INSIGHT
FP7-318225

D5.2:

Report on the integration and the evaluation of the alarming, visualization and prediction component

TU Dortmund University

August 31, 2015

Status: Final
Scheduled Delivery Date: 31/08/2015

Executive Summary

The three aspects of the INSIGHT system are: (1) Alarming, Prediction and Alarm Classification methods - discussed in WP5; (2) INSIGHT System integration and its unit testes - presented in WP2; (3) end-user assessment of the use cases described in WP6.

The report at-hand presents the progress towards the objectives of Work Package 5 (WP5). We discuss the evaluation of the Alarming, Prediction and Alarm Classification methods (mostly reported in WP3) performed on the use scenarios using previously defined test data. The Deliverable 2.3 contains the discussion of the INSIGHT system integration and unit tests, whereas the end-user assessment is presented in Deliverable 6.2.

Separate testing of these aspects is important as even with a relatively low accuracy of some prediction or event detection method the resulting alarms may provide high utility to the end-user (or vice versa).

Additionally, this deliverable reports how end-users may interact with the algorithms and how results are visualized in the user interface.

Document Information

Contract Number	FP7-318225	**Acronym**	INSIGHT
Full Title	Intelligent Synthesis and Real-Time Response using Massive Streaming of Heterogeneous Data		
Project URL	http://www.insight-ict.eu/		
EU Project Officer	Ms. Alina Lupu		

Deliverable	**Number D5.2**	Report on the integration and the evaluation of the alarming, visualization and prediction component		
Work Package	**Number**	WP5		
Date of Delivery	31/08/2015	**Actual**	31/08/2015	
Status	Final			
Nature	Report			
Distribution Type	Public			
Authoring Partner	TU Dortmund University			
QA Partner	UoA			
Contact Person	Thomas Liebig	thomas.liebig@cs.tu-dortmund.de		
	Katharina Morik	katharina.morik@cs.uni-dortmund.de		
	Phone		**Fax**	

Project Information

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number FP7-318225. The Beneficiaries in this project are:

No.	Name	Short Name	Country
1	National and Kapodistrian University of Athens	UoA	Greece
2	IBM Ireland Product Distribution Limited	IBM	Ireland
3	Fraunhofer-Gesellschaft Zur Förderung Der Angewandten Forschung E.V.	Fraunhofer	Germany
4	Technische Universität Dortmund	TUD	Germany
5	Technion - Israel Institute of Technology	Technion	Israel
6	Dublin City Council	DCC	Ireland
7	Bundesamt für Bevölkerungsschutz und Katastrophenhilfe	BBK	Germany

Table of Contents

Index of Figures

1 Introduction

This deliverable serves as a final report for the tasks in Work Package 5, "Real-Time Alarm & Prediction". We followed a task-based organization of this deliverable in order to demonstrate the achieved results regarding the WP5 goals. Hence, we present methods to: (1) predict and categorize situations by analyzing different sources of data and (2) to visualize results to the end-user. The data is used to train prediction models based on historical information.

The real-time analysis of heterogeneous data streams poses new challenges on existing methods. Whereas existing preprocessing and analysis methods could use multiple scans, real-time analysis does not have the possibility to look twice and has to perform its tasks in a single-scan. Thus, besides off-line learning from batch data, analysis and prediction methods which are capable of working on data streams are hereby presented. Another critical aspect of the prediction and anomaly detection is that it is required to be easily interpreted by the officers/decision makers of an emergency team.

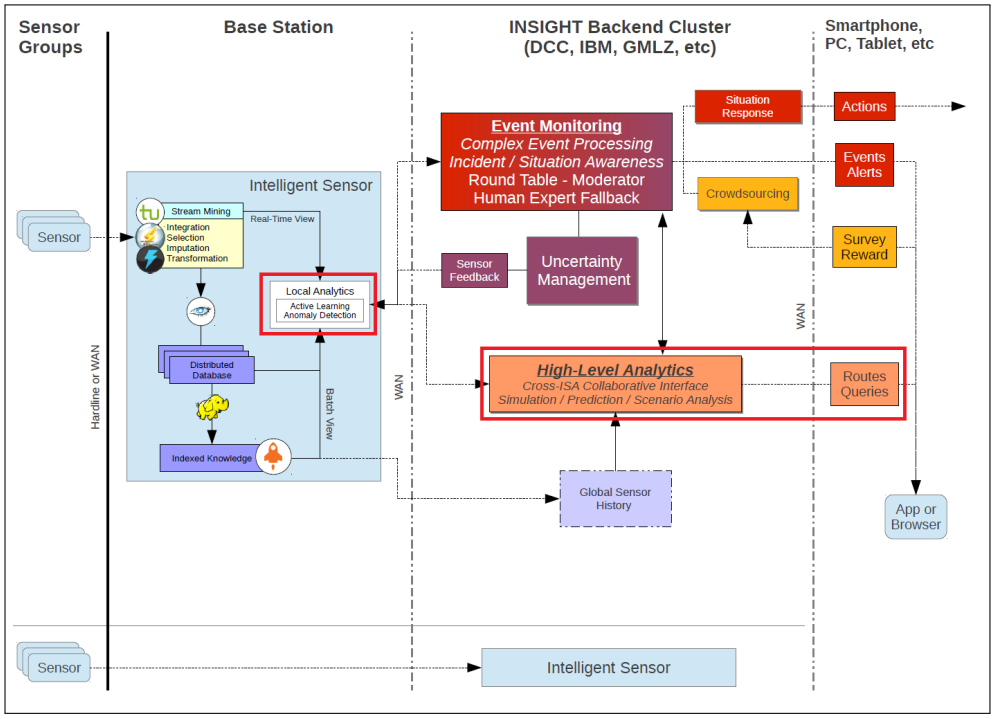


Figure 1: Deliverable D5.2 in respect to the INSIGHT Architecture as designed in D2.1.

D5.2 & the INSIGHT Architecture Figure 1 indicates the contribution of D5.2 in the context of the INSIGHT architecture as designed in Deliverable 2.1 ("System Requirements, Specifications, Standards and Guidelines for Development and Architecture"). As can be seen in this Figure, the prediction and analysis methods in D5.2 take input either in a streaming or a batch fashion. Their output is forwarded mainly to the Event Monitoring module which is implemented by the *Round Table component*. The main goal of D5.2 is to provide with methods that analyse the raw or pre-processed spatio-temporal data streams of the use-case scenarios as described in D5.1, D3.1 and D3.2 and provide evidence that an anomaly or event

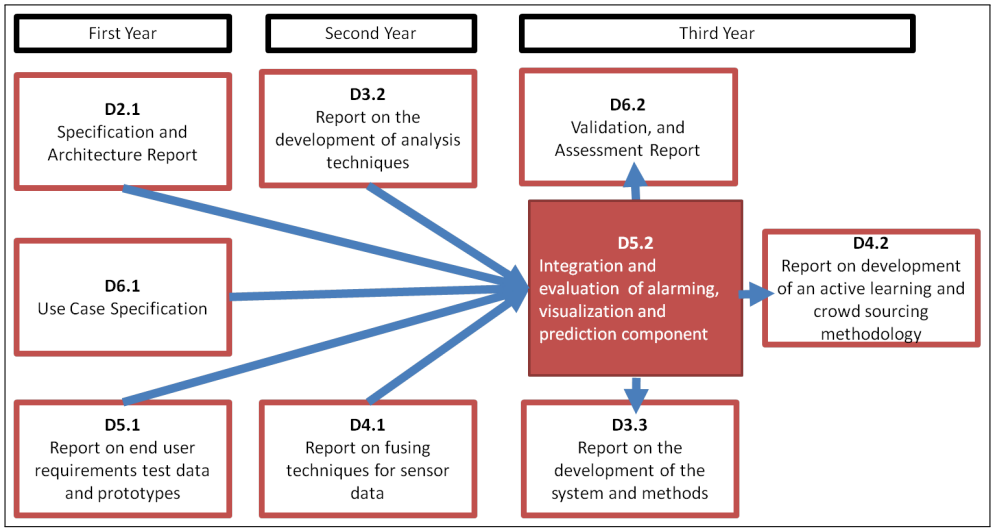


Figure 2: Dependency between D5.2 and other deliverables. Note that we only present relations with D5.2. There are many dependencies among the rest of the deliverables as well.

(see D5.1) occurred in specific place and time. As seen in the architecture, local analytics modules comprise the core of the Intelligent Sensor Agents (ISAs). In INSIGHT, multiple ISAs are maintained, each one implemented in its own lambda architecture (see D2.1). Therefore, there is a Twitter ISA, a BUS ISA, a SCATS ISA, a Mobile Phone ISA, etc. Please see D5.1 for description of the data sources and data format. The common language between the ISAs and the *Round Table*, is specified by a protocol defined in D4.1 and the *Ontology* of incidents that is defined in Deliverable D2.2.

Connection of D5.2 with other Deliverables A dependency graph of D5.2 with respect to other INSIGHT deliverables is depicted in Figure 2. As can be seen in this figure, the analysis methods are designed based on the requirements and user specifications collected in D2.1 and D6.1 respectively. Preliminary analysis and data exploration has been conducted in D5.1. Further methods for analysis of the heterogeneous data streams are conducted in D3.2, D3.3 and D4.2. The output of these local analysis models output will be fed mainly to the *Round Table* (D4.1). The crowdsourcing methods are described in D4.2, and in the report at-hand we show its user interface. Hence, the evaluation of the analysis methods developed in INSIGHT is three-fold, while D2.3 verifies that the implementation of the INSIGHT system is running and well designed, D5.2 verifies the quality of prediction, analysis and visualization methods for the use cases (reported in D6.1). Their use-case assessment will be reported in D6.2.

Connection of D5.2 with WP5 tasks

- *Task 5.2 - Alarming* Given the high volume data streams provided in the city-level and nation-wide use case scenario, requirements for online alarm detection are derived. Section 2 especially, addresses the more complex cases of heterogeneous data. In succeeding Section 3 models for online alarm detection are trained and applied to detect anomalies and complex alert events within the disaster management scenarios.

- *Task 5.3 - Prediction/Alarm Classification* In this task we developed methods for prediction and categorization of situations by analyzing different sources of data to provide correct assessment of an emergency situation (Section 3).
- *Task 5.4: - Visualization* The objective of this task was to provide a visualization of the outcomes of Alarming (T5.2) and Prediction (T5.3) to the end users. Within a city/situation map, the events and predictions are displayed to the end users, allowing for displaying alarms and predictions at different levels of granularity with respect to time and spatial dimensions (Section 4).

Methods included in Deliverable 5.2 The deliverable at-hand presents the methods included in the analysis components of the INSIGHT system and their user interface. An overview on the methods included in these ISAs are given in Table 1. The table is organized by data source.

Data Source	**Intelligent Sensor Agent (ISA)**	**Methods**	**Section**
SCATS	Prediction and Route Planning	Gaussian Process Regression, Spatio-Temporal-Random-Fields	3.1
SCATS	Outlier Detection	Statistical Analysis	3.2
BUS	Delay Time Prediction	Queueing Theory, Random Forests	3.3
BUS	Outlier Detection	Complex Rule Based Filtering	3.4
Mobile Phone Data	Outlier Detection	Statistical Analysis	3.5
Twitter	Identifying Traffic	Spatial Matching, Wordlists	3.6
Twitter	Semantic Outlier Detection	Deep Neural Networks	3.7

Table 1: Methods applied in Deliverable D5.2.

2 Requirements of Alarming and Prediction Components

The INSIGHT application scenarios demonstrate the benefits of Big Data Analytics for public safety in the area of civil protection. As a real-world test-bed, we have chosen two complementing and challenging scenarios of high public interest: traffic monitoring in cities, here the City of Dublin, and monitoring of nation-wide disasters – here: flooding in Germany.

This section is organized as follows. After a review of end-users requirements, we derive required capabilities of real-time spatio-temporal event detection, and review the methods for analysis of spatio-temporal data from deliverable D5.1. The section leads over to the next section that focuses on the prediction and alarm classification task.

2.1 End-Users Requirements to INSIGHTs Alarming Methods

Based on early demonstrators of the INSIGHT system (compare project's annual reports) the end-users identified domain specific potential of the real-time situation awareness provided by the INSIGHT system and detailed the requirements to the autonomous event identification system (EIS).

In the following we will review requirements that have been identified together with the City of Dublin City Council (DCC) and the Federal Office for Civil Protection and Disaster Assistance in Germany (BBK) [SSBS15, KSM+14]. To ensure interoperability and transferability we included city-level and nation-wide perspectives. From a professional perspective we aim at next generation Emergency Information Systems that build on top of Data Systems to be scalable, flexible, fault tolerant, secure, trustable, collaborative, relevant, and prepared for automation.

- **Scalability** has two aspects: One is the range of the EIS scaling from city-level to nation-wide foci; and the second aspect comes from a system design perspective and means horizontal scalability which states that the complexity of the underlying big data system is obscured from the application through a standard system interface. Simultaneously, this realizes resource-savings and cost-efficiency constraints as it allows to use commodity server hardware and start with a smaller sized computer cluster (easy extensibility).

- **Flexibility** refers to the system ability to include and understand new data sources as they become available and secondly, to incorporate new types of crisis events as they become known or by requisite of new end-users of the system. The system should be decoupled from the data producers (e.g. sensors) and independent of the techniques for data dissemination (e.g. via API, as RSS feed or website).

- **Fault tolerance** refers to failsafe operations of the system in case of hardware failure and the unavailability of one or many input data sources due to internal hardware failure or lost connection to the data provider. The system should continuously monitor its health and trustworthiness and report to the user. Security of the system addresses compliance issues such as privacy which should be taken into account from the beginning. Other security concerns relate to the accessibility of the data by a human operator, data retention period, and data fusion regulations.

- **Trustable EIS** ensure a high degree of certainty in all information provided and visualized to the end-user. It holds mechanisms to measure the trustworthiness of information, data sources and derived findings. Transparency of decisions is realized by a tracing mechanism that logs the event detection process.
- **Collaborative** features of EIS embrace a participatory approach for the automated management of resources and to improve emergency detection and validation of detected events as well as data enhancement in smart cities and countries. An overall goal is process automation from data collection up to information mapping, prediction and alarming. This non-exclusively comprise:
  - automated detection of relevant events and alarming (awareness)
  - automatically crowdsource selected human users for targeted measurements/ feedback
  - dispatch semi-automatic responses
  - provide labels and descriptions of events (including confidence level)
  - seamlessly integrate human experts in the decision making process
- **Relevance** of information becomes a corner stone as more and more data enters the situation centers. EIS should present information in a transparent and usable way building their own situation understanding. Preferences of end-users should also be taken into account.

The increasing importance of crowd-sourcing and social media in reality monitoring and disaster response as well as the availability of real-world sensors open up new possibilities to advance emergency information systems and to put new tools in the hands of disaster managers. In particular, a better societal management of the overall cycle of disaster monitoring and response becomes possible, citizens may become involved in emergency detection and data acquisition/validation (crowdsourcing), and advanced planning can economize resources. On the other hand we see many novel requirements and constraints when putting big data into operations. Tapping new sources of information will help us to deal with uncertainty in single-source, real-world data and to raise awareness of unforeseen anomalies for early warning. Event ontologies structure the detection process and support communications. Big Data Analytics and machine learning methods reduce manual efforts and efficiently automatize the entire emergency detection process. What is required is a big data mind-set and new skills at the information and situation centers to open up for the new possibilities in big data which the people have already utilized for their life.

