DEMO: REMI, Reusable Elements for Multi-Level Information Availability

Avigdor Gal
Nicolo Rivetti
Arik Senderovich
Technion - Israel Institute of Technology
[avigal|nrivetti|sarik]@technion.ac.il

Dimitrios Gunopulos
Ioannis Katakis
Nikolaos Panagiotou
University of Athens
[dg|katak|n.panagiotou]@di.uoa.gr

Vana Kalogeraki
Athens University of Economics and Business
vana@aueb.gr

ABSTRACT

Applications targeting Smart Cities tackle common challenges, however solutions are seldom portable from one city to another due to the heterogeneity of city ecosystems. A major obstacle involves the differences in the levels of available information. In this demonstration we present REMI, a *reusable elements* framework to handle varying degrees of information availability *by design* from two complementary angles, namely *graceful degradation* (GRADE) and *data enrichment* (DARE). In a nutshell, we develop reusable machine learning black boxes for mining and aggregating streaming data, either to infer missing data from available data, or to adapt expected accuracy based on data availability. We illustrate the proposed approach using tram data from the city of Warsaw.

CCS CONCEPTS

•**Information systems** →*Stream management; Data analytics; Data mining;*

KEYWORDS

Information Availability, Data Enrichment, Graceful Degradation

1 INTRODUCTION

Smart Cities provide an enticing use-case of Big Data event-based methods and technologies [5], demonstrating how technology can improve daily, as well as long-term, quality of life of inhabitants. Urban data is produced by a large array of sources and can be leveraged to detect disasters (*e.g.,* car accident), monitor special events (*e.g.,* festivals) or improve the city efficiency (*e.g.,* provide delay predictions in public transportation) [1]. Since these issues are common to practically any medium-to-large sized cities, any solution should be as flexible as possible in terms of the available data to allow portability despite the heterogeneity of different city ecosystems. A major obstacle to a high degree of portability involves the differences in the levels of available information, given the variety of data sources that are available in a particular city. It is worth noting that, while we focus on Smart Cities and Traffic data, similar concerns are common to many other application fields (*e.g.,* Energy Management and Environmental Monitoring, *etc.*).

The current main interest of urban environment researchers and experts is the data processing element of the city data [9], *i.e.,* how to aggregate and process data from multiple sources [1, 2] that are in general available in a streaming fashion. In this context, information fusion is of significant importance [10, 13]. For instance, [6] merges satellite, electric utility and census data for early detection of power outages. Another important element is the combination of historical and real-time (dynamic) data [1, 11]. Finally, [7] points out that urban platforms should be open, allowing citizens to easily contribute and/or retrieve data to build applications. Our work is complementary to the mentioned research efforts, focusing on the re-usability of urban data architectures and handling foreseen or unforeseen data availability in smart cities.

As an example of a Smart City application, we focus on predicting congestions in cities, based on historical and current data of public transportation. We aim at smooth portability of tools, developed for one city, to the data that is available in another. Our experience in the INSIGHT[1] and VaVeL[2] European projects shows that deploying an algorithm that was developed in one city (Dublin in this case), in another (Warsaw), becomes at times impossible due to different levels of data availability. Therefore, it is clear that not taking into account the information availability may most likely jeopardize the generalization of any Big Data event-based systems. Even across different areas of the same city we may find different levels of availability. For instance, some sensors (measuring traffic, capturing video, *etc.*) may be accessible in a suburb while not being available in another, due to either cost constraints or infrastructure failures.

In this demonstration we present REMI, a *reusable elements* framework to handle varying degrees of information availability *by design* from two complementary angles, namely *graceful degradation* (GRADE) and *data enrichment* (DARE). In a nutshell, we develop reusable machine learning black boxes for mining and aggregating streaming data, either to infer missing data from available data, or to adapt expected accuracy based on data availability. Being able to re-use these off-the-shelve modules can considerably speed-up the development of big data applications involving streaming data. On top of that, this design supports fault tolerance of smart city data architectures since it enables operation even when information sources become unavailable.

