

6.1 Related work

Several works have been published to define how to structure and share data produced in a Smart City [120]. Smart urban traffic ecosystems are identified in [121] as an example of a *big service*, “evolved from the collection of collaborating, interrelated services for handling and dealing with big data”. By collecting suitable sensor data and defining appropriate data exploitation strategies, it is possible to empower both citizens and decision-makers to improve our quality of life. However, for this dream to come true, the development of suitable data management strategies that can provide citizens and administrations with the information they need is a key issue. Thus, for example, according to [122], informational interventions are vital to encourage changes in attitudes and perceptions of people. Along the same lines, the work presented in [123] emphasizes that “Open Data can impact positively on citizens in particular and society in general”.

Traffic data can help in detecting traffic congestion, providing traffic flow prediction, and identifying traffic accidents. For this reason, several projects have published traffic information as Open Data. To lower the barrier for Open Data consumers to reuse traffic information, an Open Traffic Lights ontology has been proposed in [124]. That paper also reports a specification to publish historical and live data with Linked Data Fragments and a method to preserve the published data in the long-term.

In [125], for measuring the urban road congestion degree, that is one of the major issues in most metropolises, the estimation of a Traffic Congestion Index (TCI) of every road segment at every time slot has been proposed. An open traffic data platform has been proposed in [126] and used as a sensor data provider for different management applications.

The increased interest in Smart City data sharing for the public interest can be assessed by the number of datasets shared on the Open Data portals. When searching in the European Data Portal for “traffic”, 8949 datasets are obtained¹. Even if transport data only cover a 2.25% of the total datasets², a positive trend can be observed. Mainly, traffic Open Data are statistics that show the number of registered cars in different countries, and they are usually provided in the form of high-level vehicle fleet data in a city. On the other hand, data concerning the average daily traffic volumes on different roads and the specific traffic volume on different days and at different hours on specific road segments are not always shared with

¹Search done on August 5, 2020: <https://www.europeandataportal.eu/data/datasets?locale=en&query=traffic&page=1>.

²See the European Data Portal statistics per category at <https://www.europeandataportal.eu/catalogue-statistics/CurrentState>.

citizens. Having these data would be remarkably useful, as its exploitation could enable politicians of our cities to make more informed decisions, and the public could also be better informed about the traffic situation and use real data to promote health and environmental protection. The publication of the data as Linked Data enables a suitable and interoperable sharing of data that can facilitate the development of applications and services for smart cities [127, 128].

6.1.1 Analysis of traffic-related ontologies

Different types of existing ontologies related to the traffic of vehicles can be considered. The following ones can be highlighted:

- The *Vocabulary to Represent Data About Traffic Ontology* [129], developed by Óscar Corcho (a member of the Ontology Engineering Group at the Polytechnic University of Madrid) has been proposed for the representation of the situation of traffic in a city. It extends the Sensor Network Ontology (SSN) [130–132] to represent the intensity of traffic on the different road segments of a city. It represents road segments (concept *escjr:TramoVia*), traffic observations (concept *estrf:TrafficObservation*, which for the moment is specialized only in the subconcept *estrf:TrafficIntensityObservation*, but other subconcepts could be added in the future to represent other types of traffic observations), the sensor or sensing system used to obtain a given measurement (concept *estrf:TrafficIntensitySensor*, which is considered optional), the result of an observation (concept *TrafficIntensitySensorOutput*, which has a value –concept *estrf:TrafficIntensityObservationValue*, linked to *TrafficIntensitySensorOutput* through the property *ssn:hasValue*– and is produced by a specific sensor or sensing system –identified by a specific URI and linked to *TrafficIntensitySensorOutput* through the property *ssn:isProducedBy*–), and finally an instance *estrf:TrafficIntensity* that represents the type of property being measured (in this case, the intensity of the traffic).

This vocabulary is still work in progress, developed in the context of the working group on Transport of AENOR [133]. The authors recommend using this vocabulary in conjunction with the vocabulary proposed to represent city road maps (particularly, road segments) [134]. This proposal does not currently contemplate the modeling of traffic properties other than traffic intensities (*estrf:TrafficIntensityObservation*), but they can be easily added by extending *estrf:TrafficObservation*.

- The work in [135] presents an ontology-driven architecture that enables performing several automatic tasks to increase traffic safety and improve the comfort of the drivers.

The ontology layer is described as composed of three groups of interrelated concepts: concepts related to vehicles, concepts related to roads, and concepts related to sensors. The concepts related to vehicles describe a taxonomy of vehicles of different types (including commercial vehicles, public vehicles –buses and taxis–, private vehicles –cars, bicycles, and motorbikes– and priority vehicles –ambulances, police cars, and fire trucks–) and also allow representing information about their routes and locations. The concepts related to the infrastructure include a taxonomy of different types of roads (local roads, prefectural roads, national highways, and national expressways), as well as the representation of other parts of the infrastructure, such as the road segments, traffic lights and traffic signs, lanes, road markings (e.g., painted arrows), and other infrastructure elements (tunnels, parkings, roundabouts, bridges, gas stations, and toll stations). Finally, the concepts related to sensors are based on the use of the SSN ontology. Besides, a mapping schema is proposed to map the sensor data to semantic data, as in [136], in such a way that the sensor data can be automatically represented as instances of the SSN ontology; the property observed is *Car_flow property*.

This is a relevant work that proposes an ontological layer covering different aspects of traffic. Still, it mainly focuses on the development of an architecture that exploits such a layer to perform various actions through an agent layer. Some use case scenarios are presented: regulating the air conditioning of a car, traffic light adjustment based on the traffic flow and the weather conditions, and traffic congestion control for GPS navigators. Regarding the representation of traffic sensor data, the focus is only on the traffic flow, and, rather than proposing a new ontology or extending an existing one, the SSN ontology is directly adopted.

- The *open511 specification* [137] has been proposed as an open format for publishing road event data. Information about the road events can be provided by publishing an XML file or by allowing access to the data through a dynamic API. It supports representing elements such as events (constructions, special events –such as the celebration of a sport event–, incidents –including accidents and other unexpected events–, weather conditions, and road conditions –such as snow, ice, or fire on the road–) and geographic areas (places represented in GeoNames [138, 139]).

This work currently covers event data rather than traffic information. Nevertheless, some additional resources have also been proposed (currently as drafts that may be included in the Open511 specification in the future) to represent average historical speeds and the current speed of road segments.

- The *Road Accident Ontology* [140] focuses on the representation of information about accidents (vehicles affected, location of the accident) and the parties involved (persons involved in the accident and their insurance companies). This proposal is a draft, submitted by Daniel Dardailler for the W3C Geek Week celebrated in July 2012.

This ontology does not represent traffic, but it is included because accidents can affect traffic and even lead to traffic jams.

- Also focusing on accidents, the work in [141] proposes a lightweight *Car Accident Ontology for VANETs (CAOVA)*, that includes information about vehicles, accidents, occupants and the environment. The goal is to facilitate information about an accident to emergency vehicles.
- It is also relevant to mention the *Transportation Planning Suite of Ontologies (TPSO)* [142], which is a set of ontologies proposed for transportation planning. More specifically, eight ontologies are proposed to cover concepts related to time, meteorology, spatial locations, units of measure, changes, activities, recurring events, resources, and observations. Among these, the Observation Ontology [143] can be highlighted since it reuses the SSN Ontology to capture the concepts related to sensors, but also extends it by adding a few classes and properties for the organization of terms. Specific traffic properties (such as the traffic flow or speed) are not explicitly modeled in the proposed ontology.
- The *KM4City* [144] is an ontology for smart cities developed by the University of Florence (Italy) as a support for a platform that collects and integrates data related to the Tuscany region in Italy. It includes concepts regarding streets (Road, Node, RoadElement, AdministrativeRoad, Milestone, StreetNumber, RoadLink, Junction, Entry, EntryRule, Maneuver, Lanes, and Restriction), local public transportation (Ride, Route, RouteSection, BusStop, etc.), and sensors of traffic and different types of events (e.g., SensorSite, TrafficObservation, TrafficSpeed, TrafficConcentration, TrafficHeadway, etc.).
- Finally, some ontologies support modeling energy consumption data. Although they are not explicitly focused on traffic, they could be used as an input for traffic estimation. On the one hand, the *Smart Appliances REFERENCE (SAREF)* ontology [145] allows the representation of information related to devices (e.g., a washing machine, a temperature sensor, etc.) in a smart appliances domain as well as their functions and profiles (e.g., for energy optimization). On the other hand, the *FIEMSER ontology* [146] models the organization of building spaces (using concepts such as Building, BuildingPartition,

BuildingSpace, and BuildingZone) and the devices used in the building (defining concepts such as Device, HomeEquipment, ControlledDevice, and also more specific types such as Boiler and Radiator). Based on data provided by smart appliances, it could be possible to estimate the occupancy levels in households and buildings and thus indirectly estimate information about the traffic of vehicles outside (e.g., expected traffic variations along the day).

Summing up, for the moment, there is no comprehensive working traffic ontology extensively being applied. However, the *Vocabulary to Represent Data About Traffic Ontology* [129] commented above is very promising and can be easily extended to include all the elements that may be needed for traffic monitoring, mainly when used in conjunction with other ontologies, such as an ontology for road maps and a weather ontology to represent the weather conditions affecting the traffic observed. The KM4City ontology is particularly relevant for the purposes described in this chapter. The approach used for the semantic representation of traffic data is described in Section 6.2.

6.2 Data annotation and publishing

Within the TRAF AIR project, the conversion of the data about traffic sensors and the measurements they take over a long period into Linked Data was one of the scope. The approach implemented for this purpose is shown in Figure 6.1 and will be detailed in this section. Data related to the general information about the sensors and their measurements are stored into the TRAF AIR database. The tool Karma [147, 148] takes these data as input in CSV format and transforms it into Linked Data by using appropriate ontologies and by exploiting the Linked Geo Data ontology [149] for mapping the information of OpenStreetMap. The Linked Data produced by Karma is automatically loaded into a SPARQL endpoint, which is queried by the visualization tool called Lodview [150].

In this section, the process that starts with the data storing into the operational database and progresses until the production of the Linked Open Data is described. Firstly, Section 6.2.1 focuses on the identification of the relevant concepts and properties. Secondly, Section 6.2.2 describes the process followed for data integration. Thirdly, Section 6.2.3 explains the implementation of a SPARQL endpoint by using Virtuoso [151, 152] and how Lodview [150] has been used to make the data available online. In the end, Section 6.2.4 summarizes the technological choices made and other potential alternatives.

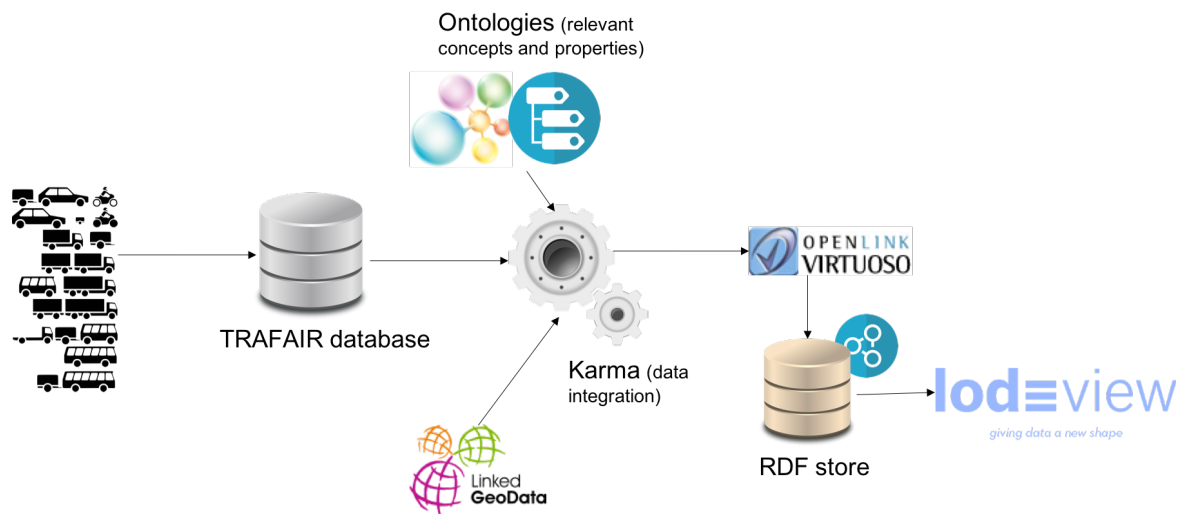


Fig. 6.1 Overview of the mappings and tools used for data annotation and publishing.

The credit of some images used has to be accredited to the creators of the software mentioned and used here for illustration purposes, others are freely available images extracted from Pixabay. The logo of Virtuoso has been extracted from <https://commons.wikimedia.org/wiki/File:Virtuoso-logo-sm.png>, contributed by Deirdre Gerhardt.

6.2.1 Identification of relevant concepts and properties

As reported in Section 6.1.1, an extensive research looking for already-existing traffic-related ontologies and vocabularies has been undertaken. At the end of this process, some of these ontologies have been selected to annotate the traffic concepts. Since no single ontology fits our needs perfectly, a combination of concepts defined in different ontologies has been used. Moreover, in some cases, it was necessary to create new classes and properties, since the available definitions were not suitable.

For the “sensor_traffic” entity type, described in Section 3.3.2 and depicted in Figure 3.2, some definitions of the Km4City ontology have been used to annotate the content of this table. In particular, the class *km4c:SensorSite* (*Traffic Sensor*) is used to identify the sensor capable of observing the traffic and the speed of the vehicles, the property *km4c:hasGeometry* to specify the point where the sensor is located, the class *km4c:Road* with the property *km4c:placedOnRoad* to refer to the name of the road where the sensor is located, the property *km4c:type* to identify the type of the traffic sensor, and the property *km4c:direction* for the direction of the vehicles counted by the sensor. It is important to specify that, to better link our data, all the streets present in our database are transformed into instances of the class *km4c:Road* giving them a URI that is, simply, the concatenation of the strings *https://trafair.eu/* and the street name. In addition, the Basic Geo (WGS84 lat/long) Vocabulary [153] has been used to represent the latitude (*geo:lat*) and the longitude (*geo:long*) of the

above-mentioned point. The Basic Geo Vocabulary is a basic RDF vocabulary that provides the Semantic Web community with a namespace for representing the latitude, longitude, and other information about spatially-located entities, using WGS84 as the reference datum.

Furthermore, the Linked Geo Data ontology has been exploited to transform the “road_section” attribute of the “sensor_traffic” table into linked data, since this attribute contains the identification number of a way in OSM and the Linked Geo Data ontology makes the information collected by the OSM project available as an RDF knowledge base according to the Linked Data principles [117]. In Linked Geo Data, the ways of OSM are dereferenceable objects at the link <http://linkedgedata.org/triplify/wayOSMID>, where *OSMID* is the identifier of the way.

The content of the “road_section” attribute has been concatenated to the link <http://linkedgedata.org/triplify/way>; therefore, if the value of the “road_section” attribute is “387989963”, then it becomes <http://linkedgedata.org/triplify/way387989963>, which is a dereferenceable object in the Linked Geo Data Knowledge Base [149]. The same approach has been used for the attribute “nearest_node”, adding the link <http://linkedgedata.org/triplify/node> to the identifier of the node, which is another element of OSM. Two new properties have been defined to connect a sensor to its way and its nearest node, called *trafair:isLocatedInOSMWay* and *trafair:hasNearestOSMNode*, respectively (see Figure 6.2). Moreover, the name of the city where the sensor is located has been added. The city has been identified by exploiting the class *dbo:Place* of the DBpedia Ontology [154], and the property *dbo:Location* to link the instance of the class *dbo:Place* to the traffic sensor.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .

<http://trafair.eu/ontology/hasNearestOSMNode>
  a rdf:Property, owl:ObjectProperty ;
  rdfs:domain geo:SpatialThing ;
  rdfs:range <http://linkedgedata.org/meta/Node> .

<http://trafair.eu/ontology/isLocatedInOSMWay>
  a rdf:Property, owl:ObjectProperty ;
  rdfs:domain geo:SpatialThing ;
  rdfs:range <http://linkedgedata.org/meta/Way> .
```

Fig. 6.2 Definition of the “isLocatedInOSMWay” and the “hasNearestOSMNode” properties.

Concerning the “sensor_traffic_observation” entity type of Figure 3.2, the Km4City ontology has been exploited. Each instance of this entity type represents an observation made by one sensor. An instance of the class *km4c:Observation* identifies the observation. Since the primary key of the corresponding “sensor_traffic_observation” table is composed of three attributes (identifier of the sensor, timestamp indicating the beginning of the observation, and type of vehicles observed), the URI of each observation has been created as the concatenation of the values of these attributes. The property *km4c:measuredBySensor* has been used to connect the observation to the sensor, while the properties *km4c:vehicleFlow* and *km4c:averageSpeed* have been exploited to indicate the number of vehicles and their average speed, respectively. The property *rdfs:label* has been used to represent the type of vehicles, which is enough for representing the names of the types of vehicles; as an alternative, an existing ontology of vehicles, like the *Vehicle Ontology* or the *Vehicle Sales Ontology*, could have been extended.³

The last two attributes hold the start and the end of the time interval of the observations. Different kinds of sensors send measurements with different time intervals, while defining how to aggregate the original data when sharing this information is up to the public administration that owns the data. The second attribute is a new attribute calculated over the first attribute adding the time interval the observation refers to. These attributes have been mapped using the *prov:startedAtTime* and *prov:endedAtTime* properties of the PROV-O ontology [155, 156].

The structure of the URIs implemented is the following:

- Instances of the class *km4c:Road* have the following URI structure: *https://trafair.eu/road/«city»/«road_name»*. It is the concatenation of the strings *https://trafair.eu/road*, the name of the city, and the road name (e.g., *https://trafair.eu/road/modena/Viale_Italia*)
- Instances of the class *km4c:SensorSite* have the following URI structure: *https://trafair.eu/sensor/«city»/«sensor_code»*. It is the concatenation of the strings *https://trafair.eu/sensor*, the name of the city, and the identifier of the sensor (e.g., *https://trafair.eu/sensor/modena/LP1*)
- Instances of the class *km4c:TrafficObservation* have the following URI structure: *https://trafair.eu/observation/«city»/«sensor_code»/«vehicle_type»/«end_date_of_the_observation»*. It is composed by the concatenation of the following items: the string

³It should be noticed that the *Vehicle Ontology* available at <https://enterpriseintegrationlab.github.io/icity/Vehicle/doc/index-en.html> does not currently define subclasses of the concept “Vehicle” and that the *Vehicle Sales Ontology* available at <http://www.heppnetz.de/ontologies/vso/ns> does not cover all the types of vehicles managed in TRAFair. However, new concepts can be added as needed.

https://trafair.eu/observation, the name of the city, the identifier of the sensor, the type of vehicles observed, and the timestamp indicating the beginning of the observation (e.g., *https://trafair.eu/lodview/observation/modena/LP1/autobus/2019-03-04T15:00:00*)

In conclusion, a new data model, based on the identification of the above explained concepts and properties, has been defined. The definition of the final data model is depicted in its entirety in Figure 6.3 and can be downloaded from <https://trafair.eu/trafair-model>. The ShEx schema is serialized using Shape Expressions Compact Syntax or ShExC⁴.

Figure 6.4 shows the exemplar triples generated for a sample traffic observation.

6.2.2 Data integration

Karma [147, 148] has been selected as the tool for representing the traffic data provided by the sensors in Linked Data. The goal is to map data stored in the TRAFAIR database by using the selected classes and properties described in Section 6.2.1. Karma is a data integration tool developed by the University of Southern California. It is an ETL (Extract, Transform, Load) tool [157, 158] which is capable of 1) retrieving data from different data sources such as files, some Relational Database Management Systems (RDBMS –MySQL, Microsoft SQL Server, Oracle, and PostgreSQL with PostGIS–), and various API services, 2) applying several kinds of transformations to the data, such as adding columns to the dataset and renaming columns, and then 3) providing the output file containing data transformed into the RDF format. This last operation is the most time-consuming one because the user has to select and load the ontologies he/she wants to use and then map each attribute of the dataset to the most appropriate class/property of the selected ontologies. This assumes the identification of the ontologies of interest according to the type of data to map. After the mapping, it is possible to download the R2RML model, which has been created, and the RDF file. R2RML [159] is a standard language proposed by the W3C RDB2RDF Working Group for expressing customized mappings from relational databases to RDF datasets. The R2RML model contains the mappings as RDF graphs which are written down in Turtle syntax. The R2RML model can be used to execute Karma in batch mode to generate RDF for large datasets and automate the transformation process. In this way, the user is not required to map the ontologies over the data every time he/she wants to make some transformation. Also in our scenario, this was the approach adopted.

⁴This is one of the possible serializations of ShEx; please, see <https://shex.io/shex-primer/> for additional information.


```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX km4c: <http://www.disit.org/km4city/schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX trafair: <https://trafair.eu/ontology/>

<Sensor> {
    km4c:type xsd:string,
    geo:lat xsd:float,
    geo:long xsd:float,
    km4c:hasGeometry xsd:string,
    km4c:direction xsd:boolean,
    rdf:type km4c:SensorSite,
    dbo:location IRI,
    km4c:placeOnRoad @<Road>,
    trafair:hasNearestOSMNode IRI,
    trafair:isLocatedInOSMWay IRI
}

<Observation> {
    rdf:type km4c:TrafficObservation,
    km4c:measuredBySensor @<Sensor>,
    rdfs:label xsd:string,
    prov:startedAtTime xsd:dateTime,
    prov:endedAtTime xsd:dateTime,
    km4c:averageSpeed xsd:string,
    km4c:vehicleFlow xsd:string
}

<Road> {
    rdf:type km4c:Road
}
```

Fig. 6.3 Data Model described though Shape Expressions.

```

@prefix km4c: <http://www.disit.org/km4city/schema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<https://trafair.eu/observation/modena/LP1/moto/2019-03-03T01:15:00>
  km4c:measuredBySensor <https://trafair.eu/sensor/modena/LP1> ;
  rdfs:label "moto" ;
  a km4c:TrafficObservation ;
  prov:startedAtTime "2019-03-03T01:00:00"^^xsd:dateTime ;
  prov:endedAtTime "2019-03-03T01:15:00"^^xsd:dateTime ;
  km4c:averageSpeed "94.2"^^xsd:string ;
  km4c:vehicleFlow "2"^^xsd:string .
    