More Information

H. Stange, S. Steenhoek, S. Bothe and F. Schnitzler, Insight-driven Crisis Information - Preparing for the Unexpected using Big Data, Proceedings of the ISCRAM 2015 Conference

D. Kinane, F. Schnitzler, S. Mannor, T. Liebig, K. Morik, J. Marecek, B. Gorman, N. Zygouras, Y. Katakis, V. Kalogeraki, and D. Gunopulos, Intelligent Synthesis and Real-time Response using Massive Streaming of Heterogeneous Data (INSIGHT) and its anticipated effect on Intelligent Transport Systems (ITS) in Dublin City, Ireland, in Proceedings of the 10th ITS European Congress, Helsinki, 2014

2.2 Required Capabilities of Real-Time Spatio-Temporal Event Detection

The increasing availability of massive heterogeneous streaming data for public organizations, governments and companies pushes their inclusion in incident recognition systems. Leveraging insights from these data streams offers a more detailed and real-time picture of traffic, communication, or social networks, to name a few, which still is a key challenge for early response and disaster management. Detecting events in spatio-temporal data is a widely investigated research area (see e.g. [Agg13] for an overview). Depending on the application, the event detection can analyze single trajectories (e.g. of persons or vehicles), group movements, spatio-temporal measurements, or heterogeneous data streams. Following examples highlight capabilities of these approaches:

- *Individual Mobility:* Within airports (or other security region) it is valuable to monitor whether individuals enter some restricted area. The analysis of stops or of sudden decelerations allows detection of unusual behaviour. Sequences of such events can be matched against predefined mobility patterns [FKMM12], e.g. to identify commuters.

- *Group Movement:* During public events the early detection of hazardous pedestrian densities gains much attention. The patterns one could distinguish and detect in group movement are *encounter*, *flock* or *leadership* pattern [DWL08].

- *Spatio Temporal Measurements:* A spatio-temporal value spans a whole region. This could be traffic flow, air pollution, noise, etc. The sudden rise or decline of these values indicates an anomaly.

- *Heterogeneous Data Streams:* The combination of previously described types of anomalies provides event filters in an urban environment based on heterogeneous data (e.g. GPS data of pedestrians, traffic loop data, mobile phone network data).

In general functions for event detection from heterogeneous data streams can be classified using a former concept of raster-geography, namely *map-algebra* [Ber09]. Both, raster geography and heterogeneous spatio-temporal data analysis consider data which is provided

in multiple layers (i.e. one layer per data stream). Functions can be applied to one or multiple layers. Thus, spatial functions split into four groups: *local*, *focal*, *zonal* and *global* ones [Ber09], illustrated in Figure 3.

- *Local functions* operate on every single cell in a layer. And the cell is processed without reference to surrounding cells. An example is a map transformation, the multiplication with a constant, or the comparison with a threshold.
- *Focal functions* process cell data depending on the values of neighboring cells. The neighborhood can be defined by arbitrary shapes. Example functions are moving averages and nearest neighbor methods.
- *Zonal functions* process cells on the base of zones, these are cells that hold a common characteristic. Zonal functions allow the combination of heterogeneous data streams in various layers by application of functions to one layer if another layer already fulfills another condition.
- *Global functions* process the entire data. Examples are distance based operations.

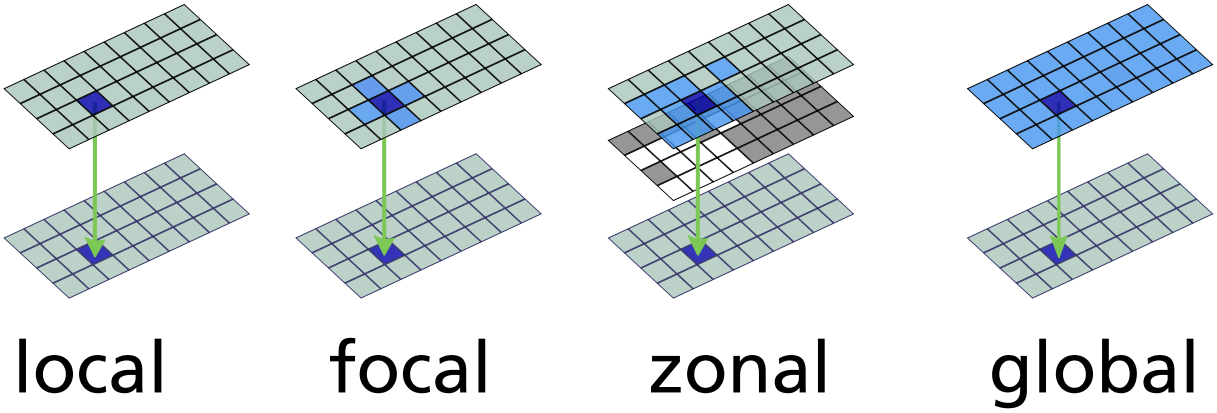


Figure 3: Function classes on Spatio-Temporal data, Dark blue highlights the currently processed location. Light blue cells indicate the regions whose values are used for computation. Best viewed in color.

For analysis of heterogeneous data streams, expressiveness of these four function types is important to *derive* low-level events (incidents), to *combine* low-level events (e.g. aggregation, clustering, prediction etc.), and to *trigger* high-level events [SL15]. For the latter, the event sequences can be matched against predefined spatio-temporal event patterns that indicate the occurrence of a high-level event.

The exploitation of spatio-temporal event patterns is a major research field in mobility mining. Event pattern matching focuses on the task to match sequences of events against event patterns and to trigger another event (which is raised for further analysis) in case the sequence matches. For this purpose, recently, pattern-graphs were introduced in [PHB12], their pattern description is capable to express the temporal relations among various occurring

events following the interval-calculus [All83]. As an example, the co-occurrence of two low-level events may trigger any high-level event. With spatio-temporal data streams also spatial relations are important to consider. The region connection calculus [RCC92] lists relations of spatial events that are essential for spatio-temporal pattern matching.

Possible frameworks for event pattern matchers are the event calculus [SPVA11], finite automaton [FKMM12] and other pattern matcher [DGP+07, PHB12] or even complex frameworks which allow application of local, focal, zonal and global functions e.g. [SG11, GKS+13]. The requirements for spatio-temporal pattern matcher in an autonomous event identification system (EIS) are:

- to operate in real-time,
- to incorporate spatial [RCC92] and temporal [All83] relations,
- to provide local, focal, zonal, and global [Ber09] predicates on the attributes, and
- to pose arbitrary queries formed of these elements (regular language [dMRS05], Kleene closure [GADI08]).

In Table 2 we compare the features of state-of-the-art event detection frameworks. The temporal expressiveness is split into the following four categories:

- *Pattern Duration* is a constraint on the temporal distance of first and last condition in a pattern.
- *Condition Duration* is a constraint on the duration of a condition to get matched.
- *Inter-Condition Duration* is a constraint on the temporal distance among succeeding conditions.
- *Complete* indicates the complete integration of the temporal relations [All83].

The Table compares the approaches from the literature against the INSIGHT *Round Table* architecture, we introduced in D2.1, D4.1 and published in [SLM+14]. This approach is inspired by the TechniBall system [GKS+13], previous works on stream data analysis [FAA+13] and follows the Lambda architecture design principles for Big Data systems [Mar13]. In this architecture every data stream is analysed individually for anomalies. In this detection functions (e.g. clustering, prediction, thresholds, etc.) on the data streams can be applied. The resulting anomalies are joined at a *Round Table*. A final Complex Event Processing component allows the formulation of complex regular expressions on the function values derived from heterogeneous data streams. As can be observed the INSIGHT architecture overcomes limitations of existing event matching frameworks. Next section will focus on the analysis of the individual data streams.

Approach	Time Algebra [All83]				Spatial Algebra [RCC92]	Regular Expressions	Spatial Functions [Ber09]				Stream Processing
	complete	condition duration	inter-condition duration	pattern duration			local	focal	zonal	global	
Mobility Pattern [dMRS05]	-	✓	-	-	-	✓	-	-	-	-	-
SASE [GWC+07]	-	✓	-	✓	-	✓	✓	✓	✓	✓	✓
SASE+ [DIG07]	-	✓	-	-	-	✓	✓	✓	✓	✓	✓
Cayuga [DGP+07]	-	-	-	✓	-	✓	✓	✓	✓	✓	✓
Spatio-Temporal Pattern Queries [SG11]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
Mobility Pattern Stream Matching [FKMM12]	-	✓	-	✓	-	✓	-	-	-	-	✓
Event calculus [SPVA11, AWG+13]	✓	✓	-	✓	-	✓	✓	✓	✓	✓	✓
Temporal Pattern Graphs [PHB12]	✓	✓	✓	✓	-	✓	✓	-	-	-	✓
INSIGHT architecture [SLM+14]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: Comparison of Spatio-Temporal event detection frameworks [SL15].

More Information

F. Schnitzler, T. Liebig, S. Mannor, G. Souto, S. Bothe, and H. Stange, Heterogeneous Stream Processing for Disaster Detection and Alarming, in IEEE International Conference on Big Data, 2014

G. Souto and T. Liebig, On Event Detection from Spatial Time series for Urban Traffic Applications, in Solving Large Scale Learning Tasks: Challenges and Algorithms, Springer International Publishing, 2015, p. (to appear).

2.3 Review of Analysis Methods from D5.1

The spatio-temporal data streams we utilize for event detection and prediction comes in a variety of forms and representations, depending on the domain, the observed phenomenon, and the observation method (compare D5.1). In principle, there are three types of spatio-temporal data: spatial time series, events, and trajectories.

- A spatial time series consists of tuples $(attribute, object, time, location)$.
- An event of a particular type $event_i$ is triggered from a spatial time series under certain conditions and contains the tuples verifying these conditions $(event_i, object_n, time_n, location_n)$.
- A trajectory is a spatial time series for a particular $object_i$. It contains the location per time and is a series of tuples $(object_i, time_n, location_n)$. Every timestamp $time_n$ is contained at most once.

The types may be transformed to accommodate different analysis tasks and goals; this is focus of WP3.

In previous report on end-user requirements, test data, and on prototype definitions (D5.1) we have identified initial tasks for spatio-temporal data analysis that are relevant for the INSIGHT applications, during the cause of the project we tailored this list further to the use cases, The analysis methods we used so far are reported in D3.2 and D3.3 as well as D4.1 and D4.2. In this report we show the application of these methods in the Intelligent Sensor Agent Components and their evaluation on the use cases test data. According to previous deliverables (D3.1, D3.2, D4.1, D4.2) the revised list of methods described in this deliverable therefore comprises:

1. Gaussian Process Regression for imputing a data point based on its neighbours,
2. Spatio-temporal Markov Random Fields for the prediction of future measurement values,
3. Statistical Tests for outlier detection,
4. Snapshot prediction for bus delay prediction,
5. Complex Event Processing for outlier detection from bus data,

6. Deep Neural Networks for semantic role labeling of Twitter messages
7. Transformation of geo-locations for map matching,
8. Text classification for identification of traffic incidents.

3 Description of Alarming and Prediction Components

This section presents how methods from Deliverables D5.1, D3.2, D3.3, D4.1 and D4.2 are applied to the use case scenarios. For evaluation purposes, the methods that are incorporated in the INSIGHT system are applied to the test data, we identified in D5.1. The section is structured by data source as highlighted in Table 3.

Data Source	Intelligent Sensor Agent (ISA)	Section
SCATS	Prediction and Route Planning	3.1
SCATS	Outlier Detection	3.2
BUS	Delay Time Prediction	3.3
BUS	Outlier Detection	3.4
Mobile Phone Data	Outlier Detection	3.5
Twitter	Identifying Traffic	3.6
Twitter	Semantic Outlier Detection	3.7

Table 3: List of Intelligent Sensor Agents by data source.

3.1 Vehicular Traffic Prediction and Route Planning

Smart route planning gathers increasing interest as Dublin and other cities become crowded and jammed. In this section, we present the trip planning ISA that incorporates future traffic hazards in routing. Future traffic conditions are predicted by a Spatio-Temporal Random Field based on a stream of sensor readings, as previously described in D3.2. In addition, our approach estimates traffic flow in areas with low sensor coverage using a Gaussian Process Regression. The conditioning of spatial regression on intermediate predictions of a discrete probabilistic graphical model allows to incorporate historical data, streamed online data and a rich dependency structure at the same time.

General architecture We give an overview of the system architecture to address the veracity, velocity and sparsity problems of urban traffic management. This section describes the input and output of the system, the individual components that perform the data analysis, and the stream processing connecting middleware.

We built the system aiming real-time streaming capabilities. Based on the streams framework, the core engine is a data flow graph that models the data stream processing of the incoming SCATS data. As can be seen in Figure 4, this data flow graph contains the SCATS data source as well as several nodes that represent preprocessing operations.

With the service layer API provided by streams, we export access to the traffic prediction model to the OpenTripPlanner component. The OpenTripPlanner provides the interface to let the user specify queries for route planning. Based on a given query (v, w) with a starting location v and a destination w, it computes the optimal route $v \to p_0 \dots p_k \to w$ based on traffic costs. Here we plug in a cost-model for the routing that is based on the traffic flow estimation and the current city infrastructure status. This cost-model is queried by OpenTripPlanner using the service layer API.

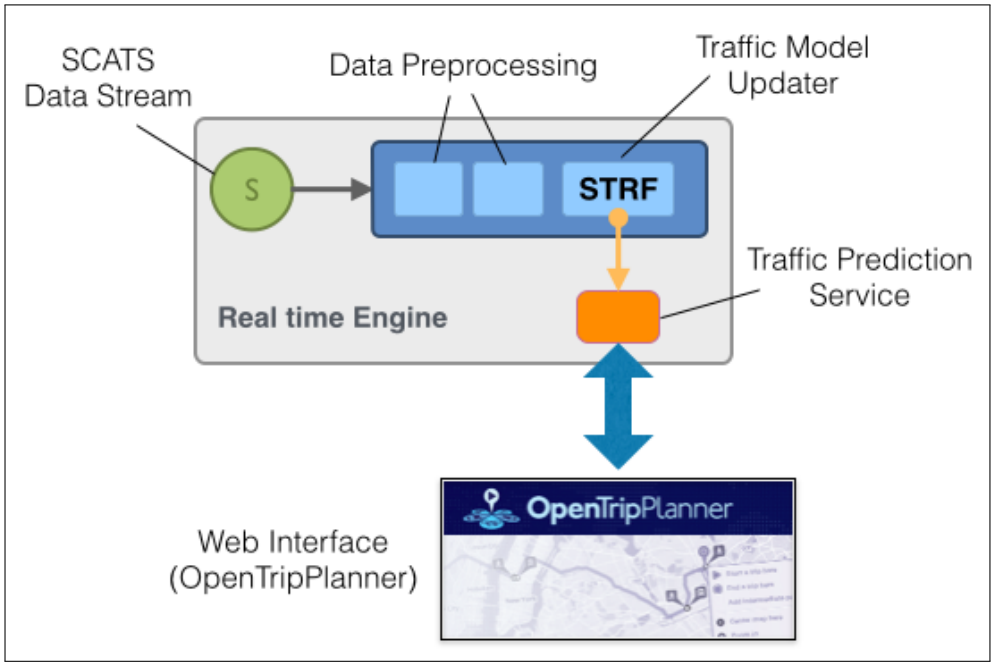


Figure 4: Architecture of Prediction and Route Planning ISA.

Traffic Model The key component of our system is the traffic model. It combines two machine learning methods in a novel way, in order to achieve traffic flow predictions for nearly arbitrary locations and points in time. This traffic model addresses multiple facets of the trip planning problem: i) sparsity of stationary sensor readings among the city, ii) velocity of real-time traffic readings and computation, and iii) veracity of future traffic flow predictions.

Based on a stream of observed sensor measurements, a Spatio-Temporal Random Field [PLM13] estimates the future sensor values, whereas values for non-sensor locations are estimated using Gaussian Processes [LXMW12]. To the best of the authors knowledge, streamed STRF+GP prediction has not been considered until now and is therefore a novel method for traffic modelling.

A Gaussian Processes denotes the covariances among the variables by a matrix which hast to be inverted for Gaussian Process Regression (GPR). This results in a high computational complexity. In [Lie14] we show that it is sufficient for imputation of an unknown measurement to consider just the closest observations (due to spatial autocorrelation of the traffic flow). This heuristic converts the GPR into a tractable problem, as the complexity of the matrix inversion can be kept constantly low by subsequent processing of unmonitored locations respecting their k nearest observations. Similar assumptions are made by STRF and kNN methods.

OpenTripPlanner OpenTripPlanner (OTP) is an open source initiative for route calculation. The traffic network for route calculation is generated using data from OpenStreetMap and (eventually) public transport schedules. Thus, OpenTripPlanner allows route calculation for multiple modes of transportation including walking, bicycling, transit or its combinations. However, vehicular routing is possible, but for data quality reasons in OpenStreetMap concerning the turning restrictions [SP12] it is not yet advisable in all regions.

