Journal of Intelligent Information Systems (2018) 51: 367–388
https://doi.org/10.1007/s10844-018-0524-5

REMI: A framework of reusable elements for mining heterogeneous data with missing information

A Tale of Congestion in Two Smart Cities

Avigdor Gal[1] · Dimitrios Gunopulos[2] · Nikolaos Panagiotou[2] · Nicolo Rivetti[1] · Arik Senderovich[3] · Nikolas Zygouras[2]

Received: 25 July 2017 / Revised: 26 July 2018 / Accepted: 31 July 2018 /
Published online: 11 August 2018

Abstract
Applications targeting smart cities tackle common challenges, however solutions are seldom portable from one city to another due to the heterogeneity of smart city ecosystems. A major obstacle involves the differences in the levels of available information. In this work, we present REMI, which is a mining framework that handles varying degrees of information availability by providing a meta-solution to missing data. The framework core concept is the REMI layered stack architecture, offering two complementary approaches to dealing with missing information, namely *data enrichment* (DARE) and *graceful degradation* (GRADE). DARE aims at inference of missing information levels, while GRADE attempts to mine the patterns using only the existing data.We show that REMI provides multiple ways for re-usability, while being fault tolerant and enabling incremental development. One may apply the architecture to different problem instantiations within the same domain, or deploy it across various domains. Furthermore, we introduce the other three components of the REMI framework backing the layered stack. To support decision making in this framework, we show a mapping of REMI into an optimization problem (OTP) that balances the trade-off between three costs: inaccuracies in inference of missing data (DARE), errors when using less information (GRADE), and gathering of additional data. Further, we provide an experimental evaluation of REMI using real-world transportation data coming from two European smart cities, namely Dublin and Warsaw.

Keywords Reusable elements · Missing information · Mining · Complex patterns · Enrichment · Graceful degradation

✉ Avigdor Gal
avigal@technion.ac.il

Dimitrios Gunopulos
dg@di.uoa.gr

Extended author information available on the last page of the article.

1 Introduction

The era of Big Data presents numerous opportunities to analyze volumes of data, continuously providing up-to-date insights as data changes. Data scientists spend much of their time struggling to identify and obtain high-quality data to support the needs of advanced data mining algorithms to enable the provision of quality outcomes. However, once reality strikes, quality data may be costly or even impossible to collect. The reasons for this may vary. At times, the data may simply not be within reach. At other times, decision makers would like to first see proven results before reaching into their wallets to make further data collection possible. Finally, run-time issues in data gathering may create temporary shortage of data, e.g., when a data source becomes temporally unavailable due to technical reasons.

The current main interest of urban environment researchers and experts is data processing of smart city data (Schieferdecker et al. 2016), focus on how to perform intelligent knowledge extraction and data mining based on multiple data sources (Artikis et al. 2014; Bockermann and Blom 2012). In this context, information fusion is of significant importance (Schnitzler et al. 2014; Zhang et al. 2016). For instance, Cole et al. (2017) merge 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 (Artikis et al. 2014; Thakur et al. 2015). Finally, Lee et al. (2013) point out that urban platforms should be open, allowing citizens to easily contribute and/or retrieve data to build applications. This paper complements these research efforts, focusing on the re-usability of urban data architectures and handling both foreseen and unforeseen data availability in smart cities.

Our work offers an in-depth investigation of the ability of Big Data projects to handle data availability issues. Specifically, we provide a framework that enables data solutions to run on less-than-perfect sources of information. We name our solution REMI for Reusable Elements for Missing Information. Our approach comprises of elements that can adapt themselves, *by design*, to the changing levels of data availability. The core of the REMI framework is the layered stack architecture, equipped with data processing units with two additional computational elements, namely DARE (*data enrichment*) and GRADE (*graceful degradation*). The former (DARE) aims at enriching a current level of data by completing missing data, while the latter (GRADE) aims at allowing an algorithm to make use of partial data with only minor reduction in performance. We allow decision makers to GATHER (*gather missing data*), instead of DARE-ing, in order to improve the quality of the solution, in case the missing information is of vast importance. Furthermore, we detail three additional components of the framework (the Repository, the Contributors, and the Runtime) and the underlying technologies enabling a real-world deployment of REMI. After introducing the REMI framework, we discuss several relevant properties of REMI, the foremost being the three (and a half) different ways software components can be re-used.

We formalize the instantiation of the REMI layered stack problem as an optimization problem (OTP) that aims at striking a balance between three choices: (**i**) investment in additional data collection (GATHER), (**ii**) (inaccurate) missing data inference (DARE), and (**iii**) using less information for answering queries (GRADE). While the first choice is expensive due to monetary investments required to add measurements, the other two choices may provide unreliable results when answering a query at hand.

To illustrate REMI, we focus on a smart city deployment, which aims at answering congestion queries using public transportation data. In the evaluation, we show that REMI supports fault tolerance of smart city data architectures since it enables query answering even when information sources become unavailable. Furthermore, we demonstrate congestion analysis based on both historical and real-time GPS data that comes from means of

public transportation (e.g., buses). A specific use-case aims at smooth portability of tools, developed for one city, to the data that is available in another. Our experience in the VaVeL[1] European project 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. Here, we present our main use-cases by telling *a tale of two cities*.