```

Fig. 6.4 Triples generated for a sample traffic observation in Turtle syntax.

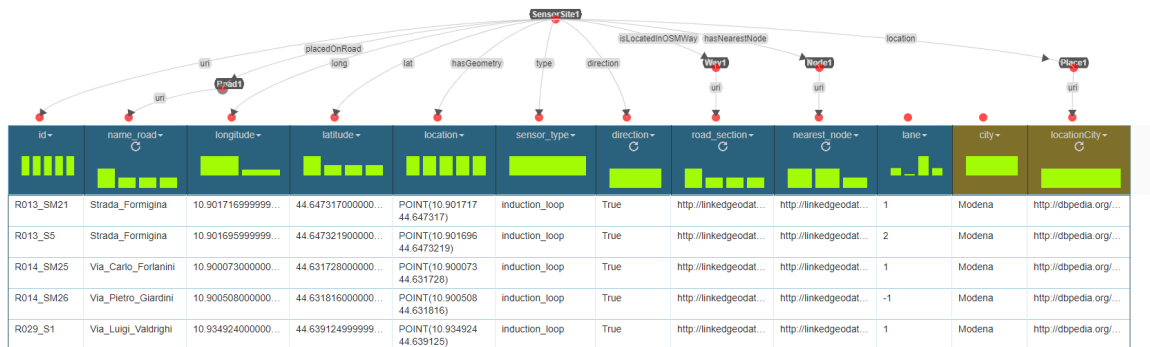


Fig. 6.5 Karma model implemented to transform data from the “sensor_traffic” table (Fig-ure 3.2) into Linked Data.

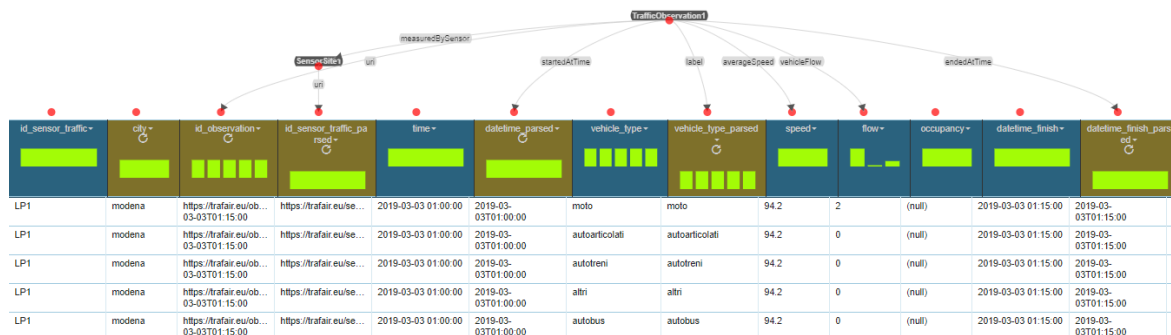


Fig. 6.6 Karma model implemented to transform data from the “sensor_traffic_observation” table (Figure 3.2) into Linked Data.

The graphical user interface of Karma has been used to create two models: one for each table. In Figure 6.5, a graphical representation of the model used to map the attributes of the “sensor_traffic” table is provided, while the mapping in Figure 6.6 is related to the “sensor_traffic_observation” table. Karma requires some input data to create the model; however, it is not necessary to develop the model importing the whole amount of data from the table. For this reason, initially, some exemplar tuples from the corresponding tables of the TRAFair database have been imported by configuring the connection to the database and using the appropriate query. Then, the selected ontologies have been uploaded and each attribute has been manually mapped to the suitable class/property. Once the mapping is concluded, the R2RML models that can be applied to a larger dataset were downloaded by using the Karma RDF Generation Service [160]. This service allows generating RDF data and publishing it on a SPARQL Endpoint. The information required for the transformation is the path of the file containing the data to be transformed (allowed formats are CSV, JSON, XML, and Excel), the URI of the R2RML model, the SPARQL endpoint, and the graph URI where the RDF data will be published.

6.2.3 Data publication and exploitation

Once the data transformation process is over, the Open Data are ready to be published. There are two main ways of publishing Linked Data on the Web: through a data dump or on a SPARQL endpoint.

On the one hand, a data dump places all dataset triples in one or more archive files. Dumps need to be downloaded entirely before they can be queried. This might be a problem since dump files can have large sizes (e.g., in our context, one year of sensor observations at 1-hour granularity takes about 3.5 GB). For this to be manageable, some policies must be established to define a suitable period granularity of each data dump (e.g., one per month, one per year, etc.). Moreover, keeping the data in each data dump up-to-date requires effort. With a solution based on data dumps, the users can download the entire dump after every update or download and apply incremental patches.

On the other hand, a SPARQL endpoint lets clients evaluate any desired (read-only) query on a server. This gives clients direct access to (only) the data they are interested in. Thus, only very little bandwidth is required, and the data is always up-to-date and can be flexibly queried.

The choice of publishing data on a SPARQL endpoint has been selected as the most convenient one. The possibility of providing data dumps was discarded because of the difficulty to foresee the needs of the users, which would be needed to set the appropriate granularity and criteria that should be considered to generate and keep the available data


dumps up-to-date. Nevertheless, data dumps could also be provided in cases where specific types of needs are foreseen, in order to cover those specific needs, along with the SPARQL endpoint for more flexible querying; complementing the SPARQL endpoint with some data dumps to cover the expected popular needs of users could be useful particularly for non-technical users with no knowledge of SPARQL.

A SPARQL endpoint is a service that can be queried in real-time and allow further data processing, for example, for traffic management systems. The RDF repository can be explored through SPARQL queries, which allow the users to express their data needs in a precise way. In this way, the user can generate his/her own data dumps (by submitting queries), adapted to his/her specific needs, rather than being forced to select among a predefined list of data dumps generated according to specific granularities and criteria set in advanced. The SPARQL endpoint is available at <https://trafair.eu/sparql>; since traffic data are not published by the City of Modena, data shared through the SPARQL endpoint are currently example data.

For the implementation of the RDF repository, Virtuoso [151, 152] has been chosen because it combines the functionality of a triple store and a SPARQL endpoint and it offers a user interface for querying the underlying data store. Indeed, every user can query the dataset in the way he/she needs, looking for the required information by using the SPARQL interface and thanks to the expressiveness of the SPARQL query language. After the installation of the tool, a new named graph called “trafair” has been created. It is possible to upload the files containing the data transformed by Karma through the graphical interface of Virtuoso. However, the Karma RDF Generation Service has been used since it allows automating the transformation and publication of the RDF data to the “trafair” graph.

A limitation of Virtuoso is that it is not able to visualize the information of a particular subject unless this is done through an ad-hoc query. This task can be particularly hard for non-skilled users. To overcome this limitation, a visual tool was installed alongside Virtuoso, called LodView [150]. LodView is a JSP web application able to offer a W3C standard compliant IRI dereferenciation. It is a Linked Data visualization tool that shows, in a tabular layout, the information of a resource, given its URI. LodView improves the end user’s experience in accessing HTML-based representations of RDF resources. In particular, it splits the information into four sections: 1) a header, containing associated images (if any), audios, videos, and a map if the resource has latitude and longitude properties; 2) the main section, that contains all the information related to the resource; 3) a section that contains all the inverse relations that LodView was able to gather from the underlying endpoint; and 4) a section that includes data related to the instances connected to the resource through the *owl:sameAs* property.

https://trafair.eu/sensor/modena/R001_SM3
<http://www.disit.org/km4city/schema#SensorSite>



geo:lat	44.6368	xsd:float
geo:long	10.9482	xsd:float
<http://www.disit.org/km4city/schema#hasGeometry>	POINT(10.948232 44.636821)	
<http://www.disit.org/km4city/schema#type>	induction_loop	
<http://www.disit.org/km4city/schema#direction>	True	
rdf:type	<http://www.disit.org/km4city/schema#SensorSite>	
dbpedia-owl:location	dbpedia:Modena	↑
<http://trafair.eu/ontology#hasNearestNode>	<http://linkedgeodata.org/triplify/node2052719336>	↑
<http://trafair.eu/ontology#isLocatedInOSMWay>	<http://linkedgeodata.org/triplify/way495300232>	↓
<http://www.disit.org/km4city/schema#placedOnRoad>	<https://trafair.eu/road/modena/Via_Giuseppe_Campi>	

RELAZIONI INVERSE

è <http://www.disit.org/km4city/schema#measuredBySensor> di 61679 risorse

Fig. 6.7 LodView representation of the station information of the traffic sensor R001_SM3.

LodView, in conjunction with a SPARQL endpoint, allows publishing RDF data according to all the defined standards for Linked Open Data. Once the user clicks on an RDF resource (for example, a URI extracted from a SPARQL query), LodView queries the dataset looking for all the information related to that specific resource and displays the data to the user. Our installation of Lodview is available at <https://trafair.eu/lodview>.

An example of the use of LodView is shown in Figure 6.7. Here the information of the traffic sensor “R001_SM3” is described: the position of the sensor is visualized on a map while other information is displayed as property-object pairs. At the bottom of the figure, the inverse relationships are listed. Figure 6.8 shows one of these relationships, which represents the observation made by the sensor “R001_SM3” on the 6th of August 2019 from 19:00 to 20:00.

all_vehicles		
https://trafair.eu/observation/modena/R001_SM3/all_vehicles/2019-08-06T19:00:00		<http://www.disit.org/km4city/schema#TrafficObservation>
rdfs:label	all_vehicles	
prov:startedAtTime	2019-08-06T18:00:00	xsd:dateTime
prov:endedAtTime	2019-08-06T19:00:00	xsd:dateTime
<http://www.disit.org/km4city/schema#averageSpeed>	17.4375	xsd:string
<http://www.disit.org/km4city/schema#vehicleFlow>	115	xsd:string
rdf:type	<http://www.disit.org/km4city/schema#TrafficObservation>	
<http://www.disit.org/km4city/schema#measuredBySensor>	<https://trafair.eu/sensor/modena/R001_SM3>	

Fig. 6.8 LodView representation of one observation made by the traffic sensor R001_SM3.

```

select ?p ?o
where {
  <https://trafair.eu/sensor/modena/R001_SM3> ?p ?o .
}

```

p	o
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.disit.org/km4city/schema#SensorSite
http://www.w3.org/2003/01/geo/wgs84_pos#lat	44.6368
http://www.w3.org/2003/01/geo/wgs84_pos#long	10.9482
http://dbpedia.org/ontology/location	http://dbpedia.org/resource/Modena
http://trafair.eu/ontology/hasNearestNode	http://linkedgeodata.org/triplify/node2052719336
http://trafair.eu/ontology/isLocatedInOSMWay	http://linkedgeodata.org/triplify/way495300232
http://www.disit.org/km4city/schema#hasGeometry	"POINT(10.948232 44.636821)"
http://www.disit.org/km4city/schema#placedOnRoad	https://trafair.eu/road/modena/Via_Giuseppe_Campi
http://www.disit.org/km4city/schema#type	"induction_loop"
http://www.disit.org/km4city/schema#direction	"True"

Fig. 6.9 SPARQL query showing information related to sensor R001_SM3.

In Figure 6.9, a simple example of a query performed on Virtuoso is shown. This query shows the data related to one traffic sensor and contains the same data shown in Figure 6.7.

As of Virtuoso 7.1, several improvements have been made to integrate the support of geospatial queries. Indeed, Virtuoso can understand representations for several types of geometric objects (points, linestrings, multilinestrings, polygons, multipolygons, and geometry collections). Besides, it supports several functions with geospatial objects, increasing compliance with GeoSPARQL and OGC (Open Geospatial Consortium) standards. Figure 6.10 shows an example of a GeoSPARQL query, which selects the number of vehicles counted by the sensors placed in the town square of Modena on January 8th, 2019.

The reader can also find other examples of possible SPARQL queries in Section 6.3. In particular, the query in Figure 6.11 counts how many sensors in our data store are located in the two cities, and the one in Figure 6.12 shows the number of vehicles counted by every sensor in our data store during a specific day. It is also possible to filter the sensors considered according to the street where they are placed. An example of this filter is shown in the query in Figure 6.13, where the results are ordered by the position of the sensors (longitude and latitude). Figure 6.14 shows an example of a GeoSPARQL query that selects how many sensors in Modena are located within the area delimited by the ring road. The last two queries mentioned show the number of vehicles counted by the sensors on January 8th, 2019.

6.2.4 Technological choices

In this section, the reasons for our choice of technological solutions (Karma, Virtuoso, and LodView) are justified.

As explained in Section 6.2.2, Karma [147, 148] has been used to represent the traffic data provided by the sensors in Linked Data. Karma is a useful ETL tool that supports the publication of data in RDF format. Besides Karma, other alternatives could have been chosen to achieve this goal [161]. Thus, a variety of tools and languages can be used to convert from different data formats to RDF-like data formats, such as OpenRefine [162], RML [163], ShExML [164], YARRML [165], and SPARQL-Generate [165]. Any of these tools could have been used instead of Karma, without much impact on the work presented in this Chapter. The main reasons why Karma was chosen for this project are the following:

- Karma allows to import data from a variety of sources other than a PostgreSQL database, and therefore our approach can be exploited even if the input data are available in other types of sources.
- Karma allows to export the data model in R2RML format, which can be applied to transform a huge amount of data in RDF. Besides, the model can be easily shared

```

SELECT ?s xsd:double(?long) as ?long xsd:double(?lat) as ?lat
       sum(xsd:integer(?count)) as ?vehicle_count
WHERE {
  ?s a <http://www.disit.org/km4city/schema#SensorSite> ;
     geo:lat ?lat ;
     geo:long ?long .
  ?o <http://www.disit.org/km4city/schema#measuredBySensor> ?s ;
     <http://www.disit.org/km4city/schema#vehicleFlow> ?count ;
     <http://www.w3.org/ns/prov#startedAtTime> ?date ;
     <http://www.w3.org/ns/prov#endedAtTime> ?enddate .
  FILTER
    ( bif:st_intersects
      ( bif:st_geomfromtext
        ("POLYGON(
          (10.919688 44.649097,
           10.925297 44.650970,
           10.934204 44.648047,
           10.929151 44.639855,
           10.918631 44.642557)
        )"
        ), bif:st_point (xsd:double(?long), xsd:double(?lat))
      )
    )
  FILTER (xsd:date(?date) = "2019-01-08"^^xsd:date) .
}
group by ?s xsd:double(?long) xsd:double(?lat)
order by DESC(?vehicle_count)

```

s	long	lat	vehicle_count
https://trafair.eu/sensor/modena/R118_SM117	10.9293	44.6419	107765
https://trafair.eu/sensor/modena/R118_SM118	10.9293	44.6419	107764
https://trafair.eu/sensor/modena/R018_SM33	10.9191	44.6439	10185
https://trafair.eu/sensor/modena/R023_SM41	10.9235	44.6421	9085
https://trafair.eu/sensor/modena/R017_SM32	10.919	44.6427	8377
https://trafair.eu/sensor/modena/R017_SM31	10.9193	44.643	8023
https://trafair.eu/sensor/modena/R026_S2	10.9291	44.6403	6163
https://trafair.eu/sensor/modena/R020_SM35	10.9204	44.6422	5911

Fig. 6.10 GeoSPARQL query showing the vehicle count of some sensors located in the town square of Modena on January 8th, 2019.

with other researchers interested in our mapping to make the same transformation; the model is independent of the data sources. In [161], Karma is compared to other tools and it is the only one that supports exporting the R2RML model.

- Karma enables importing multiple ontologies in the same project. This feature is crucial in our case since a unique ontology which includes all the classes and properties needed was not available.
- Karma offers a batch mode procedure that can be exploited for automating the conversion process given the R2RML model and a set of similar data sources. Also, it is able to interact with a Virtuoso instance and directly load the RDF data into a Virtuoso instance instead of using RDF files.

Similarly, Virtuoso [151, 152] is being used as a SPARQL endpoint (see Section 6.2.3). Alternatives that could be considered include AllegroGraph [166] or RDF4J [167], as well as graph databases such as Neo4J [168], Titan [169], GraphDB [170], or Stardog [171], to cite some examples. A benchmarking framework to evaluate and compare different data management solution for RDF and property graph data models, called LITMUS, has been proposed in [172]. Virtuoso is very popular in academia, which motivated our use in the project. Besides, according to the preliminary experimental evaluation presented in [172], where several solutions are compared (4Store, Jena, Neo4J, OrientDB, RDF3X, Sparksee, Tinker, and Virtuoso), Virtuoso achieves the best results overall (the best performance in terms of the loading time as well as regarding the cold cache execution time, and the second best one concerning the warm cache execution time). Some relevant benefits of Virtuoso include the following:

- Virtuoso is a popular tool that exposes a SPARQL endpoint for performing SPARQL queries, thus covering our fundamental need.
- Karma provides functionalities for operating with instances of Virtuoso. So, these two tools complement each other and can be easily used in conjunction.
- Virtuoso provides an open source version that is constantly being updated and improved.
- It features a backend authentication system which supports setting different privileges for different users. In this way, it is possible, for example, to block potential DELETE statements that can be sent from the Internet.

In the end, as visualization tool for RDF, in this work LodView [150] has been used (see Section 6.2.3). Other possible alternatives include Rhizomer [173, 174], LODMilla [175, 176], and LODGVis [177, 178], to cite some examples. LodView was chosen because it provides a dereferenciation system that allows the users to easily explore the relation between different instances. Some relevant benefits of LodView include the following:

- It is open-source and can be easily customized.
- It provides a simple and tabular visualization that is easy to understand.
- It is able to navigate and display the resources connected through the *owl:sameAs* relation.
- It is able to navigate and display inverse relations.
- It provides a connection with LodLive [179]. Therefore, our resources can also be visualized through the online version of LodLive, since it is able to explore the resources of a remote SPARQL endpoint. By exploiting the online version of LodLive, it was not necessary to set up a personalized instance.

Other works have focused on comparing different approaches. For example, the analysis of approaches to generate RDF data from relational data has been the subject of several studies.

Several surveys on RDF data storage/management approaches and technologies have been presented [180–184]. Finally, several surveys on visualization tools for RDF data have been published [185–188]; from these, the most recent survey is [188], where 77 linked data visualization tools have been analyzed.

6.3 Experimental evaluation

The sensors and observations R2RML models, created with Karma, have been successfully applied over traffic data of Modena and Zaragoza. The following statistics refer to a Debian 9 machine with 32 Intel(R) Xeon(R) Silver 4108 CPU at 1.80GHz processors and 64GB of RAM.

For the city of Zaragoza, in these experiments, the data of 46 static devices have been considered: the information relating to those devices resulted in 506 triples. The process for generating and loading the triples took less than 1 second. Sensors' data for the city of Modena corresponded to data about 400 traffic sensors, which resulted in 4400 triples.

Table 6.1 Sensor data performance evaluation: loading time.

City	#Sensors	#Triples	Loading Time
Zaragoza	46	506	~0.75 sec
Modena	400	4400	~5 sec

The process for generating and loading the triples took about 5 seconds. A summary of the performance of the loading process of the information about sensors is provided in Table 6.1.

Statistics related to the publication of traffic observations in the two cities are displayed in Table 6.2. Each traffic observation is transformed into a set of 7 RDF triples following the approach described in Section 6.2.1. The loading process is the entire process that loads the data into Virtuoso through Karma: a query is executed on the database, the extracted data is stored in a CSV file, and the file is processed by Karma that loads the data in Virtuoso. Each Zaragoza’s traffic sensor generates 1 observation every hour, and therefore a total of 24 observations in a day. Considering that all the sensors are working correctly, more than 1000 observations are gathered daily. From January 2019 to December 2019, the number of observations arises to 383 thousands for a total of 2.5 million triples. The triple generation and loading process took about 1.5 minutes. This information is reported in the first row of Table 6.2.

The situation in Modena is quite different due to a higher number of sensors and observations’ rate. Indeed, in Modena, 1-minute observations, which means about 1440 measurements for each sensor in a day, are gathered. In order to compare the statistics of Modena and Zaragoza, the data have been aggregated hourly. Some of the sensors in Modena collect more fine-grained data, since they can recognize up to 10 types of passing vehicles; on the other hand, not all of the sensors provide one measure per minute. Therefore, about 17500 hourly observations are gathered daily. From January to December 2019, the total number of observations arises to 6.5 million observations for a total of over 46 million of triples. This information is reported in the second row of Table 6.2.

Table 6.2 Observation data performance evaluation: loading time of hourly observations from January to December 2019

City	#Sensors	Period	#Observations	#Triples	Loading Time
Zaragoza	46	2019/01/01 - 2019/12/31	383K	2.5M	1.5 min
Modena	400	2019/01/01 - 2019/12/31	6.5M	46M	1 hour

Some tests on the loading process, to understand the capabilities of Virtuoso and Karma, the scalability of the loading process, and the variation of loading time in diverse configurations, have been performed. Traffic data generated in Modena, that are fine-grained and

can be further aggregated, have been exploited. The loading process has been tested using different granularities (1-hour aggregated observations, 15-minutes aggregated observations, and 1-minute observations) and different window lengths, i.e., the time period of data to load in one iteration (1 day, 12 hours and 3 hours). 1-hour granularity data is usual in some Open Data initiatives (e.g., the City Council of Zaragoza currently publishes pollution data with 1-hour granularity and traffic and mobility indicators with granularity not smaller than 1 hour); however, tests with lower granularity data have been performed in order to stress the system. The aim of these tests is to compare the loading time and performance. Selecting suitable granularities and window lengths for the traffic observations that have to be shared as Linked Open Data belongs to the public administration which owns the data. Currently, traffic data are not published by the city of Modena; therefore these tests are executed on real data, while the data that are shared through the SPARQL endpoint is example data (random data).

As reported in Table 6.3, in a generic day, in Modena, about 17500 1-hour observations (first row, forth column in the table), 70000 15-minutes observations (second row, forth column in the table) and 430000 1-minute observations (forth row, forth column in the table) are collected. A procedure for transforming and loading data has been deployed. This procedure divides the data in windows of 1-day length so that 365 iterations (loading stages) are needed to load the data of the whole year. In this context, hourly data were successfully handled and the time needed to upload the observations of the whole year was about 1 hour (~ 10 seconds * 365 iterations), as shown in the table. Due to scalability issues with the tool used, 15-minutes data and 1-minutes data cannot be loaded by following this approach. In Table 6.3, a loading process is denoted by “failure” in case one iteration does not end successfully within 3 hours.

The table also shows that, if the length of the temporal window is reduced to 12 hours, 15-minutes aggregated data of the whole year can be loaded in approximately 6 hours (~ 30 seconds * 730 iterations); every iteration loaded approximately 35000 observations. Due to scalability issues, 1-minute aggregated data cannot be handled when a 12-hours interval is adopted; nevertheless, by considering a window length of 3 hours, every iteration has to handle approximately 54000 observations and 1-minute data can be loaded in about 36 hours (~ 45 seconds * 2920 iterations). For more details, please see Table 6.3, which summarises the loading time of each option.

Table 6.3 Loading process performance for 1-year data under different granularity and window length conditions.

Granularity of Data	Window Length	#Iterations Required	#Observations for Each Iteration	#Generated Triples	Loading Time of a Single Iteration	Total Time	Result
1-h data	1 day	365	17,500	122.5K	14 s (avg)	1.25 h	success
15-min data	1 day	365	70,000	490K	-	-	failure
15-min data	12 h	730	35,000	245K	30 s (avg)	6 h	success
1-min data	1 day	365	430,000	3M	-	-	failure
1-min data	12 h	730	215,000	1.5M	-	-	failure
1-min data	3 h	2920	54,000	378K	45 s (avg)	36 h	success
1-min data	1 h	8760	18,000	126K	14 s (avg)	34 h	success
1-min data	1 min	525,600	200	1400	0.375 s (avg)	55 h	success

After these tests, it can be noticed that the loading process was not able to manage more than 54000 observations in a single step due to scalability issues.

However, this fact should not be considered a big issue. In fact, in case a City Council decides to publish fine-grained data (1-minute or 15-minutes data), it is likely that this data will be shared in semi-real time, therefore with a window length even less than 3 hours, so loading problems will not occur. Regarding the publication of historical data (e.g., data of the last year, or last month, every year or every month), carrying out the process with a time window of 3 hours is not a limitation.

Table 6.4 SPARQL queries response time.

Query	Short Description	Response Time	Notes
Query Fig. 6.9	Data of the sensor "R001_SM3"	300 msec	
Query Fig. 6.10	Num. of vehicles counted by sensors in Modena's square	2.6 sec	GeoSpatial
Query Fig. 6.11	Num. of sensors in each city	750 msec	
Query Fig. 6.12	Num. of vehicles counted by each sensor in the datastore	26.4 sec	
Query Fig. 6.13	Num. of vehicles counted by sensors on a street	1.86 sec	
Query Fig. 6.14	Num. of sensors within the ring road in Modena	650 msec	GeoSpatial

```
select count(?s) as ?sensors, ?o as ?city
where { ?s <http://dbpedia.org/ontology/location> ?o . }
group by ?o
```

sensors	city
46	http://dbpedia.org/resource/Zaragoza
400	http://dbpedia.org/resource/Modena

Fig. 6.11 SPARQL query showing the number of sensors in Modena and in Zaragoza.

```

select ?s sum(xsd:integer(?count)) as ?count
where {
  ?s a <http://www.disit.org/km4city/schema#SensorSite> .
  ?o <http://www.disit.org/km4city/schema#measuredBySensor> ?s .
  ?o <http://www.disit.org/km4city/schema#vehicleFlow> ?count .
  ?o <http://www.w3.org/ns/prov#startedAtTime> ?date .
  FILTER (xsd:date(?date) = "2019-01-08"^^xsd:date) .
}
order by ?s

```

s	count
https://trafair.eu/sensor/modena/LP1	14521
https://trafair.eu/sensor/modena/LP10	13278
https://trafair.eu/sensor/modena/LP11	12716
https://trafair.eu/sensor/modena/LP12	11892
https://trafair.eu/sensor/modena/LP13	0
https://trafair.eu/sensor/modena/LP14	8145
https://trafair.eu/sensor/modena/LP15	7935
https://trafair.eu/sensor/modena/LP16	9305
https://trafair.eu/sensor/modena/LP17	13844
https://trafair.eu/sensor/modena/LP18	13544

Fig. 6.12 SPARQL query showing the number of vehicles counted by each sensor on January 8th, 2019 (only a fragment of the answer is shown).

```

select ?s xsd:float(?long) xsd:float(?lat)
       sum(xsd:integer(?count)) as ?vehicle_count
where {
  ?s a <http://www.disit.org/km4city/schema#SensorSite> .
  ?o <http://www.disit.org/km4city/schema#measuredBySensor> ?s .
  ?o <http://www.disit.org/km4city/schema#vehicleFlow> ?count .
  ?o <http://www.w3.org/ns/prov#startedAtTime> ?date .
  ?s <http://www.disit.org/km4city/schema#placedOnRoad>
     <https://trafair.eu/road/zaragoza/Ronda_Hispanidad> .
  ?s <http://www.w3.org/2003/01/geo/wgs84_pos#lat> ?lat .
  ?s <http://www.w3.org/2003/01/geo/wgs84_pos#long> ?long .
  FILTER (xsd:date(?date) = "2019-01-08"^^xsd:date) .
}
group by ?s xsd:float(?long) xsd:float(?lat)
order by xsd:float(?long) xsd:float(?lat)

```

s	long	lat	vehicle_count
https://trafair.eu/sensor/zaragoza/EP28.2	-0.897157	41.6274	17850
https://trafair.eu/sensor/zaragoza/EP28.1	-0.897144	41.6276	15673
https://trafair.eu/sensor/zaragoza/EP27.1	-0.885378	41.6243	19180
https://trafair.eu/sensor/zaragoza/EP27.2	-0.885373	41.6242	20720
https://trafair.eu/sensor/zaragoza/EP26.1	-0.866296	41.6317	17569
https://trafair.eu/sensor/zaragoza/EP26.2	-0.866257	41.6316	17269
https://trafair.eu/sensor/zaragoza/EP25.1	-0.858465	41.64	16439
https://trafair.eu/sensor/zaragoza/EP25.2	-0.858313	41.64	16311
https://trafair.eu/sensor/zaragoza/EP24.1	-0.854242	41.6598	11848
https://trafair.eu/sensor/zaragoza/EP24.2	-0.854156	41.6599	10605

Fig. 6.13 SPARQL query showing the list of sensors located on the street named “Ronda Hispanidad” and the number of vehicles counted by these sensors on January 8th, 2019.

In the end, it could also be interesting to analyse the response time of the queries. Different SPARQL queries have been performed during the test phase and very fast responses have been obtained, even given the high number of observations stored in the endpoint. In particular, Table 6.4 contains some statistics related to the queries presented in Section 6.2.3 and the queries reported in Figure 6.11, 6.12, 6.13, and 6.14.