DEBS'17, Barcelona, Spain

DOI: http://

[1] http://www.insight-ict.eu/
[2] http://www.vavel-project.eu/

We showcase reusable elements in a transportation use-case, in particular congestion analysis in the city of Warsaw based on event recordings emitted by tram GPS system, the city road network, stop positions and tram schedule.

The rest of the paper is organized as follows. Section 2 explains the basics of reusable elements. System architecture of REMI is given in Section 3, followed by the details of the demonstration (Section 4) and concluding remarks (Section 5).

2 REUSABLE ELEMENTS

Reusable elements should be designed to adapt to various information levels, depending on the available data sources. Therefore, prior to algorithmic solutions, we consider the layering of possible information availability. This involves identifying the minimum data requirements, denoted layer L_0, which is necessary to provide a meaningful answer to the targeted problem and that can be safely assumed to be available in any deployment setting. On top of layer L_0, one can design levels of increased information availability (layers $L_i, i > 0$), adding new data sources that may be less available or, alternatively, more expensive.

Figure 1 shows an implementation of the aforementioned information layering approach, as applied to our transportation use-case (further details in Section 3.1). We assume that the basic information needed is a stream of trams' GPS positioning, without which it would be impossible to perform on-line spatio-temporal analysis. At the next layer, we add the network of roads, which is often a publicly available information. The third layer introduces locations of tram stations, and finally in the last layer consider the schedule (the planned arrival and departure times of trams into and from stations). Note that the data that is often available in Smart Cities complies to one of the aforementioned four levels of information.

Constructed on top of the information layering approach, the REMI solution that we propose in this demo allows to seamlessly analyze congestion given various information levels using (1) graceful degradation and/or (2) data enrichment.

2.1 Graceful Degradation

The graceful degradation (GRADE) approach involves decreasing the accuracy of the output to cope with the unavailability of some data sources. For each information layer, we replace the design of an ad-hoc algorithm with reusable elements. The analysis algorithm A_{L_i} at layer L_i takes as input both the new data source in this layer and the output of analysis algorithm $A_{L_{i-1}}$ of layer L_{i-1} and refines the output of $A_{L_{i-1}}$ given the new data. Each layer is also equipped with a translator T_{L_i} that bridges between the layer algorithm

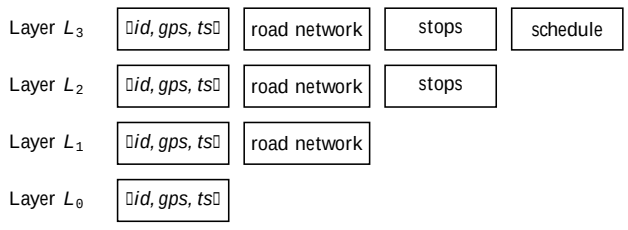

Figure 1: Layers of information availability.

output and the output interface that is shared by all layers. The error in analyzing congestion via GRADE stems from the lack of information at layer L_i.

Figure 2 illustrates the concept of graceful degradation using Smart Cities data. Every layer performs congestion analysis that refines the algorithm of the previous layer. For example, consider the case of the first two layers from the beginning of the current section: L_0 (GPS locations of trams), and L_1 (GPS locations of trams combined with the road network). Algorithm A_{L_0} is unaware of the roads. Hence, it considers slowdowns in tram movements to classify a set of trajectories (and sub-trajectories) as congestion. Adding the road network on-top allows for a proper interpolation of these locations into congested road segments.

2.2 Data Enrichment

An alternative approach to GRADE is data enrichment (DARE). In DARE, we use the available data in L_i to infer the next (missing) layer L_{i+1}. Unlike GRADE, the error when analyzing congestion with DARE comes from the inherent inference inaccuracies (e.g., due to statistical errors) when moving from L_i to L_{i+1} rather than from unavailable information at layer L_i.