1.1 A tale of two cities: The use-case of dublin and warsaw

> "It was the best of times, it was the worst of times, ... in short, ... its noisiest authorities insisted on its being received..."
>
> *A Tale of Two Cities*
> *Charles Dickens*

Our use-case for this work comes from the domain of smart cities. Smart cities provide an enticing use-case of big data event-based methods and technologies (Chen et al. 2014), 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 city efficiency (e.g., provide congestion predictions during rush hours) (Artikis et al. 2014). Since these issues are common to practically any medium-to-large sized cities, any solution should be as flexible as possible in terms of available data to allow portability despite the heterogeneity of different smart city ecosystems.

We consider the use-case of public transportation analysis in *two* large European cities, namely Dublin (Ireland) and Warsaw (Poland). A key query to be answered in this setting is that of congestion analysis, which takes the form of mining delay patterns that indicate congestion from a stream of incoming events. The events are collected in real-time from GPS systems that are installed on board various means of public transportation.

The two cities share commonalities when it comes to analyzing congestion, and collect similar low-level event data that comes from GPS systems. However, the two datasets exhibit major differences in the level of available information. While Dublin bus data is enriched with features that indicate stopping at stations, delays with respect to the timetable, and congestion levels, Warsaw data comprises raw spatial-temporal observations that merely indicate buses locations and the corresponding timestamps. Warsaw's City Council faces the dilemma of collecting additional information that would raise the quality of their tram data. As an alternative approach, they consider to infer the missing information using data mining methods, or to use existing congestion mining algorithms that use low-level GPS data.

1.2 Contribution and Structure

The novelty of this work is four-fold:

– We present the REMI layered stack architecture for big data applications, equipped with capabilities to support effective processing in settings where only partial data is available (Section 2.1).

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

– We detail the REMI framework and the underlying technologies supporting the layered stack (Section 2.2).
– We show that the proposed architecture can be developed incrementally, is fault tolerant and highly reusable (Section 2.3).
– We propose a mapping of the proposed framework into an optimization problem (OTP), highlighting a trade-off of missing data handling modes and supporting decision making (Section 3).

We have implemented a prototype of our solution in an urban transportation setting, and experimented with Warsaw data to provide a proof-of-concept.

The rest of the paper is organized as follows. Section 2 presents the REMI architecture, as well as GRADE, DARE, and the REMI framework. In Section 3, we provide a mapping from REMI into an OTP to decide between DARE, GRADE, and GATHER. The implementation and an empirical evaluation are outlined in Section 4, while Section 5 concludes the paper.

2 Architecture

In this section we present REMI, a framework of reusable elements for mining event data, when facing information unavailability. Alongside the architecture (cf., Fig. 1) we introduce notation that would serve us in the definition of the OTP in Section 3. We also specify the components that enable a real-world deployment of REMI and discuss several useful properties of the architecture.

2.1 REMI layered stack

The REMI layered stack, illustrated in Fig. 1, is a conceptual description of data availability with $\mathcal{L}_\alpha = \{l_0, \ldots, l_{n_\alpha}\}$, a finite set of data layers. α corresponds to the query that we mine using REMI, e.g., a congestion query ($\alpha = cong$). We enumerate the layers in $\mathcal{L}_\alpha$ with n_α being the maximal number of layers in a REMI deployment to mine a pattern α. In what follows, for the sake of clarity, we omit the subscript α when clear from the context.

For every layer in $\mathcal{L}_\alpha$ we consider a corresponding set of data sources denoted $\mathcal{S}_\alpha = \{s_0, \ldots, s_{n_\alpha}\}$. The starting data source, denoted s_0, is the minimal amount of information that is necessary in order to mine the pattern of interest. Hence, we assume that it is available in any REMI deployment that corresponds to $\mathcal{L}_\alpha$. On top of source s_0, we place

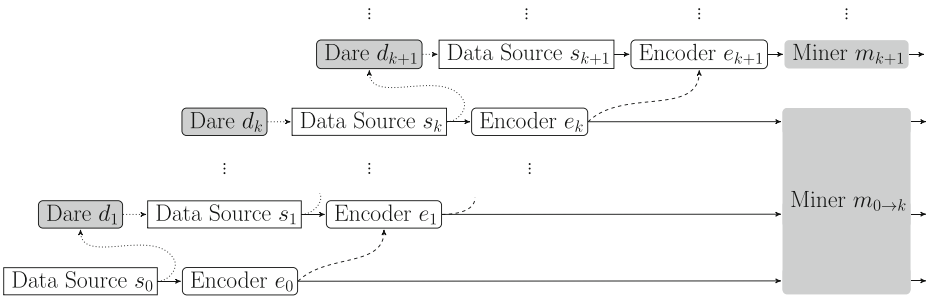


Fig. 1 REMI: DARE, GRADE, and what is in between

additional data sources s_i ($i > 0$) in an increasing order that corresponds to increasing data availability and/or increasing cost of data gathering. Note that in practice, constructing a total order between information sources is far from trivial and in most cases a designer must strike a balance between two criteria, namely cost of data gathering and inaccuracies due to information unavailability.