```

SELECT COUNT(?s) AS ?sensors_within_ringroad
WHERE {
  ?s a <http://www.disit.org/km4city/schema#SensorSite> .
  ?s geo:lat ?lat .
  ?s geo:long ?long .
  FILTER
    (bif:st_intersects
      (bif:st_geomfromtext
        ("POLYGON(
          (10.911314 44.671688, 10.902193 44.665496,
          10.888511 44.664096, 10.869854 44.657755,
          10.865605 44.650308, 10.886023 44.618224,
          10.928934 44.599777, 10.963760 44.647728,
          10.920020 44.672499, 10.911314 44.671688) )"
        ), bif:st_point (xsd:float(?long), xsd:float(?lat))
      )
    )
}
ORDER BY DESC(?sensors_within_ringroad)

```

sensors_within_ringroad
326

Fig. 6.14 GeoSPARQL query showing the number of sensors in Modena located inside the area delimited by Modena's ring road.

This approach has proven to be very effective in publishing Linked Data. The robustness provided by OpenLink Virtuoso (open source edition) allows efficiently managing large quantities of triples. Furthermore, its ability to manage geospatial data makes it a valuable tool for our purposes. To the best of our knowledge, Virtuoso open source edition does not support the creation of RDF views over external databases, so the Karma tool has been used in our pipeline to convert relational data into RDF triples. Moreover, an instance of LodView has been adopted to obtain graphical representations of the data hosted in the Virtuoso

endpoint. In the end, this completely open-source approach is well suited for handling a large amount of geospatial data and will be the base for further improvements.

Part II

Safety Solution

Chapter 7

Urban crime analysis from news streams

In most recent years, along with the concept of Smart City, the idea of Safe City is spreading. It is related to the strategies that aim to help the government in the development of a city security system for reducing the possibility of crime and providing an environment where people feel safe and comfortable. Safe city concept aims at reducing the crime problem and the violence in urban areas [189, 190].

The first step for the scope of Safe City should be the analysis of the crime events distribution in the involved city, i.e., where and when crimes occur, identifying the causes. Sound crime analysis is paramount to this success. Crime analysis [191] is not merely crime events counting; it is an in-depth examination of the different criminogenic factors (e.g., time, place, socio-demographics) that help understanding why the crime occurs. It consists of a set of systematic, analytical processes for providing timely and pertinent information relative to crime patterns and trend correlations to assist the police in crime reduction, prevention, and evaluation. Data-driven policing and associated crime analysis are still dawning. Focusing resources on high-crime places, high-rate offenders, and repeated victims can help police effectively reducing the crime rate in their communities. Police can take advantage of knowing when, where, and how to focus its resources, as well as how to evaluate the effectiveness of their strategies. The use of Geographic Information System (GIS) techniques in crime mapping [192, 193] helps crime analysis and allows localizing crimes to identify the high-risk areas.

If police reports are made public, they could be analyzed and geolocalized for the above-mentioned scopes, and made accessible to citizens. Several countries provide statistics on crime, but they are often available with some delay. The delay between the occurrence of the event and the report publication can reach some days, months or even years. In most cases, they are provided as aggregated data, not as single crime events. An example are the

Open Data about crime and policing of England, Wales and Northern Ireland. The data are collected by police force and made available at street-level with a delay of around 2 months.¹

Unfortunately, police reports are usually private documents. In other countries, authorities do not provide free access to updated datasets containing information about crimes happening in the cities. Therefore, although they are very useful documents, police reports cannot be considered a possible source for timely crime analysis for citizens. In those cases, newspapers are a valuable source of authentic and timely information. In Italy, police crime reports are not available to citizens, only some aggregated analyses are published yearly. The reports of the Italian National Institute of Statistics (ISTAT)² provide a clear picture of the types of crime happen in each province during the year. The information provided is aggregated by time and space and become available after (at least) one year from the crime event happening. Based on these official statistics, it is not possible to perform an up-to-date analysis of the local situation in each neighborhood. Two Italian cities provide updated crime datasets on their Open Data portals. Torino AperTO Open Data Portal includes information about the type of crime, the location and the date of the crime events occurred, but with a delay of two years³, while Trento provides the annual burglary rate of the previous year⁴. In both cases, data are not timely, and, in Trento, they are also aggregated.

Extracting crime events from news articles published on the web by local newspapers can help overcome the lack of crime up-to-date information. Indeed, several works concerning crime analysis exploit news articles [194–197, 194, 198, 196]. Detailed information about the crime events can be extracted through the application of Natural Language Processing (NLP) techniques to the news articles' text. Newspapers provide reliable, localized, and timely information (the time delay between the occurrence of the event and the publication of the news does not exceed 24/48 hours). The main drawback is that newspapers do not collect and publish all the facts related to crimes, but only the most relevant, i.e., the ones that arouse the interest of the readers. Therefore, there is a percentage of police reports that will not be turned into news articles and is lost.

The Italian Crime Analysis framework proposed in this thesis aims at extracting crime data from news articles, enriching them with semantic information and providing useful visualization. The strategy employs several techniques: crime categorization, named entity extraction, 5W+1H extraction through question answering, linked data mapping, geolocalization, time expression normalization, and de-duplication. The novelty of such a framework

¹On the police Open Data portal <https://data.police.uk/>, it was possible to download data about November 2021 on the 13th January 2022.

²<https://www.istat.it/en/>

³<http://aperto.comune.torino.it/>

⁴European Data Portal <https://www.europeandataportal.eu/it>

is the integration of multiple techniques, previously used in different contexts, for solving various sub-problems into a common framework for crime analysis. A java application integrates the different techniques of the framework. The framework has been applied and tested on news articles related to the Modena province (see Chapter 9). However, with the proper configuration it can be adapted to manage data of other cities.

In the following, Section 7.1 presents an overview of other crime analysis projects and the main approaches for news categorization and document similarity. The general approach proposed is described in Section 7.2.

7.1 Related work

In the last decades, methods for news content extraction have gained relevant interest [199, 200], and several online platforms have been developed to visualize the results of the extraction, such as the Europe Media Monitor (EMM) NewsExplorer⁵ and NewsBrief,⁶ the Thomson Reuters Open Calais,⁷ and the Event Registry.⁸ These platforms download news articles from the web and exploit different language processing algorithms to detect entities, group news articles into clusters according to their topics [201, 202]. In particular, a lot of scientific research is devoted to crime data mining, and new software applications have been created for detecting and analyzing crime data. In [203], an approach is described to extract important entities from police narrative reports written in plain text by using a SOM (self-organizing map) clustering method. Crime analysis methods are applied to find trends [204] or to predict crime events [205], by using neural networks, Bayesian networks, and algorithms as K-nearest-neighbour, boosted decision tree, K-means. An interesting example of crime analysis and mapping in the city of Chicago⁹ is explained in [206]. In this case, crime data are extracted from the Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting) system, composed of relational databases that allow law enforcement officials to cross-reference available information in investigations and to analyze crime patterns using a geographic information system (GIS). To protect the privacy of crime victims, addresses are shown at the block level only, and specific locations are not identified.

⁵<https://emm.newsexplorer.eu/>

⁶<https://emm.newsbrief.eu/>

⁷<https://www.crunchbase.com/organization/opencalais>

⁸<https://eventregistry.org/>

⁹<https://data.cityofchicago.org/Public-Safety/Crimes-Map/dfnk-7re6>

The main research areas of the Italian Crime Analysis Framework presented in this thesis are the categorization of news articles based on the type of crime described and the analysis of similarity among documents.

7.1.1 Crime categorization

The scope of news categorization is to assign a news article to a crime category. It can be addressed following several approaches, such as text classification, community or topic detection [207–210]. In the proposed framework, this problem is considered as a text classification task which consists of automatically assigning text documents to one of the predefined categories. Due to the information overload, this is a well-proven way to organize free document sources, to improve browsing or to identify related content by tagging content or products based on their categories. Newspaper articles are part of the increasing volume of textual data generated every day together with company data, healthcare data, social network contents, and others. News categorization can be useful to organize them by topic for generating statistics [211] or detect fake news [212–214].

Automatic text classification has been widely studied since 1960s and over the years different types of approaches to this problem have arisen. Recent surveys [215, 216] mainly distinguish between two categories of approaches: *conventional methods* (or *shallow learning methods*) and *deep learning-based methods*.

Conventional methods are those that need a pre-processing step to transform the raw text input into flat features which can be fed to a Machine Learning model. In the literature, there are several feature extraction techniques, such as term frequency (TF), term frequency-inverse document frequency (TF-IDF), N-grams, Bag-of-words and word embeddings. Among these, word embedding is one of the most recent text representations which is swiftly growing in popularity. Word embedding is a continuous vector representation of words that encodes the meaning of the word, such that the words that are closer in the vector space are supposed to be similar in the meaning. The use of word embeddings as additional features improves the performance in many NLP tasks, including text classification [217–225]. There are different Machine Learning algorithms that can be trained to derive these vectors, such as Word2Vec [226], FastText [227, 228], Glove [229].

In the last decade, deep neural network have been overcoming state-of-the-art results in many fields, including Natural Language Processing. This success relies on their capacity to model complex and non-linear relationships within data. This has led to an increasing development of deep learning-based methods also in text classification. They exploit many of the most known deep learning architectures, such as CNNs [230–232], RNNs [233, 234], LSTMs [235–237] and the most recent Transformers [238, 239]. Unlike conventional

methods, they do not need designing rules and features by humans, since they automatically provide semantically meaningful representations. These advantages involve a great deal of complexity and computational costs.

Many of the works regarding news categorization fall in the category of the conventional methods mentioned above. Keyword extraction, term frequency, document frequency, TF-IDF, POS tagging are mainly used as feature extraction methods along with the traditional Machine Learning models as classification methods, such as Naive Bayes, Decision Tree, Support Vector Machine or K-Nearest Neighbour [240–242]. There are some examples also for the Italian language [243–245]. However, none of them exploit word embeddings for feature extraction. In [246, 247] two multi-label classification approaches are described; in these works, the feature extraction methods leverage on topic modeling through Latent Dirichlet Allocation, which has the advantage to make text dimension reduced before getting the features. More recent works include the use of deep learning architecture, in particular CNNs [248–250] and BERT [251, 252].

In literature, there are very few works on the categorization of crime news articles. The authors of [253] proposed an approach for classifying Thailand online crime news involving TF-IDF as feature extraction method and tested six different Machine Learning algorithms for classification. The classifiers with the best results are Support Vector Machine and Multinomial Naive Bayes which reach an F-measure around 80%. In [236] better results (98.87% of accuracy) are achieved by using LSTM to classify Spanish news texts deriving the text representation from a pre-trained Spanish Word2Vec model.

A thorough search of the relevant literature did not yield works devoted to developing methods for the automatic classification of criminal and violent activities from Italian documents.

7.1.2 Document similarity

Calculating the similarity among textual documents may be useful for different goals. In the framework proposed in this thesis it is used to detect news articles related to the same crime events.

The literature contains plenty of works that propose similarity measures [254, 255, 255–257]. There are two types of similarity, i.e. lexical similarity and semantic similarity. In the first case, the similarity is calculated taking into account the sequence of characters in the compared texts; while in the second case a deeper analysis of the meaning of each word in the text is required, including the replacement of synonyms, the understanding of the context.

In [258], the authors classify the approaches used for text similarity in four classes: string-based, corpus-based, knowledge-based, and hybrid solutions. String-based algorithms

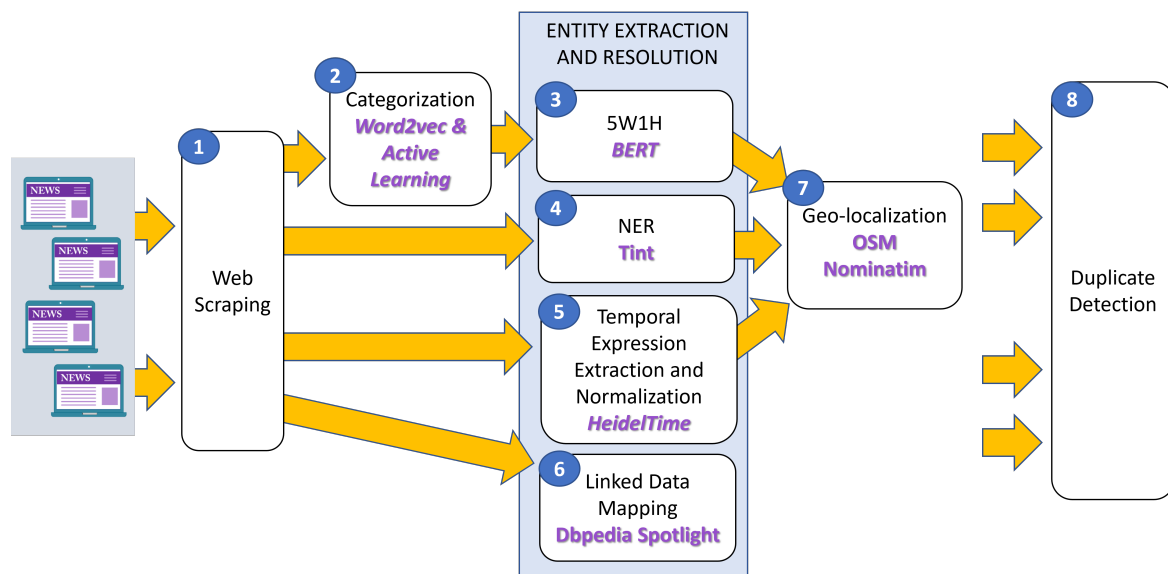


Fig. 7.1 The method implemented to extract, store, analyze the news articles (the numbers in the circles represent the phases described in Section 7.2).

focus on the lexical similarity and calculate a metric through string matching or comparison. Exemplar distance measures are the Euclidean distance, the Cosine Similarity, the Jaccard similarity, Jaro, the Damerau-Levenshtein, the Longest Common Substring. On the other hand, corpus-based and knowledge-based algorithms allows to calculate the semantic similarity. Popular solutions are the Latent Semantic Analysis [259], the Explicit Semantic Analysis [260], the Pointwise Mutual Information [261].

Besides, distance measures can be calculated on the text as it is or on a numerical representation, such as word embedding obtained by a specific model, like Word2Vec [226] or FastText [228]. The second section of this paper will examine the text representation, which represents the text as numerical features that can be calculated directly.

7.2 The proposed approach

The approach used by the framework to extract semantic information related to an event starting from news articles published on the web consists of 8 phases. The output of each phase is stored in a database (Section 7.3). It is a general approach that can be applied to any information content that describes events. The phases should be executed mainly in sequence for each news article (except for some phases where the execution can be run in parallel); in any case, different news articles can be processed in parallel.

The entire process is executed periodically (i.e. with a certain frequency) to extract the latest published news articles, analyze them and add information to the up-to-date database, and compare them with the other news articles in the database to find duplicates. The execution starts analyzing the most recent news article as first, and avoids to analyze the news articles already present in the database. Since the process is quite fast (around 10 minutes to analyze 100 news articles and execute 1,300 comparisons) the frequency of execution depends on the need of having a real-time up-to-date database and how often the selected online newspapers publish news articles.

Figure 7.1 illustrates the process and steps. Firstly, data extraction is performed by harvesting online newspapers (*web scraping*). The content of each news article is labeled and structured to be stored in a database and to be semantically annotated. Some web content may already expose a predefined structure. For example, HTML pages encoded with the Document Object Model (DOM), have a tree structure wherein each node is an object representing a part of the document. If this is not the case, other methods can be used, such as RSS Feed, API, and so on.

The second phase is the categorization of the event. This phase is crucial to map a news article w.r.t. a type of event (business, sports, crime, politics, arts, culture, etc.). Given some pre-categorized news articles, i.e., annotated training data, Machine Learning algorithms can be applied to uncategorized news articles to assign them a type of event.

The phases from 3 to 6 are related to the entity extraction and resolution, that means identifying the entities in the news articles' text and label them according to their type. The third phase is the identification and extraction of the 5W+1H. This phase depends on the category the news article belongs to since the questions can change according to the type of event described in the news article.

The fourth phase is the Named Entity Extraction (NER). This phase exploits the text of the news articles extracted during the first phase and does not depend on the output of other phases. Therefore, it can be executed in parallel with the second phase. NER identifies the reference to persons, organizations, and places, and temporal expressions in the text of the news article. Its results can intersect the output of 5W+1H phase.

The fifth phase is the extraction and normalization of temporal expressions. By analyzing the news article's body, temporal expressions can be identified (for example, words like "yesterday", "two days ago", "this morning" are identified as temporal expressions) and then normalized in date format. This operation allows to identify the date of the event.

The sixth phase is the Linked Data Mapping; during this phase, the entities, such as persons, organizations, and locations are linked to entities (URI) available in Linked Datasets.

The seventh phase is the geographical localization; in this phase, the entities that have been identified as locations in phase 4 or as answers to “where” in phase 3 are processed to be geo-referenced. Different methods can be applied; moreover, if a location is not specified in the news article, organizations (identified in phase 4) can also be exploited to geolocate the event.

The eighth, and last, phase is the identification of duplicates. This phase should be applied not only in the case that the input sources (online newspapers) are more than one since it is possible to find the same event described in more news articles also within the same newspaper where updates about one event are published along time. The duplicate detection phase is carried out by identifying possible duplicates (candidates) and then making a comparison to the news’ text to understand if they are duplicates or not. At the end of this phase, after identifying the duplicates, it is also possible to merge the information of the events to enrich the information that will be stored on the database.

The use of semantic technologies is a key point in the presented approach for detecting events from news articles and enriching them with information automatically extracted from the text.

In Chapter 8 the solutions implemented for each aforementioned phase are described in detail.

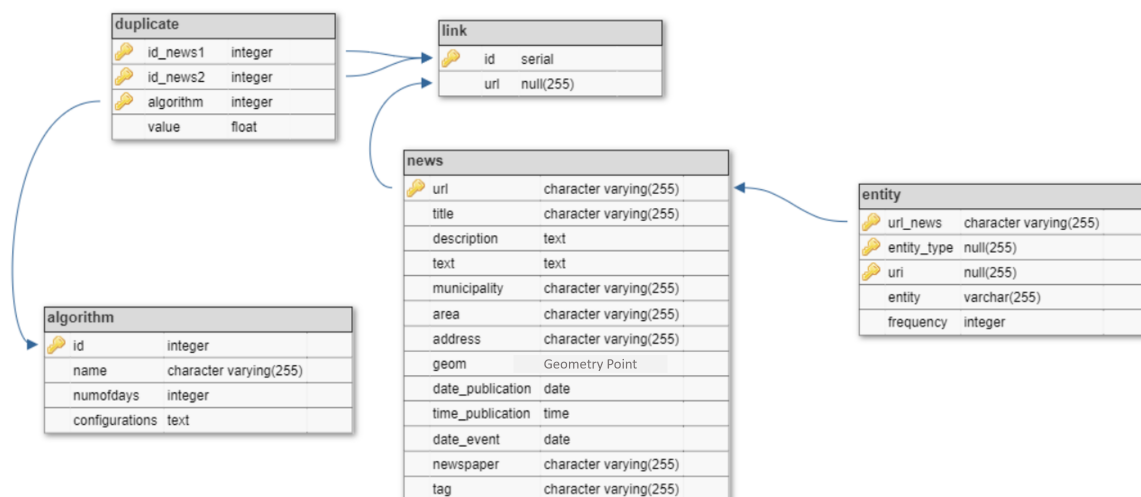


Fig. 7.2 Structure of the data platform used to store data extracted from the news articles.

7.3 Crime data platform

The structure of the data platform used to store crime-related news is displayed in Figure 7.2. This database is a PostgreSQL database that uses the extension PostGIS to store geospatial data. The “news” table stores the data related to the news articles, some of these data are extracted from the newspaper directly, while the other ones are obtained by the application of NLP algorithms. The “link” table is used by the web scraping phase to store the url of web pages containing the news articles and avoid analyzing the same news article more than one time. The tables “algorithm” and “duplicate” collect the configuration of deduplication algorithms and their output, respectively. More algorithms can be applied to the same group of news articles and can find different set of duplicates. In the end, the “entity” table stores the references to the Linked Open Data resources corresponding to the entities detected in the news articles’ text.

Chapter 8

The Italian crime analysis framework

Each phase of the approach proposed in the previous chapter has been implemented to develop a crime analysis framework for Italian documents. In this Chapter, the solutions for each phase are described in details.

This work has been published in [195, 262, 263] and accepted for publication in [264]. The code of the Italian Crime Analysis frameworks has been selected by the *ISWC Reproducibility Initiative*. The code of the applications is open source¹ and it is reproducible by others following the instruction provided with the code.

8.1 Web scraping

The scope of this phase is the extraction of basic components of the news articles, i.e., title, sub-title, body, and publication date. These components are always present in news articles since they are the key elements, and they are crucial for the following phases of the approach.

In this phase, the Document Object Model (DOM), i.e., the HTML structure of the web pages, is exploited. Each online newspaper follows a specific structure to publish news articles in the web. This structure allows to identify the aforementioned key elements since each element is encapsulated in a predefined HTML tag. The HTML tags used in the web page depend on the newspaper, however, the HTML structure is quite similar for all the online newspapers. An exemplar structure is shown in the code of Listing 8.1. Here, some components can be highlighted: title, description (sub-title), keywords, municipality, district, address, publication date, and body. As an example, the title of the news article can be automatically extracted considering the attribute “content” of the HTML tag “meta” combined with the attribute “name=title”. In the same way, also the other data can be

¹<https://github.com/federicarollo/Crime-event-localization-and-deduplication>

extracted. The process of web scraping here developed takes in input a list of URLs of the main web page of the newspapers to consider. Since this page contains multiple news articles related to different topics the URLs related to each news article are automatically retrieved. Then, the process accesses each URL and extracts available information from the HTML tags by using the java web crawler Jsoup.² A brief analysis of one news article of the selected newspapers allows to understand the HTML structure used and customize the process of web scraping. Some newspapers can encapsulate in HTML tags not only the basic components of the news article (title, sub-title, body, and publication date), but also other data, such as details on the location of the event (see “municipality”, “district” and “address” in Listing 8.1).

Exploiting the HTML structure of the web pages for accessing data makes the goal suitable for multiple data sources. Indeed, other solutions like RSS Feed or proprietary API are not always available and need customized extraction processes. Moreover, in recent years, interest in RSS Feeds has gone down with greater growth of Social Networks and fewer newspapers make RSS Feeds available.

Listing 8.1 Exemplar HTML structure of a web page containing a news article related to a theft.

```
<html lang="it">
  <head>
    <meta name="title" content="Man steals a car
    and runs on board">
    <meta name="description" content="The man
    was a convicted felon[...]">
    <meta name="keywords" content="theft">
    [...]
  </head>
  <body>
    <span class="municipality">
      <a href="/news/Modena" class="link">
        Modena
      </a>
    </span>
    <span class="district">
      <a href="/chronicle/accademia-militare"
      class="link">Accademia militare</a>
    </span>
```

²<https://jsoup.org/>