The default routing algorithm in OTP is the A* algorithm [HNR68] which utilizes a cost-

heuristic to prune the Dijkstra search [Dij59]. At every considered intermediate location (between start and target location) the cost-heuristic estimates a lower bound of the remaining travel costs to the target. The cost estimate for traversing this intermediate location is calculated using the sum of the costs to the location and the estimated remaining costs.

OpenTripPlanner consists of two components an API and a web application which interfaces the API using RESTful services. The API loads the traffic network graph, and calculates the routes. The web application provides an interactive browser based user interface with a map view. A user of the trip planner can form a trip request by selecting a start and a target location on the map, see Section 4.1.4 for description of the user interface. Besides the web application there exist OpenTripPlanner user interfaces for mobile devices. The variety of existing user interfaces stresses the sustainability of our decision for OpenTripPlanner.

Empirical Evaluation For evaluation, we used real data streams obtained from the SCATS sensors of Dublin city. The data was collected between January and April 2013 and comprises ≈ 9GB of data, compare D5.1. The SCATS dataset includes 966 sensors. SCATS sensors transmit information on traffic flow every six minutes.

For the experiments in Dublin, the traffic network is generated based on the OpenStreetMap[1] data. In the preprocessing step the network is restricted to a bounding window of the city size. Next, every street is split at any junction in order to retrieve street segments. In result we obtain a graph that represents the traffic network. The SCATS locations, are mapped to their nearest neighbours within this street network.

In the preprocessing step the sensor readings are aggregated within fixed time intervals. We tested various intervals and decided for 30 minutes, as lower aggregates are too noisy, caused by traffic lights and sensor fidelity.

The spatial graph that is required for the STRF is constructed as k-nearest-neighbor (kNN) graph of the SCATS sensor locations, compare D3.2. In what follows, a 7NN graph is used, since a smaller k induces graphs with large disconnected components and a larger k results in more complex models without improving the performance of the method. The fact that no information about the actual street network is used to build G_0 might seem counterintuitive, but undirected graphical models like STRF do not use or rely on any notion of flow. They rather make use of conditional independence, i.e. the state of any node v can can be computed if the states of its neighboring nodes are known. Thus, the kNN graph can capture long-distance dependencies that are not represented in the actual street network connectivity. The maximum traffic flow value that is measured by each SCATS sensor in each 30-minutes-window is discretized into one of 6 consecutive intervals. A separate STRF model for each day of the week is constructed and each day is further partitioned into 48 snapshot graphs, since we can divide a day into 48 blocks of 30 minutes length. The model parameters are estimated on SCATS data between January 1 and March 31 2013 and evaluated on data from April 2013.

The evaluation data is streamed as observed nodes into the STRF which computes a new conditioned prediction for all unobserved vertices of the spatio-temporal graph whenever time proceeds to the next temporal snapshot. The discrete predictions are then de-discretized by taking the mean of the bounds of the corresponding intervals and subsequently forwarded to

[1] OpenStreetMap: `http://www.openstreetmap.org`

the Gaussian Process which uses these predictions to predict values at non-sensor locations. Notice that although the discretization with subsequent de-discretization seems inconvenient at a first glance, it allows the STRF to model any non-linear temporal dynamics of the sensor measurements, i.e. the flow at a fixed sensor might change instantly if the sensor is located close to a factory at shift changeover.

Application of Gaussian Processes requires a joint multivariate Gaussian distribution among the considered random variables. In our case, these random variables denote the traffic flow per junction. Literature on traffic flow theory [Lay09, Dav97] tested traffic flow distributions and supports a hypothesis for a joint lognormal distribution. We test our dataset for this hypothesis. Thus, we apply the Mardia [Mar70] normality test to the preprocessed data set. The test checks multivariate skewness and kurtosis. We apply the implementation contained in the R package MVN [KG13]. The tests confirmed the hypothesis that the recorded traffic flow (obtained from the SCATS system) is lognormal distributed. Thus, application of Gaussian Processes to log-transformed traffic flow values is possible. The hyper-parameters for the GP are chosen in advance using a grid search. Best performance was achieved with $\alpha = 1/2$ and $\beta = 1/2$. The STRF provides complete knowledge on future sensor readings which is necessary for our GP. As the STRF model performs well [PLM13], we set the noise among the sensor data in our GP to a small variance of 0.0001.

The OpenTripPlanner creates a query for the costs at a particular coordinate in space-time. The query is transmitted from the route calculation to the traffic model. There, the query is matched to the discrete space. The spatial coordinates are encoded in the WGS84 reference system [Nat00]. To avoid precision problems during the matching between the components, the spatial coordinate is matched with a nearest neighbour method using a KDTree data structure [Moo91]. The nearest neighbor matching offers also the possibility to query costs for arbitrary locations. The timestamp of the query is discretized to one of the 48 bins we applied in the STRF.

Figure 5, shows different routes which are calculated for a particular Monday (8th April 2013) among a fixed start and a fixed target at different time stamps. The figure reveals that different routes are calculated depending on the traffic situation. In [SLM15] we considered an alternative distributed architecture for traffic flow distribution.

Figure 5: Results of route calculations for fixed start and target at different timestamps (from left to right: 7:00, 8:00, 8:30). Best viewed in color.

More Information

T. Liebig, Speed-Up heuristics for the traffic flow estimation with Gaussian Process Regression, in Proceedings of the 11th Symposium on Location-Based Services, 2014, pp. 136-138.

M. Stolpe, T. Liebig, K. Morik, Communication-Efficient Privacy-Preserving Label Aggregation for Spatio-Temporal Learning, in Proceedings of the Workshop on Parallel and Distributed Computing for Knowledge Discovery in Data Bases held at ECML/PKDD, 2015, (to appear)

T. Liebig, N. Piatkowski, C. Bockermann, and K. Morik, Route Planning with Real-Time Traffic Predictions, in Proceedings of the 16th LWA Workshops: KDML, IR and FGWM, 2014, pp. 83-94.

3.2 Anomaly Detection in Vehicular Traffic Streams

The SCATS data stream contains meaningful information regarding the traffic condition at different junctions in Dublin city. Monitoring the SCATS data could help in the detection of traffic anomalies. We created simple rules that check whether the streaming data exceeds a specific value. More specifically we check whether the moving average of the last 5 reported degree of saturation values for every SCATS sensor exceeds a predefined value. This threshold was tuned accordingly, using DCC traffic operators feedback. We decided to check the moving average, as it is less noisy and at the same time it is possible to reduce the number of false positive reported alarms. This approach identifies traffic anomalies at a specific lane of a junction that is facing high congestion. An example of an identified events is presented in Figure 6.

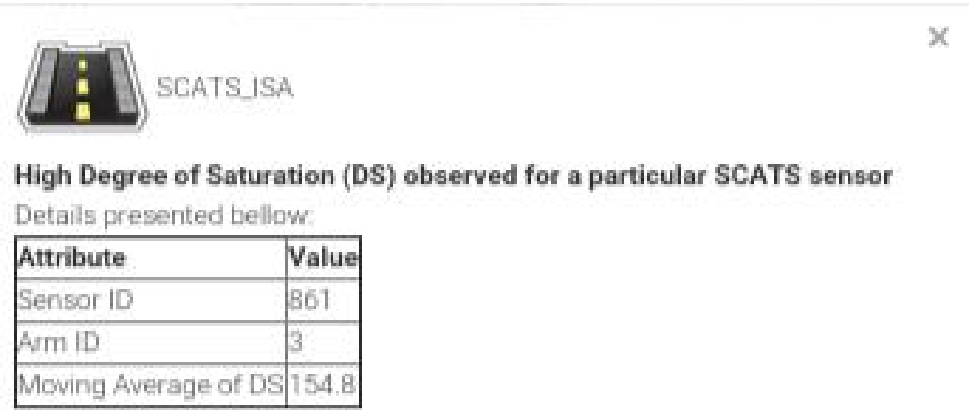


Figure 6: Message generated from the SCATS analysis detection system.

Related Approaches Stolpe et. al. propose [SBDM13] a Vertically Distributed Core Vector Machines (VDCVM) algorithm for anomaly detection which is based on Core Vector Machine

(CVM) algorithm [BC08]. The VDCVM has two components, the Central Node P_0 which coordinates the entire system and the Data Node $P_1...P_k$ which detects the anomalies in a distributed manner. The Data Node has two more sub-components, the Worker and Data Repository. The anomaly is detected locally by each Worker through a local model and sent to the Central Node along with a small sample of all observations. Then, the Central Node trains a global model on such a sample and used to define whether the sent observation is an anomaly or not. The advantage of this work is the good communication cost between Workers and the Central Node in the training phase, but this approach cannot detect anomalies which are global due to a combination of features, and that is its disadvantage. In [YKB14], Yang et. al present a non-parametric Bayesian method, or Bayesian Robust Principal Component Analysis (RPCA) - BRPCA, to detect traffic events on road. This method takes the traffic observations as one dimension data (1-D) and converts it into a matrix format which in turn decomposes it into a superposition of low-rank, sparse, and noise matrices. The idea of BRPCA is to improve the traffic detection by sharing a sparsity structure among multiple data streams affected by the same events. Such an approach uses multiple homogeneous data streams and a static weather data source in the detection process. The advantage of this work is the generating of a Ground Truth by 3 expertises in the traffic domain which reviewed different plots. However, the approach is limited to detect only 3 types of traffic events which are Slow down, Unexpected high traffic volume and traffic jam. Guo et al. [GHW14] propose a traffic flow outlier detection approach which focuses on the pattern changing detection problem to detect anomalies in traffic conditional data streams. The traffic data comes from inductive loop sensors of four regions in United State and United Kingdom, as well as this works makes use of a short-term traffic condition forecasting system to evaluate the proposed approach. This approach performs the analysis of the incoming data point after the data point be processed by Integrated Moving Average filter (IMA) which captures the seasonal effect on the level of traffic conditional series, and then Kalman Filter picks up the local effect flow levels after IMA, and GARCH Filter models and predict time-varying conditional variance of the traffic flow process. These Filters constitute together the integrated forecast system aforementioned. Trilles et al. [Trilles et al. 2015] propose a variation of CUmulative SUM (CUSUM) algorithm in Storm Framework to detect anomalies in data streams near to Real-Time. This approach is only applied when the observations are in-control, that is, the data is normal distributed. The events are detected, if the cumulative sum exceeds the threshold (CUSUM control charts), then it is an Up-Event due its increase and if the cumulative sum is lower than a threshold (CUSUM control charts), then it is a Decrease-Event. However, the work does not present experiments with a data source which has high refresh rate such as SCATS data stream.

Other works also propose solutions to detect anomaly traffic events such as [LZC+11, YL11, PCLZ13, PZWS13, YKB14]. However, these solutions make use of moving sensors such as GPS, and we have been focusing on static sensors.

3.3 BUS - Delay Time Prediction

In this part, we focus on prediction of traveling times, which can be announced to the passenger that uses Dublin buses, in an online fashion. Here, we developed a prediction engine that, given a scheduled bus journey (route) and a 'source/destination' pair, provides a prediction for the traveling time, while considering both Machine Learning techniques

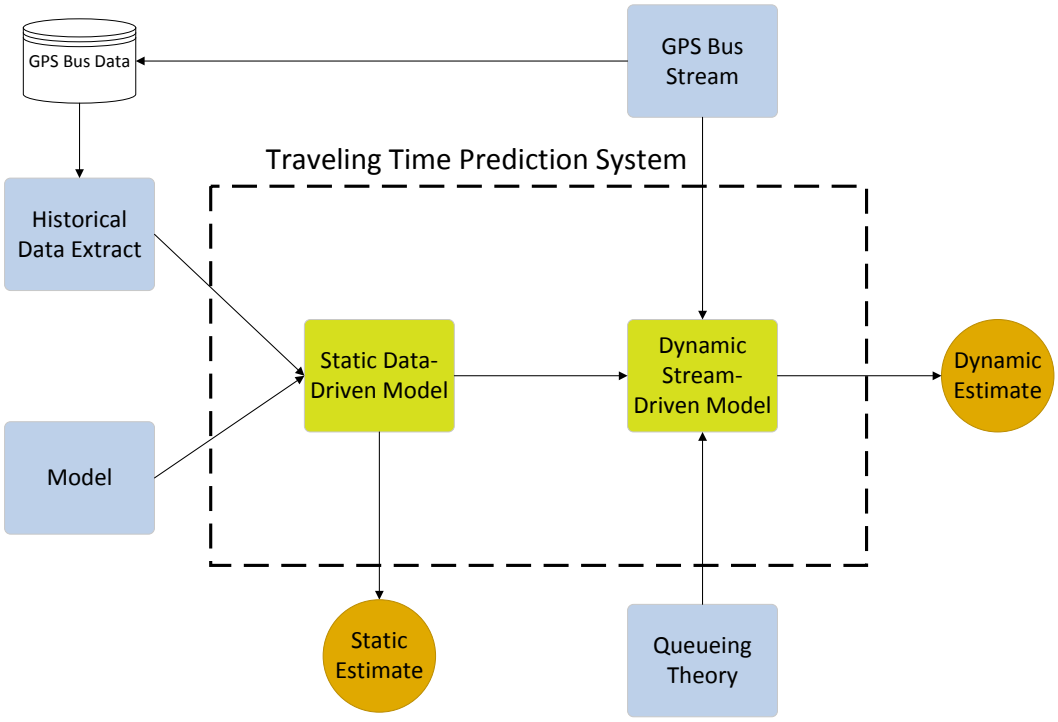


Figure 7: A real-time prediction engine.

that generalize from historical data, and methods that are grounded in Queueing Theory, which use real-time streams of information that are transmitted from the bus to correct the history-based prediction.

System's Architecture. Figure 7 presents the proposed two-component system, its inputs, outputs and the flow of information throughout the system. The first input into our system is a historical extract of the GPS Bus Data. During every bus journey, the Global Positioning System (GPS) location of the bus, along with other parameters, are transmitted (as real-time streams) and stored into a repository (managed by Dublin City Council). The data is then available for analysis in an off-line manner; however, the streams of data may also be used in real-time in what we refer to in Figure 7 as 'GPS Bus Stream'. Further information about the data repository is available in Deliverable 5.1[2].

The second input to our system is a Machine-Learning based prediction model for traveling times, denoted 'Model' (e.g. decision trees). Given the data and a model, an off-line static prediction component is constructed; we refer to it as 'Static Data-Driven Model', since it is based only on past data and is not dynamically updated. The component provides an estimate for the traveling time that accounts for regular events, i.e. hours in which changes in traffic are predictable.

[2] http://www.insight-ict.eu/sites/default/files/deliverables/D5-1.pdf

However, it turns out that the static model is often insufficient, since unpredictable changes in traffic (e.g. accidents, social events) may occur during a bus journey. These clearly influence the remaining traveling time for that journey and must therefore be accounted for in real-time. The second components of the proposed system, the 'Dynamic Stream-Driven Model', first identifies these unpredictable variants of regular hours and consequently updates the prediction in a dynamic fashion by using principles of Queueing Theory.

Data Modeling and Cleaning. Prediction of traveling time may exploit historical data on scheduled journeys or real-time streams of information on recent movements of vehicles. This section defines a common model for these types of information by means of the notion of a *journey log* (J-Log). A J-Log is a set of sequences of recorded journey events of scheduled bus trips, each sequence being partially ordered by the timestamp that indicates the occurrence time of an event. A J-Log is a particular type of an event log, as they are known, for instance, in the field of business process mining.

We illustrate the notion of a J-Log using data of the bus network in the city of Dublin.[3] Here, location of buses is sampled in intervals of 5 to 300 seconds (20 seconds on average), depending on the current location of the bus. For each event the following data is submitted to a monitoring system:

- A timestamp of the event.
- A vehicle identifier for the bus.
- A bus stop relating the bus to the stop on its journey with maximal proximity. Hence, every event has a bus stop identifier, even when the bus is not at the stop.
- A journey pattern that defines the sequence of bus stops for a journey.