In this work we assume that the available data sources are non-redundant. However, the model could be easily extended to include redundant data sources. In particular, each data source s_i is redefined as a set of redundant data sources. The most straightforward approach is to hide the redundancy from the framework selecting a single data-source to be used at deployment time. A more refined approach would involve the application of multi-view learning techniques (Xu et al. 2013; Cuzzocrea et al. 2015) on the whole set of redundant data sources to improve the layer performance.

To demonstrate the layered architecture and corresponding data sources, let us consider the Dublin use-case. To this end, we instantiate the set $\mathcal{S}$ as $\mathcal{S} = \{s_0, s_1, s_2\}$ such that s_0 corresponds to GPS location data, s_1 represents the road network, and s_2 is a dataset of bus stop GPS locations (cf., Fig. 2). Although real-time GPS data is an expensive data source, it comprises an essential building block for many existing congestion mining algorithms; hence it is considered to be part of s_0. Furthermore, GPS data that stem from mobile sources is becoming more common in Smart City settings. The road network (s_1) is often a publicly available resource (for instance through OpenStreeMap (OpenStreetMap Foundation)). Finally, accurate bus stop positions may not be available in all settings. Specifically, for the Warsaw use-case, bus stop GPS locations were not readily available at the beginning of the project and had to be gathered by connecting to an external data source.

Beyond the data source layering, for each layer $l_i \in \mathcal{L}$ we assume the existence of two corresponding software components, namely encoders ($e_i \in \mathcal{E}$) and miners ($m_i \in \mathcal{M}$). An encoder is a software component that takes as an input a data source s_i and the output of encoder from the previous layer, e_{i-1}, $i > 0$.

The output of an encoder e_i is data in the input format expected by the corresponding miner, m_i. Hence, encoder e_i combines the data from all previous layers (starting l_0). Further, the input to miner m_i is based on all available data sources up to layer i (i.e., $\bigcup_{j=0}^{i} s_j$) (Fig. 1 shows only the links between consecutive layers to avoid cluttering the figure). Therefore, the output of m_i is expected to be more accurate than the output of m_{i-1} ($i > 0$). It is worth noting that the architecture does not assume that every layer implements the mining algorithm from scratch; it may leverage software components (i.e., miners) from previous layers.

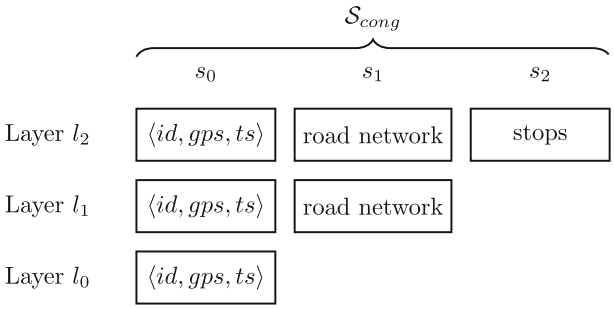


Fig. 2 $\mathcal{L}_{cong}$ Data sources layering

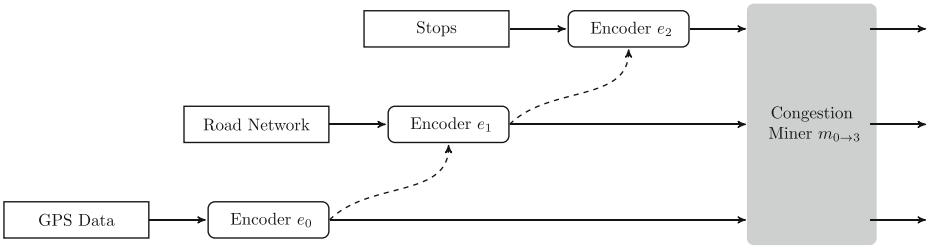


Fig. 3 REMI layered stack instantiation in Dublin

To demonstrate the deployment of REMI, we return to the Dublin use-case and consider all its REMI components (cf., Fig. 3). Specifically, a single mining algorithm $m_{0\to2} = m_0 = m_1 = m_2$ detects congestion patterns. The miner is modular in the sense that it can work with s_0 (GPS data only), as well as with other data sources that are added on-top (i.e., road network and station GPS locations). To this end, $m_{0\to2}$ exploits *slowdown* events to detect congestion. Clearly, the more information is available, the better. For example, what is detected as congestion at l_0 can be a slowdown due to an arrival of a bus into a crowded junction or station. Hence, with additional information on the road map, and on bus stop locations, $m_{0\to2}$ performs better in estimating congestion.

REMI supports two approaches for handling missing layers of information. The first approach is termed data enrichment (DARE). It attempts to complete missing information by inferring its data source. Once the data source is reconstructed, one can apply the corresponding encoder and miner. The second approach is referred to as graceful degradation (GRADE), and it attempts to operate the encoder and miner of the highest layer available. For example, in the Warsaw use-case (cf., Fig. 4) only layers l_0 and l_1 are available, while l_2 is DARE-ed. Alternatively, GRADE l_2 implies answering the query without using stop locations. Finally, the missing information can be collected, assuming the dataset for layer l_2 exists. The two approaches, DARE and GRADE, are discussed in more details next, Sections 2.1.1 and 2.1.2, respectively.