As our first example of the DARE approach, we consider a scenario where L_0 information is available, i.e., GPS locations of traveling trams are known over time, yet the road network is missing. Here, one may apply techniques proposed in [4, 8] to infer an approximate road map, which enables the application of A_{L_1}.

As an additional example, consider the information required to perform congestion analysis via A_{L_2}. Specifically, A_{L_2} requires the availability of station locations. However, the available information in L_1 involves only GPS locations of trams over time, as well as the road network. We may apply unsupervised learning to discover points of interest (stations in our case) by employing techniques from [3, 14]. Subsequently, one may apply A_{L_2} for the desired congestion analysis.

3 DEMONSTRATION FRAMEWORK

We now detail the demonstration framework. Specifically, we start with the data model (Section 3.1), followed by the system's architecture in Section 3.2.

3.1 Data Model

In this section, we present a model of the data sources we use to produce a congestion analysis, as well as their different degrees of information availability. The most basic layer of information (Layer L_0, *cf.,* Figure 1) contains the GPS sensor readings from Warsaw trams. These can be represented as a massive distributed stream of tuples (*i.e.,* events) $\sigma = \langle t_1; \dots; t_j; \dots; t_m \rangle$, where each tuple is a quadruple $\langle id; line; \partial ps; ts \rangle$ encoding the identifiers of the tram $id \in \mathbb{N}^+$ and of the line $line \in \mathbb{N}^+$, the GPS location $\partial ps \in \mathcal{P}$ (where $\mathcal{P} = (lon\partial; lat) \in \mathbb{R}^+ \times \mathbb{R}^+$) and a timestamp $ts \in \mathbb{N}^+$ (milliseconds from `epoch`). For the sake of simplicity, we assume that σ has already gone through the classical data cleaning steps (*i.e.,* duplicates and outliers detection). It is worth noting that prefixes of σ of length j, denoted as $\sigma^{h(j)}$, can be stored and used for off-line analysis, while the congestion analysis is performed on the online stream σ.

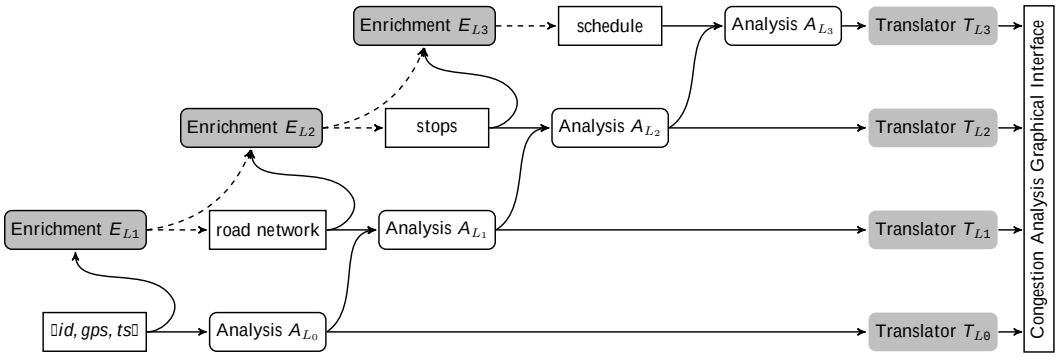

Figure 2: Reusable Elements: GRADE vs. DARE.

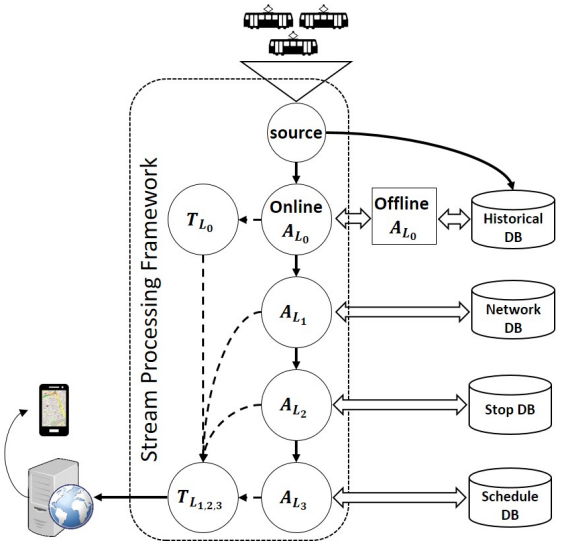

Figure 3: Demonstration Architecture.