Based on this input data, the construction of a J-Log as defined above is trivial; timestamps, bus stops and journey patterns are given directly in the input. To partition the events into journeys, a combination of the vehicle identifier and the journey pattern is used. An excerpt of the J-Log for the bus data from the city of Dublin is presented in Table 4. It features intuitive values for the attributes (e.g., stop "Parnell Square") as well as their numeric representation according to our formal model (e.g., 264 identifies the stop "Parnell Square").

Model of Segmented Journeys. To address the problem of traveling time prediction, as a first step, we construct a model that establishes a relationship between different journeys by means of visited bus stops. To this end, journeys are modeled as *segments of stops* (Figure 8). This segmented model, in turn, allows for fine-granular grounding of the prediction of traveling time: instead of considering only journeys that follow the same sequence of stops, we consider all journeys that share some segments can be used for prediction.

Below, we first describe the segmented model for journeys. A journey log (J-Log) is built of multiple journeys, where a journey is a sequences of events that are emitted by a bus as part of a particular trip. In order to rely on the aforementioned segmented model as the basis for the traveling time predictors, in a first step, we transform the J-Log into a Segmented J-Log that is built of timing information for segments.

[3] See also `http://www.dublinked.ie/` and `http://www.insight-ict.eu/`

Table 4: Example J-Log from buses in Dublin.

Event Id	Journey Id	Timestamp	Bus Stop	Journey Pattern
1	36006	1415687360	Leeson Street Lower (846)	046A0001
2	36012	1415687365	North Circular Road (813)	046A0001
3	36009	1415687366	Parnell Square (264)	046A0001
4	36006	1415687381	Leeson Street Lower (846)	046A0001
5	36009	1415687386	O'Connell St (6059)	046A0001
6	36012	1415687386	North Circular Road (814)	046A0001
7	36006	1415687401	Leeson Street Upper (847)	046A0001
8	36009	1415687406	O'Connell St (6059)	046A0001

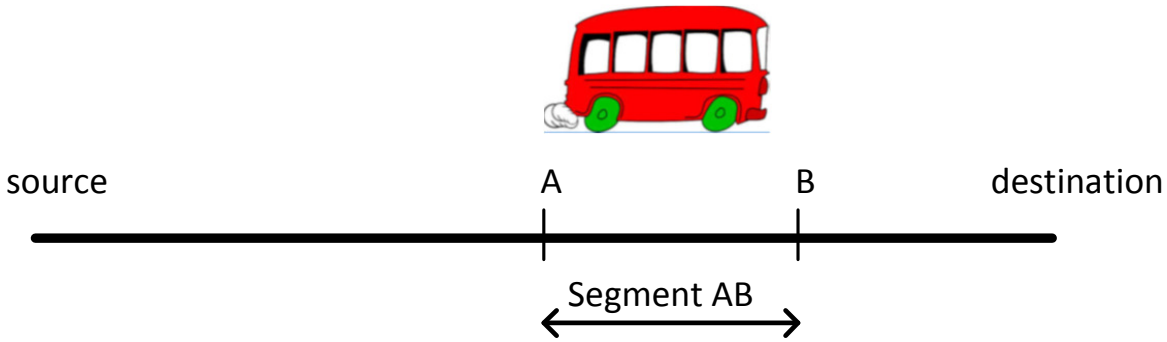


Figure 8: A segmented model of traveling times.

A Segmented J-Log is a sequence of *segment events* that capture information on the start and end bus stop of the segment, the journey from which the segment event was derived, the respective journey pattern, and the start and end timestamps observed for the segment. The last two elements are computed using the earliest time the particular journey reached the start and end bus stop. A Segmented J-Log is constructed from the journey events of all journeys in a J-Log. That is, a segment event is derived from two journey events of the same journey, such that (1) the journey events refer to two successive bus stops of the journey pattern, and (2) the journey events are the earliest events referring to these two bus stops. We capture this construction as follows.

A Segmented J-Log can be trivially constructed from a J-Log. However, in many real-world applications, the recorded data is incomplete due to data loss, unavailability of data recording devices, or data sampling. Then, it may be impossible to construct a segment event for each pair of successive bus stops of all journeys. For instance, for the data of the bus network in the city of Dublin described in the aforementioned data example, due to data sampling, the raw data does not necessarily contain a journey event for each bus stop of the respective journey pattern.

For a journey pattern and a journey recorded in the J-Log, we consider the following three cases of missing sequences. There is a sequence of missing journey events:

- between two consecutive recorded journey events.
- that includes the first bus stop.

- that includes the last bus stop.

In this section, we show how to use complementary information on the geographical distances between bus stops and principles of *kinematics* (relating distances to velocity and time) to impute missing journey events and their corresponding timestamps.

Prediction Techniques - Static vs. Dynamic. As we mentioned before, the prediction methods we present can be divided into two types. The first type comes from Machine Learning and is based on decision trees. The feature space includes also recent information that is considered by the first-type predictor. Both prediction methods make use of the segmented model of journeys and on the Segmented J-Log. The second type consists of a method that comes from Queueing Theory and approximates systems in heavy-traffic. The predictor is non-learning, in the sense that it does not generalize prediction from historical events, but rather uses recent events to predict future traveling times. We now introduce the snapshot principle for traveling time prediction

The *snapshot principle* is a heavy-traffic approximation that refers to the behavior of a queueing model under limits of its parameters, as the workload converges to capacity. In our context it means that a bus that passes through a segment, will experience the same traveling time as another bus that has just passed through that segment (not necessarily of the same type, line, *etc.*). In real-life settings, the heavy-traffic approximation is not always plausible and thus the applicability of the snapshot principle predictors should be tested ad-hoc, when working with real-world data sets.

In our case however, the snapshot predictor needs to be lifted to a network setting. Incidentally, based on results from Queueing Theory, the snapshot principle holds for networks of queues, when the routing through this network is known in advance. Clearly, in scheduled transportation such as buses this is the case as the order of stops (and segments) is predefined. We hypothesize that the snapshot predictor performs better whenever recent buses are in time proximity to the current journey. We test this hypothesis in the empirical evaluation of our techniques.

Now, we describe the use of Machine Learning and, more specifically, of regression techniques to predict the bus traveling time. As opposed to the snapshot method described above, Machine Learning techniques exploit past journey logs to learn a prediction model, and then use this model to make a prediction on new instances of the problem, in our case, traveling times as part of current journeys. Below, we discuss the features that we use, as the basis to the Machine Learning techniques. Then, we briefly describe the generic regression algorithms that we apply to solve the problem. Finally, we integrate the snapshot predictor with Machine Learning methods.

The features we consider are

- The travel time of the last bus that used that segment;
- The interval between the time the last bus left the segment;
- The day of the week; and
- The time of the day (hours, minutes, seconds).

The first two features are computed from the Segmented J-Log and therefore depend on the information available when the prediction is made, at time t.

Once the features are selected, we are ready to briefly present the regression algorithms applied to the aforementioned features. These algorithms all output ensembles $\Psi_M = \{\psi_m\}_{m=1}^M$ of M regression trees. A regression tree is a tree where each internal node is a test on the value of a feature, and where each leaf corresponds to a value of the target variable, in our case traveling time. An instance goes down from the root to a leaf by selecting at each internal node the branch corresponding to the result of the test of that node. The predicted value for that instance is the value associated with the leaf it reaches. Ensembles are sets of regression trees. The value predicted by the ensemble is typically the (potentially weighted) average of the values predicted by each tree of the ensemble:

$$\Psi_M(.) = \sum_{m=1}^{M} \lambda_m \psi_m(.) \ ,$$

where λ_m is the weight of tree ψ_m. Using an ensemble rather than a single model typically leads to an improvement in accuracy . We briefly describe below the methods we considered to build ensembles. We note that our selection of these methods cannot be comprehensive. However, since our focus is on a comparative analysis with the snapshot-based method for prediction, a selection that covers a variety of different techniques is sufficient.

A **random forest** (RF) is an ensemble built by learning each tree on a different bootstrap replica of the original learning set. A bootstrap replica is obtained by randomly drawing (with replacement) original samples and copying them into the replica. Each tree is learned by starting with a single leaf and greedily extending the tree. Extension consists of considering all possible tests (features and values) at all leafs and splitting the leaf using the test that maximizes the reduction in quadratic error. The tree weights are all equal $\lambda_m = 1/M$.

Extremely randomized trees (ET) is an ensemble where each tree was learned by randomizing the test considered during greedy construction. Instead of considering all values of the features for the split test, only a value selected at random is considered for each feature (and leaf). The tree weights are all equal $\lambda_m = 1/M$.

AdaBoost (AB) builds an ensemble iteratively by reweighting the learning samples based on how well their target variable is predicted by the current ensemble. The worse the prediction is, the higher the weight becomes. Therefore, the next tree constructed focuses on the most 'difficult' samples. Given the m^{th} model $\psi_m : \mathcal{X} \to \mathcal{Y}$ learned from a learning set $\{x_k, y_k\}_{k=1}^N$ with weights w_k^m, the next weights w_k^{m+1} are computed as follows:

$$\begin{aligned}
L_k &= (y_k - \psi_m(x_k))^2 / \max_j (y_j - \psi_m(x_j))^2 \\
\bar{L} &= \sum_k L_k w_k^m / \sum_j w_j^m \\
\beta_m &= \bar{L}/(1-\bar{L}) \\
w_k^{m+1} &= w_k^m \beta_m^{1-L_k} \ .
\end{aligned}$$

The value predicted is the weighted median of the predictions of the trees, where the weight of each tree ψ_m is $-\log \beta_m$. Initial weights are all equal to $1/N$. AB is typically used with

weak models, that do not model the data well outside an ensemble. For this reason, the depth (number of tests before reaching a leaf) of regression trees is typically limited when they are combined using AdaBoost. In our experiments, we tried both a depth of 1 and 3. The latter was almost always better, so we will only report the corresponding results. Except for this limitation, trees are learned greedily on the re-weighted learning sets.

Gradient tree boosting (GB) is another boosting algorithm. Instead of weighting the samples, GB modifies modifies the target variable value for learning each tree. The values used to learn the m^{th} tree are given by

$$\tilde{y}_k = y_k - \Psi_{m-1}(x_k) \ . \tag{1}$$

The new tree is trained on the prediction error $\tilde{y}_k$. In this algorithm, the model weights are replaced by leaf weights λ_m^l, where l is a leaf index. The leaf weight is given by $\lambda_m = \nu\gamma_m^l$. ν is a regularization term (equal to 0.1 in our experiments). γ_m^l is optimized by line search:

$$\gamma_m^l = \arg\min_{\gamma} \sum_{k:reach(x_k,\psi_m,l)} (y_k - [\Psi_{m-1}(x_k) + \gamma\psi_m(x_k)])^2$$

where $reach(x_k, \psi_m, l)$ is true if x_k reaches leaf l in the tree ψ_m. This ensemble is initialized by a tree learned on the unweighted learning set. We also considered a more robust version of this algorithm, denoted by GBLAD, that optimizes the absolute deviation error instead of the mean quadratic error. The most important changes are that each tree is constructed on a learning set $\{x_k, sign(y_k)\}$, and the value of each leaf is the median of the prediction errors of the training samples that reach it.

Now, we show a combination of the snapshot predictor and the Machine Learning techniques. The snapshot method stems from Queueing Theory and was demonstrated to perform well in practice, for delay prediction in various settings where heavy traffic (resource queues) produces these delays. Since bus delays are often induced by car traffic, it is tempting to use it as a baseline and try to improve over it. Boosting algorithms appear particularly suited for that task, since they construct ensembles of models sequentially, based on the results of the previous models in the ensemble. Following this line of reasoning, we modify the three boosting algorithms discussed above (AB, GB and GBLAD). That is, the first model considered in the boosting is the snapshot model. We respectively denote the three resulting algorithms S+AD, S+GB and S+GBLAD.

Empirical Evaluation. Here, we introduce our experimental setting, and discuss the main findings of our empirical evaluation. We shall now describe the construction of two datasets, the training set and the test set that we used for our experiments. First, the training set was used to train our Machine Learning methods and construct learning sets for each of the segments. Then, a test set of all trips (and sub-trips) was constructed to test both the Machine Learning and the Snapshot predictors.

The training set in our experiments consists of 8 days of bus data, between September and October of 2014. Each day contains approximately 11500 traveled segments. Essentially, the learning set is a Segmented Journey Log for those 8 days. Note that the first trip for a certain day will not include the last bus to go through a segment during that day. The data comes from all buses that share segments with line $046A$, since it is the only line that we analyze via

the test set. Hence, we exploited traveling time information from additional lines that share segments with $046A$.

The test set comprises bus data from a single day, September 22nd, 2014. We considered actual trips of line $046A$, which is one of the lengthiest lines in the city of Dublin (58 stops). First, the line travels through city center, where traffic can become extremely hectic and a large number of passengers may cause stopping delays. Then, it goes through a highway section and lastly it visits a suburban area where the delays are mostly due to frequent get-offs of passengers. During that day, the line has traveled through 111 journeys, all of which were included in our test set.

For the Machine Learning methods, we used the scikit-learn (Python) implementation of the algorithms to create ensembles of regression trees. Also, we relied on ensembles of $M = 100$ trees, unless otherwise stated. The algorithms that combine the snapshot method with Machine Learning (S+AD, S+GB and S+GBLAD), therefore contain 99 trees in addition to the snapshot model.

The main results of our experiments are:

- Prediction methods that combine the snapshot principle and Machine Learning techniques are superior in quality of prediction to both snapshot predictors and Machine Learning methods (that do not include the snapshot predictor).
- The prediction error increases with the number of bus stops per journey. However, when considering the relative error, it is stable for all trip lengths, i.e. the predictors do not deteriorate proportionally to length of the journey (in stops).
- Surprisingly, the snapshot predictor does not deteriorate for longer trips, therefore contradicting the hypothesis that the snapshot predictor would be more precise for journeys with higher temporal proximity to the current journey.

More Information

Gal, A., Mandelbaum, A., Schnitzler, F., Senderovich, A., Weidlich, M. (2014). Traveling Time Prediction in Scheduled Transportation with Journey Segments. Technical report, Technion-Israel Institute of Technology.

Senderovich, A., Weidlich, M., Gal, A., Mandelbaum, A., Kadish, S., Bunnell, C. A. Discovery and Validation of Queueing Networks in Scheduled Processes.

3.4 BUS - Outlier Detection

The buses that move in Dublin city periodically transmit their position and information regarding their route to the Dublin City traffic control center. The buses can be thought as moving sensors that provide useful information especially for areas that could not be covered from static sensors (SCATS, CCTV cameras). It is very important to collect and analyse the received raw bus data in order to automatically identify anomalous events or events of an emergency that happen in Dublin.

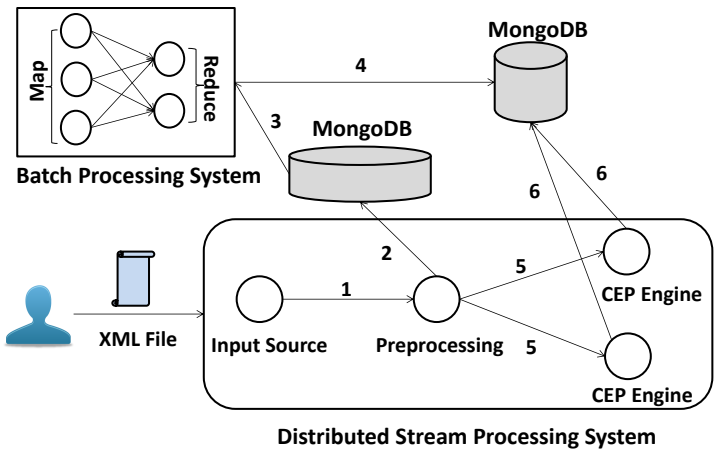


Figure 9: BUS ISA architecture.