2.1.1 Graceful degradation

The graceful degradation (GRADE) approach involves decreasing the accuracy of the output to cope with data source unavailability. Let $\widehat{S}$ denote the set of available data sources in an instantiation of a REMI deployment $\mathcal{L}_\alpha$ such that $\widehat{S} \subset \mathcal{S}_\alpha$ (there are missing data sources). Let k be the largest index in $\mathcal{S}_\alpha$ that is still available in $\widehat{S}$, i.e.,, $\forall i \leq k, s_i \in S$. Then, to GRADE is to use the output of m_k, while ignoring the increased accuracy provided by

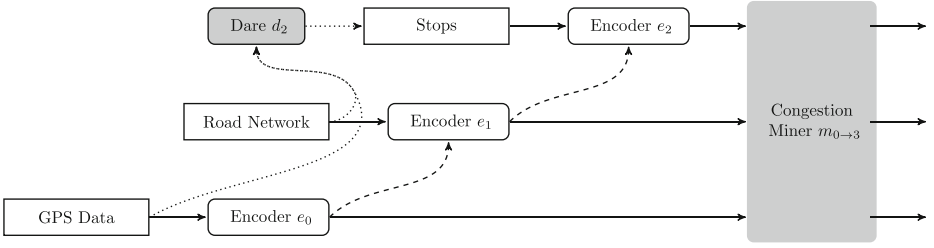


Fig. 4 REMI layered stack instantiation in Warsaw

higher-than-k layers (and their miners), since these require additional data sources. In other words, when GRADE-ing, we 'climb' the dashed lines that appear in Fig. 1, until we reach a point of a missing data source. For the Warsaw use-case where stop locations are not available, we use only layers l_0 and l_1 to mine congestion.

The error in analyzing congestion via GRADE stems from the lack of information at layer l_i with respect to the highest possible layer l_{n_α}. To quantify accuracy loss we define the array $P \in [0, 1]^{n_\alpha}$, which stores accuracy values achieved by the miners at the various layers. We use $P[i]$ for the i-th component of P. Accuracy measures are normalized with respect to layer l_{n_α} (i.e., $P[n_\alpha] = 1$). We shall assume that higher layers yield higher accuracy values, i.e., $\forall l_i \in \mathcal{L}_\alpha \setminus \{l_0\} : P[i-1] \leq P[i]$.

Consider again the Dublin use-case and its first two layers: l_0 (GPS locations), and l_1 (road network). Miner m_0 is unaware of the road network. Hence, it considers the slow-downs of buses nearby to classify events as congestion. Adding the road network allows, among other things, a proper interpolation of bus locations into congested road segments. Thus, one can avoid interference caused by buses on other close-by road segments. The array P can be estimated by Dublin, for which all information layers are available. Then, upon implementation of REMI in Warsaw, the estimated P can be used.

2.1.2 Data enrichment

An alternative approach for handling missing data is DARE (Data Enrichment). We consider a set of algorithms $\mathcal{D}_\alpha = \{d_1, \cdots, d_{n_\alpha}\}$ to impute (i.e., build an estimation of the) missing layers for $\mathcal{L}_\alpha$. Here, d_i analyzes the data available at layers l_{i-1} (i.e., $s_0, \ldots, s_{i-1}$) in order to infer the missing data source s_i at layer l_i ($i > 0$). Unlike GRADE, the error in DARE stems from the inherent inference inaccuracies (e.g., due to statistical errors) when s_i is imputed by d_i. To incorporate the inaccuracies that result from data imputation we expand the array P that stores accuracy values. Let $\mathbf{y}$ ($y_i \in \{0, 1\}$) be a vector of size n_α, where $y_i = 1$ if the data was missing at layer l_i and the corresponding DARE algorithm, d_i, was used to impute s_i. We define $\gamma = \sum_{y_i \in \mathbf{y}} y_i 2^i$ to be the value of the binary representation of $\mathbf{y}$ that uniquely identifies each vector $\mathbf{y}$ and thus any combination of DARE-d layers. Then, P is defined as matrix of size $n_\alpha \times 2^{n_\alpha}$, where $P[i, \gamma]$ is the accuracy achieved at layer l_i when using the DARE approach for all layers l_j with $y_j = 1$.

For example, consider the Dublin use-case in a scenario where only s_0 is available, i.e., GPS locations of traveling buses are known over time, yet the road network (s_1) is missing. Here, one may apply techniques proposed in Chen et al. (2016) and Rogers et al. (1999) to infer an approximate road map, which enables the application of m_1.

As an additional example, consider the information required to perform congestion analysis via m_2. Specifically, m_2 requires the availability of GPS location of bus stops. However, in Warsaw the available information involves only GPS locations of buses over time (s_0), and the road network (s_1). We may apply unsupervised learning to discover points of interest (stops in our case) by employing techniques from Zheng et al. (2009) and Cao et al. (2010). Subsequently, one may apply m_2 for the desired congestion analysis (cf., Fig. 4).

2.2 REMI framework

The REMI framework comprises four components, with the core part being the aforementioned layered stack architecture (Section 2.1). We next introduce three additional components of the framework, namely the Repository, the Contributors, and the Runtime (cf., Fig. 5).