```
<span class=`` address">
  Vittorio Emanuele Road, 13
</span>
<time class=`` datestamp">
  September 23, 2020 9:30
</time>
<div class=`` entry-content" id=`` article-body">
  <p>The episode happened last Saturday,
  in the evening, while [...]</p>
</div>
</body>
</html>
```

8.2 News categorization

The categorization of news articles consists of identifying the type (or category) of events described in each news article, i.e. social events, crimes, sport events, and so on. Applying NLP techniques to the body of news articles to categorize them means approaching categorization as a text classification task.

To enhance the accuracy of category prediction, different approaches have been evaluated and compared. Firstly, some supervised and unsupervised algorithms have been applied to the word embeddings (also called word vectors) generated by the Word2Vec model. Then, given the promising results of supervised algorithms, active learning has been exploited with supervised categorization to improve the results. Also, different pre-processing techniques have been compared.

In the following, the method used to represent news articles through word embeddings is explained in details and the supervised and unsupervised categorization algorithm are compared and the application of active learning to the supervised categorization is depicted.

The results shown in this Section are also published in [263, 264].

8.2.1 Text representation extraction

To start with, the text of the news articles is pre-processed following some consecutive phases, as illustrated in Figure 8.1. The first phase is the *tokenization* (Point 1), which returns the list of the words that are present in the text, then the *stop word removal* phase (Point 2), a commonly used technique before performing NLP tasks, removes the stop words (e.g., articles, prepositions, conjunctions) from the above list since they usually occur a lot of times

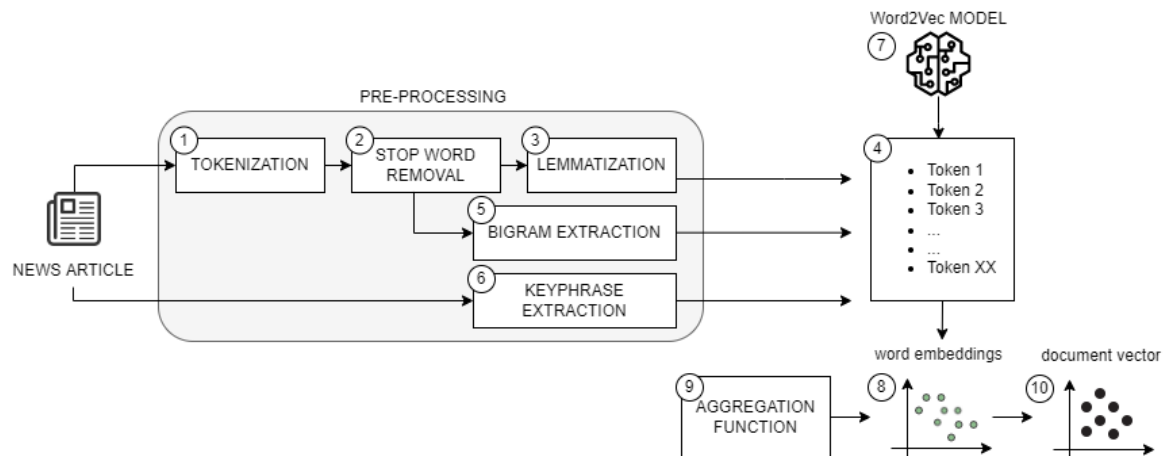


Fig. 8.1 Process of document vector extraction from news articles.

in texts and do not provide any relevant information. The result is a list of the most relevant words that are present in the text. Then, the *lemmatization* (Point 3) is applied for replacing the words in the list with their lemma. At the end of these phases, the final result is a list of meaningful *tokens* for every news article (Point 4).

In addition, bigrams and keyphrases are extracted to identify the most frequent sequences of two adjacent words (*bigram*) (Point 5) and the most relevant expressions that can contain multiple words (*keyphrase*) (Point 6). The bigrams are extracted from the list of tokens after removing the stop words, considering a minimum number of co-occurrences of the two words (*min_count*) and a threshold of the score [265] obtained by the following the formula:

$$score = \frac{L * (bigram_count - min_count)}{count(X) * count(Y)}$$

where L is the number of unique tokens in the text of the news article, *bigram_count* if the number of occurrences of the bigram, and X and Y are the two words of the candidate bigram. The keyphrases are identified in the news articles' text by using the RAKE (Rapid Automatic Keyword Extraction) algorithm [266] that is an unsupervised and domain-independent method. Both bigrams and keyphrases are added to the list of *tokens* (Point 4). The news articles with an empty list of tokens are removed.

Then, each token is replaced by its corresponding word embedding using a trained *Word2Vec model* (Point 7 and 8). Word2Vec [226] is based on the idea that each word can be represented by a fixed-length real-valued vector (*word embedding*). By using Word2Vec, operations on documents are transformed into mathematical operations in the vector space. The semantic similarity of two documents is the similarity of two vectors which represent that documents. Word2vec allows to extract the word embeddings that can be exploited

in several natural language processing tasks, such as clustering, categorization, sentiment analysis, synonyms identification, and so on. Word2Vec is based on a shallow neural network whose input data are generated by a window sliding on the text of the training corpus. This window selects a context within which a target is obscured to be predicted based on the rest of the selected context. Through this “fake task” internal parameters of the network are learned to build word embeddings, that is the real objective of training. There are two fake tasks: *Continuous Bag-of-Words (CBOW)* when the target is the central word in the sliding window and the surrounding words are the context, and *Skip-gram* when the central word is the context and the surrounding words are the target. If a token in the list is not found in the vocabulary of the model, it is simply discarded from the list without any replacement. Consequently, an *aggregation function* is applied to the obtained word embeddings to get the document vector of each news article (Point 9). As the authors of [224] suggest, two vector representations can be extracted:

- A1** the simple average of the word vectors,
- A2** the average of the word vectors weighted by the TF-IDF score of each word computed on the text of the news articles in the dataset. This representation gives more importance to those vectors that are related to words with a high frequency in the text of a news article and a low frequency in the others.

The obtained *document vectors* (Point 10) are the input data for any categorization algorithm.

8.2.2 Supervised and unsupervised categorization

After obtaining the document vectors of each news article in the dataset, several algorithms can be used to identify the category each news article belongs to. Both supervised and unsupervised techniques can be taken into account.

The *supervised text categorization* algorithms predict the topic of a document within a predefined set of categories, named labels. In the use case described in this thesis, the labels are the crime categories and the documents are the texts of the crime news articles which are represented by the document vectors.

The *unsupervised text categorization*, also known as clustering, is the task of grouping a set of objects in such a way that objects in the same group (called cluster) are more similar to each other than those in the other groups. The use of clustering for crime categorization consists of feeding the obtained document vectors into an algorithm and checking if the final clusters have a correspondence with some predefined crime categories. As suggested by the

authors of [267], to address the unbalance problem of the input dataset, the *Synthetic Minority Oversampling Technique* (SMOTE) [268] is employed. The approach is to oversample the elements in the minority class. Starting from an imbalanced dataset, this technique creates new samples for the classes that are present in minority in order to equal the number of elements in the most present category. The algorithm works in the feature space, then the new points do not correspond to real data. SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b .

8.2.3 Active learning

Active learning is a special case of Machine Learning in which a learning algorithm can interactively query some information source to label new data points with the desired outputs. Active learning has been employed mainly to improve the categorization of news articles or to extend the categorization, which may already be applied to some news articles, to news articles without an annotated category.

Active learning is an iterative process. Figure 8.2 illustrates every step of an iteration. The first phase is the training phase. The input (*labeled data*) consists of a list of feature vectors and the corresponding category (also known as label or class) which the news article (represented by that feature vector) belongs to. The feature vectors are the ones obtained through the application of Word2Vec. Training a categorization algorithm on the aforementioned four types of feature vectors, four categorization models are obtained. Each model is applied to the test set which consists of the four types of feature vectors of unlabeled news articles (i.e. news articles with no specified category). For each element of the test set, there are four different predictions, that belong to the event categories. At this point, a criterion is needed to select the most reliable predictions. This criterion can take into account the prediction confidence (i.e. the probability score returned by the categorization model with the predicted category) and can consider if the four predictions are all the same. Then, the documents that match the criterion are added to the labeled data and removed from the unlabeled data to start a new iteration. Accordingly, the models are retrained with the increased training set, and new predictions are made on the updated test set. The process stops when there are no unlabeled data to annotate or the confidence of the predictions is very low.

This mechanism should improve the confidence's prediction at each step because at each iteration the uncertain predictions (low confidence) are discarded, waiting for the following

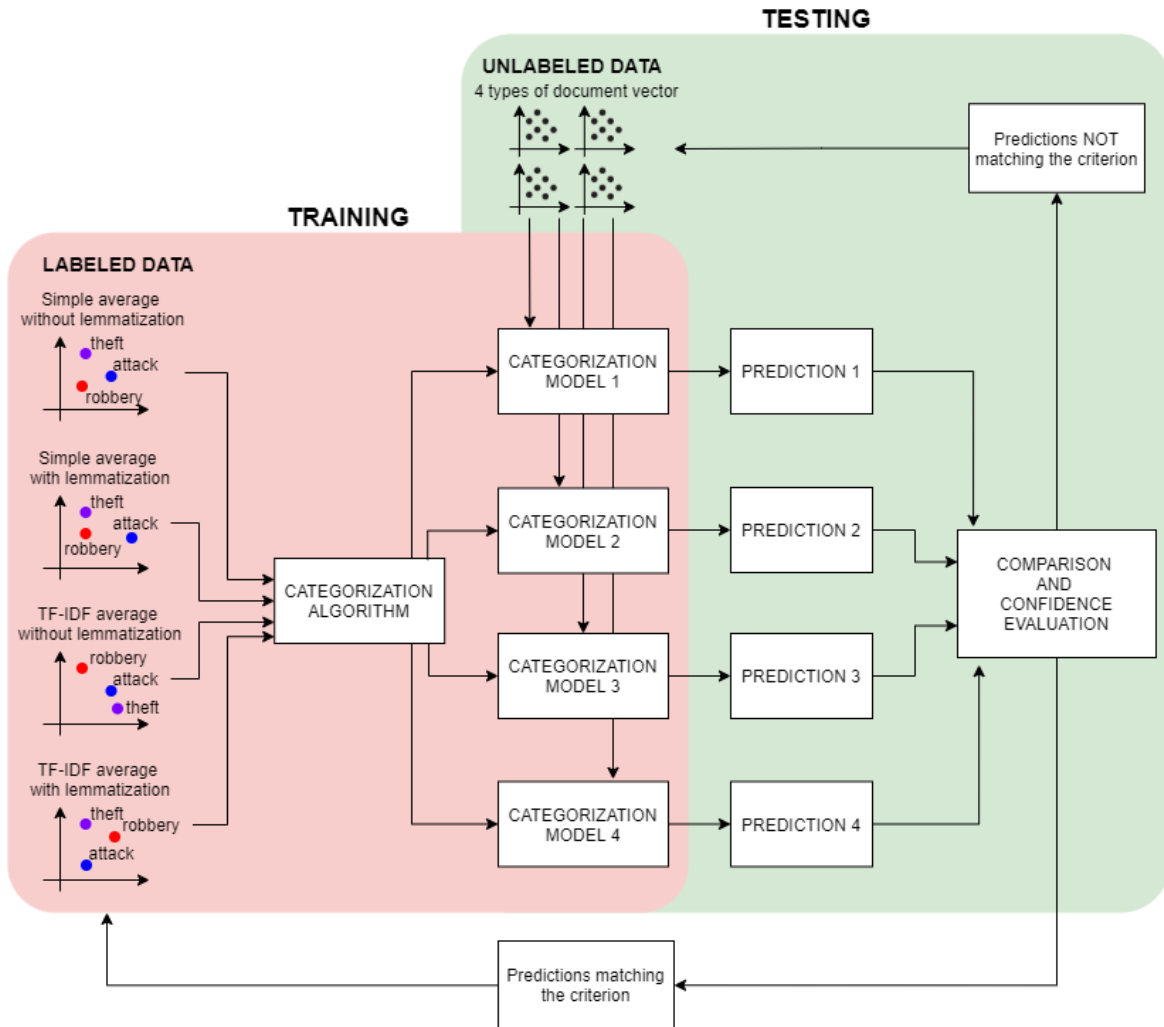


Fig. 8.2 Active learning process.

iterations, that use re-trained models, to annotate them with higher confidence. At each step the models are enhanced and provide more accurate annotations. Machine learning models are affected by the size of the training set, therefore, in general, a limited size can lead to low accuracy results. However, with active learning, it is possible to cope with a limited training set, since the training set is increased at each iteration.

8.3 Entity extraction and mapping

The scope of this phase is to extract entities from the body of the news articles and classify them according to predefined classes. An entity is a real-world object, such as a location, organization, or person, that can be denoted with a proper name, but it can be also an abstract

concept, such as the temporal reference (time) of an event. Referring to the crime event, the scope is to identify where the event is happened (location) and when (time), if some organization is involved in the event (organization), who has been involved (who are the victim and the author of the crime), and if there are some objects that are relevant in the event. Some examples of objects can be the weapon used in the robbery or object stolen during the theft. Identifying these entities means extracting the semantic information of the news articles.

8.3.1 NER

The scope of the NER phase is the extraction of semantic information from the text of the news articles. References to location, organization and person have been extracted from the news articles' body through Tint (The Italian NLP Tool)³ [269], an open-source collection of modules for the NLP of Italian texts based on the Stanford CoreNLP. Tint was chosen since compared with the three most popular NLP tools for Italian language (Tanl [270], TextPro [271], and TreeTagger [272]), it reported the highest values of speed, precision, and F-measure in the recognition of entities [269]. The NER module of Tint uses the Conditional Random Field (CRF) sequence tagger included in the Stanford CoreNLP, and it was trained on the ICAB dataset, which contains 180K words taken from the Italian newspaper "L'Adige" [269].

This phase takes in input the body of each news article and store the textual reference to the entity in the "entity" table of the database (see Figure 7.2) including the entity type and its frequency, i.e how many times it appears in the text of that news article.

8.3.2 Time extraction and normalization

To detect when the crime event described in the news articles happened, an algorithm of temporal expression extraction and normalization is applied to the body of each news article. The HeidelTime temporal tagger [273] was used. This tool can extract temporal expressions from documents in natural language and normalize them according to the TIMEX3 annotation standard. Using the date of publication of the news articles and the result of the temporal expression normalization, the date of the crime event is calculated. When multiple event dates are detected, the selected one is the date with higher frequency. In case of multiple event dates with the same frequency, the date closest to the publication date is taken into account.

³<http://tint.fbk.eu/>

8.3.3 Linked Open Data mapping

In mapping entities w.r.t. Linked Open Data (LOD) the scope is to find the resources of DBpedia [154] and Linked Geo Data [274] which correspond to the entities mentioned in the text of each news article.

This task has been performed through the use of DBpedia spotlight [275].⁴ This is an open source web service able to disambiguate and annotate mentions of DBpedia resources in natural language text. The API of DBpedia spotlight can be exploited to send a GET request through the function “annotate”. The API takes in input the text to annotate and some optional configuration parameters to filter the results. The parameters are the minimum limit of confidence score of the annotation, the value of support (how prominent is the entity in Lucene Model, i.e. number of inlinks in Wikipedia), the type of entity to be annotated, and the SPARQL query to select the resources of interest. In addition, user can provide a list of allowed URIs for annotation. The request returns the URIs of annotated entities and other information in json or html format. Also, the type of entity is returned.

DBpedia spotlight for the Italian language runs on Docker and the GET request to the API is set in Java application through the “jsoup” library. Text of the news articles is provided as input and the value of confidence was set to 0.7 to avoid inaccurate annotations. Then, the result of the request is parsed to retrieve the URIs of the entities. This list of entities can contain the entities identified by Tint, however, additional entities can be detected. The data to store in the data platform in the “entity” table include the URI of the LOD resource, its frequency for that news article, the name of the entity (i.e. the reference in the text), and its type (i.e. location, organization, person). To know the type of each entity some additional SPARQL queries are performed since this information is not always available in the request response. The Java class for SPARQL queries exploits the query engine ARQ.⁵ Firstly, a SPARQL query is performed to retrieve the value of the property *rdf:type* of the extracted URIs. The research refers to the objects of that property which refer to the DBpedia ontology (*dbo*), e.g. *dbo:Location*, *dbo:PopulatedPlace*, *dbo:Organization*, *dbo:Person*. If the property is not available or none of the results refer to *dbo*, the properties *rdf:subClassOf* and *owl:equivalentClass* are exploited in combination with *rdf:type*, as shown in Query 1-2 of Table 8.1. Let consider in these exemplar queries “<<http://dbpedia.org/resource/Modena>>” the resource of DBpedia representing the city of Modena. If Queries 1 and 2 do not return any result, the property *wikiPageRedirects* of DBpedia ontology is used to find the resource which the annotated entity is redirected to (Query 3). In the end, if the goal fails with the previous queries, an entity which has the annotated entity as its Italian translation is

⁴<https://www.dbpedia-spotlight.org/>

⁵<https://jena.apache.org/documentation/query/>

searched. An example is the resource `<http://dbpedia.org/resource/Cocaina>` which does not contain the expected property. However, another DBpedia resource report *Cocaina* as its Italian translation; therefore, it is checked if this resource has the *rdf:type* property (Query 4). Another SPARQL query is used to find the resource of Linked Geo Data (*lgdt*) which corresponds to the entity identified by DBpedia spotlight. The name of this resource is specified through the property *owl:sameAs* (Query 4).

8.3.4 Geo-localization

The GPS coordinates of the event locations are retrieved by using the OpenStreetMap API, and in particular, Nominatim.⁶ Geolocation is an iterative process; it starts by evaluating whether municipality and address are extracted during the web scraping phase and trying to geo-localize them.

If that information is not present or its geo-localization fails,⁷ the GPS coordinates of the entities identified during the process of Linked Data mapping are extracted through a SPARQL query, as shown in Table 8.2.

If this query does not provide any results, the iterative process continues exploring the entities extracted during the NER phase. When multiple locations are detected, the choice is to with the one with higher frequency. If more locations with the same frequency are available, the first occurrence in the news article is taken into account. This does not affect significantly the result of the geolocation; indeed, checking manually some cases, in the majority of the cases, multiple references to locations in a news article were all close together. If the geolocation of locations failed, the same iteration is performed with organizations. In the end, the GPS coordinates and the related address are stored in the “news” table.

8.4 Duplicate detection

Duplicate detection (or *deduplication*) is an interesting task when ingesting news articles, especially if more sources are crawled. This phase consists of identifying news articles related to the same event. In this way, it is possible to build the storyline of the event and merge information to provide a full description of the event. To give a few examples, if a newspaper publishes news articles related to social events, it could be published firstly a

⁶<https://nominatim.openstreetmap.org/>

⁷The query is performed through the function `OpenStreetMapUtils.getInstance().getCoordinates(location)` where *location* is a string that can be generated by the municipality and the address. This function provides the latitude and the longitude of the location. The success of this function depends on how the address is stored in Open Street Map and how the location is reported in the news article.

Table 8.1 SPARQL queries used to retrieve the type of annotated entities.

```

Prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix rdfs: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix dbo: <http://dbpedia.org/ontology/>
Prefix owl: <http://www.w3.org/2002/07/owl#>

# Query 1
select distinct ?type where {
  <http://dbpedia.org/resource/Modena> rdfs:subClassOf{0,5}/a ?type .
}

# Query 2
select distinct ?type where {
  <http://dbpedia.org/resource/Modena> owl:equivalentClass{0,1}/a ?type .
}

# Query 3
select distinct ?type where {
  <http://dbpedia.org/resource/Modena> dbo:wikiPageRedirects ?new_resource .
  ?new_resource a ?type .
}

# Query 4
select distinct ?type where {
  ?s rdfs:label ``Cocaina"@it .
  ?s a ?type
}

# Query 5
select distinct ?type where {
  <http://dbpedia.org/resource/Modena> owl:sameAs{0,1} ?x .
}

```

Table 8.2 SPARQL queries used to retrieve the GPS coordinates of annotated entities.