System's Architecture: In order to monitor the bus data stream and identify anomalies in real-time we used a system architecture similar with the one described in D3.2 at Section 3.2, named "A Scalable Architecture for Traffic Monitoring" and presented in Figure 9. The proposed system consisted of the following components:

- A distributed Stream Processing System (Storm) that is responsible to route the incoming streaming bus data to the appropriate processing unit.
- A batch processing framework (Hadoop) that is responsible to process the batch data and identify the normal traffic behaviour for different areas of the city.
- A distributed database where the received data are stored (MongoDB).
- A storage medium where the statistics, calculated from the Hadoop jobs, are stored (MongoDB).
- Multiple Complex Event Processing (Esper) engines distributed at different cluster nodes, initialized with different rules and used in order to detect events of interest at the city of Dublin.

Data Preprocessing: In order to extract meaningful information from the raw data we preprocessed them and we enhanced them with new features. When a new tuple was received from a particular bus, new information was calculated such as the vehicle's speed and moving direction of the bus. In addition when a bus stopped at a bus stop we calculated the time needed to travel from the previous bus stop to the current. This time is sent to the Esper engine in order to find whether increased times to travel between two consecutive bus stops are observed.

Supporting Dynamic Rules: Given that we look for abnormalities during the course of the day for different pairs of bus stops, we compute statistics continuously and update the rules

accordingly. These statistics are calculated using Hadoop jobs. The jobs are invoked periodically, e.g., every week, to compute statistics for the different bus stops' pairs. Specifically, the job calculates the mean and standard deviation of the time needed to go from one stop to the next for different hours of the day and different types of days (weekdays or weekends). In the map phase the historical data are retrieved from the distributed MongoDB and then they are emitted to the reduce tasks. The reducers aggregate the parameters values for the different spatial locations and then compute the mean and the standard deviation for the time needed to travel two consecutive bus stops for different hours and days. The calculated statistics are stored in another collection of the MongoDB server and are retrieved during the online processing to be used as thresholds for the running rules.

Rules Description: The rules that we applied follow the general rule template that was described in D3.2. The main rule that we use checks whether the average observed time, needed to travel two consecutive bus stops, exceeds the expected time. This rule uses as input the times reported from the last 10 buses that moved between these two consecutive bus stops. The Esper code of the previously described rule is presented in Listing 1 and contains three streams, defined in the **FROM** clause. The first stream contains the last time from stop to stop that was arrived in the system. The second stream contains the last 10 times from stop to stop, for each pair of consecutive bus stops, that arrived in the system. The last stream, named $statsS2S$, contains all the statistics for all the possible pair of stops for different hours of day and for weekdays and weekends. This stream which contains the thresholds is loaded in the Esper Enginge in advance, when the engine is initialized. The streaming data join with this stream in order to retrieve their thresholds. The rule is fired when the average value is greater than the threshold for that particular hour and day, as it is defined in the **WHERE** clause. The threshold is set to be 3 times the standard deviation away from the mean. When this rule is fired it sends to the listener the information defined in the **SELECT** clause. More specifically it will return (i) the timestamp of the event (ii) the hour of the day (iii) the type of day (weekday or weekend) (iv) the bus stop ID where the bus started (v) the bus stop ID where the bus ended up (vi) the average time required from the last 10 buses to travel these two bus stops (vii) the mean value of the time required to travel these two bus stops as it is calculated from the Hadoop job (viii) the standard deviation of the time required to travel these two bus stops as it is calculated from the Hadoop job. Finally this information is collected from the listener that creates a message that is forwarded to the traffic operators and is presented in Figure 10.

Listing 1: Esper Rule template.

```
SELECT
        timeS2S.timestamp as timestamp,
        timeS2S.currentHour as currentHour,
        timeS2S.dateType as dateType,
        timeS2S.stopFrom as stopFrom,
        timeS2S.stopTo as stopTo,
        avg(timeS2S_Win.time) as averageTime,
        statisticsS2S.meanTime as meanTime,
        statisticsS2S.stdvTime as stdvTime
FROM
        timeS2S.std:lastevent() as timeS2S,
        timeS2S.std:groupwin(stopFrom,stopTo).win:length(10)
           as timeS2S_Win,
        statisticsS2S.win:keepall() as statsS2S
WHERE
        timeS2S.stopFrom = timeS2S_Win.stopFrom and
        timeS2S.stopTo = timeS2S_Win.stopTo and
        timeS2S.stopFrom = statsS2S.stopFrom and
        timeS2S.stopTo = statsS2S.stopTo and
        timeS2S.currentHour = statsS2S.currentHour and
        timeS2S.dateType = statsS2S.dateType and
        avg(timeS2S_Win.time) > statsS2S.meanTime+3*statsS2S.stdvTime
GROUP BY
        timeS2S.stopFrom,
        timeS2S.stopTo
```

BUS_ISA

Position = (53.39725, -6.224199999999996)
Bus Anomaly:Increased time required to go from bus stop 4597 to bus stop 6078. More specifically the average time required to go from one stop to the other currently is 393 sec. While the expected time from the historical data was 236 sec and Standard Deviation was 1 sec for weekdays at 8:00

Figure 10: Message generated from the Bus ISA.

Faulty Sensors: Additionally we set up rules that check whether the buses report noisy measurements. More specifically we set up rules that inform the traffic operators when one of the following happen:

- If the buses report wrong bus stop: Fire an alarm if the euclidean distance between the bus and its reported bus stop is greater than 1500m.
- If the bus moves and at stop field is TRUE: Fire an alarm if the bus reports that it is at stop and the distance covered from the time that it reported that entered the bus stop exceeds 30m.
- If the buses report extreme speeds: Fire an alarm if the bus reports that moves with speed greater than 120km/h.

More Information

N. Zygouras, N. Zacheilas, V. Kalogeraki, D. Kinane, and D. Gunopulos, Insights on a Scalable and Dynamic Traffic Management System, EDBT 2015, pp 653-664.

3.5 Mobile Phone Data - Outlier Detection

The mobile network intelligent sensor agent (ISA) is responsible for detecting abnormal activity volumes in a mobile phone network. As part of the INSIGHT system it continuously monitors the load of all cell towers. To detect these abnormal readings it constructs a model of normal readings and compares current readings to this. The detection process is split into a batch analysis and an in-stream speed analysis. With this approach we follows the general idea of the Lambda architecture as proposed in [MW15]. If the detection process discovers an abnormal state, an event candidate is reported to the upstream component of the INSIGHT system, the *Round Table* [SLM+14]. For an overview of the components see Figure 11.

Input Data Streams and Transport

Available data The data for the mobile phone ISA originates from Vodafone, the second largest network provider in Germany. The data is obtained from 55.000 mobile cell towers, covering all places in the country. Every time there is a record of the current load on the specific cell. Within the network this information is continuously available, but due to technical reasons it is not preserved for historical analysis in general. We gained access to hourly snapshots of this information which we use for development of the mobile phone ISA.

Geographical Referencing The mobile phone network consists of individual cells. Each of the mobile network cells has a identifier (ID). These ID codes can be used to determine the spatial position of the cell tower. To determine the geographical position we utilize a free web service, OpenCellId (see [ope]). Given a cell identification, it provides the position in WGS84 [Nat00] with latitude and longitude coordinates. The values are stored in a `csv` file for further processing.

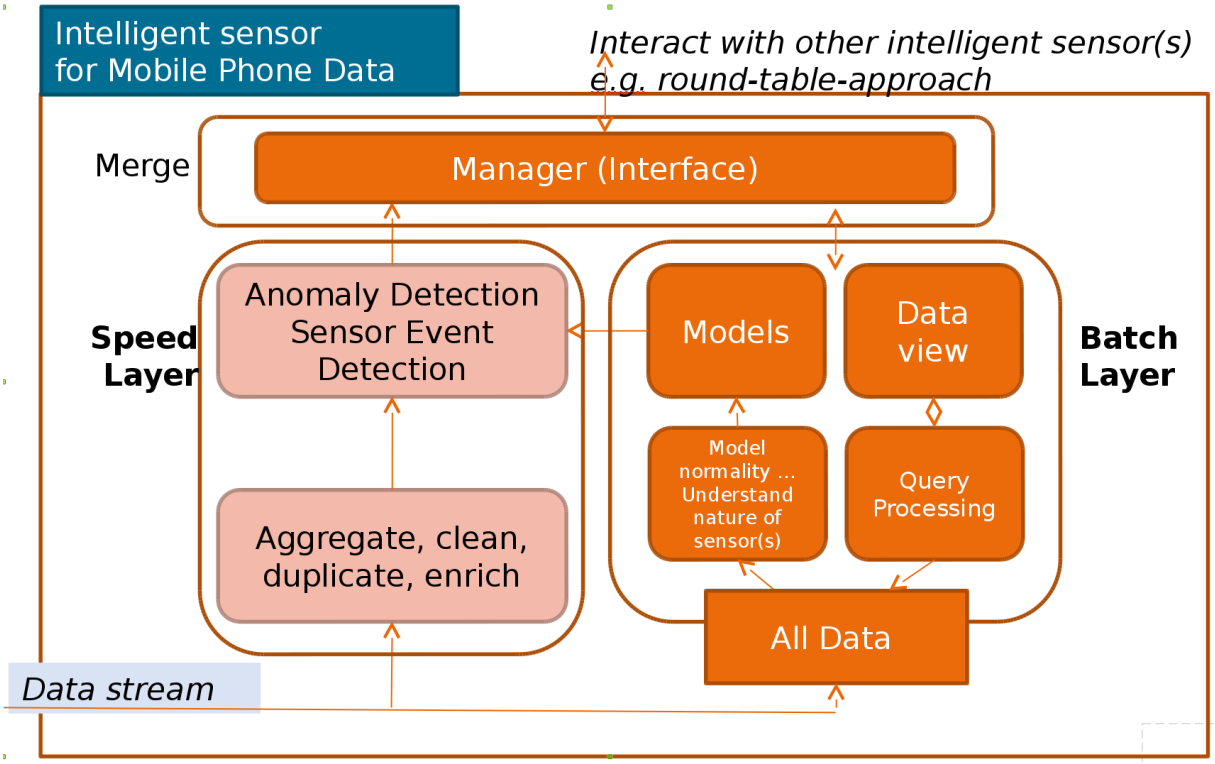


Figure 11: Overview on Mobile Network ISA.

Data Stream Transport and Routing The described input data is a stream of distributed sensor readings (cell towers are the sensors). For analysis the lambda architecture provides two layers, i.e. the speed layer and the batch processing layer. Both layers need to be supplied with the required data. Therefore, we stream the original data into a scalable distributed fault-tolerant message queuing system, Apache Kafka (see [kaf]). A message with a certain topic is inserted to Kafka and remains available up to a configured lifetime. For this purpose Kafka follows the publish subscribe abstraction. Each message can be retrieved by multiple consumer processes before expiration happens. These properties make Kafka an excellent candidate for feeding our two processing layers.

The batch processing in the lambda architecture holds a historical set of data relevant for the application, briefly referred to as *All Data*. To maintain this set it is required to periodically transfer new data from the Kafka queue to the batch processing system. A system capable of this transport is Camus [cam]. The Camus system connects to a Kafka system and incrementally appends the new data to the target batch storage system.

Batch Processing and Data Storage

In many practical applications the Apache Hadoop [had] system is commonly used for batch processing. We use the provided components of this open source software for carrying out tasks in the batch layer. Namely, we use the Hadoop distributed file system (HDFS) for storing the historical reference data (all data). The detected abnormal events are stored to HDFS as well as the models of normality. For the derivation of models we use the Hadoop execution

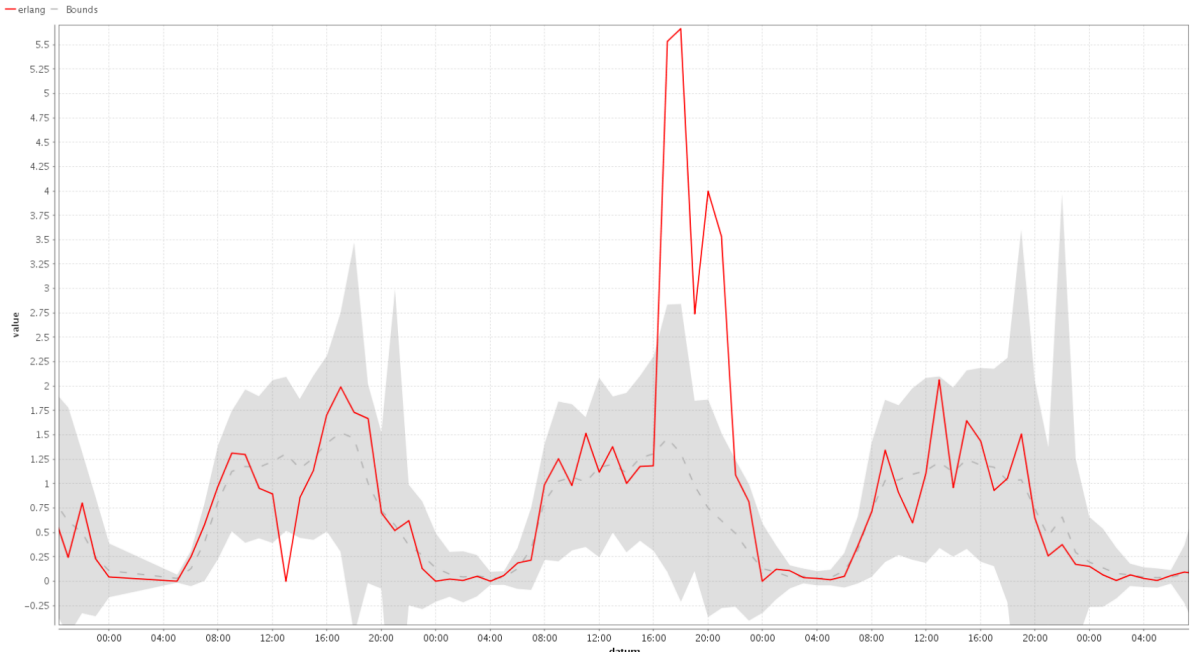


Figure 12: Model of normality. Different times have different mean values as well as acceptable corridors.

engine Map Reduce.

Modeling of Normality For outlier detection, we make the following assumptions. The data has normal state, aberrations thereof occur at random and the distribution of readings follows a normal distribution. We partition the data according to the following scheme: *day of week* (dow): Monday-Sunday and *hour of the day* (hod): 0-24. For each partition we define a context of N values for the calculation of a mean $\mu_{dow,hod}$ and a variance σ. This gives us a day and time sensitive model of normality for each individual cell. In Figure 12 the corridor acceptable for a normal value is depicted as a grey corridor around the green line of mean values. Please note, the mean and variance are different for different days and times.

A data item for time t belongs to multiple different contexts. It is duplicated and associated to each of them. Afterwards, each context is processed by an aggregation operation. Our implementation uses the Cascading library. Within this aggregation, the corresponding mean and variance values are calculated in each context. After the statistics are calculated, the outlier detection checks whether the variance is within the confidence corridor. In case a data point is outside, a marker for this outlier is created.

Once all the markers have been created, they are sorted and the duration of the events can be calculated. If the duration exceeds a preset threshold T_{evt} an event candidate is created and stored to a file in HDFS.

Quality Assurance The Cascading library allows to create Junit tests. A data tuple can be artificially inserted in a processing flow and results can be compared to an expected outcome. Testing is applied on different levels. We explicitly test the outcome of functions, for instance the calculation of statistical values. We insert a sequence of records with a known

abnormal event to the process chain and check if the generated anomaly is detected and the upstream event candidate is identified and created.

Query Processing The incoming data is stored in HDFS. We register it as data source in Lingual. This allows for analytic queries using ANSI SQL. For instance, it is possible to query aggregate counts per time unit. Also an analysis based on geographical location is possible using bounding boxes.

Data View The derived events are stored in a `csv` table. The structure thereof is as follows (units are given in brackets):

- timestamp (as unix time in ms),
- duration of event (hours),
- maximal deviation (in σ units),
- latitude, longitude (geographical coordinates according to WGS84 reference system [Nat00]).

Pushed to a mysql database for consumption by the GUI.

Real Time Processing Initially, the input stream is split by cellids to allow parallel processing. As streaming framework we chose the popular Apache Storm [sto] system.