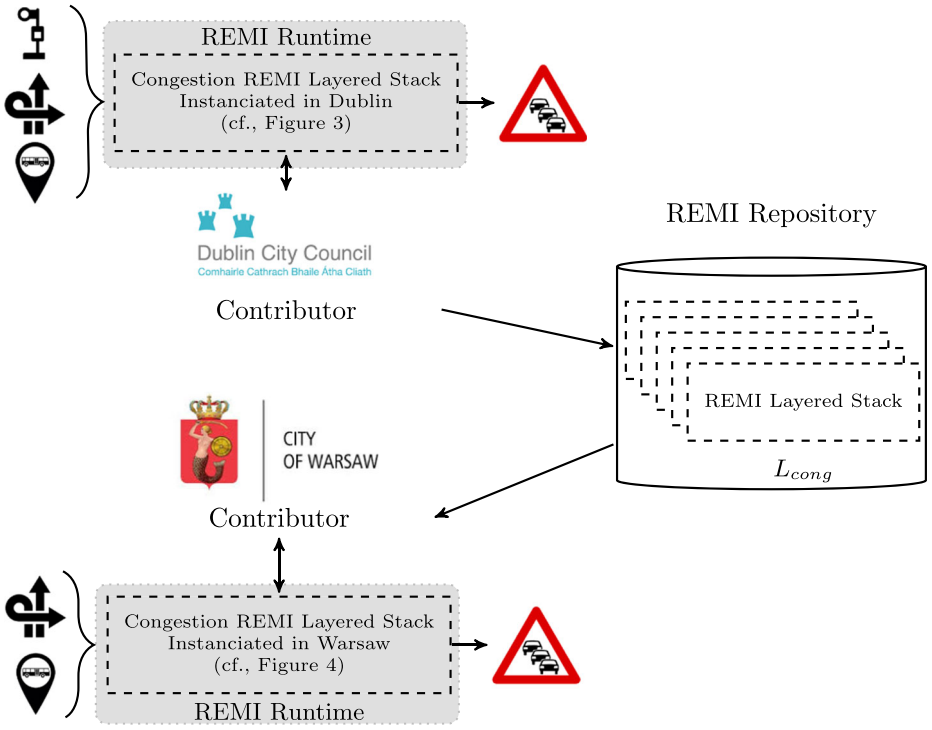


Fig. 5 REMI: Framework

The Repository is, as its name compels, an on-line accessible storage for the REMI layered stack software, uniquely identified, components (i.e., encoders, miners, and DARE algorithms) The Repository also collects the REMI layered stack specifications as a septuple $\langle id, name, \mathcal{L}, \mathcal{S}, \mathcal{E}, \mathcal{M}, P\rangle$, where id uniquely identifies the REMI layered stack, $name$ is a commodity reference name, $\mathcal{L}$ is the set of layers, $\mathcal{S}$ is the set of sources, $\mathcal{E}$ is the set of encoders, $\mathcal{M}$ is the set of miners, and P is the precision matrix.

The Contributors are entities, registered at the Repository, with the ability to retrieve or store REMI software components and specifications from or to the Repository, given some mechanism of access control and ownership. A Contributor can initiate the development of a new REMI stack of layers to solve a specific problem, and can also provide the actual initial implementation of the software components, i.e., the encoders and miners for the initial specification. Contributors can also retrieve REMI stacks and deploy them in their own target setting. Let $\widehat{\mathcal{S}}$ be the set of available data-sources for the retrieved REMI stack, then we have either (**i**) exactly the same data-source $\widehat{\mathcal{S}} = \mathcal{S}$ or (**ii**) a subset of the data-source $\widehat{\mathcal{S}} \subset \mathcal{S}$.

In the first case, the Contributor can trivially deploy the REMI layered stack as is. In the second case, the Contributor can either GATHER, GRADE or DARE. If the daring algorithm is not already available in the Repository, then the Contributor can implement and push it to the Repository.

Contributors also provide or fetch the P matrix. As in any collaborative framework, we expect goodwill participation from the Contributors with a richer deployment setting. These Contributors can compare the performances of the stack when using the actual data or the

data reconstructed by the DARE algorithms, and can provide the related value in the P matrix. The precision values in P from a single Contributor are an empiric estimation of the precision in another deployment. As such, in practice the Repository will store multiple estimations of the values of P as a collection for each entry. Contributors leveraging P to take decision (cf., Section 3) can than apply a number of aggregation function (e.g., min, max, average, etc.) to extract a more sensible value.

Revisiting our Smart Cities use-cases, Dublin is a Contributor that, given all the available data source in the city (GPS buses position, road network, and bus stop GPS locations), has developed the REMI stack to mine congestion patterns $\mathcal{L}_{cong}$ with layers l_0, l_1, and l_2. The encoder and miner software components and the REMI layered stack specification is then stored in the Repository. Warsaw is another Contributor, taking the $\mathcal{L}_{cong}$ REMI stack specification and the available software components from the Repository. Without the bus stop positions, we may choose to either (**i**) use only the first 2 layers of $\mathcal{L}_{cong}$ (GRADE), (**ii**) collect s_2 (GATHER), or (**iii**) implement the required DARE algorithm d_2 to provide an approximation for stop positions. The last option enables the use of all layers in $\mathcal{L}_{cong}$. Dublin is required to provide the accuracy measures for the DARE algorithm designed by Warsaw. Furthermore, Warsaw may leverage its bus timetable to further enhance congestion analysis, thus adding a fourth (l_3) layer to $\mathcal{L}_{cong}$.