```

Prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>

select ?lat ?long where {
  <http://dbpedia.org/resource/Modena> geo:lat ?lat .
  <http://dbpedia.org/resource/Modena> geo:long ?long .
}

```

news article to promote the event some days before, then another article close to the event providing more detailed information, and again another news article after the event to give a report of the event. If it is possible to understand that the news articles are related to the same event, the storyline of the event can be built by linking the news articles each other and merging information where they are missing.

The duplicate detection is performed in three steps: (1) applying the blocking technique to a set of news articles, (2) comparing the news articles in pairs, and (3) merging the information previously extracted.

8.4.1 Blocking technique

The application of the blocking technique allows avoiding some comparisons in the next step. The comparisons between news articles that is improbable to refer to the same event are discarded. This task exploits the information extracted in the previous phases: the publication date, the event date, the municipality, and the type of event described.

Firstly, only news articles with the same event date are taken into account for comparison. If no event date has been extracted, the comparisons are performed on the news articles published in a date slot which considers the publication date. It is assumed that news articles related to the same event might be published with a few days of delay in different newspapers or in the same newspaper. If it is considered a d -days slot and an average of n news articles published on the first newspaper per day and m on the second newspaper, the number of pairwise comparisons is obtained by the following formula:

$$total_comparisons = (m * d) * (n * d) = m * n * d^2$$

This could be a very high number and could make the process very time consuming due to its quadratic complexity. For this reason, other filters are applied. The blocking technique compares only news articles classified in the same type of event (retrieved by the categorization task) and related to events that happened in the same municipality, including cases in which municipality is not specified.

8.4.2 News articles comparison

After the identification of the news articles to compare, different techniques can be applied to find duplicates.

The similarity between two news articles is computed by using the following multi-variables formula:

$$\textit{similarity} = \alpha * X + \beta * Y + \gamma * Z$$

where X is the similarity between the titles of the two pieces of news, Y is the similarity between the sub-titles, Z is the similarity between the bodies, and α , β , γ are the weights to be assigned according to the importance of each similarity. The threshold T determines if the two compared pieces of news are related to the same crime event: if the similarity computed by the multi-variables formula is equal or greater than T the news are labelled as “duplicates”, otherwise the news are considered as related to different events.

The proposed method combines the Cosine Similarity and the shingling technique. The shingling technique [276] is computed between set of shingles, instead of words. A shingle is a portion of text, also known as “n-gram” where n is the number of consecutive words in the text. In the sentence “*The mugger escaped with the bag*”, if 3-Shingling technique is used, the shingles will be “the mugger escaped”, “mugger escaped with”, “escaped with the”, “with the bag”. For comparison, the Jaccard index and the cosine similarity methods are applied to a set of manually labelled 50 news reports. The cosine similarity computed by using the shingling technique is the best option, since it provided better results of recall, precision, and accuracy. Indeed, this is also the expected result, since the Jaccard index is not affected by the number of word occurrences. The cosine similarity instead takes into account how many times the shingle appears in a specific news and its “importance” thanks to the TF-IDF (Term Frequency - Inverse Document Frequency) function. The “importance” of a shingle is made by how many times it appears in the news report (i.e. *term frequency*), compared to how many times it appears in all the collection of news reports to be compared (i.e. *document frequency*). The framework implements the library “java-string-similarity”,⁸ a Java library implementing different string similarity and distance measures, including Shingle based algorithms with cosine similarity.

To determine the best value of the configuration parameters, a validation test has been made on a dataset containing 358 manually labelled news articles related to only one type of crime (robbery) published between February 2019 and May 2019 in two Italian newspapers. The process found 44 duplicates.

The first choice is the dimension of the shingles. Obviously, the smaller the shingle size, the greater the amount of storage capacity and space complexity required. In a previous work [277], Alonso et al. discovered that the optimal shingle dimension without losing precision were between 1 and 3. Based on these tests, the length of the shingle was set to 2 for the title and the sub-title and 3 for the body of the news. Then, the weights α , β , γ are determined.

⁸<https://github.com/tdebatty/java-string-similarity>

Several tests are executed on manually labelled news reports to select the best values of these parameters, the best values of precision, recall, and accuracy have been obtained by using the values $\alpha = 1$, $\beta = 1.25$, and $\gamma = 4$. Therefore, these values are selected as the best values. Tests have been performed making comparisons between news published in the 3-days and the 5-days slot (*day slot = 3* and *day slot = 5*). The news pairs selected by the reduction algorithm to be compared are 376, instead of $196 * 162 = 31752$. The time required by the duplicate detection algorithm is a little more than one minute. Table 8.3 shows the evaluation of tests using the slot of 3-days with different values of threshold T . Tests described in Table 8.4, instead, are made using the slot of 5-days. In both cases the best results are obtained with the threshold set to 0.70. The difference in accuracy, precision, recall, and F1-measure is minimal.

Table 8.3 Validation test of the *3-days slot* duplicate detection.

T	Accuracy	Precision	Recall	F1-score
0.71	0.92	1.00	0.36	0.53
0.70	0.94	0.79	0.68	0.73
0.69	0.92	0.68	0.68	0.68
0.68	0.91	0.63	0.70	0.67
0.67	0.91	0.60	0.82	0.68
0.66	0.87	0.49	0.84	0.62
0.65	0.86	0.47	0.98	0.64

Table 8.4 Validation test of the *5-days slot* duplicate detection.

T	Accuracy	Precision	Recall	F1-score
0.71	0.91	0.89	0.37	0.52
0.70	0.93	0.72	0.70	0.71
0.69	0.92	0.66	0.70	0.68
0.68	0.91	0.61	0.72	0.67
0.67	0.90	0.57	0.84	0.68
0.66	0.86	0.46	0.86	0.60
0.65	0.84	0.43	1.00	0.60

8.4.3 Information merging

After discovering duplicates, information related to the two examined news articles can be compared and merged. In particular, this phase is related to the address extracted from the text of the news articles and, consequently, the geolocalization, the data of 5W+1H, and

the event date obtained by the application of HeidelTime. This operation allows to create a detailed dataset and to highlight some possible errors in the detection of duplicates or the entity extraction phase. Indeed, if the above-mentioned data are not shared by the two news articles, it may be possible that the detection of duplicates may have wrongly identified newspaper articles as duplicates. This could happen in very few cases due to the presence of the blocking technique.

Each field related to the address (city, municipality, name of the road) is compared; if the first news article has some empty field and the others are the same of the second news article (as expected since the news articles have been detected as duplicates), the empty fields are replaced with the information of the second news article, or vice versa. The same approach is applied to the data of 5W+1H and the event date. If some data is in contrast, the news articles are labeled for a manually control.

Chapter 9

Use case in Modena

The Italian Crime Analysis framework described in Chapter 8 has been tested to collect and analyze the news articles related to the crimes occurred in the province of Modena.

The data of the Italian Department of Public Security of the Minister of the Interior (published by Sole24Ore¹) classifies the city of Modena at the 12th position among the other Italian cities based on the number of crimes reported to the police. The total number of police reports in Modena in 2021 was 26,328 (3,722 reports per 100,000 inhabitants). The first city in the national ranking is Milan with 159,613 reports (4,866 reports per 100,000 inhabitants).

According to the latest report of ISTAT², the most frequent crimes in Modena from 2016 to 2020 are thefts, damages, scams and computer fraud, threats, and wilful injury. Figure 9.1 reports the number of the mentioned crimes reported to the authorities. As can be seen, the number of thefts decreases from 2019 to 2020. This is probably due to the lockdown caused by the Covid-19 emergency. In fact, because of the pandemic, the government had imposed severe restrictive measures, allowing only essential displacements. As a result, the population was often at home and worked from home if allowed. On the other hand, the number of computer fraud increased (from 1985 in 2019 to 2773 in 2020). Due to the protracted pandemic emergency, ISTAT expects an increase of computer fraud also in 2021.

Since the city of Modena does not provide any Open Data about crimes, crime analysis for citizens can rely on local newspapers. The Italian Crime Analysis framework collected 17,500 reports from June 2011 to December 2021 (approximately 10 years), and is currently running to analyze news articles published every day by two local newspapers. On 17,500 reports, the framework was able to geolocalize almost 100% of the crime events and normalize the time expressions on 83% of the news articles. The results produced allows performing

¹<https://lab24.ilsole24ore.com/indice-della-criminalita/?Modena>

²http://dati.istat.it/Index.aspx?DataSetCode=dccv_delittips

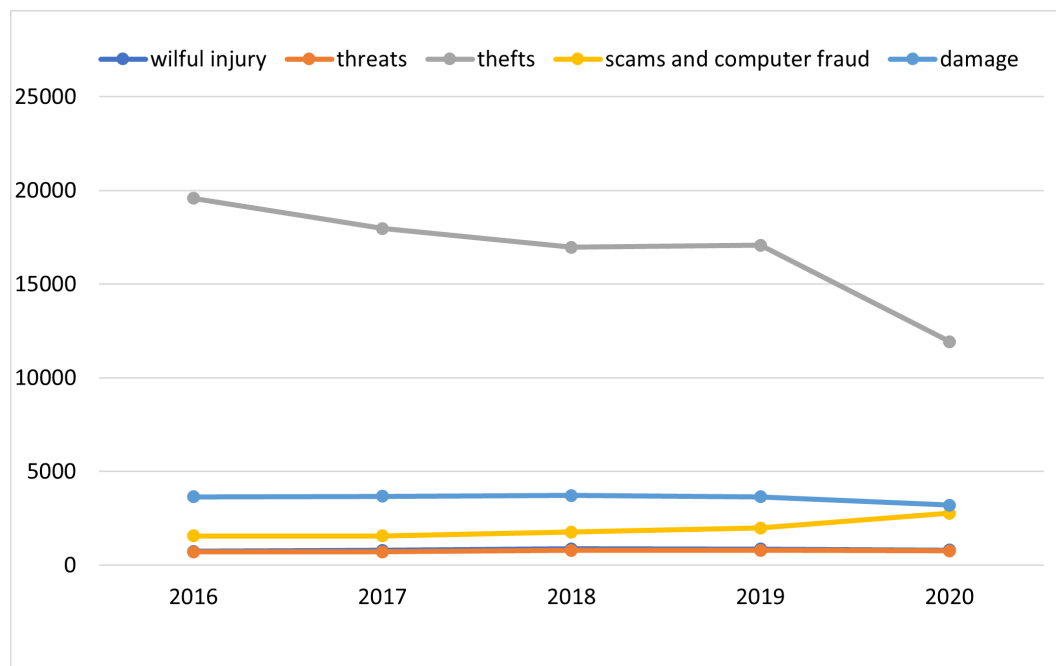


Fig. 9.1 Crimes reported to the authorities in the province of Modena from 2016 to 2020 [SOURCE: ISTAT, data of the Italian Ministry of the Interior].

crime mapping studies and the identification of crime hot spots in semi real-time. Some visualizations of these results are shown through the “Modena Crime” web application.³

In the following, the configuration of the framework to build the crime dataset of Modena is described (Section 9.1). Then, the experiments conducted on this dataset for each phase of the framework are evaluated (from Section 9.2.1 to Section 9.2.4). The impact and the scalability of the Italian Crime Analysis framework are discussed in Section 9.3. In the end, Section 9.4 provides some exemplar visualization of the collected data.

The experiments described in this chapter have been published in [262, 263] and accepted for publication in [264].

9.1 Configuration of the framework

The Italian Crime Analysis framework has been set up to collect news articles from two online newspapers of the province of Modena: “ModenaToday”⁴ and “Gazzetta di Modena”⁵. The selection of these newspapers is motivated by their popularity. They publish on average 850 news articles per year related to crimes in the Modena province. There exist other 3

³<https://dbgroup.ing.unimo.it/modenacrime>

⁴“ModenaToday” newspaper: <https://www.modenatoday.it/>

⁵“Gazzetta di Modena” newspaper: <https://gazzettadimodena.gelocal.it/modena>

minor online newspapers, however integrating them will not substantially change the results since they cover just 5% of the total news articles already collected, and their news articles are in almost all cases duplicated with respect to the news reported by the two main newspapers.

The data extracted from the newspapers include the *URL* of the web page containing the news article, the *title* of the news article, the *sub-title*, the *text*, the information related to the place where the crime occurred (*municipality*, *area*, and *address*), the *publication_datetime* that is the date and the time of publication of the news article, and the *event_datetime* that refers to the date of crime event. Part of these data is automatically extracted from the web page of the news articles, the other ones are identified by applying NLP techniques to the text of the news articles. The first newspaper already classifies the news articles according to the type of event described by using appropriate keywords in the URL (this classification is done manually by the journalist, author of the news articles). For example, the web page at the link <https://www.modenatoday.it/tag/furti/> on the “ModenaToday” newspaper contains only news articles describing thefts (in Italian, the word “furti” means thefts). The second newspaper does not provide a similar categorization; however, there is a research option on the website which allows to filter news articles based on the words appearing in the news articles’ text. For example, the web page at the link <https://gazzettadimodena.gelocal.it/ricerca?query=furti> on the “Gazzetta di Modena” newspaper shows only news articles containing the word “furti” (thefts) in the text. If the word indicating a category of crime appears in the text of a news article, it is very probably that the news article describes a crime event related to that category. Each news article is assigned to a specific crime category. The list of categories is based on two lists of crimes: the annual crime reports of the Italian National Institute of Statistics (ISTAT) and the data of the Italian Department of Public Security of the Minister of the Interior (published by Sole24Ore⁶). The ISTAT annual report⁷ shows, aggregated by time and space, the number of crimes divided by category that happen in each Italian province. The crime hierarchy of ISTAT is very detailed with 53 types of crimes organized in a hierarchy. The Italian Department of Public Security of the Minister of the Interior publishes a list of the most frequent crimes in each province, also Modena. It uses 37 categories of crimes. For the city of Modena in 2021, only 13 categories have a number of complaints greater than 0.4 on 100,000 inhabitants. Based on those two lists, and on the broader categories of crimes used by newspapers, the final list of crimes was chosen. The total number of categories is 13: “furto” (theft), “rapina” (robbery), “omicidio” (murder), “violenza sessuale” (sexual violence), “maltrattamento” (mistreatment), “aggressione” (aggression), “spaccio” (illegal sale, most commonly used to refer to drug trafficking), “droga” (drug dealing),

⁶<https://lab24.ilsole24ore.com/indice-della-criminalita/?Modena>

⁷http://dati.istat.it/Index.aspx?DataSetCode=dccv_delittips

“truffa” (scam), “frode” (fraud), “riciclaggio” (money laundering), “evasione” (evasion), and “sequestro” (kidnapping).

The current dataset contains 17,500 news articles published in the two selected newspapers from 2011 to December 2021 (approximately 10 years). The dataset is imbalanced on the category of the crimes that are described in the news articles. During the web scraping phase, in some cases, it was possible to collect also the municipality, area, and address of the location where the crime happened.

9.2 Experimental results

This section is devoted to the evaluation and discussion of the experiments related to each phase of the Italian Crime Analysis framework proposed in Chapter II on the built dataset described in the previous section. In particular, the results refer to the news categorization using word embeddings through supervised algorithms, unsupervised algorithms and active learning (Section 9.2.1), the application of the Named Entity recognition (Section 9.2.2), the extraction of the event date (Section 9.2.3), and the deduplication 9.2.4).

9.2.1 News categorization

Considering that the news articles in the dataset are written in Italian, three Word2Vec models have been chosen for the experiments:

- M1** a pre-trained model [278], whose dimension is 300. The dataset used to train Word2Vec was obtained exploiting the information extracted from a dump of Wikipedia, the main categories of Italian Google News and some anonymized chats between users and the customer care chatbot Laila.⁸ The dataset (composed of 2.6 GB of raw text) includes 17,305,401 sentences and 421,829,960 words.
- M2** a Skip-Gram model trained from scratch on the crime news articles of the dataset for 30 epochs (*window_size=10, min_count=20, negative_sampling=20, embedding_dim=300*).
- M3** a Skip-Gram model which has been trained on the crime news articles of the dataset for 5 epochs, starting from the embeddings of M1 (*window_size=10, min_count=20, negative_sampling=20, embedding_dim=300*).

⁸<https://www.laila.tech/>

The experiments have been conducted employing all the three models to extract the word embeddings separately. In addition, three different configurations of the pre-processing phase have been set up to allow a comparison of the results:

- P1** pre-processing with tokenization and stop word removal. The result is a list of relevant words for each news article.
- P2** pre-processing with tokenization, stop word removal, and lemmatization. The result is a list of relevant lemmatized words for each news article.
- P3** pre-processing with tokenization, stop word removal, lemmatization, and bigram and keyphrase extraction. The result is the list of P2 with the integration of bigrams and keyphrases.

In the end, the two aggregation functions (A1 and A2) have been applied to the word embeddings. Concluding, 18 different combinations of pre-processing, word embeddings' average, and Word2Vec model have been obtained.

Evaluation metrics

Precision, recall and accuracy are the most common metrics when evaluating a categorization task. They are obtained by the following formula:

$$precision = \frac{TP}{TP + FP}, recall = \frac{TP}{TP + FN}, accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

where, given a category, TP is the number of samples that are correctly assigned to that category (*true positives*), FP is the number of samples that are associated to that category, but they belong to a different one (*false positives*), FN represents the number of samples of that category that are assigned by the algorithm to another one (*false negatives*), and TN indicates the number of samples that are correctly not assigned to that category (*true negatives*). Using the precision and the recall values, F1-score can be calculated:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

For the supervised algorithms, these metrics are calculated for each category, and then averaged. Since the built dataset is very imbalanced, the average is weighted by the support, i.e. the number of news articles for each category.

Before calculating the same metrics for the unsupervised categorization, it is necessary to find the best match between the class labels and the cluster labels, i.e. to assign a category

of crime to each cluster. To start with, the highest number of samples for a certain category in a cluster is found, and the category is assigned to that cluster. Then, the process continues with the other clusters and the other categories, again starting from the highest number of samples. The process assigns only one category to each cluster, and a category cannot be assigned to multiple clusters. For each cluster, the values of precision, recall, accuracy, and F1-score are calculated and then the average of these values is found for the overall values.

In addition to the above-mentioned metrics, the Silhouette Coefficient [279] is used in unsupervised categorization to assess the quality of clusters, and determine how well the clusters fit the input data. This metric evaluates the density of clusters generated by the model. The score is computed by averaging the silhouette coefficient for each sample, that is computed as the difference between the average intra-cluster distance (a), i.e. the average distance between each point within a cluster, and the mean nearest-cluster distance (b), i.e. the average distance between all clusters, normalized by the maximum value:

$$silhouette = \frac{1}{N} * \sum_{k=1}^N \frac{(b_k - a_k)}{\max(a_k, b_k)}$$

where N is the number of generated clusters. The Silhouette Coefficient is a score between 1 and -1, where 1 means that there are highly dense clusters and clearly distinguished, while -1 stands for completely incorrect clustering. A value near 0 represents overlapping clusters with samples very close to the decision boundary of the neighboring clusters. Different distance metrics can be used to calculate a and b , the most common distances are the euclidean distance, the manhattan distance, canberra, cosine, jaccard, minkowski. The euclidean distance has been employed.

Supervised categorization

Different supervised Machine Learning algorithms have been exploited to compare their performances. For each algorithm, around 65% of the news articles in the dataset is used as the training set, while the remaining is used as the test set. Both sets contain articles from both newspapers. Table 9.1 shows the number of news articles for each category that are included in each set. As can be noticed, there is a considerable imbalance of the categories. The dominant category is “theft”.

Table 9.2 shows the values of precision and recall of 15 supervised algorithms trained on the document embeddings obtained by the 18 different combinations of pre-processing configuration, word embeddings’ average and Word2Vec model. In the table, the first column contains the name of the categorization algorithm employed, and the highlighted cells with the

Table 9.1 The number of news articles in the training and test sets for each category in supervised categorization.

Category	Training Set	Test Set
Theft	7,062	3,658
Drug dealing	1,180	617
Illegal sale	769	382
Aggression	619	301
Robbery	500	303
Scam	414	204
Mistreatment	225	125
Evasion	196	92
Murder	169	81
Kidnapping	162	85
Money laundering	99	56
Sexual violence	106	39
Fraud	42	14
Total	11,543	5,957

number in bold indicate the best values of precision or recall (values greater than or equal to 0.78). As can be seen, there are six algorithms with the highest values: Linear SVC ($C = 1.0$), SVC (RBF kernel, $C = 1.0$, $\text{gamma} = \text{'scale'}$), SGD (both configurations), Bagging, and XGBoost. Considering the performance of these algorithms in the different configurations, it can be noticed the lowest values are found when model M1 is used. Therefore, even if the embeddings of M1 are trained on a dataset that largely includes news articles and contains contexts very similar to the ones of the dataset, M2 and M3 outperform M1 in terms of precision and recall. This is probably due to the fact that the word embeddings of M2 and M3 are learned from the same documents that are then categorized (indeed, both M2 and M3 are trained on the crime news articles). This makes certain words more discriminative for certain contexts, and therefore, for certain crime categories. Comparing models M2 and M3, it can be noticed that the use of lemmatization and the extraction of bigram and keyphrase has little influence on the performances. The same consideration can be done comparing the simple average and the TF-IDF weighted average.

Table 9.3 shows in detail the results of the best algorithm (Linear SVC) using the embeddings of M3, the pre-processing with tokenization and stop word removal, and the simple average for each category. The first column contains the name of the crime category, while the second column indicates the number of news articles in the test set for that category. The values of precision, recall and f1-score show that the algorithm suffers from the imbalance of the training set. The less the category is present in the dataset, the more the recall (sometimes also the precision) decreases. In some cases, recall is equal to zero, this

Table 9.2 Precision (P) and recall (R) of the application of different categorization algorithms on the embeddings derived from the three selected models.

		P1						P2						P3					
		M1		M2		M3		M1		M2		M3		M1		M2		M3	
		A1	A2	A1	A2	A1	A2	A1	A2	A1	A2	A1	A2	A1	A2	A1	A2	A1	A2
<i>Linear SVC</i> (<i>C=1.0</i>)	P	.77	.75	.80	.79	.80	.79	.77	.74	.80	.79	.80	.79	.74	.72	.79	.77	.79	.77
	R	.78	.76	.81	.80	.81	.79	.78	.76	.81	.80	.81	.79	.76	.73	.80	.78	.80	.78
<i>SVC (RBF, C=1, gamma='scale')</i>	P	.75	.74	.79	.79	.81	.79	.74	.74	.79	.79	.78	.79	.69	.71	.79	.79	.78	.78
	R	.75	.74	.80	.80	.80	.80	.74	.74	.80	.80	.79	.80	.73	.72	.80	.80	.79	.79
<i>SGD (L2 norm, Hinge loss)</i>	P	.76	.75	.80	.77	.80	.79	.74	.74	.79	.78	.80	.78	.71	.72	.79	.79	.79	.79
	R	.76	.74	.80	.78	.81	.79	.74	.76	.80	.79	.80	.79	.74	.73	.79	.73	.79	.77
<i>XGBoost</i>	P	.73	.71	.77	.76	.77	.76	.72	.70	.77	.76	.77	.76	.69	.68	.76	.75	.77	.76
	R	.74	.73	.78	.77	.78	.78	.74	.72	.78	.77	.78	.78	.72	.70	.77	.76	.78	.78
<i>Bagging (KNN(n=5))</i>	P	.72	.70	.76	.76	.77	.76	.72	.70	.77	.76	.77	.76	.67	.65	.76	.75	.76	.75
	R	.74	.72	.77	.77	.78	.77	.74	.72	.78	.77	.78	.77	.70	.68	.77	.76	.77	.76
<i>KNN (k=5)</i>	P	.71	.70	.76	.75	.76	.76	.71	.70	.76	.75	.76	.75	.65	.65	.75	.74	.76	.75
	R	.73	.72	.77	.76	.77	.77	.73	.72	.78	.77	.77	.77	.69	.68	.77	.76	.77	.76
<i>SGD (L1 norm, Perceptron)</i>	P	.80	.68	.76	.76	.80	.77	.73	.73	.83	.75	.78	.75	.73	.69	.79	.77	.72	.77
	R	.57	.69	.73	.76	.73	.75	.69	.70	.68	.76	.75	.74	.70	.69	.74	.72	.71	.68
<i>KNN (k=3)</i>	P	.70	.69	.75	.74	.76	.75	.69	.68	.76	.75	.76	.75	.66	.64	.74	.73	.74	.73
	R	.72	.71	.77	.76	.77	.76	.72	.71	.77	.76	.77	.76	.69	.67	.75	.75	.75	.75
<i>KNN (k=1)</i>	P	.69	.68	.74	.73	.75	.73	.69	.67	.75	.74	.75	.73	.64	.63	.73	.72	.73	.73
	R	.69	.67	.74	.73	.75	.74	.69	.67	.75	.74	.75	.74	.65	.63	.73	.72	.73	.74
<i>Random Forest (n=100)</i>	P	.71	.66	.74	.73	.75	.74	.64	.63	.75	.73	.75	.74	.62	.61	.71	.71	.74	.74
	R	.70	.68	.74	.72	.75	.74	.69	.67	.75	.73	.75	.75	.67	.65	.72	.71	.74	.74
<i>Bagging (Decision Tree)</i>	P	.64	.62	.70	.68	.70	.70	.62	.61	.71	.69	.72	.72	.60	.60	.68	.67	.71	.70
	R	.69	.67	.72	.71	.73	.72	.68	.66	.73	.72	.74	.73	.66	.66	.71	.70	.73	.72
<i>Adaboost (Decision Tree)</i>	P	.63	.64	.71	.71	.70	.71	.62	.59	.72	.69	.71	.70	.59	.56	.67	.66	.70	.70
	R	.67	.68	.73	.72	.72	.73	.67	.65	.74	.72	.73	.72	.65	.64	.71	.69	.72	.72
<i>BernoulliNB</i>	P	.69	.69	.74	.74	.74	.74	.69	.68	.74	.74	.74	.75	.67	.66	.74	.74	.73	.74
	R	.55	.54	.62	.62	.58	.58	.58	.55	.63	.63	.61	.61	.56	.54	.60	.61	.58	.58
<i>GaussianNB</i>	P	.73	.71	.76	.75	.76	.75	.73	.70	.76	.76	.76	.76	.71	.70	.74	.73	.74	.73
	R	.52	.43	.60	.58	.59	.57	.52	.39	.61	.58	.60	.58	.50	.41	.60	.59	.58	.57
<i>Decision Tree</i>	P	.57	.55	.62	.62	.64	.64	.56	.55	.64	.62	.65	.63	.54	.52	.61	.60	.63	.62
	R	.56	.55	.62	.61	.63	.63	.55	.54	.63	.61	.64	.62	.53	.51	.61	.59	.62	.62

means that the number of true positives is zero or there are a lot of false negatives, i.e. the algorithm was not able to identify the most news articles of that category. On the other hand, when the precision is equal to 1 it means that the number of false positives is zero, i.e. no news article of other categories has been mislabeled with that category.

After some analysis, it was discovered that the annotation for the news articles of “Gazzetta di Modena” in the dataset is not so accurate, therefore these tests on categorization are “dirty”. Then, the test was performed again by using the embeddings of M2 and M3 and the best six categorization algorithms of the previous examples only on the news articles published in “ModenaToday”. Table 9.4 shows the values of precision and recall achieved by the best categorization algorithms on “ModenaToday” news articles. Comparing these values to the values of Table 9.2, slightly higher values are highlighted. The highest values are found when Linear SVC is applied to the document embeddings obtained by the word

Table 9.3 Precision, Recall and F1-score for each crime category obtained using the embeddings of model M3 with pre-processing P1 and average A1, and Linear SVC.

Category	#news articles	precision	recall	f1-score
Theft	525	.96	.96	.96
Drug dealing	177	.81	.81	.81
Illegal sale	173	.82	.82	.82
Robbery	143	.87	.78	.82
Aggression	90	.72	.81	.76
Scam	81	.80	.86	.83
Murder	42	.80	.93	.86
Kidnapping	41	.79	.83	.81
Mistreatment	22	1	.27	.43
Sexual violence	8	0	0	0
Money laundering	7	0	0	0
Evasion	5	1	.40	.57

embeddings of model M3 using the simple average and the pre-processing with tokenization and stop word removal (P1).

In conclusion, further steps in pre-processing, such as lemmatization, bigram extraction and keyphrase extraction do not seem to be beneficial because the performances in terms of precision and recall do not improve. Also, the simple average (A1) shows better results than the TF-IDF average (A2). The model M3 is the preferable model for two reasons:

- the training of a Word2Vec model from scratch on the dataset requires 15 minutes, while the use of transfer training learning for M3 requires less than 3 minutes for retraining,
- the pre-trained model has a wider vocabulary. It could be useful the document embeddings extraction for new news articles which contain words that do not appear in the training corpus. However, it is highly likely that all those words that are discriminative for crime categories are already present in the vocabulary of M2.

Unsupervised categorization

Clustering test has been performed on the features obtained by M3, according to the results of the supervised categorization. Only the news articles published in the “ModenaToday” newspaper are used since the annotations available for this newspaper are more reliable than the ones available for the “Gazzetta di Modena” newspaper. The dataset contains 5,896 news articles and is imbalanced. Table 9.5 shows the number of news articles for each category in the dataset. Again, the most present category is “theft”, while the least present category

Table 9.4 Precision (P) and recall (R) of the application of the best six algorithms on the embeddings of M2 and M3 on “ModenaToday” news articles.

		P1				P2				P3			
		M2		M3		M2		M3		M2		M3	
		A1	A2	A1	A2	A1	A2	A1	A2	A1	A2	A1	A2
<i>Linear SVC (C=1.0)</i>	P	.85	.82	.86	.83	.84	.82	.85	.83	.83	.80	.83	.81
	R	.85	.82	.86	.83	.85	.82	.85	.83	.83	.81	.84	.81
<i>SVC (RBF, C=1, gamma='scale')</i>	P	.84	.84	.84	.83	.83	.83	.83	.83	.82	.81	.81	.81
	R	.85	.85	.85	.84	.84	.83	.83	.84	.83	.83	.82	.82
<i>SGD (L2 norm, Hinge loss)</i>	P	.85	.83	.85	.84	.83	.84	.85	.83	.82	.81	.81	.83
	R	.85	.83	.84	.83	.83	.83	.85	.82	.83	.81	.80	.80
<i>SGD (L1 norm, Perceptron)</i>	P	.84	.81	.85	.82	.84	.81	.85	.83	.81	.79	.82	.83
	R	.79	.81	.81	.82	.80	.80	.79	.80	.80	.79	.80	.81
<i>XGBoost</i>	P	.80	.80	.81	.81	.81	.80	.80	.81	.79	.78	.80	.79
	R	.81	.80	.82	.81	.81	.81	.80	.81	.80	.79	.80	.80
<i>Bagging (KNN(n=5))</i>	P	.78	.79	.79	.80	.81	.79	.81	.80	.79	.77	.79	.78
	R	.78	.78	.79	.79	.80	.79	.81	.80	.79	.77	.79	.79

is “fraud” with only 3 news articles. SMOTE has been applied to overcome the unbalance problem, generating new points in order to achieve the number of “theft” instances in all the other categories. In the end, in the test, there are 30,888 points in the feature space (2,376 points for each category). Four unsupervised algorithms are chosen for the experiments:

- K-means
- Mini Batch K-means
- Agglomerative Clustering
- Spectral Clustering

For all these algorithms, the number of clusters n has to be established in advance. To start with, n was set to 13, that is the number of crime categories extracted from the newspapers.

According to the results of the supervised categorization, clustering was applied to the document embeddings obtained by model M3 with the simple average of the word embeddings considering all the three different pre-processing phases. Table 9.6 shows the values of precision, recall, f-score, and accuracy in each test, the two highest values of each metric are highlighted. Precision, recall, and f1-score are always low (the highest value is 0.52), while accuracy reaches high values (from 0.86 to 0.93). This means that the number of false positives and false negatives w.r.t. the true positives is very high.

Summing up, the highest metric values are highlighted in the second type of the pre-processing phase using K-means. The value of the silhouette coefficient in this test is 0.132,

Table 9.5 The number of news articles from “ModenaToday” newspaper for each category.

Category	#news articles
Theft	2,376
Drug Dealing	810
Illegal sale	739
Robbery	616
Aggression	427
Scam	415
Murder	185
Kidnapping	168
Mistreatment	85
Evasion	35
Sexual Violence	20
Money Laundering	17
Fraud	3
Total	5,896

that is a low value, while the highest value (0.138) corresponds to the K-means algorithm with the first pre-processing type. In Figure 9.2 the histograms show the number of news articles in each cluster for each crime category. For a better visualization, only the names of dominant categories are shown. As can be seen, in all the clusters there are few (maximum 3) dominant categories, as expected by the value of the silhouette coefficient. Figure 9.3 displays the silhouette coefficient for each sample, visualizing which clusters are dense and which are not. The red line indicates the average (0.132). This plot allows understanding the cluster imbalance. In all the clusters, except cluster 4, there are some instances with negative coefficient, this means that the instances are in the wrong cluster. The highest coefficients are related to some samples in cluster 3, indeed, looking at the histograms, in that cluster there are the most samples of “murder” (almost 2,000 samples) and this category is present also in cluster 9 and 12 but with a very low number of samples (around 10).

Table 9.6 Evaluation of unsupervised categorization using the document embeddings obtained by model M3 with the simple average and the three different pre-processing phases.

	<i>precision</i>			<i>recall</i>			<i>f1-score</i>			<i>accuracy</i>		
	<i>P1</i>	<i>P2</i>	<i>P3</i>	<i>P1</i>	<i>P2</i>	<i>P3</i>	<i>P1</i>	<i>P2</i>	<i>P3</i>	<i>P1</i>	<i>P2</i>	<i>P3</i>
<i>K-means</i>	.44	.52	.47	.47	.51	.46	.45	.51	.47	.90	.92	.89
<i>Mini Batch K-means</i>	.51	.49	.49	.50	.50	.49	.50	.50	.49	.91	.90	.92
<i>Agglomerative Clustering</i>	.45	.47	.47	.49	.48	.39	.47	.48	.43	.90	.92	.92
<i>Spectral Clustering</i>	0	.40	.33	.40	.50	.46	0	.45	.38	.93	.89	.86

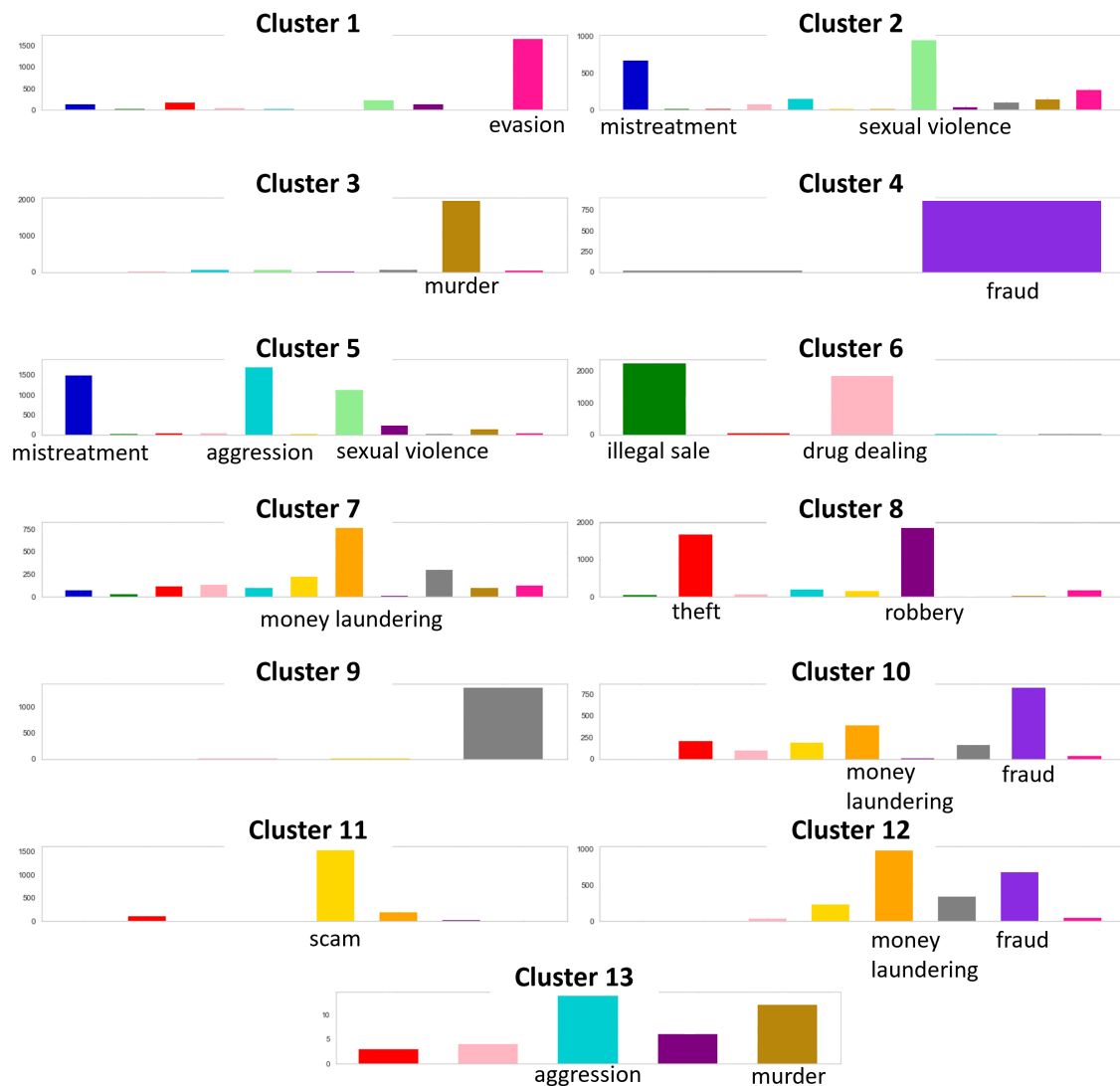


Fig. 9.2 Histograms of the cluster distribution obtained with K-means ($n = 13$) applied to the embeddings of model M3, simple average and second pre-processing type.

Analyzing in detail the results of this experiment, it can be noticed that the clusters group together categories that are semantically similar. Based on this consideration, a test was run by grouping together semantically similar categories in macro-category. The chosen macro-categories are seven:

- “Kidnapping”,
- “Murder”,
- “Robbery”, “Theft”,

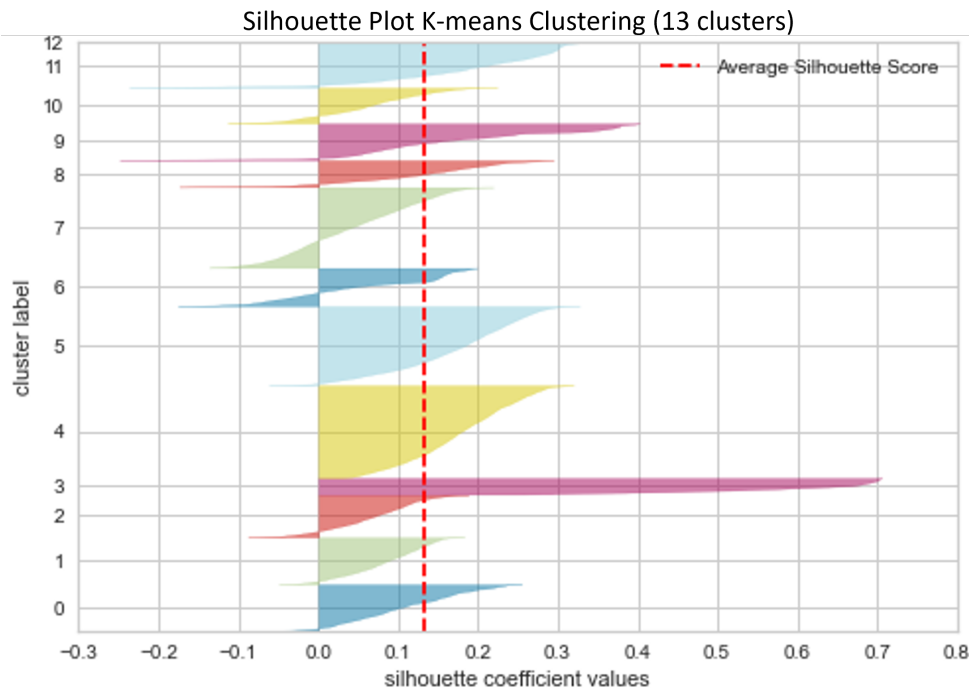


Fig. 9.3 Plot of the silhouette coefficient in the clusters of K-means ($n = 13$) on the embeddings of model M3, simple average (A1) and pre-processing with lemmatization (P2).

- “Mistreatment”, “Aggression”, “Sexual Violence”,
- “Scam”, “Fraud”, “Money Laundering”,
- “Illegal Sale”, “Drug Dealing”,
- “Evasion”.

All the four algorithms tested before are re-used to perform categorization with macro-categories. In this case, the best result in terms of silhouette coefficient is given by the Spectral Clustering using the document embeddings generated by the simple average and the third pre-processing that includes also bigram and keyphrase extraction. The results are shown in Table 9.7, the numbers in bold are the number of instances of the assigned category for the corresponding cluster. Considering the Table row by row, it can be noticed that each macro-category is dominant in only one cluster. However, clusters 1 and 6 have two dominant macro-categories. In addition, while clusters 1, 2, 3, and 6 contain a high number of samples, in the other clusters there are few samples. Following the procedure previously described to assign a category to each cluster, the assigned category for cluster 4 is “evasion” which has no instance in that cluster. The overall accuracy achieved in this experiment is 0.90.

Table 9.7 Results of unsupervised text categorization obtained by Spectral Clustering ($n=7$) applied to the document embeddings of model M3, simple average and the third pre-processing type.

Macro-category	1	2	3	4	5	6	7
Robbery and theft	258	3959	354	11	33	137	0
Drug dealing and illegal sale	119	156	4226	2	7	240	2
Fraud, scam and money laundering	98	1354	90	0	13	5573	0
Aggression, mistreatment and sexual violence	5940	937	132	5	11	99	4
Kidnapping	156	77	41	0	19	2075	8
Murder	2123	209	0	0	11	16	17
Evasion	1694	147	346	0	0	189	0

Active learning

Based on the results of supervised categorization, the choice was to use M3 and the supervised method Linear SVC with active learning following the procedure described in Section 8.2.3 [280]. Active learning has been exploited to re-annotate the news articles of the dataset coming from “Gazzetta di Modena”. Therefore, the training set is constituted by “Modena Today” news articles, as shown in Table 9.8.

Table 9.8 The number of news articles in the training and test sets for each category.

Category	Training Set	Test Set
Theft	2314	7217
Drug dealing	794	761
Illegal sale	675	296
Aggression	416	366
Robbery	599	173
Scam	400	157
Mistreatment	85	163
Murder	177	49
Evasion	35	208
Kidnapping	160	64
Money laundering	17	95
Sexual violence	18	75
Fraud	3	44
Total	5693	9668

The first test applies the following criterion for the annotation: “the data that are inserted in the new training set after each iteration are those with at least a prediction whose confidence is greater than 0.80; they are re-annotated with the value of that prediction”. Examining in detail the results of the first iteration, it was noticed that that the predictions are more reliable than the annotation provided by the newspaper. After 5 iterations, there are 1,553 news articles in the test test, while the remaining 83.93% has been re-annotated, as shown in Table 9.9.

Table 9.9 Final results of active learning with the first configuration.

iteration	Training	Test	at least one prediction $C \geq 0.80$
1	5693	9668	6238
2	11931	3430	1015
3	12946	2415	488
4	13434	1927	240
5	13674	1687	134
re-annotated data	8115 / 9668 (83,93%)		

Then, a new criterion for annotation has been defined. It is an extension of the first criterion which also includes the news articles with prediction confidence lower than 0.80 and all four predictions that agree with each other. The results are reported in Table 9.10. The second criterion achieves a greater coverage than the first one. After the last iteration, the news articles in the test set are 637.

Table 9.10 Final results of active learning with the second configuration.

iteration	Training	Test	at least one $C \geq 0.80$	same prediction $C \geq 0.80$
1	5693	9668	6238	1613
2	13544	1817	452	413
3	14409	952	68	1630
4	14640	721	26	58
re-annotated data	9031 / 9668 (92,54%)			

9.2.2 NER effectiveness

The NER was evaluated on 530 news articles manually labeled, published on both “Modena-Today” and “Gazzetta di Modena” newspapers from January 2020 to March 2020 related to all the types of crime taken into consideration. The manual annotation of the news articles identifies only one location for each news article.

Table 9.11 Evaluation of the coverage in detecting location reference through the application of NER and the integration of Linked Geo Data

<i>method</i>	location references found	references localized in Modena province	location references not found
NER locations	474	454	55
+ NER organizations	+37 (511)	+46 (500)	-37 (18)
+ Linked Geo Data	+18 (529)	+29 (529)	-18 (0)

To prove that the adoption of the NER and the mapping w.r.t. Linked Geo Data enhances the geolocalization, the data obtained without the application of these methodologies have been checked. Without NER, only the tags on the HTML documents can be used. In a newspaper like “ModenaToday”, only the city or the area within the city to which the news refers are provided. In this case, a correspondence with the municipality can be established for 95% of news articles. This reference is very loose because it does not allow us to locate the crime event to a specific point but only to refer it to an area/city. On the other hand, by using NER, persons, organizations and locations can be extracted from each news article. The geolocalization of the location entities allows us to retrieve the GPS coordinates in 479 news (90%). If no location is discovered, the geolocalization is applied to the organizations, finding 34 geolocated entities. Searching in Linked Geo Data the locations/organizations, a location for each news is found. Moreover, with Linked Geo Data, each crime event is enriched and linked to a URI referencing its location. Table 9.11 shows how the application of NER and the integration with Linked Geo Data increase the possibility of retrieving the GPS coordinates. In the table, the first column explains the method used, the second column is the number of news articles where the location reference is found, the third column shows how many of these locations are geolocated in the area of Modena province, and in the end, the last column is the number of news articles where a location reference is not found. As can be seen, Linked Geo Data allows finding a location reference for all news in the dataset and revises the coordinates geolocated out from the area of Modena province. Without the use of Linked Geo Data, the locations and organizations, identified with the NER, allow finding location reference in 96% of the news articles in the dataset.

All the locations retrieved are geolocated in Modena province thanks to the integration with Linked Geo Data. Due to the incompleteness of some addresses found by the NER locations and organizations, the OpenStreetMap API is not able to retrieve the GPS coordinates. In some cases, the crime events are geolocated out from the Modena province.

Linked Geo Data revises these errors. Comparing the results with the manual annotations, the geolocalization performs with a 90% precision.

Considering all the news in the dataset discovered by the approach, 12,482 news articles among the 17,500 news articles have been geolocated exploiting the locations found by NER, 515 of these GPS coordinates were out of the Modena province. On the remaining news, almost the total of news articles has been located through organizations (293 out of the Modena province) and 13,725 thanks to the integration with Linked Geo Data (1 out of the Modena province).

9.2.3 Time extraction and normalization effectiveness

The use of the HeidelTime tagger allowed extracting temporal expressions in 11,426 news articles (83% of the total number of stored news articles). To evaluate the precision of such an approach, a manual evaluation was performed on the same dataset used for the evaluation of the NER effectiveness. One event date has been manually identified for each news. The algorithm has extracted and normalized time reference in 440 news articles (83%). All the time references found are correctly normalized. In the remaining 90 news articles, the identification of time reference was a bit challenging since it was embedded in expressions like “after six months”, or “at dawn”, or other similar expressions. In other cases, the news is about some updates of an event very far in time; therefore, only the reference to the year is found.

9.2.4 Duplicate detection

The duplicate detection algorithm was evaluated on 470 news articles coming from the two different newspapers, and related to different types of crime (165 thefts, 127 robberies, and 178 attacks). The dataset was manually labeled, and 49 duplicates were found. The reduction algorithm selected 261 news articles pairs to be compared (43 about robberies, 100 attacks, and 118 thefts); the time required to find duplicates was two seconds. The deduplication has been applied using three alternatives: 3-days and 5-days time slots on the news date, and the event date. Using the time slots performed similarly with the threshold of 0.70. The results are shown in Table 9.12. The precision is a bit higher for the 3-days slot, where it reaches the 91%, while the recall is better for the 5-days slot, where it is 88%. The third evaluation was done exploiting the results of the Temporal Expression Normalization phase. On the test set, 440 news articles (94%) have a time expression that has been extracted, and normalized, and used to define the event date. The duplicate detection considering the event date in the

Table 9.12 Duplicate detection algorithm on different *day slots*.

<i>T</i>	<i>day slot</i>	Accuracy	Precision	Recall	F1-measure
0.70	3-days	0.96	0.91	0.65	0.76
0.70	5-days	0.97	0.84	0.88	0.86

pairwise comparison allowed discovering 95% of the duplicates. Thus, improving the results of the duplicate detection algorithm.

9.3 Impact and scalability of this research

The major limitation in using newspapers for crime analysis is related to the fact that newspapers do not publish news articles for each crime happened in Modena since some crimes are not of public interest. To evaluate the impact of the method employed by the Italian Crime Analysis framework, the number of crimes in the built Modena crime dataset and the number of crimes published in the official report of ISTAT (i.e. the crimes reported to the police) have been compared. The report related to the period from 2018 till 2018 footnote <http://dati.istat.it/Index.aspx?QueryId=25097&lang=en> has been taken into account. The information is only quantitative; the types of crime are reported per province as a rate out of 100,000 inhabitants. No information about where the crime happened (the municipality, the district, or the address), and the period of the year (season, month, or date) is provided. The classification of crimes includes 50 types of crimes and is very exhaustive, including categories that are not taken into consideration in the approach. For providing a comparison between the two datasets, only the crime categories in common have been taken into account. The number of crimes is considered after the deduplication. Since a location-based comparison was not possible because ISTAT provides a unique report for the entire province, the number of crimes for the city of Modena in each year has been calculated by using the total population registered in the same year (from 184700 till 186300 inhabitants). With the total number of crimes in the city of Modena of 7107 from 2014 till 2018, the dataset covers around 21% of the crimes reported by ISTAT. A hypothesis on this low coverage can be attributed to the fact that not all the criminal events recorded by ISTAT, and therefore in the police reports, are of high impact. Therefore, not all of them are reported in local news articles.

The most frequent crimes in both datasets are thefts for each year from 2014 to 2018. Figure 9.4 shows the total number of the top three types of crimes recorded in the report of ISTAT and compared to the number collected by the framework. As can be seen, the lower coverage is reported in thefts (the percentage average is 18%), while the higher one is in

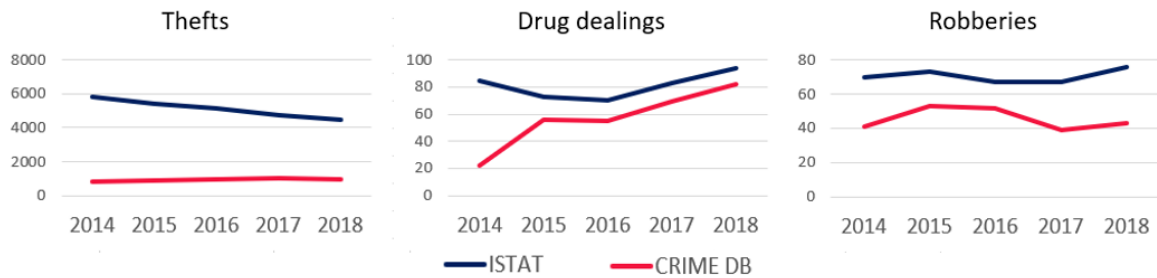


Fig. 9.4 Distribution of crime reports from 2014 to 2018 in different neighborhoods of Modena.

drug dealings (70%). It is very interesting to see that the trend over time of drug dealing is very similar in the two distributions, excluding the year 2014. Robberies have quite a similar trend in both the datasets (the dataset contains the 64% of the total robberies).

The approach here described is currently in use to identify the crimes occurring in Modena and its province in real-time. The visualization of crime-related data is available online⁹. This application is a Python web application which uses Tornado as a web framework. Everybody wants to avoid finding ourselves involved in unpleasant incidents. The citizens and the visitors of a city need to be aware of crime statistics; this may affect where they will stay, go for a walk, live, and work. On the other hand, city councils have the responsibility of identifying crime hot spots to employ appropriate monitoring and controlling mechanisms. Regarding the impact of this approach, even if the approach has been applied in a medium area, it highlights its potentiality. In Italy it is not possible to collect real-time crime information from official sources, since official criminal statistics are reported annually with a delay of 6 months. The proposed approach can be applied everywhere also in small or medium cities/areas since there will be always one or more newspapers that report the main crimes to happen in that place.

A first scalability test has been executed to ingest all the news articles related to crimes happened in the entire Emilia-Romagna region. Other 9 newspapers which publish news related to the 9 provinces of the Emilia-Romagna region were selected. All the available news articles, from 2011 till now, which refer to the eleven crime types have been collected. The total number of news articles is 35,000 (on average 3,900 news articles for each province). The crime ingestion can be run in parallel for different newspapers and for different crime type. Therefore, 99 ingestion processes have been executed in parallel to extract, analyze

⁹Crime Visualization App - <https://dbgroup.ing.unimo.it/modenacrime>

and store data of the region. The total loading time, that depends on the loading time of the province with the higher number of news from 2011, is 3 hours for 35,000 news articles¹⁰.

9.4 Event visualization

The data of the Modena Crime dataset can be exploited to make an analysis of the distribution of crimes in the city, discover trends, highlight areas at risk.

Some analyses are shown in Modena Crime Dashboard¹¹ that is a multi-language web application implemented in Python. All the data are available in Italian and English. This dashboard allows to provide a semi real-time overview and some analysis of the crime situation in the Modena province.

Figure 9.5 shows an overview of the main visualizations of the Modena Crime Dashboard. The dashboard provides interactive maps with the geolocated crimes, heatmaps to discover the high-crime areas, statistics and annual reports. The interactivity of the dashboard allows to select the crimes of interest and visualize them in the map, the same operation is possible on the heatmap. The option “Stats” in the menu allows to create histograms and other plots based on the selection of municipality in the province of Modena, month, year and crime of interest. In the end, annual reports can be downloaded through the button “Reports” in the menu.

This dashboard represents an important contribution since there is no similar visualization tool in Italy.

¹⁰The test has been performed on a Microsoft Windows 10 Pro with 16GB RAM.

¹¹Modena Crime Dashboard: <https://dbgroup.ing.unimo.it/modenacrime>



Fig. 9.5 Overview of the Modena Crime Dashboard.

Chapter 10

Conclusions

This thesis contributed to designing data-driven approaches for urban and social challenges. Two main topics have been addressed: the implementation and management of sensor networks for the monitoring of road traffic and air quality at an urban scale, and the development of a comprehensive framework for crime analysis from news articles. These two topics refer to the concepts of *Smart City* and *Safe City*, that identify cities employing technology to ensure a good quality of life for citizens, good services, and safety.

In the first part of the thesis, the TRAFAIR project has been presented. The project aims to estimate the level of pollution at an urban scale by producing real-time air pollution estimates in 6 European cities and by developing a service for forecasting urban air quality based on weather forecasts and tailored traffic flow estimates. This two-year project started on the 1st November 2018 and has been implemented in 6 cities of different sizes: Zaragoza (600,000 inhabitants), Florence (382,000), Modena (185,000), Livorno (160,000), Santiago de Compostela (95,000) and Pisa (90,000). The project has provided a flexible and adaptable service to study the current and predictive air quality. Since it relied only on open source software, the project can be replicated in other cities. The implementation of the Smart City data platform to manage all the data necessary for the scopes of the project has been discussed, and the choice of using a PostgreSQL database has been motivated by explaining the challenges in defining a unified data model shared by all the partners of the project. The other research activities in this thesis that concerns the TRAFAIR project are mainly focused on the use case of the city of Modena, a medium-size city in the north of Italy with a network of around 400 traffic sensors and 13 low-cost air quality sensors. In this thesis, a detailed description of the sensor data collection in Modena and an exploratory analysis of the generated data have been provided.

An interactive dashboard for the air quality sensor data visualization and management has been successfully developed according to the technical requirements provided by the environmental experts. SenseBoard is a flexible tool that can be integrated into specific IoT environments. It is a multi-purpose tool: to manage and maintain the air quality sensor network control and to supervise the calibration process and the identification of anomalies. The dashboard integrates a big amount of heterogeneous data, both geo-spatial and time-series data. The flexibility and scalability of SenseBoard have allowed monitoring a dynamic sensor network since the sensors are moved frequently.

Different methodologies to detect anomalies in sensor data streams have been described and evaluated. One filter and three novel approaches have been applied to the traffic sensor data providing extensive experimental results. A flow-speed correlation filter has been implemented on the traffic sensor data to remove unrealistic observations where the number of counted vehicles (flow) in a certain time interval is not consistent with the corresponding average speed. Currently, the filter is implemented as a trigger each time a new observation is stored in the data platform. A novel data cleaning process based on the Seasonal-Trend Decomposition using Loess (STL) and the Interquartile Range (IQR) analysis has been developed. The experimental results have confirmed the efficiency of such an approach, allowing the sensor faults removal from the input of the traffic simulation model, improving the performance of the model, and ensuring a better emulation of the real urban traffic conditions. The proposed solution is currently employed in real-time to detect anomalies on traffic sensor data in Modena. An additional proposed approach for anomaly detection has combined the correlation study between vehicle flow and speed performed by FFIDCAD and the prediction of flow based on the trend analysis of the ARIMA model. This method has allowed identifying as anomalies the sensor data which deviate from the predicted values. The approach used in another method is to cluster spatio-temporal data to detect two types of anomalies: contextual point anomalies and contextual collective anomalies. The adopted algorithm combines ST-BOF and ST-BDBCAN in cascade and has several parameters which have to be heuristically optimized. Several tests have been needed to define the set of parameters suitable for the application and the type of anomalies that need to be detected. A Python implementation of the combination of the two algorithms has been released. The results of the experiments were promising and showed the potential of considering the geographical features of the data in anomaly detection. Thanks to this work, some unresolved challenges can be highlighted: managing the spatio-temporal distance is quite complicated and could benefit from more sophisticated distance functions able to capture the topology of the street, the traffic correlations, and assign optimized weights to the features. Still, this work could be a baseline for future improvements. A majority voting

method has been applied to the air quality sensor data. Since usually anomaly detection on low-cost sensors is performed on concentrations obtained after the calibration process, this appears to be the first research effort to address anomaly detection on raw air quality data improving calibration performances on multivariate time-series. This has allowed reducing the error on the calibrated values. Moreover, the flow for porting 12 unsupervised Machine Learning models to execute anomaly detection on edge IoT devices has been described. Preliminary tests have been conducted on a partially synthetic dataset of air quality data. This work is a result of the collaboration with the researchers of the National University of Ireland Galway (NUIG) where I spent three months.

Since one of the scopes of TRAF AIR is the release of Open Data, a method for semantically annotating and publishing traffic sensor data of Modena and Zaragoza has been described. The proposed approach has proven to be feasible and easily adaptable to be applied in different fields, especially in a Smart City context. This work represents a relevant and compelling use case concerning the collection and exploitation of semantic sensor data in real-world scenarios.

In the second part of the thesis, a framework for performing crime analysis through the extraction of semantic information from news articles is described.