Layout of the Storm Topology Input is a (replicated) Kafka spout. Connected to detection bolts using cellid as key. Due to the replication, each of the processing bolts processes a fraction of cellids. Any state associated to a bolt is volatile in Storm. The speed layer is considered to be corrected by the results from the batch layer, therefore additional mechanisms to make the states resilient are skipped in favor of low latency. Detected event candidates are reported to an output bolt, which creates the connection to the speed view. For the mobile phone ISA this is achieved using mySQL as data storage and using JDBC to connect the Storm bolt to the database.

Processing Steps and Methods

1. *Step1:* Floating average and mean per cell tower. Monitor 3σ corridor.

```
def online_variance(data):
    n = 0
    mean = 0.0
    M2 = 0.0

    for x in data:
        n = n + 1
        delta = x - mean
```

```
        mean = mean + delta/n
        M2 = M2 + delta*(x - mean)

    if n < 2:
        return float('nan');
    else:
        return M2 / (n - 1)
```

The online algorithm as described in [Knu97].

2. *Step2:* Check the condition that current measurements are within the corresponding confidence interval.

3. *Step3:* If the condition is violated, i.e. there is an abnormal reading detected, there are two possible situations: (1) There is no context open for that cellid. In that case we create a new context window to store the abnormal sensor events. (2) There is an open window for the cell at-hand. In this case two cases are distinguished again. If the current reading is *not consecutive* in time the context is discarded and a new one is created. Otherwise, if it is a subsequent reading the context is extend by the current reading. Further, the number of events in the context is checked against the minimal even duration threshold. If the threshold is exceeded an event candidate is created and emitted to the output bolt. Please note, that another event may be created with a longer duration after reception of an additional abnormal reading for that cell.

Output The events detected by the Mobile Phone ISA are depicted to the user as shown in Figure 13.

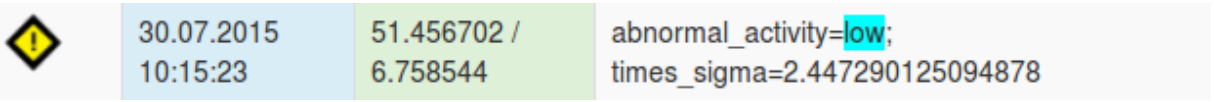


Figure 13: User interface of the Mobile Phone ISA.

3.6 Identifying Traffic by Twitter Analysis

Twitter is the largest micro-blogging service currently available with more than 316 million active users per month[4] (as of July 2015) resulting in a document stream consisting of 500 million tweets per day. Focusing on the city of Dublin, more than 15000 geo-tagged tweets located in the city of Dublin are posted per day. A small proportion of these tweets often refers to a traffic incident that takes place in the city. These tweets often originate from Twitter accounts specialized on reporting traffic incidents such as Livedrive or AARoadWatch or from users interacting with these services about an incident they just observed. Such tweets are highly valuable since they provide real-time information and updates about events happening at Dublin and as a result their identification is crucial. Inspired by this necessity, our Twitter

[4] https://about.twitter.com/company

anomaly detection system is able to detect traffic as well as flood related problems in Dublin using the Twitter stream. Example of event tweets detected by our system are presented in Table 5.

Data Preprocessing In order to identify meaningful tweets that refer to various incidents a set of reprocessing steps should be applied on the incoming data. More specifically for every tweet received from the system:

- We extract the tweet text and convert it to lowercase discarding any non alphanumeric characters keeping only a small subset of commonly used symbols. Also we remove any non English word.
- We remove all mentions. However, we keep all hashtags since a hashtag itself provides valuable information
- We remove English stop-words and terms that appear with a frequency more than a threshold as common words
- We identify Dublin specific highway abbreviations in the text and we replace them with predefined tokens (e.g. Abbreviations M50 and N11 will be converted to the tokens ROAD_M and ROAD_N respectively)

Location Extraction The Tweets retrieved by the Twitter API often do not contain exact coordinates or location meta-data. However, before we perform event detection on the incoming data we would like to ensure that even an approximate location could be extracted from the text. For that reason, we search within the tweets text for location references using an algorithm described in [DLB13] with the support of a lucene index of Dublin streets, highways and POIs obtained from Open Street Maps[5]. Using this algorithm we are able to detect textual references to a place and assign approximate geo-coordinates to the non geo-located tweets.

Feature Extraction After the above pre-processing steps are performed the tweets text should be converted to an appropriate representation. We decided to follow the bag of words representation in order to convert the tweets to the vector space model. That is, every document is represented as a vector representing the appearance of a term. In addition, the TF-IDF weighting scheme is used instead of using simple term counts since we observed that it leads to better detection performance.

Anomaly Detection The core component of the Twitter anomaly detection system is a machine learning supervised classifier. This classifier receives as input a TF-IDF weighted vector representing a tweet and decides if this tweet is related to a traffic incident or not. Different classifiers including Support Vector Machines, Logistic Regression and Multinomial Naive Bayes have been trained. Each of them with different training time requirements and end performance. The top performing models were the Logistic Regression and the Support Vector Machine with nearly similar detection accuracy. As an example, the Logistic Regression

[5] `https://www.openstreetmap.org`

Id	Tweets
1	Knockmarron Hill outbound is very sow following an earlier breakdown which has been cleared.
2	N11: We have reports of a collision on the N11 inbound just after Loughlinstown rounddabout
3	@paraicgallagher There was a collision before J9, now cleared. Those are the delays.
4	@LiveDrive the M50 collision has the N4 inbound backed up to hermitage golf club. Expect a long wait if you're trying to get on M50 south
5	@dublinbusnews Are you experiencing delays with 16 service? Been waiting ages at stop 278 with next bus @ 3mins for last 15mins.

Table 5: Example tweets that have been retrieved from the Twitter Anomaly detection system.

classifier scored a Precision of 0.70 and a Recall of 0.79 under the evaluation phase. The time-consuming model training has been performed offline. The resulting model, receives data in a streaming fashion and performs event detection in real-time.

Dataset Description: Every supervised classifier requires a labeled dataset that will be used for training. In order to face this requirement, for the traffic incident event detection case the decision was to gather tweets posted from Dublin traffic services as well as tweets from citizens interacting with these services, more specifically, we gathered tweets from the LiveDrive, AARoadWatch and GardaTraffic accounts during the years 2013 and 2014 (Services dataset). In addition, we also gathered mentions of citizens at these accounts during the year 2014 (Interaction dataset). Finally, we gathered a dataset from all the tweets located at the city of Dublin during the year 2014 (Location dataset). We automatically assigned labels using the following assumption:
Tweets originating from the Services and the Interaction datasets are assumed to be traffic related while tweets originating from the Location dataset are assumed irrelevant to traffic.
The final training set consisted of approximately 20000 traffic related tweets and 100000 non traffic related randomly selected. For the flood event detection case we have a used a manually labeled data-set from United Kingdom tweets described in [SVGA15]. We found that it could be applied to the Dublin city too with a descend performance.

System Architecture The Twitter anomaly detection system's architecture, presented also on figure 14 consists of the following components.

- A Twitter crawler that receives data using the Twitter Filter Stream API[6]. This stream contains tweets posted in the area of Dublin, tweets that contain specific keywords (e.g. M50) and tweets posted from a set of Dublin users obtained from [KLG14].
- A Pre-Processor that receives the data and performs the required pre-processing steps. Then converts the documents to the appropriate vector space representation.

[6] `https://dev.twitter.com/streaming/reference/post/statuses/filter`

- A Lucene[7] index that contains a list of the Streets, Highways and POIs in the city of Dublin used in order to perform geo-tagging.
- The text classifier that is responsible for identifying documents that refer to traffic incidents.
- A MongoDB[8] database for storing the detected events as well as the raw data.

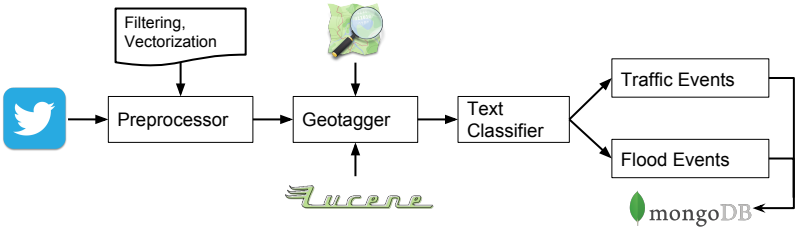


Figure 14: The architecture of the Twitter event detection system.

3.7 Monitoring Emergencies in Social Media

Task of Twitter ISA In the nation-wide disaster monitoring use-case of INSIGHT, Twitter data analysis is used to learn about potential disasters. Core tasks in this domain are (i) to reliably detect potential disasters as early as possible while minimizing false alarm, (ii) to verify the existence of a disaster, (iii) to predict the characteristics and consequences of a disaster.

The basic assumption is that Twitter user send information on new disasters even before official channels are informed about a disaster. Uncovering this hidden information will thus lead to a crucial time advantage for disaster management. A disaster can be considered as an event, which is described by a number of characteristics: What did happen, where did it happen, when did it happen, are there any casualties, is there a danger for additional people, is there material damage, etc. Twitter messages do include event descriptions but usually describe only one or two aspects of an event. Consequently there is a need for aggregation of event properties. Moreover tweets include a lot of noise as users can send messages about virtually any topic, e.g. what they eat for lunch. This information is uncontrolled and difficult to assimilate. The Twitter ISA has the task to analyze the Twitter data stream in real-time. It has to identify events in very unstructured text snippets, extract the characteristics of events and their urgency, follow events over time, and detect the development of new vocabularies. Detecting Word Senses and their Similarity

A very simple approach to event detection is the search of keywords (e.g. flood, explosion) in the Twitter message stream. The problem, however, is that many words like flood, explosion, have several different meanings which vary by context. Flood, for instance may refer to a (disastrous) overflow of water, or figuratively to a large number or quantity. Words with identical writing and different meaning are called homonyms. Since the latter sense of flood frequently appears in tweets we have to disambiguate between the different meanings before

[7] https://lucene.apache.org/core/
[8] https://www.mongodb.org/

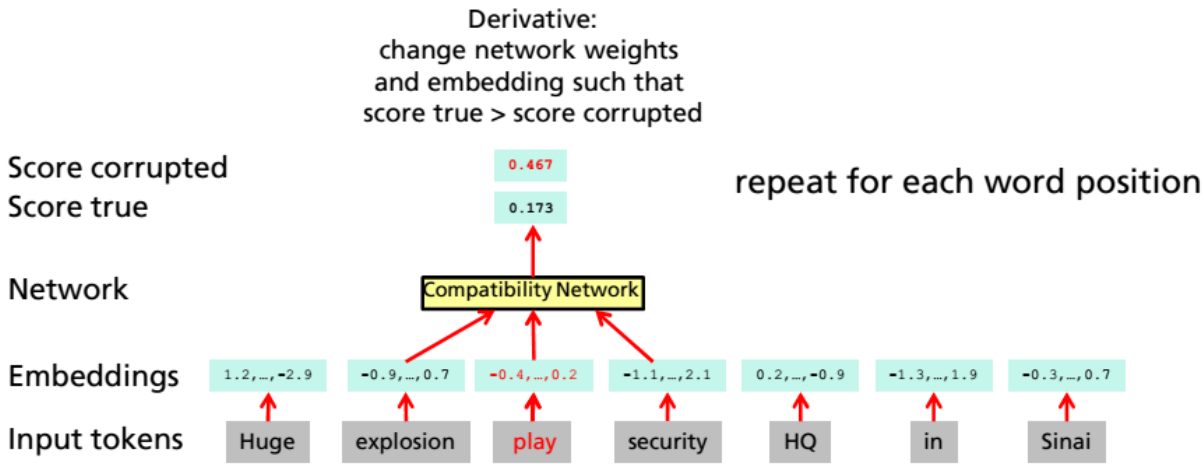


Figure 15: Computing word meanings by embeddings using deep neural networks on AKKA.

crime	explosion	killed	evacuation	lol	stupid
violence	accident	rescued	bombardment	lmao	dumb
suicide	eruption	kidnapped	interdiction	haha	nasty
homicide	wreckage	attacked	invasion	hahaha	ignorant
burglary	outbreak	murdered	extermination	smh	fat
police	barrage	hanged	bombing	LOL	ugly
corruption	fog	arrested	encirclement	lmfao	ratchet
theft	explosions	captured	replenishment	hahah	dirty
narcotics	iceberg	imprisoned	inspection	hahahaha	simple
fraud	earthquake	fired	redeployment	lmaoo	lame

Figure 16: Word similarity matrix.

we can assume to have some message on a real flood disaster. Moreover language provides many alternative ways to express a flood, e.g. "the level of river Elbe is rising".

We implemented a system representing word meanings by embeddings according to [CWB+11]. It uses a neural network and is trained by contrastive divergence. We designed a new extension of this system to represent several meanings for each word. This system was parallelized on a compute-cluster using the Akka[9] toolkit for distributed and resilient message driven applications.

The resulting word representations are 50-dimensional real vectors. We may use Euclidean distance as a similarity measure for representations. The next Figure 16 shows the nearest neighbors of words (first row) according to their representation similarity. "Accident", for instance has "eruption" as nearest neighbor.

[9] http://www.akka.io

Deriving Semantic Roles between Phrases Semantic role labeling consists of the detection of the semantic arguments associated with the predicate (verb) of a sentence and of their classification into their specific roles. For example, given a sentence like "Bank robber kills police officer with a gun", the task would be to recognize the verb "to kill" as representing the predicate, "Bank robber" as representing the killer (agent), "police officer" as representing the victim (patient), and "a gun" as representing the instrument. This is an important step towards making sense of the meaning of a sentence. A semantic representation of this sort is at a higher-level of abstraction than a syntax tree. For instance, the sentence "The police officer was killed by the bank robber with a gun" has a different syntactic form, but the same semantic roles.

We used the derived embeddings and implemented a new algorithm for semantic role labeling for English extending (Collobert et al. 2011). As a basis we used syntactic parse trees generated by the Stanford parser. In addition we derived semantic representations of phrases corresponding to specific nodes in the parse tree and used them as input to the classifier algorithm. The details are published in (Paass Pratap 2014).

Extracting Clusters of Relevant Event Tweets It is very hard for a person to spot events in Twitter without being overwhelmed by an endless stream of redundant tweets. We implemented a system to detect novel events as they are published on Twitter. Provided with a Twitter stream that is initially filtered by a list of seed terms corresponding to known events (e.g., flood, explosion, overflow, submersion) the system automatically mines the social stream, to provide a set of headlines that summarize the topics for a number of time slots of interest. In Twitter there are a few new factors that make the problem more challenging, e.g., different language styles between Twitter and traditional news media, the fragmented and possibly ambiguous nature of tweets due to their 140 character length constraint, the high amount of noise in the user-generated content and the real-time data processing aspect.

The event detection approach has the following ingredients: a combination of aggressive data preprocessing, hierarchical clustering of tweets, time dependent n-gram and cluster ranking and headlines re-clustering. For data preprocessing we normalize the text to remove URLs, user mentions and hashtags, as well as digits and other punctuation. Next, we tokenize the remaining clean text by white space, and remove stop words. In order to prepare the tweet corpus, in each time window, for each tweet, we first append the user mentions, the hashtags and the resulting clean text tokens. We check the structure of the resulting tweet, and filter out tweets that have more than 2 user mentions or more than 2 hashtags, or less than 4 text tokens. These tweets usually do not carry enough news-like content, or are generally very noisy. Using a language identification system (langid.py) we removed all non-German tweets. Subsequently we tagged all words with German Part-of-speech tags using Treetagger (http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/). Thes POS-tags were used as inputs for a Conditional Random Field to tag the tweet phrases with named entities Person, Location, Organization. These phrases later got a higher weight in the clustering algorithm. For each time window from the window tweet corpus, we create a (binary) tweet-term matrix, where we remove user mentions (but keep hashtags), and the vocabulary terms are only bi-grams and tri-grams, that occur in at least a prescribed number of tweets.

Using this term-matrix we compute a hierarchical clustering employing the fastcluster library (http://danifold.net/fastcluster.html) that can efficiently deal with thousands of tweets/terms.