Finally, The Runtime (cf., Fig. 6) is the set of software components that eases the deployment of REMI layered stacks. Since each software component of REMI can perform on- and/or off-line computation, encoders, miners, and DARE algorithms run on top of Apache Flink (The Apache Software Foundation) Stream and Batch-Processing System as `DataStream` or `DataSet` operators (`map`, `filter`, etc.) The messaging to and from the Runtime, as well as between the REMI layered stack software components and the Runtime is handled through RabbitMQ. The data sources and miners output are made available to the REMI layered stack and to an external observer through RabbitMQ queue brokered by the Runtime, respectively. An adapter manages the setup and tear-down of the REMI layered stack, as well as the synchronization with the Runtime. The Runtime and the REMI layered stack are developed as dockerized (Docker https://www.docker.com/) components, easing the deployment on a single machine or on a cluster.

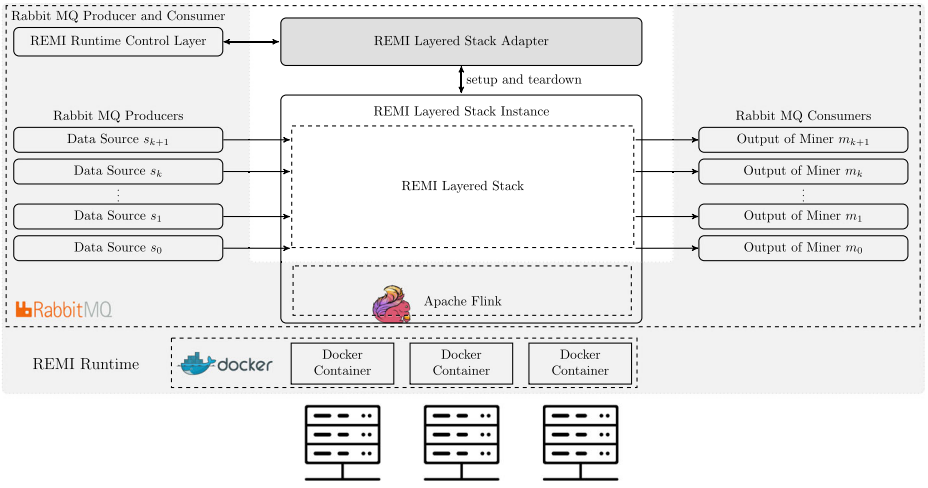


Fig. 6 REMI: Runtime

2.3 Fault tolerance, incremental development and 3.5 degrees of reusability

REMI layered stack has several interesting properties. First, the combination of two abilities, namely to gracefully degrade (GRADE) the accuracy of the mining outcome, and to produce an approximation of missing data (DARE), is not restricted neither to design time nor to off-line computations. If a failure results in data unavailability for some of the sources, our REMI layered stack is able to react and guarantee business continuity. This capability of *fault tolerance* makes the REMI layered stack resilient with respect to data unavailability, which is a fault that is seldom considered in the literature.

Second, the layered architecture also explicitly prompts for an *incremental* implementation of the software stack. This property allows decision makers to commission only an initial subset of the entire stack, and continue with further development only when satisfied with the results of mining low-layer patterns.

Finally, and most importantly, the REMI layered stack allows to reuse software components in three (and a half) dimensions, as follows:

1. *Portability*. The REMI layered stack is developed for a given deployment setting in a specific domain. However, it can be seamlessly deployed in a different target deployment at the same domain (i.e., porting it from one smart city to another).
2. *Inter-stack Reusability.* The second dimension concerns the minimization of the number of ad-hoc software components for a single layer. Specifically, the encoders for all layers (where data is available or provided by a DARE algorithm) are fully used and miners can be shared among layers.
3. *Intra-stack Reusability.* The common modular structure of REMI's layered stack provides the opportunity to re-use software components (i.e., DARE algorithms, encoders, and miners) in more than a single REMI layered stack. Further, intra-stack usability is two-fold: (**i**) it allows for the use of the same modules across different mining problems, and (**ii**) it allows to solve different problems in other domains. For instance, in a hospital use-case where patients and physicians are tracked using real-time locating system (RTLS), one may be interested to infer examination times. To this end, it is possible to re-use the previously introduced road network inference algorithm to estimate the indoor layout of the hospital, thus imputing s_1 on top of the RTLS data source (s_0).

The cross-domain usage of mining algorithms implemented in different REMI stacks can easily be compared to transfer learning (Pratt 1993; Mihalkova et al. 2007) with one major difference. Transfer learning leverages the training from an initial (source) problem, to speed up the training on another (target) problem. For instance, considering neural networks, it means to use the network weights obtained through on a source problem in order to speed up the training on a target problem. Here we do not argue to reuse the model, but rather the algorithm to learn the model.