The proposed framework addresses multiple techniques for solving various sub-problems: extracting crime events from news articles, structuring information into a data model, geolocating them, linking them to Linked Data resources, and deduplicating them. The Italian Crime Analysis framework has been successfully employed in the province of Modena and has allowed collecting a consistent dataset of 17,500 news articles about 13 types of crimes. The success project has led the interest of a local newspaper in the city of Modena to exploit the results of the research. A comparison with the official data provided by the Italian National Institute of Statistics gave way to discover that this approach has allowed collecting about 21% of the crime events available in the official reports of ISTAT. This can be considered a satisfactory result since it is known a priori that newspapers do not publish news articles about all the crimes that happen in a city.

Extensive experiments have been conducted on the categorization of news articles based on the type of crime. The best configuration of the proposed approach has been detected in the use of word embeddings extracted by a retrained Word2Vec model¹ and the active learning applied to Linear SVC. The use of NER combined with the Linked Data mapping has enhanced the semantic information of each crime and the geolocalization of the events.

¹A new Word2Vec model has been released at <https://github.com/SemanticFun/Word2Vec-for-text-categorization/>.

The time expression normalization has allowed detecting the date of the event improving the performance of the duplicate detection algorithm. The extracted data have been published online in the Modena Crime dashboard.² It is the only dashboard that allows citizens of Modena to have a look at the crime data in near real-time and recent statistics on crime trends, providing a time-space analysis unveiling these critical data to the citizen.

The approach is domain-independent; it can be applied to any kind of news article. For new sources, an individual wrapper/extractor has to be built to retrieve title, subtitle, and so on. This is the only part that needs to be built (and in some cases only adapted from other wrappers) to connect a new source with the application. All the other phases can be re-used as they are provided in the open-source code.³ The code of the Italian Crime Analysis frameworks has been selected by the *ISWC Reproducibility Initiative*. The NER algorithm can be applied to new sources provided that the text is in Italian, while the temporal expression extraction and normalization can also be performed on text of other languages. The geolocation process can geolocate addresses all over the world. The duplicate detection can be applied to text in different languages, since the similarity measure is not affected by the language. The crime categorization approach can be applied also in other contexts different from crime news articles, and is suitable for documents in languages different from Italian. However, since the trained Word2Vec model is language-dependent, it is necessary to use the appropriate Word2Vec model (if exists) or train the model on the documents in the specific language. Also, it is possible to test this approach on word embeddings generated by using other models, such as Glove or FastText.

²<https://dbgroup.ing.unimo.it/modenacrime>

³<https://github.com/federicarollo/Crime-event-localization-and-deduplication>

References

- [1] Laura Po, Federica Rollo, José Ramón Ríos Viqueira, Raquel Trillo Lado, Alessandro Bigi, Javier Cacheiro López, Michela Paolucci, and Paolo Nesi. TRAFAIR: understanding traffic flow to improve air quality. In *2019 IEEE International Smart Cities Conference, ISC2 2019, Casablanca, Morocco, October 14-17, 2019*, pages 36–43. IEEE, 2019.
- [2] United Nations General Assembly. Transforming our world: The 2030 agenda for sustainable development, 2015. Available at http://www.un.org/ga/search/view_doc.asp?symbol=A/RES/70/1&Lang=E.
- [3] A. Bigi, G. Veratti, S. Fabbi, L. Po, and G. Ghermandi. Forecast of the impact by local emissions at an urban micro scale by the combination of lagrangian modelling and low cost sensing technology: The trafair project. Harmonisation within Atmospheric Dispersion Modelling for Regulatory Purposes, HARMO, 2019. cited By 2.
- [4] Nurul Nasuha and Munzilah Rohani. Overview of application of traffic simulation model. *MATEC Web of Conferences*, 150:03006, 01 2018.
- [5] Sinem Coleri, Sing Yiu Cheung, and Pravin Varaiya. Sensor networks for monitoring traffic. In *Allerton Conference on Communication, Control and Computing*, pages 32–40, 2004.
- [6] Sergio Ilarri, Ouri Wolfson, and Thierry Delot. Collaborative sensing for urban transportation. *IEEE Data Engineering Bulletin*, 37(4):3–14, December 2014. Special Issue on Urban Informatics.
- [7] Chiara Bachechi and Laura Po. Implementing an urban dynamic traffic model. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2019, Thessaloniki, Greece, October 14-17, 2019*, pages 312–316, Thessaloniki, Greece, 2019. ACM.
- [8] Chiara Bachechi and Laura Po. Traffic analysis in a smart city. In *IEEE/WIC/ACM International Conference on Web Intelligence - Companion Volume, WI '19 Companion*, page 275–282, New York, NY, USA, 2019. Association for Computing Machinery.
- [9] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo - simulation of urban mobility. *Int. Journal on Advances in Systems and Measurements*, 5, 2012.
- [10] Daniel Krajzewicz, Georg Hertkorn, Christian Feld, and Peter Wagner. Sumo (simulation of urban mobility); an open-source traffic simulation. *4th Middle East Symposium on Simulation and Modelling (MESM2002)*, pages 183–187, 01 2002.

- [11] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wiessner. Microscopic traffic simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582, 2018.
- [12] B. Bessagnet, A. Hodzic, R. Vautard, M. Beekmann, S. Cheinet, C. Honoré, C. Liousse, and L. Rouil. Aerosol modeling with chimere—preliminary evaluation at the continental scale. *Atmospheric Environment*, 38(18):2803 – 2817, 2004.
- [13] J. Steppeler, G. Doms, U. Schättler, H. W. Bitzer, A. Gassmann, U. Damrath, and G. Gregoric. Meso-gamma scale forecasts using the nonhydrostatic model lm. *Meteorology and Atmospheric Physics*, 82(1):75–96, 2003.
- [14] G. Bonafè. pescos: pescos 0.3.0 (version v0.3.0), 2015.
- [15] M. Mircea, L. Ciancarella, G. Briganti, G. Calori, A. Cappelletti, I. Cionni, M. Costa, G. Cremona, M. D’Isidoro, S. Finardi, G. Pace, A. Piersanti, G. Righini, C. Silibello, L. Vitali, and G. Zanini. Assessment of the ams-minni system capabilities to simulate air quality over italy for the calendar year 2005. *Atmospheric Environment*, 48:178 – 188, 2014.
- [16] D. Oetl. Evaluation of the Revised Lagrangian Particle Model GRAL Against Wind-Tunnel and Field Observations in the Presence of Obstacles. *Boundary-Layer Meteorology*, 155:271–287, 2015.
- [17] Sergio Ibarra-Espinosa, Rita Ynoue, Shane O’Sullivan, Edzer Pebesma, María de Fátima Andrade, and Mauricio Osses. Vein v0.2.2: an r package for bottom-up vehicular emissions inventories. *Geoscientific Model Development*, 11(6):2209–2229, 2018.
- [18] Chiara Bachechi, Laura Po, and Federica Rollo. Big data analytics and visualization in traffic monitoring. *Big Data Research*, 27:100292, 2022.
- [19] Postgresql. <https://www.postgresql.org>. Accessed: 2020-12-06.
- [20] PostGIS. <https://postgis.net>. Accessed: 2020-12-06.
- [21] Piotr Grzesik and Dariusz Mrozek. Comparative analysis of time series databases in the context of edge computing for low power sensor networks. In *20th International Conference, Amsterdam, The Netherlands, June 3-5, 2020*, volume 12141 of *Lecture Notes in Computer Science*, pages 371–383. Springer, 2020.
- [22] I. Petre, R. Boncea, C.Z. Radulescu, A. Zamfiroiu, and I. Sandu. A time-series database analysis based on a multi-attribute maturity model. *Studies in Informatics and Control*, 28(2):177–188, 2019.
- [23] Rob Kiefer. Timescaledb vs. postgresql for time-series: 20x higher inserts, 2000x faster deletes, 1.2x-14,000x faster queries, 2017. Available at <https://blog.timescale.com/blog/timescaledb-vs-6a696248104e/>, accessed 3 February 2021.

- [24] OpenStreetMap contributors. Planet dump retrieved from <https://planet.osm.org> . <https://www.openstreetmap.org>, 2017.
- [25] Google. Google Maps. <http://maps.google.com/>. Accessed: 2020-07-22.
- [26] Apple. Apple Maps. <https://maps.apple.com/>. Accessed: 2020-07-22.
- [27] HERE Technologies. Here. <https://www.here.com/>. Accessed: 2020-07-22.
- [28] TomTom International BV. Tomtom. <https://www.tomtom.com>. Accessed: 2020-07-22.
- [29] Mordechai Haklay. How good is Volunteered Geographical Information? a comparative study of OpenStreetMap and ordnance survey datasets. *Environment and Planning B: Planning and Design*, 37(4):682–703, August 2010.
- [30] Silvana Camboim, João Bravo, and Claudia Sluter. An investigation into the completeness of, and the updates to, OpenStreetMap data in a heterogeneous area in Brazil. *ISPRS International Journal of Geo-Information*, 4(3):1366–1388, August 2015.
- [31] Jesús Almendros-Jiménez and Antonio Becerra-Terón. Analyzing the tagging quality of the spanish OpenStreetMap. *ISPRS International Journal of Geo-Information*, 7(8):323, August 2018.
- [32] Maria Brovelli and Giorgio Zamboni. A new method for the assessment of spatial accuracy and completeness of OpenStreetMap building footprints. *ISPRS International Journal of Geo-Information*, 7(8):289, July 2018.
- [33] Alaa Alhamwi, Wided Medjroubi, Thomas Vogt, and Carsten Agert. OpenStreetMap data in modelling the urban energy infrastructure: a first assessment and analysis. *Energy Procedia*, 142:1968–1976, 2017. Ninth International Conference on Applied Energy.
- [34] ISO. *ISO 19156:2011 - Geographic information – Observations and measurements*. 01 2011.
- [35] Laura Po, Federica Rollo, Chiara Bachechi, and Alberto Corni. From sensors data to urban traffic flow analysis. In *2019 IEEE International Smart Cities Conference, ISC2 2019, Casablanca, Morocco, October 14-17, 2019*, pages 478–485. IEEE, 2019.
- [36] Chiara Bachechi, Federica Rollo, Federico Desimoni, and Laura Po. Using real sensors data to calibrate a traffic model for the city of modena. In Tareq Z. Ahram, Waldemar Karwowski, Alberto Vergnano, Francesco Leali, and Redha Taiar, editors, *Intelligent Human Systems Integration 2020 - Proceedings of the 3rd International Conference on Intelligent Human Systems Integration (IHSI 2020): Integrating People and Intelligent Systems, February 19-21, 2020, Modena, Italy*, volume 1131 of *Advances in Intelligent Systems and Computing*, pages 468–473. Springer, 2020.
- [37] Chiara Bachechi, Federica Rollo, and Laura Po. Real-time data cleaning in traffic sensor networks. In *17th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2020, Antalya, Turkey, November 2-5, 2020*, pages 1–8. IEEE, 2020.

- [38] Federica Rollo and Laura Po. Senseboard: Sensor monitoring for air quality experts. In Constantinos Costa and Evaggelia Pitoura, editors, *Proceedings of the Workshops of the EDBT/ICDT 2021 Joint Conference, Nicosia, Cyprus, March 23, 2021*, volume 2841 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021.
- [39] Federica Rollo, Bharath Sudharsan, Laura Po, and John G. Breslin. Air quality sensor network data acquisition, cleaning, visualization, and analytics: A real-world iot use case. In Afsaneh Doryab, Qin Lv, and Michael Beigl, editors, *UbiComp/ISWC '21: 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and 2021 ACM International Symposium on Wearable Computers, Virtual Event, September 21-25, 2021*, pages 67–68. ACM, 2021.
- [40] ITS-teknik. Inductive loops. Technical report, its-teknik, 2018.
- [41] Federal Highway Administration. *Traffic Detector Handbook: Third Edition—Volume I*. U.S. Department of Transportation, 2006.
- [42] M. B. Priestley. *Non-Linear and Non-Stationary Time Series*. London : San Diego : Academic Press, 1988.
- [43] Mehmet Ali Ertürk, Muhammed Ali Aydin, M. Talha Buyukakkaslar, and Hayrettin Evirgen. A survey on lorawan architecture, protocol and technologies. *Future Internet*, 11(10):216, 2019.
- [44] Xiangtao Liu, Tianle Zhang, Ning Hu, Peng Zhang, and Yu Zhang. The method of internet of things access and network communication based on mqtt. *Computer Communications*, 153:169 – 176, 2020.
- [45] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. Deep learning for anomaly detection: A review. *ACM Comput. Surv.*, 54(2):38:1–38:38, 2021.
- [46] Laura Erhan, Maryleen U. Ndubuaku, Mario Di Mauro, Wei Song, Min Chen, Giancarlo Fortino, Ovidiu Bagdasar, and Antonio Liotta. Smart anomaly detection in sensor systems: A multi-perspective review. *Inf. Fusion*, 67:64–79, 2021.
- [47] Gavneet Singh Chadha, Intekhab Islam, Andreas Schwung, and Steven X. Ding. Deep convolutional clustering-based time series anomaly detection. *Sensors*, 21(16):5488, 2021.
- [48] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. *IEEE Access*, 9:120043–120065, 2021.
- [49] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, and Fatima Mohamad Dakalbab. Machine learning for anomaly detection: A systematic review. *IEEE Access*, 9:78658–78700, 2021.
- [50] Ane Blázquez-García, Angel Conde, Usue Mori, and José Antonio Lozano. A review on outlier/anomaly detection in time series data. *ACM Comput. Surv.*, 54(3):56:1–56:33, 2021.

- [51] Xing Wang, Jessica Lin, Nital Patel, and Martin Braun. Exact variable-length anomaly detection algorithm for univariate and multivariate time series. *Data Min. Knowl. Discov.*, 32(6):1806–1844, 2018.
- [52] A. Gaddam, T. Wilkin, and M. Angelova. Anomaly detection models for detecting sensor faults and outliers in the iot - a survey. In *2019 13th International Conference on Sensing Technology (ICST)*, pages 1–6, 2019.
- [53] Muhammad Fahim and Alberto Sillitti. Anomaly detection, analysis and prediction techniques in iot environment: A systematic literature review. *IEEE Access*, 7:81664–81681, 2019.
- [54] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M. Shamim Hossain. Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet Things J.*, 8(8):6348–6358, 2021.
- [55] Andrew A. Cook, Goksel Misirli, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet Things J.*, 7(7):6481–6494, 2020.
- [56] Frank E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.
- [57] Chiara Bachechi, Federica Rollo, and Laura Po. Detection and classification of sensor anomalies for simulating urban traffic scenarios. *Cluster Computing Journal*, 2021.
- [58] Chiara Bachechi, Federica Rollo, Laura Po, and Fabio Quattrini. Anomaly detection in multivariate spatial time series: A ready-to-use implementation. In Francisco José Domínguez Mayo, Massimo Marchiori, and Joaquim Filipe, editors, *Proceedings of the 17th International Conference on Web Information Systems and Technologies, WEBIST 2021, October 26-28, 2021*, pages 509–517. SCITEPRESS, 2021.
- [59] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research (JAIR)*, 45, 11 2012.
- [60] Anitha Ramchandran and Arun Kumar Sangaiah. Chapter 11 - unsupervised anomaly detection for high dimensional data—an exploratory analysis. In Arun Kumar Sangaiah, Michael Sheng, and Zhiyong Zhang, editors, *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, Intelligent Data-Centric Systems, pages 233 – 251. Academic Press (2018).
- [61] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009.
- [62] Tony Luo and Sai Nagarajany. Distributed anomaly detection using autoencoder neural networks in wsn for iot. pages 1–6, 05 2018.
- [63] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19*, page 3009–3017, New York, NY, USA, 2019. Association for Computing Machinery.

- [64] Sheraz Naseer, Yasir Saleem, Shehzad Khalid, Muhammad Khawar Bashir, Jihun Han, Muhammad Munwar Iqbal, and Kijun Han. Enhanced network anomaly detection based on deep neural networks. *IEEE Access*, 6:48231–48246, 2018.
- [65] Benjamin Staar, Michael Lütjen, and Michael Freitag. Anomaly detection with convolutional neural networks for industrial surface inspection. *Procedia CIRP*, 79:484–489, 2019. 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy.
- [66] Jinbo Li, Hesam Izakian, Witold Pedrycz, and Iqbal Jamal. Clustering-based anomaly detection in multivariate time series data. *Appl. Soft Comput.*, 100:106919, 2021.
- [67] Mete Celik, Filiz Dadaser-Celik, and Ahmet Dokuz. Anomaly detection in temperature data using dbscan algorithm. 06 2011.
- [68] Julian von Schleinitz, Michael Graf, Wolfgang Trutschnig, and Andreas Schröder. VASP: an autoencoder-based approach for multivariate anomaly detection and robust time series prediction with application in motorsport. *Eng. Appl. Artif. Intell.*, 104:104354, 2021.
- [69] Sabyasachi Basu and Martin Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *Knowl. Inf. Syst.*, 11(2):137–154, 2007.
- [70] David Hill and Barbara Minsker. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling & Software*, 25:1014–1022, 09 2010.
- [71] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLOS ONE*, 11(4):1–31, 04 2016.
- [72] Mohammad Braei and Sebastian Wagner. Anomaly detection in univariate time-series: A survey on the state-of-the-art. *CoRR*, abs/2004.00433, 2020.
- [73] Qingsong Wen, Jingkun Gao, Xiaomin Song, Liang Sun, Huan Xu, and Shenghuo Zhu. Robuststl: A robust seasonal-trend decomposition algorithm for long time series. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 5409–5416, 2019.
- [74] İsmail Sezen, Alper Unal, and Ali Deniz. Anomaly detection by stl decomposition and extended isolation forest on environmental univariate time series. In *EGU General Assembly 2020, Online, 4–8 May 2020*, 2020.
- [75] Xi Wang and Chen Wang. Time series data cleaning: A survey. *IEEE Access*, 8:1866–1881, 2020.
- [76] Qin Yu, Lyu Jibin, and Lirui Jiang. An improved arima-based traffic anomaly detection algorithm for wireless sensor networks. *International Journal of Distributed Sensor Networks*, 12(1):9653230, 2016.
- [77] H. Zare Moayedi and M. A. Masnadi-Shirazi. Arima model for network traffic prediction and anomaly detection. In *2008 International Symposium on Information Technology*, volume 4, pages 1–6, 2008.

- [78] Siqi Liu, Adam Wright, and Milos Hauskrecht. Online conditional outlier detection in nonstationary time series. In *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2017, Marco Island, Florida, USA, May 22-24, 2017*, pages 86–91. AAAI Press.
- [79] Sooyeon Lee and Huy Kang Kim. Adsas: Comprehensive real-time anomaly detection system. In Brent ByungHoon Kang and Jin Soo Jang, editors, *Information Security Applications - 19th International Conference, WISA 2018, Jeju Island, Korea, August 23-25, 2018, Revised Selected Papers*, volume 11402 of *Lecture Notes in Computer Science*, pages 29–41. Springer.
- [80] Hongzhi Wang, Mohamed Jaward Bah, and Mohamed Hammad. Progress in outlier detection techniques: A survey. *IEEE Access*, 7:107964–108000, 2019.
- [81] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Trans. Knowl. Data Eng.*, 26(9):2250–2267, 2014.
- [82] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: identifying density-based local outliers. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 93–104. ACM, 2000.
- [83] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96*, page 226–231. AAAI Press, 1996.
- [84] Lian Duan, Lida Xu, Feng Guo, Jun Lee, and Baopin Yan. A local-density based spatial clustering algorithm with noise. *Inf. Syst.*, 32(7):978–986, 2007.
- [85] Maria Bala Duggimpudi, Shaaban Abbady, Jian Chen, and Vijay Raghavan. Spatio-temporal outlier detection algorithms based on computing behavioral outlierness factor. *Data & Knowledge Engineering*, 122:1–24, 07 2019.
- [86] Zheng Chen, Xinli Yu, Yuan Ling, Bo Song, Wei Quan, Xiaohua Hu, and Erjia Yan. Correlated anomaly detection from large streaming data. In *IEEE International Conference on Big Data, Big Data 2018*, pages 982–992, 2018.
- [87] Gustavo Souto and Thomas Liebig. Analyzing the correlation among traffic loop sensors to detect anomalies in traffic loop data streams. In BRACIS, editor, *Proceedings of the 3rd Symposium on Knowledge Discovery and Machine Learning*, 2015.
- [88] Nikolaos Zygouras, Nikolaos Panagiotou, Nikos Zacheilas, Ioannis Boutsis, Vana Kalogeraki, Ioannis Katakis, and Dimitrios Gunopulos. Towards detection of faulty traffic sensors in real-time. In *MUD@ICML*, volume 1392, 01 2015.
- [89] Zhongmin Wang, Guo-Hao Song, and Cong Gao. An isolation-based distributed outlier detection framework using nearest neighbor ensembles for wireless sensor networks. *IEEE Access*, 7:96319–96333, 2019.

- [90] L. Zhu, R. Krishnan, A. Sivakumar, F. Guo, and J. W. Polak. Traffic monitoring and anomaly detection based on simulation of luxembourg road network. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 382–387, 2019.
- [91] Javier Martínez, Angeles Saavedra, P. J. García Nieto, J. I. Piñeiro, Carla Iglesias, Javier Taboada, J. Sancho, and J. Pastor. Air quality parameters outliers detection using functional data analysis in the langreo urban area (northern spain). *Appl. Math. Comput.*, 241:1–10, 2014.
- [92] Jun Shen, Minhua Yang, Bin Zou, Neng Wan, and Yufang Liao. Outlier detection of air temperature series data using probabilistic finite state automata-based algorithm. *Complex.*, 17(5):48–57, 2012.
- [93] M.-F. Harkat, M. Mansouri, M. Nounou, and H. Nounou. Enhanced data validation strategy of air quality monitoring network. *Environmental Research*, 160:183–194, 2018.
- [94] N. Shaadan, A.A. Jemain, M.T. Latif, and S.M. Deni. Anomaly detection and assessment of pm10 functional data at several locations in the klang valley, malaysia. *Atmospheric Pollution Research*, 6(2):365–375, 2015.
- [95] L.-J. Chen, Y.-H. Ho, H.-H. Hsieh, S.-T. Huang, H.-C. Lee, and S. Mahajan. Adf: An anomaly detection framework for large-scale pm2.5 sensing systems. *IEEE Internet of Things Journal*, 5(2):559–570, 2018.
- [96] Thor-Bjørn Ottosen and P. Kumar. Outlier detection and gap filling methodologies for low-cost air quality measurements. *Environmental science. Processes & impacts*, 21 4:701–713, 2019.
- [97] Yue Cong. Anomaly detection in streaming data from air quality monitoring system. 2015.
- [98] Robert Cleveland. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6, 1990.
- [99] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5):561–580, October 2007.
- [100] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03*, pages 359–370, Seattle, Washington, USA, 1994.
- [101] Masud Moshtaghi, Christopher Leckie, Shanika Karunasekera, James C. Bezdek, Sutharshan Rajasegarar, and Marimuthu Palaniswami. Incremental elliptical boundary estimation for anomaly detection in wireless sensor networks. In Diane J. Cook, Jian Pei, Wei Wang, Osmar R. Zaiane, and Xindong Wu, editors, *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 467–476. IEEE Computer Society, 2011.

- [102] Masud Moshtaghi, James C. Bezdek, Timothy C. Havens, Christopher Leckie, Shanika Karunasekera, Sutharshan Rajasegarar, and Marimuthu Palaniswami. Streaming analysis in wireless sensor networks. *Wireless Communications and Mobile Computing*, 14(9):905–921, 2014.
- [103] A.M. Bianco, M. García Ben, E.J. Martínez, and V.J. Yohai. Outlier detection in regression models with arima errors using robust estimates. *J. Forecast.*, 20:565–579, 2001.
- [104] Antonio D’Ortona. Anomaly detection in low-cost air quality sensor data. Master’s thesis, 2020.
- [105] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.
- [106] Christopher Olah. Understanding lstm networks. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>, August 27, 2015.
- [107] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002, Helsinki, Finland, August 19-23, 2002, Proceedings*, volume 2431 of *Lecture Notes in Computer Science*, pages 15–26. Springer, 2002.
- [108] Markus Goldstein and Andreas R. Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. 2012.
- [109] Mia Hubert, Michiel Debruyne, and Peter J. Rousseeuw. Minimum covariance determinant and extensions. *WIREs Computational Statistics*, 10(3):e1421, 2018.
- [110] Jonathon Shlens. A tutorial on principal component analysis. *CoRR*, abs/1404.1100, 2014.
- [111] Zhana Bao. Robust subspace outlier detection in high dimensional space. *CoRR*, abs/1405.0869, 2014.
- [112] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 413–422. IEEE Computer Society, 2008.
- [113] Bernhard Schölkopf, Robert C. Williamson, Alexander J. Smola, John Shawe-Taylor, and John C. Platt. Support vector method for novelty detection. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 582–588. The MIT Press, 1999.
- [114] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. Angle-based outlier detection in high-dimensional data. In Ying Li, Bing Liu, and Sunita Sarawagi, editors, *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 444–452. ACM, 2008.

- [115] Bharath Sudharsan, John G Breslin, and other. Porting and execution of anomalies detection models on embedded systems in iot: Demo abstract. In *International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2021.
- [116] Principles - international open data charter. <https://opendatacharter.net/principles/>. International Open Data Charter; accessed: 2020-08-02.
- [117] Christian bizer and tom heath and tim berners-lee, linked data: Principles and state of the art. <https://www.w3.org/2008/Talks/WWW2008-W3CTrack-LOD.pdf>. Talk at the 17th International World Wide Web Conference W3C Track, at the WWW 2008; accessed: 2020-12-06.
- [118] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.
- [119] Federico Desimoni, Sergio Ilarri, Laura Po, Federica Rollo, and Raquel Trillo-Lado. Semantic traffic sensor data: The trafair experience. *Applied Sciences*, 10(17), 2020.
- [120] Julián Andrés Rojas Meléndez, Brecht Van de Vyvere, Arne Gevaert, Ruben Taelman, Pieter Colpaert, and Ruben Verborgh. A preliminary open data publishing strategy for live data in flanders. In Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, pages 1847–1853. ACM, 2018.
- [121] Xiaofei Xu, Quan Z. Sheng, Liang-Jie Zhang, Yushun Fan, and Schahram Dustdar. From Big Data to Big Service. *Computer*, 48(7):80–83, July 2015.
- [122] Shiraz Ahmed, Muhammad Adnan, Davy Janssens, Erika Brattich, Ansar ul Haque Yasar, Prashant Kumar, Silvana di Sabatino, and Elhadi M. Shakshuki. Estimating pro-environmental potential for the development of mobility-based informational intervention: a data-driven algorithm. *Personal and Ubiquitous Computing*, 23(5-.6):653–668, November 2018.
- [123] Francisco R. Soriano, Juan Javier Samper-Zapater, Juan Jose Martinez-Dura, Ramon V. Cirilo-Gimeno, and Javier Martinez Plume. Smart mobility trends: Open data and other tools. *IEEE Intelligent Transportation Systems Magazine*, 10(2):6–16, 2018.
- [124] Brecht Van de Vyvere, Pieter Colpaert, Erik Mannens, and Ruben Verborgh. Open traffic lights: a strategy for publishing and preserving traffic lights data. In Sihem Amer-Yahia, Mohammad Mahdian, Ashish Goel, Geert-Jan Houben, Kristina Lerman, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 966–971. ACM, 2019.
- [125] Mingqi Lv, Tieming Chen, Yifan Li, and Yinglong Li. Urban traffic congestion index estimation with open ubiquitous data. *J. Inf. Sci. Eng.*, 34(3):781–799, 2018.

- [126] Klaus Pollhammer, Thomas Novak, Philip Raich, Wolfgang Kastner, Albert Treytl, and Gabor Kovacs. Open traffic data platform for scenario-based control. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, October 23-26, 2016*, pages 4677–4682. IEEE, 2016.
- [127] Sergio Consoli, Valentina Presutti, Diego Reforgiato Recupero, Andrea G. Nuzzolese, Silvio Peroni, Misael Mongiovi, and Aldo Gangemi. Producing Linked Data for smart cities: The case of Catania. *Big Data Research*, 7:1–15, 2017.
- [128] Marijn Janssen, Ricardo Matheus, and Anneke Zuiderwijk. Big and Open Linked Data (BOLD) to create smart cities and citizens: Insights from smart energy and mobility cases. In *International Conference on Electronic Government (EGOV 2015)*, volume 9248 of *Lecture Notes in Computer Science*, pages 79–90. Springer, 2015.
- [129] LocaliData) Óscar Corcho (Ontology Engineering Group Universidad Politécnica de Madrid. Vocabulary to represent data about traffic (vocabulario para la representación de datos sobre tráfico). <http://vocab.linkeddata.es/datosabiertos/def/transporte/trafico>. Accessed: 2020-12-06.
- [130] Semantic sensor network ontology. <https://www.w3.org/TR/vocab-ssn>. W3C Recommendation, 19 October 2017; accessed: 2020-12-06.
- [131] Michael Compton, Payam Barnaghi, Luis Bermudez, Raúl García-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, Vincent Huang, Krzysztof Janowicz, W. David Kelsey, Danh Le Phuoc, Laurent Lefort, Myriam Leggieri, Holger Neuhaus, Andriy Nikolov, Kevin Page, Alexandre Passant, Amit Sheth, and Kerry Taylor. The SSN ontology of the W3C Semantic Sensor Network incubator group. *Journal of Web Semantics*, 17:25–32, 2012.
- [132] Krzysztof Janowicz, Armin Haller, Simon J.D. Cox, Danh Le Phuoc, and Maxime Lefrançois. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 56:1–10, May 2019.
- [133] AENOR (Spanish Association for Normalization). <https://www.aenor.com>. Accessed: 2020-12-06.
- [134] Jesús Vera, Manuel Tobarra, María Jesús Fernández, Óscar Corcho, and Víctor Morlán. Vocabulary to represent data of a city roadmap (vocabulario para la representación de datos de un callejero). <http://vocab.linkeddata.es/datosabiertos/def/urbanismo-infraestructuras/callejero>. Accessed: 2020-12-06.