The next layer (Layer L_1, *cf.,* Figure 1) adds the network of roads in Warsaw. In particular, we model this data as a function $d : \mathcal{P} \times \mathcal{P} \rightarrow \mathsf{R}^+$ that returns the travel distance using the road network, which is different from bird's fly distance, from one GPS position to another.

The third layer of information (Layer L_2, *cf.,* Figure 1) contains the positions of the trams stops across the city. We model this information as a function $p : \mathsf{N}^+ \times \mathcal{P} \rightarrow \mathsf{N}^+ \cup \{-1\}$ that, given a line identifier and a GPS position, returns either the identifier of the stop or -1 if it is not a stop position.

The top most layer (Layer L_3, *cf.,* Figure 1) adds the scheduled arrival and departure time for each tram line. The schedule is modeled as a function $\mathtt{s} : \mathsf{N}^+ \times \mathsf{N}^+ \times \mathsf{N}^+ \times \mathsf{N}^+ \rightarrow \mathsf{N}^+ \times \mathsf{N}^+$, that given a day (days from `epoch`), a line and stop identifier and a time, returns the expected arrival and departure time.

3.2 System Architecture

Figure 3 shows the general architecture of the demonstration. The additional data for $L_{i \geq 1}$ is stored in databases, namely: `Network DB`, `Stop DB` and `Schedule DB`. The business logic wraps these databases with the functions defined in the previous section. The stream σ of GPS readings is collected and pushed into a stream processing framework (in this implementation we use Apache Flink [12]).

The source duplicates the stream that is (**i**) stored in a database (`Historical DB`) ($\sigma^{h(j)}$) and (**ii**) forwarded into the the first layer analysis algorithm A_{L_0}. A_{L_0} is split into off-line (on top of the `Historical DB`) and on-line analysis. The former provide the tools to the latter to analyze (on-line) the incoming GPS readings. The analyzing algorithm for the following layers are simply pipelined, and their output is processed by the translation operator before reaching the Web server. It is worth noting that only one of the dashed path is operational at any given time, depending on which layer of information availability we want to display. Finally, the GUI is based on a Web page, displaying the city map with the results of the congestion analysis.

4 DEMONSTRATION DETAILS

In this section we provide more details on the implementation of the reusable elements (Figure 2) and how increasing the information availability improves the accuracy of the congestion analysis (Figure 4).

Layer $\mathsf{L_0}$ At layer L_0, the analyzer A_{L_0} uses a combination of clustering algorithms and rule based pattern mining on location and time on the prefix $\sigma^{h(j)}$ (previously) stored in the `Historical DB` to extract and tune a set of rules that are then applied on the on-line data. Then, A_{L_0} takes as input the stream σ of sensors readings and outputs a stream $\sigma_{A_{L_0}}$ of tuples $\langle line; ts; \partial ps_{start}; \partial ps_{end}; delay \rangle$ encoding the line identifier, a timestamp, a starting and ending gps positions as well as the delay between them. Each of these tuples are the result of an aggregation over several gps readings from the original σ stream. Translator T_{L_0} computes the speed observed between ∂ps_{start} and ∂ps_{end} on each tuple of σ_{L_0}, and classifies them into 3 degrees of congestion severity (*i.e.,* none, medium, high). The output is a stream $\sigma_{T_{L_0}}$ of tuples $\langle \partial ps_{start}; \partial ps_{end}; congestion; ts \rangle$, which are ingested by the GUI and shown on the map.

Layer $\mathsf{L_1}$ Layer L_1 introduces the knowledge of the road network stored in the `Network DB` and offered to A_{L_1} as a distance function