3 Decision making support with REMI

The REMI framework lands itself well to coping with different levels of information availability, while at the same time providing well-grounded tools to support decision making, as we demonstrate next.

3.1 Optimization problem

Recall that $\mathcal{S} = \{s_0, \ldots, s_n\}$ being the possible data sources in a REMI layered stack. We define the set $\widehat{S} \subseteq \mathcal{S}$ to be the available datasets in the target deployment and assume that $s_0 \in \widehat{S}$. A decision maker is then faced with several ways to instantiate the stack, as there are several approaches that she can use for each layer (i.e., GRADE, DARE, and GATHER). A set of tools that support this decision making would then be welcomed by decision makers. In this section, we show how one can formalize a multi-objective optimization problem (MOTP), based on a REMI layered stack. Since the MOTP is intractable, we relax it in the form of a Linear Programming Problem (LPP). This formalization can then be easily solved and leveraged to drive future decision making.

Let $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ be three vectors of decision variables, indicating whether we use GATHER (1), or DARE (2) for layer l_i whenever the data source s_i is missing. Alternatively, one can USE (3) the layer l_i when the data source s_i is available. Specifically,

$$x_i = \begin{cases} 0, \text{ do not Gather layer } s_i \notin \widehat{S} \\ 1, \text{ Gather layer } s_i \notin \widehat{S} \end{cases} \tag{1}$$

$$y_i = \begin{cases} 0, \text{ do not Dare layer } s_i \notin \widehat{S} \\ 1, \text{ Dare layer } s_i \notin \widehat{S} \end{cases} \tag{2}$$

$$z_i = \begin{cases} 0, \text{ do not Use } s_i \in \widehat{S} \\ 1, \text{ Use } s_i \in \widehat{S} \end{cases} \tag{3}$$

Furthermore, we define two assistive decision variables. First, $\gamma = \sum_{y_i \in \mathbf{y}} y_i 2^i, \gamma \in \{0, \ldots, 2^{n_\alpha}\}$, is the value of the binary representation of $\mathbf{y}$ which uniquely identifies each vector $\mathbf{y}$ and thus any combination of DARE-d layers. Second, $k = \sum_{s_i \in \mathcal{S}} x_i + y_i + z_i, k \in \{0, \ldots, n_\alpha\}$ is the sum of vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, which can be interpreted as the maximum reached layer, the one that is used for the analysis.

We define a multi-objective optimization problem that aims at minimizing the cost of gathering missing data, and the cost of reduced accuracy due to both GRADE and DARE. The REMI-based MOTP is:

$$\min_{\mathbf{x},\mathbf{y}} \langle 1 - P[k, \gamma], \sum_{l_i \in \mathcal{L}} x_i g_i \rangle \tag{4}$$

$$s.t.\ \gamma = \sum_{y_i \in \mathbf{y}} y_i 2^i, \tag{5}$$

$$k = \sum_{s_i \in \mathcal{S}} x_i + y_i + z_i, \tag{6}$$

$$\forall s_i \in \mathcal{S} : x_i + y_i + z_i - x_{i-1} - y_{i-1} - z_{i-1} \leq 0, \tag{7}$$

$$\forall s_i \in \mathcal{S} \setminus S : x_i + y_i \leq 1, \tag{8}$$

$$\forall s_i \in S : x_i = y_i = 0, \tag{9}$$

$$\forall s_i \in \mathcal{S} \setminus S : z_i = 0, \tag{10}$$

where $\mathbf{g}$ is the cost vector of gathering data, P is a matrix returning the accuracy (in [0, 1]) achieved using layer k and DARE-d layers $\mathbf{y}$ (cf., Section 2.1).

Constraints (5) and (6) are used to express the assistive decision variables γ and k, respectively the identifier of the DARE-d layers and the maximum layer reached by the current solution. Constraint (7) enforces the pipelined structure of the REMI layered stack where no gaps are allowed in the layers. Therefore, if we use s_i, gather it or dare it, then the previous layer data source s_{i-1} must be available, has been gathered or dared. Constraints (8), (9) and (10) forbid the gathering and daring of the same layer, gathering or daring when the data source is available, and using a data source that is not available.

Since the MOTP is intractable (Deb 2014), we relax it into a Linear Programming Problem (LPP). The LPP and MOTP are identical in terms of constraints. However, their score functions differ, as in the LPP the score function is linear, and is defined as:

$$\min_{\mathbf{x},\mathbf{y}} c(1 - P[k, \gamma]) + \sum_{l_i \in \mathcal{L}} x_i g_i, \tag{11}$$

with c being the scaling factor that normalizes (in)accuracies and gathering costs. The problem is linear[2] since all its constraints are linear, while its score function depends a linear combination of the decision variables.

Recall that (cf., Section 2.2) the Contributors are responsible (and able) to fill the matrix P. The decision makers may not be confident in the values of P retrieved from the Repository, and thus on the LPP solution. Then, they can collect a sample of the data that has to be GATHER-ed or DARE-d, either to validate the solution or to solve the LPP with the resulting values for P. Considering $\mathbf{g}$ and c, the former can be easily provided by the decision maker itself. On the other hand, c may turn out to be hard to asses. A common approach to handle these constant in bi-objective optimization problem is to run the linear program with several values of c, creating a Pareto curve (Deb 2014). The Pareto curve guides decision makers to either choose the solution or value of c that they deem more relevant.