The idea behind tweet clustering is that tweets belonging to the same event will cluster together, and thus we can consider each cluster as a detected event. We cut the resulting dendrogram at a prescribed distance threshold which was determined by experiment. Next, we introduce a modified term weighting, based on the frequency in the time window, as well as the boosting factor for named entities. For the frequency based weight, we use a formula from that discounts terms that already occurred in previous time steps emphasizing novel terms. This adapted measure was used to recompute the clustering to arrive at the final tweet-event clusters. These were presented in the Twitter GUI.

More Information

Paass, G., Pratap, B. (2014) Semantic Role Labeling Using Deep Neural Networks. Workshop on "Representation Learning" at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases 2014 (ECML/PKDD)

4 Description of Integration and Visualization

Previous section describe the methods integrated in the INSIGHT system. In the system, every analysis method is encapsulated in an ISA that communicates to the *Round Table*. The *Round Table* combines the anomalies detected in various data sources (compare D4.1 and D2.3). The resulting events are presented to the end-user.

The INSIGHT visualization system gives the operator the ability to inspect in real-time events that take place in the city of Dublin or in Germany. The interface is built as a web application and is presented on a user friendly web page. The requirement from the end user, in order to user the interface, is the existence of a web browser on his system. The interface is divided in layers, whose visibility may be toggled separately, in order to organize the information provided and to avoid overloading the screen with objects.

The user interface is able to provide visualizations about the raw data, anomalous data identified by the different Intelligent Sensor Agents (ISA) and events detected by the *Round Table* Manager. Finally, the interface provides the ability for the user to send direct feedback to system about the events displayed on the map.

This section describes the user interface of the INSIGHT system and the crowdsourcing capabilities of the mobile application.

4.1 INSIGHT Visualization in City-Level Use Case at Dublin

The INSIGHT interface for the city-level use case at Dublin is presented in Figure 17. The layers are presented at the top-left corner of the screen. The available layers are the following:

- RT-Events: Presents all the *Events* identified from the *Round Table* Manager
- ISA-Anomalies: Presents all the anomalies identified by the different Intelligent Sensor Agent (ISA) processing units.
- Live-Bus: Presents in realtime the current location of *delayed* buses.
- Live-Scats: Presents all the SCATS sensors reporting increased Degree of Saturation value during their last measurement.

One or more layers may be active at the same time. When a layer is selected, the traffic inspector needs to select what kind objects that represent an anomaly or event would like to see on the map. The available objects include *Icons* and *Polygons*. Icons are located exactly where the events happen while the polygons represent the affected area of the event. Further information related with the anomaly or the event is provided when the user clicks on an icon or polygon. According to the type of the incident a different icon is used. Figure 18 shows the icons used from the RT-Events layer while Figure 19 shows the icons used from the ISA-Anomalies layer. In the case of the polygons a different color and shading is used in order to differentiate incidents from different sources.

4.1.1 Online data plots

The INSIGHT system is able to provide to the user data visualizations in the form of line-plots for the raw data available. In this way, a user after inspecting the data could have a more clear

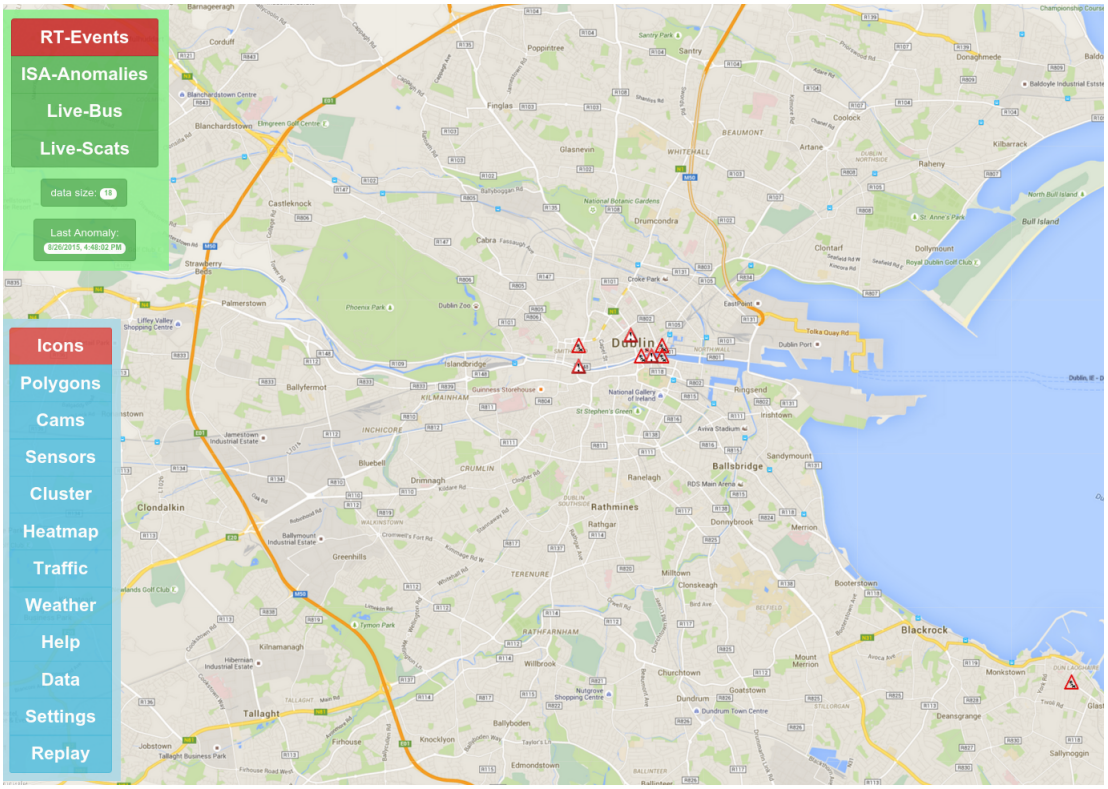


Figure 17: The Dublin INSIGHT interface.

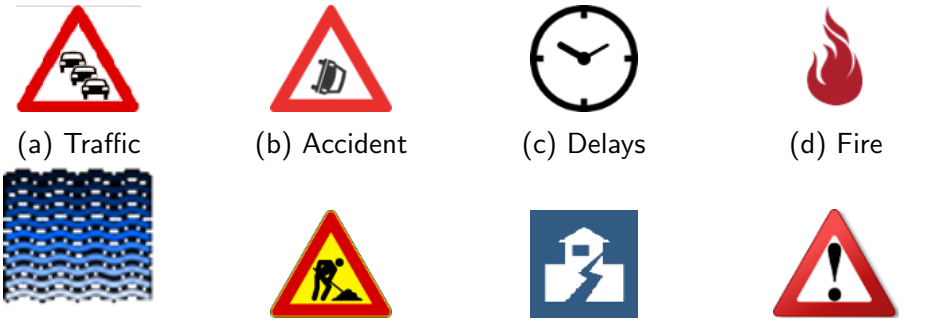
image about ambitious alarms and events. This feature is available for the SCATS-ISA and the Bus-ISA raw data. The system supports auto-complete capabilities to help the user find the sensors he is looking for and also supports multi-line plots allowing the user to combine sensors in one plot. Also moving average smoothing is supported in cases where the sensor data are very noisy. On Figure 20 an example plot for some scats sensors is presented.

4.1.2 Historical tweets

The INSIGHT interface provides the user with the ability to search for historical Tweets. The user is able to search for tweets during a specific period of interest. In addition, the user is able to retrieve tweets that have an event probability more than a threshold that he sets in order to easily access event tweets. Finally, the interface supports keyword based filtering on the list of tweets returned according to the user criteria. This way, the user can easily check if event tweets detected by the Twitter-ISA refer to an area of interest.

4.1.3 Replay

The interface supports the ability to reprocess the historical data gathered and then to replay them with a considerable speedup. When the user decides to replay the INSIGHT analysis for a specific period he is also able to select to run the different parameters required from the ISA as well as from the *Round Table* Manager components. This way one will be able to replay the analysis in cases of important events, decide on new parameters and finally tune the system.



(a) Traffic (b) Accident (c) Delays (d) Fire

(e) Flood (f) Constructions (g) Earthquake (h) Major Event

Figure 18: The different event icons used from the RT-Events layer.

(a) Scats (b) Bus (c) Tweets

Figure 19: The different event icons used from the ISA-Anomalies layer.

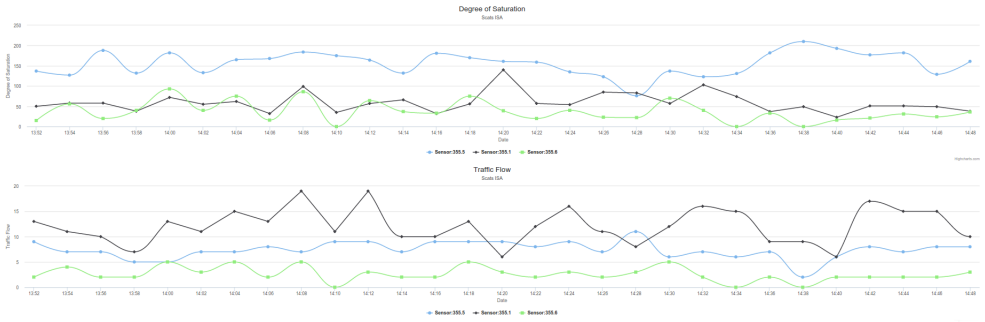


Figure 20: Example of multi-plots of some SCATS sensors generated by the INSIGHT interface.

4.1.4 Trip Planner

The traffic dependent trip computation is provided to the end-user as a website[10]. Similar to existing trip planners, the user interface allows selection of a start and target location on the map or in a textbox. Route computation is triggered on pressing the "Plan Your Trip" button, and incorporates real-time traffic predictions as described in Section 3.1.

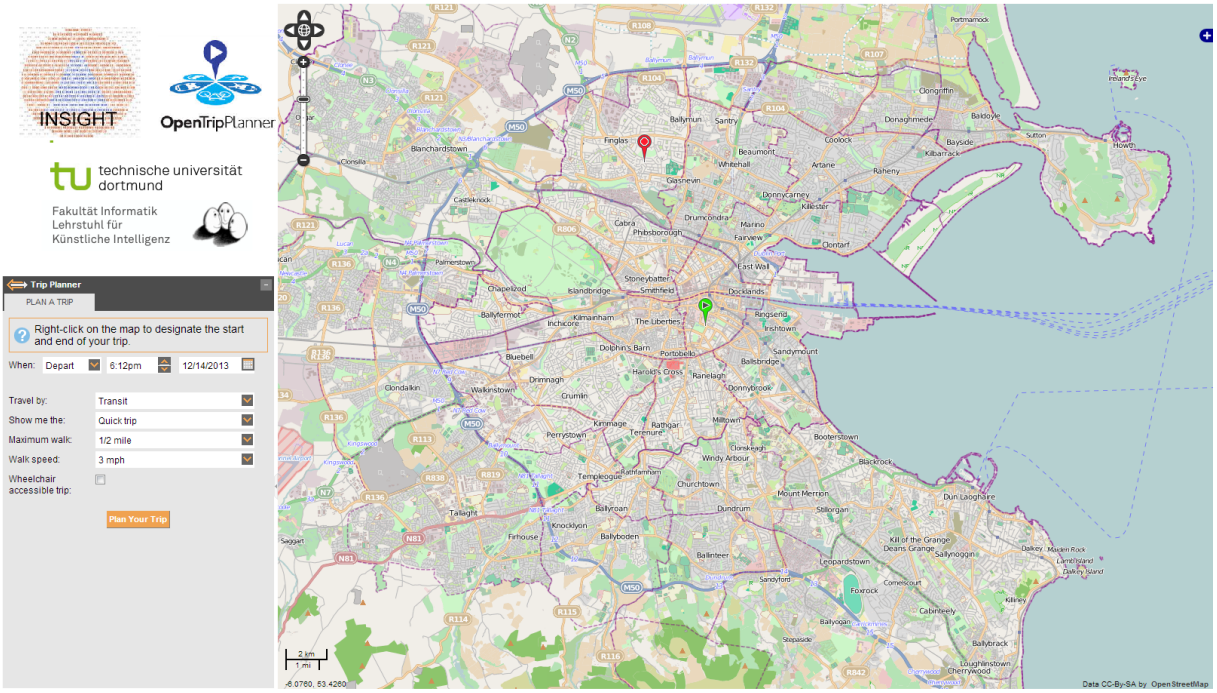


Figure 21: OpenTripPlanner User Interface. Map view is on the right side including a green pin which indicates the start location and a red pin that indicates the target. Best viewed in color.

4.2 INSIGHT Visualization in Nation-Wide Use Case

The INSIGHT interface for the nation-wide use case at BBK is presented in Figure 22. The layers are presented at the top-right corner of the screen. The available layers are the following:

- Warnings: Presents all *events* to the user.
- HeatMap: Presents the HeatMap related to the number of warnings per location to the user.
- Roads: Presents current traffic state to the user.

The navigator box below can be used to center the map at a certain location (in this case the city of Duisburg). Details of a selected event are shown below that navigator.

[10]The service utilizes the OptenTripPlanner framework.

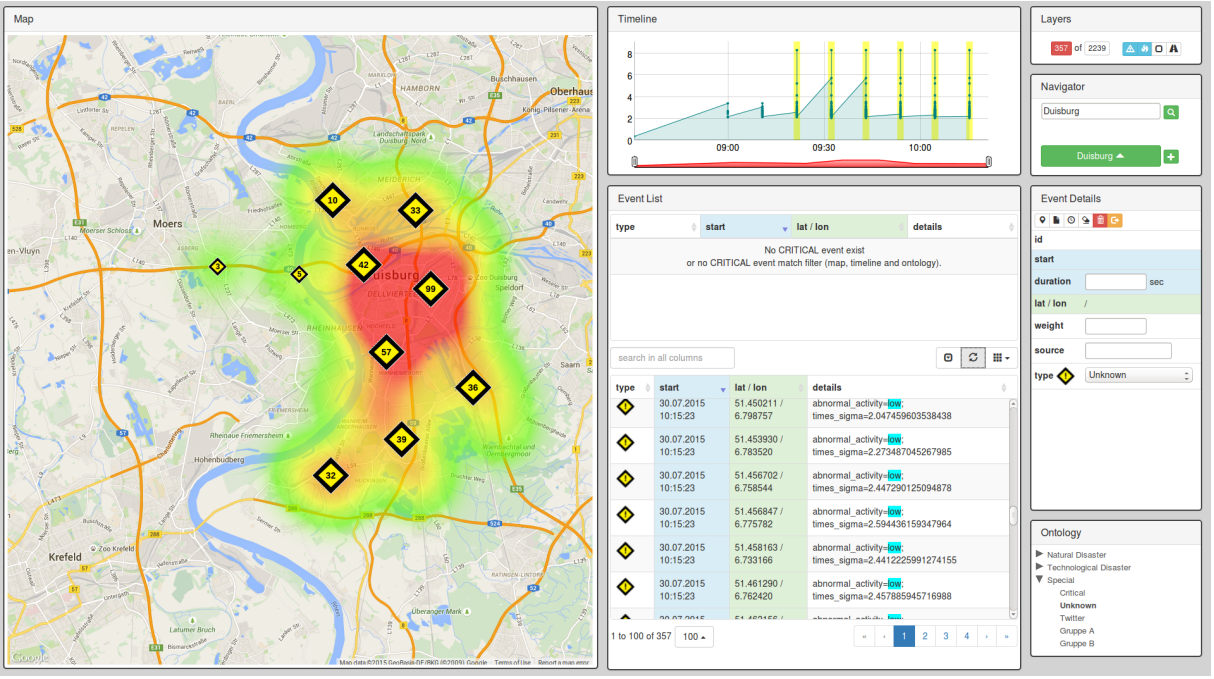


Figure 22: The BBK INSIGHT interface.

To the left a map depicts spatial distribution of the detected warnings and events. Depending on the zoom of the map events are clustered to prevent cluttering of the map display. The number of detected events is visually emphasized by a heat-map which is in the background of the map and supports visual inspection of affected regions.

The boxes Event List and Timeline provide information on the sequence of detected events. This includes: timestamp, number of detected events per timestamp, location, originating ISA, details on the event condition.