- [135] Susel Fernandez, Rafik Hadfi, Takayuki Ito, Ivan Marsa-Maestre, and Juan Velasco. Ontology-based architecture for intelligent transportation systems using a traffic sensor network. *Sensors*, 16(8):1287, August 2016.
- [136] Xiaoming Zhang, YUnping Zhao, and Wanming Liu. A method for mapping sensor data to SSN ontology. *International Journal of u- and e- Service, Science and Technology*, 8(9):303–316, September 2015.
- [137] Open North. Open511 specification. <http://www.open511.org>. Accessed: 2020-12-06.

- [138] GeoNames. <https://www.geonames.org>. Accessed: 2020-12-06.
- [139] Dirk Ahlers. Assessment of the accuracy of GeoNames gazetteer data. In *Seventh Workshop on Geographic Information Retrieval (GIR 2013)*, pages 74–81. Association for Computing Machinery, 2013.
- [140] Daniel Dardailler. Road accident ontology – draft. <https://www.w3.org/2012/06/rao.html>. Accessed: 2020-12-06.
- [141] Javier Barrachina, Piedad Garrido, Manuel Fogue, Francisco J. Martinez, Juan-Carlos Cano, Carlos T. Calafate, and Pietro Manzoni. CAOVA: A car accident ontology for VANETs. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1864–1869, 2012.
- [142] Megan Katsumi and Mark Fox. An ontology-based standard for transportation planning. In *The Joint Ontology Workshops (JOWO 2019)*, volume 2518 of *CEUR Workshop Proceedings*, September 2019.
- [143] Enterprise Integration Lab, University of Toronto. Observations ontology. <http://ontology.eil.utoronto.ca/icity/Observations/1.0>. Accessed: 2020-12-06.
- [144] Pierfrancesco Bellini, Paolo Nesi, and Mirco Soderi. Km4City – the knowledge model 4 the city smart city ontology. <http://www.disit.org/5606>, <http://www.disit.org/km4city/schema>, 2018. Accessed: 2020-12-06.
- [145] María Poveda Villalón and Raúl García-Castro. Smart Appliances REference (SAREF). <https://ontology.tno.nl/saref>. Accessed: 2020-12-06.
- [146] Laura Daniele. FIEMSER ontology. <https://sites.google.com/site/smartappliancesproject/ontologies/fiemsers-ontology>. Accessed: 2020-12-06.
- [147] University of Southern California (USC). Karma – a data integration tool. <https://usc-isi-i2.github.io/karma>. Accessed: 2020-12-06.
- [148] Shubham Gupta, Pedro Szekely, Craig A. Knoblock, Aman Goel, Mohsen Taheriyani, and Maria Muslea. Karma: A system for mapping structured sources into the semantic web. In Elena Simperl, Barry Norton, Dunja Mladenic, Emanuele Della Valle, Irini Fundulaki, Alexandre Passant, and Raphaël Troncy, editors, *The Semantic Web: ESWC 2012 Satellite Events*, pages 430–434, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
- [149] Agile Knowledge Engineering and Semantic Web (AKSW) research group – University of Leipzig, Institute for Applied Informatics (InfAI). The Linked GeoData knowledge base. <http://linkedgeodata.org>.
- [150] LodLive Team. LodView. <https://lodview.it/>, <https://github.com/LodLive/LodView>. Accessed: 2020-12-06.
- [151] OpenLink Software. Virtuoso. <https://virtuoso.openlinksw.com>. Accessed: 2020-12-06.

- [152] Orri Erling and Ivan Mikhailov. RDF support in the virtuoso DBMS. In *Studies in Computational Intelligence*, pages 7–24. Springer, 2009.
- [153] W3C Semantic Web Interest Group. Basic Geo (WGS84 lat/long) Vocabulary. <https://www.w3.org/2003/01/geo>. Accessed: 2020-12-06.
- [154] DBpedia. The DBpedia Ontology. <https://wiki.dbpedia.org/services-resources/ontology>. Accessed: 2020-12-06.
- [155] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. PROV-O: The PROV Ontology. *W3C recommendation*, 30, 2013.
- [156] PROV-O: The PROV Ontology. <https://www.w3.org/TR/prov-o>. W3C Recommendation, 30 April 2013; accessed: 2020-12-06.
- [157] Panos Vassiliadis. A survey of extract–transform–load technology. *International Journal of Data Warehousing and Mining*, 5(3):1–27, July 2009.
- [158] Panos Vassiliadis, Alkis Simitsis, and Eftychia Baikousi. A taxonomy of ETL activities. In *ACM Twelfth International Workshop on Data Warehousing and OLAP (DOLAP 2009)*, pages 25–32. ACM, 2009.
- [159] R2RML: RDB to RDF Mapping Language. <https://www.w3.org/TR/r2rml>. W3C Recommendation, 27 September 2012; accessed: 2020-12-06.
- [160] University of Southern California (USC). Karma rdf generation service. <https://github.com/usc-isi-i2/Web-Karma/tree/master/karma-web-services/web-services-rdf>. Accessed: 2020-12-06.
- [161] Anna Kokolaki and Yannis Tzitzikas. Facetize: An interactive tool for cleaning and transforming datasets for facilitating exploratory search. *CoRR*, abs/1812.10734, 2018.
- [162] Metaweb Technologies, Inc. OpenRefine. <https://openrefine.org>, <https://github.com/OpenRefine>. Accessed: 2020-21-07. Created by Metaweb Technologies, Inc. and originally written and conceived by David Huynh, OpenRefine is now an open source project with several contributors.
- [163] Anastasia Dimou and Miel Vander Sande. RDF Mapping Language (RML). <https://rml.io/specs/rml>. Accessed: 2020-21-07. W3C, unofficial draft 15 July 2020. Ghent University – iMinds – Multimedia Lab.
- [164] Herminio García. ShExML. <http://shexml.herminiogarcia.com>. Accessed: 2020-21-07. WESO Research Group, University of Oviedo.
- [165] Pieter Heyvaert, Ben De Meester, and Anastasia Dimou. YARRML. <https://rml.io/yarrml>. Accessed: 2020-21-07. imec – Ghent University – IDLab.
- [166] Franz Inc. AllegroGraph. <https://allegrograph.com/products/allegrograph>. Accessed: 2020-21-07.
- [167] Eclipse Foundation, Inc. . RDF4J. <https://rdf4j.org>. Accessed: 2020-21-07.

- [168] Neo4j, Inc. Neo4J. <https://neo4j.com>. Accessed: 2020-21-07.
- [169] DataStax. Titan. <https://titan.thinkaurelius.com/>, <https://github.com/thinkaurelius/titan>. Accessed: 2020-21-07.
- [170] Ontotext. GraphDB. <http://graphdb.ontotext.com>. Accessed: 2020-21-07.
- [171] Stardog Union. Stardog. <https://www.stardog.com>. Accessed: 2020-21-07.
- [172] Harsh Thakkar, Yashwant Keswani, Mohnish Dubey, Jens Lehmann, and Sören Auer. Trying not to die benchmarking: Orchestrating RDF and graph data management solution benchmarks using LITMUS. In *13th International Conference on Semantic Systems*, Semantics2017, pages 120–127. ACM, 2017.
- [173] Universitat de Lleida. Rhizomer. <http://rhizomik.net/html/rhizomer/>. Accessed: 2020-21-07. Rhizomik initiative, GRIHO (Human-Computer Interaction and data integration) research group.
- [174] Josep Maria Brunetti, Roberto García, and Sören Auer. From overview to facets and pivoting for interactive exploration of semantic web data. *International Journal on Semantic Web & Information Systems*, 9(1):1—20, January 2013.
- [175] Andras Micsik. LODMilla. <https://www.dbpedia.org/community/lodmilla/>, 2016. Accessed: 2020-21-07.
- [176] András Micsik, Zoltán Tóth, and Sándor Turbucz. LODmilla: Shared visualization of Linked Open Data. In *Theory and Practice of Digital Libraries (TPDL) – Selected Workshops*, volume 416 of *Communications in Computer and Information Science*, pages 89–100. Springer, 2014.
- [177] José Negr ao. LODGVis. <https://github.com/joseolimpio/LODBrowser>. Accessed: 2020-12-06.
- [178] Danilo Barbosa Coimbra, José Olímpio Mouzinho Negrão, and Frederico Araujo Durão. LODGVis: An interactive visualization for linked open data navigation. In *25th Brazilian Symposium on Multimedia and the Web (WebMedia)*, pages 433–440. ACM, 2019.
- [179] Diego Valerio Camarda, Silvia Mazzini, and Alessandro Antonuccio. Lodlive, exploring the web of data. In *I-SEMANTICS '12*, 2012.
- [180] David C. Faye, Olivier Curé, and Guillaume Blin. A survey of RDF storage approaches. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, 15:11–35, 2012.
- [181] Gianfranco E. Modoni, Marco Sacco, and Walter Terkaj. A survey of RDF store solutions. In *2014 International Conference on Engineering, Technology and Innovation (ICE)*, pages 1–7, 2014.
- [182] Zongmin Ma, Miriam A. M. Capretz, and Li Yan. Storing massive Resource Description Framework (RDF) data: a survey. *The Knowledge Engineering Review*, 31(4):391–413, 2016.

- [183] Kiyoshi Nitta and Iztok Savnik. Survey of RDF storage managers. In *Sixth International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA)*, pages 148–153. IARIA, 2014.
- [184] M. Tamer Özsu. A survey of RDF data management systems. *Frontiers of Computer Science*, 10(3):418–432, April 2016.
- [185] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising Linked Data: A survey. *Semantic Web*, 2(2):89–124, April 2011.
- [186] Karwan Jacksi, Nazife Dimililer, and Subhi R. M. Zeebaree. State of the art exploration systems for Linked Data: A review. *International Journal of Advanced Computer Science and Applications*, 7(11):155–164, November 2016.
- [187] F. Antoniazzi and F. Viola. RDF graph visualization tools: a survey. In *23rd Conference of Open Innovations Association (FRUCT)*, pages 25–36. IEEE, 2018.
- [188] Federico Desimoni and Laura Po. Empirical evaluation of Linked Data visualization tools. *Future Generation Computer Systems*, 112:258–282, 2020.
- [189] Jozef Ristvej, Maros Lacinák, and Roman Ondrejka. On smart city and safe city concepts. *Mob. Networks Appl.*, 25(3):836–845, 2020.
- [190] Devi Mega Risdiana and Tony Dwi Susanto. The safe city: Conceptual model development - a systematic literature review. *Procedia Computer Science*, 161:291–299, 2019. The Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia.
- [191] Giles Oatley and Brian Ewart. Data mining and crime analysis. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 1(2):147–153, 2011.
- [192] Aries M. Gelera and Edgardo S. Dajao. Crime mapping approach for crime pattern identification: A prototype for the province of cavite. In Xin-She Yang, Simon Sherratt, Nilanjan Dey, and Amit Joshi, editors, *Proceedings of Sixth International Congress on Information and Communication Technology - ICICT 2021, London, UK, Volume 2*, volume 236 of *Lecture Notes in Networks and Systems*, pages 899–909. Springer, 2021.
- [193] Alina Ristea and Michael Leitner. Urban crime mapping and analysis using GIS. *ISPRS Int. J. Geo Inf.*, 9(9):511, 2020.
- [194] Srinivasa K and P. Santhi Thilagam. Crime base: Towards building a knowledge base for crime entities and their relationships from online news papers. *Information Processing and Management*, 56(6), 2019.
- [195] Laura Po and Federica Rollo. Building an urban theft map by analyzing newspaper crime reports. In *13th International Workshop on Semantic and Social Media Adaptation and Personalization, SMAP Zaragoza, Spain*, pages 13–18, 2018.

- [196] Tirthankar Dasgupta, Abir Naskar, Rupsa Saha, and Lipika Dey. Crimeprofiler: Crime information extraction and visualization from news media. In Amit P. Sheth, Axel Ngonga, Yin Wang, Elizabeth Chang, Dominik Slezak, Bogdan Franczyk, Rainer Alt, Xiaohui Tao, and Rainer Unland, editors, *Proceedings of the International Conference on Web Intelligence, WI '17*, page 541–549, New York, NY, USA, 2017. Association for Computing Machinery.
- [197] Prathap Rudra Boppuru and Ramesha Kenchappa. Geo-spatial crime analysis using newsfeed data in indian context. *Int. J. Web Based Learn. Teach. Technol.*, 14(4):49–64, 2019.
- [198] Vasu Sharma, Rajat Kulshreshtha, Puneet Singh, Nishant Agrawal, and Akshay Kumar. Analyzing newspaper crime reports for identification of safe transit paths. In Rada Mihalcea, Joyce Yue Chai, and Anoop Sarkar, editors, *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 17–24. The Association for Computational Linguistics, 2015.
- [199] Kaihang Zhang, Chuang Zhang, Xiaojun Chen, and Jianlong Tan. Automatic web news extraction based on DS theory considering content topics. In *Proc. of International Conference*, volume 10860 of *Lecture Notes in Computer Science*, pages 194–207. Springer, 2018.
- [200] Chandrakala Arya and Sanjay K. Dwivedi. Content extraction from news web pages using tag tree. *Int. J. Auton. Comp.*, 3(1):34–51, 2018.
- [201] Basanta Chaulagain, Aman Shakya, Bhuwan Bhatt, Dip Kiran Pradhan Newar, Sanjeeb Prasad Panday, and Rom Kant Pandey. Casualty information extraction and analysis from news. In *Proc. of the International Conference on Information Systems for Crisis Response and Management*. ISCRAM Association, 2019.
- [202] Jakub Piskorski, Vanni Zavarella, Martin Atkinson, and Marco Verile. Timelines: Entity-centric event extraction from online news. In *Proc. of Text2Story - Third Workshop on Narrative Extraction From Texts*, volume 2593 of *CEUR Workshop Proceedings*, pages 105–114. CEUR-WS.org, 2020.
- [203] Mohammad Reza Keyvanpour, Mostafa Javideh, and Mohammad Reza Ebrahimi. Detecting and investigating crime by means of data mining: a general crime matching framework. *Procedia Computer Science*, 3:872 – 880, 2011.
- [204] Tong Wang, Cynthia Rudin, Daniel Wagner, and Rich Sevieri. Learning to detect patterns of crime. In *Machine Learning and Knowledge Discovery in Databases*, pages 515–530, 2013.
- [205] GC Oatley, John Zeleznikow, and BW Ewart. Matching and predicting crimes. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 19–32. Springer, 2004.
- [206] A. Alqahtani, A. Garima, and A. Alaiad. Crime analysis in chicago city. In *International Conference on Information and Communication Systems (ICICS)*, pages 166–172, 2019.

- [207] Laura Po, Federica Rollo, and Raquel Trillo Lado. Topic detection in multichannel italian newspapers. In Andrea Cali, Dorian Gorgan, and Martín Ugarte, editors, *Semantic Keyword-Based Search on Structured Data Sources - COST Action IC1302 Second International KEYSTONE Conference, IKC 2016, Cluj-Napoca, Romania, September 8-9, 2016, Revised Selected Papers*, Lecture Notes in Computer Science, pages 62–75, 2016.
- [208] Federica Rollo. A key-entity graph for clustering multichannel news: student research abstract. In *Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco*, pages 699–700, 2017.
- [209] Sonia Bergamaschi, Laura Po, and Serena Sorrentino. Comparing topic models for a movie recommendation system. In Valérie Monfort and Karl-Heinz Krempels, editors, *WEBIST 2014 - Proceedings of the 10th International Conference on Web Information Systems and Technologies, Volume 2, Barcelona, Spain, 3-5 April, 2014*, pages 172–183. SciTePress, 2014.
- [210] Laura Po and Davide Malvezzi. Community detection applied on big linked data. *J. UCS*, 24(11):1627–1650, 2018.
- [211] David B. Bracewell, Jiajun Yan, Fuji Ren, and Shingo Kuroiwa. Category classification and topic discovery of japanese and english news articles. *Electron. Notes Theor. Comput. Sci.*, 225:51–65, 2009.
- [212] Tao Jiang, Jian Ping Li, Amin Ul Haq, Abdus Saboor, and Amjad Ali. A novel stacking approach for accurate detection of fake news. *IEEE Access*, 9:22626–22639, 2021.
- [213] Tien Huu Do, Marc Berneman, Jasabanta Patro, Giannis Bekoulis, and Nikos Deligiannis. Context-aware deep markov random fields for fake news detection. *IEEE Access*, 9:130042–130054, 2021.
- [214] Rohit Kumar Kaliyar, Anurag Goswami, and Pratik Narang. Fakebert: Fake news detection in social media with a bert-based deep learning approach. *Multim. Tools Appl.*, 80(8):11765–11788, 2021.
- [215] Ankita Dhar, Himadri Mukherjee, Niladri Sekhar Dash, and Kaushik Roy. Text categorization: past and present. *Artif. Intell. Rev.*, 54(4):3007–3054, 2021.
- [216] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. A survey on text classification: From shallow to deep learning. *CoRR*, abs/2008.00364, 2020.
- [217] Congcong Wang, Paul Nulty, and David Lillis. A comparative study on word embeddings in deep learning for text classification. In *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval, NLPPIR 2020*, page 37–46, New York, NY, USA, 2020. Association for Computing Machinery.
- [218] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. Word-class embeddings for multiclass text classification. *Data Min. Knowl. Discov.*, 35(3):911–963, 2021.

- [219] Awet Fesseha, Shengwu Xiong, Eshete Derb Emiru, Moussa Diallo, and Abdelghani Dahou. Text classification based on convolutional neural networks and word embedding for low-resource languages: Tigrinya. *Inf.*, 12(2):52, 2021.
- [220] Anton Borg, Martin Boldt, Oliver Rosander, and Jim Ahlstrand. E-mail classification with machine learning and word embeddings for improved customer support. *Neural Comput. Appl.*, 33(6):1881–1902, 2021.
- [221] Evripides Christodoulou, Andreas Gregoriades, Maria Pampaka, and Herodotos Herodotou. Application of classification and word embedding techniques to evaluate tourists’ hotel-revisit intention. In Joaquim Filipe, Michal Smialek, Alexander Brodsky, and Slimane Hammoudi, editors, *Proceedings of the 23rd International Conference on Enterprise Information Systems, ICEIS 2021, Online Streaming, April 26-28, 2021, Volume 1*, pages 216–223. SCITEPRESS, 2021.
- [222] Piotr Semberecki and Henryk Maciejewski. Deep learning methods for subject text classification of articles. In Maria Ganzha, Leszek A. Maciaszek, and Marcin Paprzycki, editors, *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017, Prague, Czech Republic, September 3-6, 2017*, volume 11 of *Annals of Computer Science and Information Systems*, pages 357–360, 2017.
- [223] Martin Vita and Vincent Kríz. Word2vec based system for recognizing partial textual entailment. In Maria Ganzha, Leszek A. Maciaszek, and Marcin Paprzycki, editors, *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, FedCSIS 2016, Gdańsk, Poland, September 11-14, 2016*, volume 8 of *Annals of Computer Science and Information Systems*, pages 513–516. IEEE, 2016.
- [224] Tom Lin. Performance of different word embeddings on text classification. <https://towardsdatascience.com/nlp-performance-of-different-word-embeddings-on-text-classification-de648c6262b>, 2019. Accessed: 7 June 2021.
- [225] Joseph Lilleberg, Yun Zhu, and Yanqing Zhang. Support vector machines and word2vec for text classification with semantic features. In Ning Ge, Jianhua Lu, Yingxu Wang, Newton Howard, Philip Chen, Xiaoming Tao, Bo Zhang, and Lotfi A. Zadeh, editors, *14th IEEE International Conference on Cognitive Informatics & Cognitive Computing, ICCI*CC 2015, Beijing, China, July 6-8, 2015*, pages 136–140. IEEE Computer Society, 2015.
- [226] Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.
- [227] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 07 2016.
- [228] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomáš Mikolov. Bag of tricks for efficient text classification. In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the*

- Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 427–431. Association for Computational Linguistics, 2017.
- [229] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, volume 14, pages 1532–1543, 01 2014.
- [230] Yoon Kim. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL, 2014.
- [231] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2267–2273. AAAI Press, 2015.
- [232] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657, 2015.
- [233] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 562–570. Association for Computational Linguistics, 2017.
- [234] Adji B. Dieng, Chong Wang, Jianfeng Gao, and John W. Paisley. Topicrnn: A recurrent neural network with long-range semantic dependency. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [235] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566. The Association for Computer Linguistics, 2015.
- [236] Mireya Tovar Vidal, Emmanuel Santos Rodríguez, and José Alejandro Reyes-Ortíz. Classification of criminal news over time using bidirectional LSTM. In Yue Lu, Nicole Vincent, Pong Chi Yuen, Wei-Shi Zheng, Farida Cheriet, and Ching Y. Suen, editors, *Pattern Recognition and Artificial Intelligence - International Conference, ICPRAI*

- 2020, Zhongshan, China, October 19-23, 2020, *Proceedings*, volume 12068 of *Lecture Notes in Computer Science*, pages 702–713. Springer, 2020.
- [237] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 551–561. The Association for Computational Linguistics, 2016.
- [238] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.
- [239] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764, 2019.
- [240] Abhishek Sanwaliya, Kripa Shanker, and Subhas C. Misra. Categorization of news articles: A model based on discriminative term extraction method. In Fritz Laux and Lena Strömbäck, editors, *The Second International Conference on Advances in Databases, Knowledge, and Data Applications, DBKDA 2010, Menuires, France, 11-16 April 2010*, pages 149–154. IEEE Computer Society, 2010.
- [241] Mayy Tahrawi. Arabic text categorization using logistic regression. *International Journal of Intelligent Systems and Applications*, 7:71–78, 05 2015.
- [242] Rini Wongso, Ferdinand Ariandy Luwinda, Brandon Christian Trisnajaya, Olivia Rusli, and Rudy. News article text classification in indonesian language. In *ICCSCI*, pages 137–143, 2017.
- [243] Pietro Totis and Manfred Stede. Classifying italian newspaper text: news or editorial? In Elena Cabrio, Alessandro Mazzei, and Fabio Tamburini, editors, *Proceedings of the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Torino, Italy, December 10-12, 2018*, volume 2253 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.
- [244] Francesco Camastra and Gennaro Razi. Italian text categorization with lemmatization and support vector machines. In Anna Esposito, Marcos Faúndez-Zanuy, Francesco Carlo Morabito, and Eros Pasero, editors, *Neural Approaches to Dynamics of Signal Exchanges*, volume 151 of *Smart Innovation, Systems and Technologies*, pages 47–54. Springer, 2020.

- [245] Alessandro Bondielli, Pietro Ducange, and Francesco Marcelloni. Exploiting categorization of online news for profiling city areas. In *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS 2020, Bari, Italy, May 27-29, 2020*, pages 1–8. IEEE, 2020.
- [246] Yiqi Bai and Jie Wang. News classifications with labeled LDA. In Ana L. N. Fred, Jan L. G. Dietz, David Aveiro, Kecheng Liu, and Joaquim Filipe, editors, *KDIR 2015 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015), Volume 1, Lisbon, Portugal, November 12-14, 2015*, pages 75–83. SciTePress, 2015.
- [247] Zhenzhong Li, Wenqian Shang, and Menghan Yan. News text classification model based on topic model. In *15th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2016, Okayama, Japan, June 26-29, 2016*, pages 1–5. IEEE Computer Society, 2016.
- [248] Chunhui He, Yanli Hu, Aixia Zhou, Zhen Tan, Chong Zhang, and Bin Ge. A web news classification method: Fusion noise filtering and convolutional neural network. In *SSPS 2020: 2020 2nd Symposium on Signal Processing Systems, Guangdong China, July, 2020*, pages 80–85. ACM, 2020.
- [249] Yunlong Zhu. Research on news text classification based on deep learning convolutional neural network. *Wireless Communications and Mobile Computing*, 2021:1–6, 12 2021.
- [250] Jiajia Duan, Hui Zhao, Wenshuai Qin, Meikang Qiu, and Meiqin Liu. News text classification based on MLCNN and bigru hybrid neural network. In *3rd International Conference on Smart BlockChain, SmartBlock 2020, Zhengzhou, China, October 23-25, 2020*, pages 137–142. IEEE, 2020.
- [251] Dohyung Kim, Jahwan Koo, and Ung-Mo Kim. Envbert: Multi-label text classification for imbalanced, noisy environmental news data. In Sukhan Lee, Hyunseung Choo, and Roslan Ismail, editors, *15th International Conference on Ubiquitous Information Management and Communication, IMCOM 2021, Seoul, South Korea, January 4-6, 2021*, pages 1–8. IEEE, 2021.
- [252] Kuncahyo Setyo Nugroho, Anantha Yullian Sukmadewa, and Novanto Yudistira. Large-scale news classification using BERT language model: Spark NLP approach. *CoRR*, abs/2107.06785, 2021.
- [253] Tipajin Thaipisutikul, Suppawong Tuarob, Siripen Pongpaichet, Amornsri Amornvatcharapong, and Timothy K. Shih. Automated classification of criminal and violent activities in thailand from online news articles. In *13th International Conference on Knowledge and Smart Technology, KST 2021, Bangsaen, Chonburi, Thailand, January 21-24, 2021*, pages 170–175. IEEE, 2021.
- [254] Lisna Zahrotun. Comparison jaccard similarity, cosine similarity and combined both of the data clustering with shared nearest neighbor method. *Computer Engineering and Applications*, 5:11–18, 2016.

- [255] Yasunao Takano, Yusuke Iijima, Kou Kobayashi, Hiroshi Sakuta, Hiroki Sakaji, Masaki Kohana, and Akio Kobayashi. Improving document similarity calculation using cosine-similarity graphs. In Leonard Barolli, Makoto Takizawa, Fatos Xhafa, and Tomoya Enokido, editors, *Advanced Information Networking and Applications - Proceedings of the 33rd International Conference on Advanced Information Networking and Applications, AINA 2019, Matsue, Japan, March 27-29, 2019*, volume 926 of *Advances in Intelligent Systems and Computing*, pages 512–522. Springer, 2019.
- [256] Jiapeng Wang and Yihong Dong. Measurement of text similarity: A survey. *Information*, 11(9), 2020.
- [257] Dimas Wibisono Prakoso, Asad Abdi, and Chintan Amrit. Short text similarity measurement methods: a review. *Soft Comput.*, 25:4699–4723, 2021.
- [258] Wael Hassan Gomaa and Aly A. Fahmy. A survey of text similarity approaches. *International Journal of Computer Applications*, 68:13–18, 2013.
- [259] Peter Foltz. Latent semantic analysis for text-based research. *Behavior Research Methods*, 28:197–202, 02 1996.
- [260] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 1606–1611, 2007.
- [261] Karl-Michael Schneider. Weighted average pointwise mutual information for feature selection in text categorization. In Alipio Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama, editors, *Knowledge Discovery in Databases: PKDD 2005, 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, October 3-7, 2005, Proceedings*, volume 3721 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2005.
- [262] Federica Rollo and Laura Po. Crime event localization and deduplication. In Jeff Z. Pan, Valentina A. M. Tamma, Claudia d’Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal, editors, *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II*, volume 12507 of *Lecture Notes in Computer Science*, pages 361–377. Springer, 2020.
- [263] Giovanni Bonisoli, Federica Rollo, and Laura Po. Using word embeddings for italian crime news categorization. In Maria Ganzha, Leszek A. Maciaszek, Marcin Paprzycki, and Dominik Slezak, editors, *Proceedings of the 16th Conference on Computer Science and Intelligence Systems, Online, September 2-5, 2021*, pages 461–470, 2021.
- [264] Po L. Rollo F., Bonisoli G. Supervised and unsupervised categorization of an imbalanced italian crime news dataset. *Lecture Notes in Business Information Processing*, 2022. (to appear).
- [265] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In

- Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.
- [266] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. *Automatic Keyword Extraction from Individual Documents*, chapter 1, pages 1–20. John Wiley & Sons, Ltd, 2010.
- [267] Lov Kumar, Mukesh Kumar, Lalita Bhanu Murthy Neti, Sanjay Misra, Vipul Kocher, and Srinivas Padmanabhuni. An empirical study on application of word embedding techniques for prediction of software defect severity level. In Maria Ganzha, Leszek A. Maciaszek, Marcin Paprzycki, and Dominik Slezak, editors, *Proceedings of the 16th Conference on Computer Science and Intelligence Systems, Online, September 2-5, 2021*, pages 477–484, 2021.
- [268] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 06 2002.
- [269] A. Palmero Aprosio and G. Moretti. Italy goes to Stanford: a collection of CoreNLP modules for Italian. *ArXiv e-prints*, 2016.
- [270] Giuseppe Attardi, Stefano Dei Rossi, and Maria Simi. The tanl pipeline. In *Proc. of the Workshop on Web Services and Processing Pipelines in HLT, co-located LREC*, 2010.
- [271] Emanuele Pianta, Christian Girardi, Roberto Zanolli, and Fondazione Bruno Kessler. The textpro tool suite. In *Proc. of LREC-08*, 2008.
- [272] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing, Manchester, UK*, 1994.
- [273] Jannik Strötgen and Michael Gertz. Heideitime: High quality rule-based extraction and normalization of temporal expressions. In *Proc. of the International Workshop on Semantic Evaluation, SemEval@ACL*, pages 321–324, 2010.
- [274] Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. Linkedgeodata: A core for a web of spatial open data. *Semant. Web*, 3:333–354, October 2012.
- [275] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In Chiara Ghidini, Axel-Cyrille Ngonga Ngomo, Stefanie N. Lindstaedt, and Tassilo Pellegrini, editors, *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, ACM International Conference Proceeding Series, pages 1–8. ACM, 2011.
- [276] Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer networks and ISDN systems*, 29(8-13):1157–1166, 1997.

-
- [277] Omar Alonso, Dennis Fetterly, and Mark Manasse. Duplicate news story detection revisited. In Rafael E. Banchs, Fabrizio Silvestri, Tie-Yan Liu, Min Zhang, Sheng Gao, and Jun Lang, editors, *Information Retrieval Technology*, pages 203–214, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [278] Giovanni Gennaro, Amedeo Buonanno, Antonio Girolamo, Armando Ospedale, Francesco Palmieri, and Gianfranco Fedele. *An Analysis of Word2Vec for the Italian Language*, pages 137–146. 01 2021.
- [279] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.
- [280] Giovanni Bonisoli. Crime analysis using word embeddings: the case of an italian news dataset. Master’s thesis, 2021.