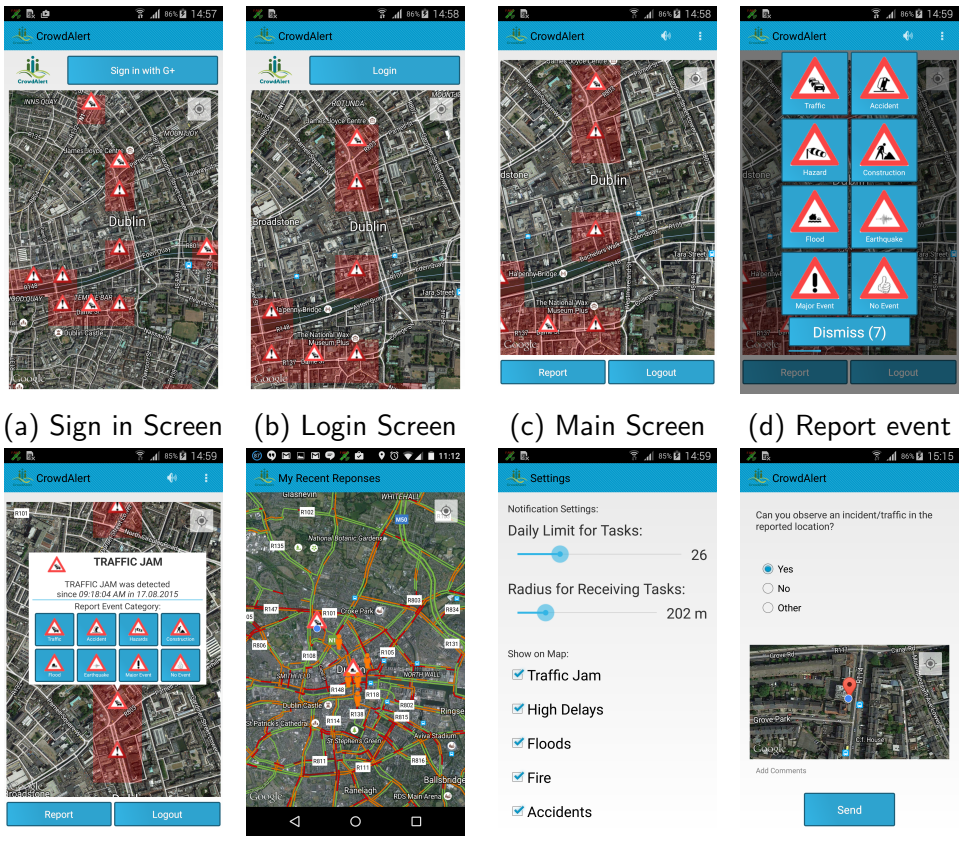
4.3 CrowdAlert Application

CrowdAlert is a free app, developed from the INSIGHT project and it is designed to enable users to receive traffic information and unusual events to meet the growing needs of businesses, residents, commuters and shoppers. CrowdAlert uses real-time data from road sensors, bus sensors and human crowd sensors.

At the main screen (Figure 23a) the user can observe the events that have been identified from the INSIGHT system, summarized in Figure 24. However, the user should register/login using his/her Google account in order to receive the respective rewards for his/her contributions.

After logging in (Figure 23c), the user can provide real-time information. The app supports 2 functionalities for providing such information:

- Reporting a new event in the user current location using the Report button (Figure 23d)
- Categorize an event identified by the INSIGHT System by clicking on the respective marker and selecting the type of the event (Figure 23e)



(a) Sign in Screen (b) Login Screen (c) Main Screen (d) Report event

(e) Categorize event (f) Recent Responses (g) User Settings Menu (h) Crowdsourcing Tasks

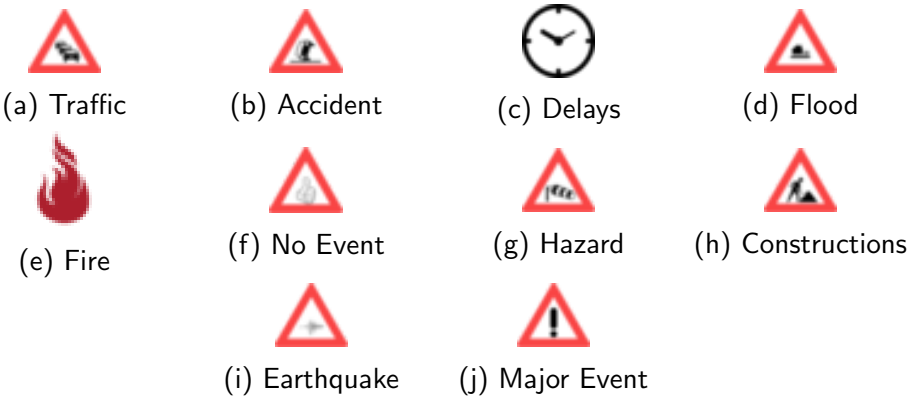
Figure 23: CrowdAlert app.

The user is also able to:

- Logout using the Logout button
- Pause/resume receiving nearby crowdsourcing tasks using the mute button in his up right corner
- Navigate to the "Settings" and "Recent Responses" from the menu button (up right corner)

Every user can also determine his/her preferences from the settings menu (Figure 23g) and observe the responses he/she has provided over the last 7 days (Figure 23f).

Finally, users will be receiving geo-located crowdsourcing tasks (Figure 23h) from the INSIGHT system. The users are prompted to answer the question that refers to the presented location that provides feedback to the INSIGHT system regarding an ongoing event.



(a) Traffic (b) Accident (c) Delays (d) Flood

(e) Fire (f) No Event (g) Hazard (h) Constructions

(i) Earthquake (j) Major Event

Figure 24: CrowdAlert Events.

5 Conclusion

During the third year of the project, the consortium worked together towards all task of WP5. After reviewing previous requirement elicitation, this document summarizes all developments regarding the adjustment of Alarming, Prediction and Classification methods to INSIGHTs use cases. The presented methods are encapsulated in Intelligent Sensor Agents and included in the INSIGHT system. The visualization of the analysis results and parameters to the end-users of the INSIGHT system (through the user interface) is also reported in this deliverable.

References

[Agg13] Charu C. Aggarwal. *Outlier Detection*. Springer, New York, NY, USA, 2013.

[All83] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[AWG+13] Alexander Artikis, Matthias Weidlich, Asaf Gal, Vana Kalogeraki, and Dimitrios Gunopulos. Self-adaptive event recognition for intelligent transport management. In *Big Data, 2013 IEEE International Conference on*, pages 319–325, Oct 2013.

[BC08] Mihai Badoiu and Kenneth L. Clarkson. Optimal core-sets for balls. *Comput. Geom.*, 40(1):14–22, 2008.

[Ber09] Joseph K. Berry. Gis modeling and analysis. In M. Madden, American Society for Photogrammetry, and Remote Sensing, editors, *Manual of Geographic Information Systems*, pages 527–585. American Society for Photogrammetry and Remote Sensing, 2009.

[cam] Camus. `https://github.com/linkedin/camus`. Accessed: 2015-08-01.

[CWB+11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[Dav97] Gary A. Davis. Estimation theory approach to monitoring and updating average daily traffic. Technical Report mn/rc 97-05, minnesota department of transportation, office of research administration, january 1997.

[DGP+07] Alan J Demers, Johannes Gehrke, Biswanath Panda, Mirek Riedewald, Varun Sharma, Walker M White, et al. Cayuga : A General Purpose Event Monitoring System. *Publish*, pages 412–422, 2007.

[DIG07] Yanlei Diao, Neil Immerman, and Daniel Gyllstrom. Sase + : An agile language for kleene closure over event streams. *Analysis*, (UM-CS-07-03):1–13, 2007.

[Dij59] Edsgar W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[DLB13] Elizabeth M Daly, Freddy Lecue, and Veli Bicer. Westland row why so slow?: fusing social media and linked data sources for understanding real-time traffic conditions. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 203–212. ACM, 2013.

[dMRS05] Cedric du Mouza, Philippe Rigaux, and Michel Scholl. Efficient evaluation of parameterized pattern queries. In Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken, editors, *CIKM*, pages 728–735. ACM, 2005.

[DWL08] Somayeh Dodge, Robert Weibel, and Anna-Katharina Lautenschütz. Towards a taxonomy of movement patterns. *Information visualization*, 7(3-4):240–252, 2008.

[FAA+13] Georg Fuchs, Natalia Andrienko, Gennady Andrienko, Sebastian Bothe, and Hendrik Stange. Tracing the german centennial flood in the stream of tweets: First lessons learned. 2013.

[FKMM12] Simona Florescu, Christine Körner, Michael Mock, and Michael May. Efficient mobility pattern stream matching on mobile devices. In *Proc. of the Ubiquitous Data Mining Workshop (UDM 2012)*, pages 23–27, 2012.

[GADI08] Daniel Gyllstrom, Jagrati Agrawal, Yanlei Diao, and Neil Immerman. On supporting kleene closure over event streams. In Gustavo Alonso, José A. Blakeley, and Arbee L. P. Chen, editors, *ICDE*, pages 1391–1393. IEEE, 2008.

[GHW14] Jianhua Guo, Wei Huang, and Billy M. Williams. Real time traffic flow outlier detection using short-term traffic conditional variance prediction. *Transportation Research Part C: Emerging Technologies*, pages 1–13, July 2014.

[GKS+13] Avigdor Gal, Sarah Keren, Mor Sondak, Matthias Weidlich, Christian Bockermann, and Hendrik Blom. TechniBall: DEBS2013 Grand Challenge. In *Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems*, page in press. ACM Press, 2013.

[GWC+07] Daniel Gyllstrom, Eugene Wu, Hee-Jin Chae, Yanlei Diao, Patrick Stahlberg, and Gordon Anderson. SASE : Complex Event Processing over Streams. *Science*, 1:407–411, 2007.

[had] Apache Hadoop. `https://hadoop.apache.org/`. Accessed: 2015-08-01.

[HNR68] Peter E. Hart, NilsJ. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[kaf] Apache Kafka. `http://kafka.apache.org/`. Accessed: 2015-08-01.

[KG13] Selcuk Kormaz and Dincer Goksuluk. *MVN: Multivariate Normality Tests*, 2013. R package version 1.0.

[KLG14] Dimitrios Kotzias, Theodoros Lappas, and Dimitrios Gunopulos. Addressing the sparsity of location information on twitter. In *EDBT/ICDT Workshops*, pages 339–346, 2014.

[Knu97] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[KSM+14] Dermot Kinane, François Schnitzler, Shie Mannor, Thomas Liebig, Katharina Morik, Jakub Marecek, Bernard Gorman, Nikolaos Zygouras, Yannis Katakis, Vana Kalogeraki, et al. Intelligent synthesis and real-time response using massive streaming of heterogeneous data (insight) and its anticipated effect on intelligent transport systems (its) in dublin city, ireland. In *Proceedings of the 10th ITS European Congress, Helsinki*, 2014.

[Lay09] Maxwell G. Lay. *Handbook of Road Technology, Fourth Edition*. Taylor & Francis, 2009.

[Lie14] Thomas Liebig. Speed-up heuristics for the traffic flow estimation with gaussian process regression. In *Proceedings of the 11th Symposium on Location-Based Services*, pages 136–138, 2014.

[LXMW12] Thomas Liebig, Zhao Xu, Michael May, and Stefan Wrobel. Pedestrian quantity estimation with trajectory patterns. In *Machine Learning and Knowledge Discovery in Databases*, volume 7524 of *Lecture Notes in Computer Science*, pages 629–643. Springer Berlin Heidelberg, 2012.

[LZC+11] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1010–1018, New York, NY, USA, 2011. ACM.

[Mar70] Kanti V. Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57:519–530, 1970.

[Mar13] Nathan Marz. *Big data : principles and best practices of scalable realtime data systems*. O'Reilly Media, 2013.

[Moo91] Andrew Moore. An introductory tutorial on kd-trees. Technical Report Technical Report No. 209, Computer Laboratory, University of Cambridge, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1991.

[MW15] Nathan Marz and James Warren. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2015.

[Nat00] National Imagery and Mapping Agency. Department of Defense World Geodetic System 1984: its definition and relationships with local geodetic systems. Technical Report TR8350.2, National Imagery and Mapping Agency, St. Louis, MO, USA, january 2000.

[ope] OpenCellID gps positions for gsm base stations. `http://opencellid.org/`. Accessed: 2015-08-01.

[PCLZ13] Linsey Xiaolin Pang, Sanjay Chawla, Wei Liu, and Yu Zheng. On detection of emerging anomalous traffic patterns using gps data. *Data Knowl. Eng.*, 87:357–373, September 2013.

[PHB12] Sebastian Peter, Frank Höppner, and Michael R. Berthold. Learning pattern graphs for multivariate temporal pattern retrieval. In Jaakko Hollmen, Frank Klawonn, and Allan Tucker, editors, *Advances in Intelligent Data Analysis XI*, volume 7619 of *Lecture Notes in Computer Science*, pages 264–275. Springer Berlin Heidelberg, 2012.

[PLM13] Nico Piatkowski, Sangkyun Lee, and Katharina Morik. Spatio-temporal random fields: compressible representation and distributed estimation. *Machine Learning*, 93(1):115–139, 2013.

[PZWS13] Bei Pan, Yu Zheng, David Wilkie, and Cyrus Shahabi. Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 344–353, New York, NY, USA, 2013. ACM.

[RCC92] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In Bernhard Nebel, Charles Rich, and William R. Swartout, editors, *KR*, pages 165–176. Morgan Kaufmann, 1992.

[SBDM13] Marco Stolpe, Kanishka Bhaduri, Kamalika Das, and Katharina Morik. Anomaly Detection in Vertically Partitioned Data by Distributed Core Vector Machines. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases ECML PKDD 2013*, page in press. Springer, 2013.

[SG11] Mahmoud Aattia Sakr and Ralf Hartmut Güting. Spatiotemporal pattern queries. *GeoInformatica*, 15(3):497–540, 2011.

[SL15] Gustavo Souto and Thomas Liebig. On event detection from spatial time series for urban traffic applications. In *Solving Large Scale Learning Tasks: Challenges and Algorithms*, page (to appear), 2015.

[SLM+14] François Schnitzler, Thomas Liebig, Shie Mannor, Gustavo Souto, Sebastian Bothe, and Hendrik Stange. Heterogeneous stream processing for disaster detection and alarming. In *IEEE International Conference on Big Data*, pages 914–923. IEEE Press, 2014.

[SLM15] Marco Stolpe, Thomas Liebig, and Katharina Morik. Communication-efficient privacy-preserving label aggregation for spatio-temporal learning. In *Proceedings of the Workshop on Parallel and Distributed Computing for Knowledge Discovery in Data Bases held at ECML/PKDD*, page (to appear), 2015.

[SP12] Simon Scheider and Jörg Possin. Affordance-based individuation of junctions in open street map. *Journal of Spatial Information Science*, 4(1):31–56, 2012.

[SPVA11] Anastasios Skarlatidis, Georgios Paliouras, George A. Vouros, and Alexander Artikis. Probabilistic event calculus based on markov logic networks. In Frank Olken,

Monica Palmirani, and Davide Sottara, editors, *RuleML America*, volume 7018 of *Lecture Notes in Computer Science*, pages 155–170. Springer, 2011.

[SSBS15] Hendrik Stange, Sylvia Steenhoek, Sebastian Bothe, and François Schnitzler. Insight-driven crisis information – preparing for the unexpected using big data. In *Proceedings of the ISCRAM 2015 Conference*, 2015.

[sto] Apache Storm. `https://storm.apache.org/`. Accessed: 2015-08-01.

[SVGA15] Antonia Saravanou, George Valkanas, Dimitrios Gunopulos, and Gennady Andrienko. Twitter floods when it rains: A case study of the uk floods in early 2014. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 1233–1238. International World Wide Web Conferences Steering Committee, 2015.

[YKB14] Shiming Yang, Konstantinos Kalpakis, and Alain Biem. Detecting road traffic events by coupling multiple timeseries with a nonparametric bayesian method. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):1936–1946, March 2014.

[YL11] Su Yang and Weihua Liu. Anomaly detection on collective moving patterns. *IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing*, 7:291–296, October 2011.