The linear problem can be slightly modified in order to shift the focus on either the financial or the accuracy cost. Replacing the objective function (4) with (12) and adding Constraint (13), we maximize the accuracy with a bounded budget $\overline{g}$. Replacing the objective function (4) with (14) and adding constraint (15), we minimize the financial cost with a bounded accuracy loss $1 - \overline{p}$.

$$\min 1 - P[k, \gamma] \tag{12}$$

$$\sum_{i \in \mathcal{L}} x_i g_i \leq \overline{g} \tag{13}$$

$$\min \sum_{i \in \mathcal{L}} x_i g_i \tag{14}$$

$$1 - P[k, \gamma] \leq 1 - \overline{p} \tag{15}$$

3.2 Assessing development and data gathering

In Section 2.3 we discussed how the REMI layered stack structure can naturally develop in an incremental fashion. However, additional information is needed to decide whether it is worthwhile (accuracy wise) and affordable to develop an additional layer in a stack. From the presented LPP we can easily extract (16) and (17), to return the increase in accuracy

[2]The objective function is indeed linear since we can define an assistive decision variable $w_{k',\gamma'} \in \{0, 1\}$ that equals 1 whenever the value of k is set to k' and the value of γ is set to γ'. We can then rewrite $c(1 - P[k, \mathbf{y}])$ as $c \sum_{k' \in \{0,\ldots,n_\alpha\}} \sum_{\gamma' \in \{0,\ldots,2^{n_\alpha}\}} (1 - w_{k',\gamma'} P[k', \gamma'])$.

($\Delta_{p_{k+1}}$) and cost (Δ_g) to be expected when moving from layer l_k to layer l_{k+1}, either using the DARE or the GATHER approaches, respectively.

$$\Delta_{p_{k+1}} = \begin{cases} P[k+1, \gamma + 2^{k+1}] - P[k, \gamma] & , \text{Dare} l_{k+1} \\ P[k+1, \gamma] - P[k, \gamma] & , s_0 \in S \vee \text{Gather} l_{k+1} \end{cases}, \tag{16}$$

$$\Delta_g = \begin{cases} 0 & , s_0 \in S \vee \text{Dare} l_{k+1} \\ g_{k+1} & , \text{Gather} l_{k+1} \end{cases} \tag{17}$$

Note that the development of DARE algorithms may have associated costs, which we ignored for the sake of simplicity. However, it can be easily incorporated into the LPP.

4 Implementation and experimental evaluation

This section details the implementation of the REMI architecture, with respect to two smart city use-cases, and presents the results of the experimental evaluations. The first use case is the running example presented so far, i.e., detecting congestion in the cities of Dublin and Warsaw. The second use case shows the use of REMI to estimate the arrival time of buses in Dublin.

4.1 Datasets of two cities

To evaluate the approaches, we used data from both Dublin and Warsaw. Both datasets comprised GPS recordings with approximately 20 seconds between two consecutive readings per vehicle. Data was collected from public transportation travelling through the two cities.

Table 1 presents a sample of the Dublin data. Every recorded event consists of a trip identifier, a timestamp, and a GPS location. Trips are uniquely identified every time a bus departs from a terminal stop.

In addition, we received a set of stop positions and extracted the corresponding road networks (expressed by the nearest stop in Table 1). Subsequently, this resulted in a deployment of REMI at layer l_2 for both cities. For the experiments, we have used two datasets, termed CONG and ETA. The CONG dataset consists of a single month of Dublin data, namely September 2014. Every day consists of approximately 1.5 million GPS recordings.

Table 1 Sample of bus data from Dublin

Event Id	Trip_Id Id	Timestamp	(Lon,Lat)	Nearest_Station Pattern
1	36006	1415687360	(−6.266066,53.338269)	Leeson Street Lower (846)
2	36012	1415687365	(−6.266332,53.408386)	North Circular Road (813)
3	36009	1415687366	(−6.130316,53.254202)	Parnell Square (264)
4	36006	1415687381	(−6.446961,53.254202)	Leeson Street Lower (846)
5	36009	1415687386	(−6.341833,53.289558)	O'Connell St (6059)
6	36012	1415687386	(−6.314403,53.340444)	North Circular Road (814)
7	36006	1415687401	(−6.251677,53.340444)	Leeson Street Upper (847)
8	36009	1415687406	(−6.46544,53.356864)	O'Connell St (6059)

For the Warsaw use-case we used a single month, July 2016, with nearly 970000 observations per day. Note that in Dublin data, congestion information (whether an event occurred in a congested area) exists, and serves as our ground truth for the DARE experiment.

The estimated time of arrival (ETA) dataset is created from the GPS measurements reported by the buses that move within the city of Dublin. More specifically, the buses fleet in the city of Dublin consists of 911 vehicles and each of them periodically reports its position every 20 seconds. The ETA dataset is constructed from the trajectories of the line 046A, one of the longest lines in the city of Dublin. The same bus line was also studied in Gal et al. (2017)

Trajectories for the period from 05/02/2018 to 14/02/2018 were used for the evaluation of all the proposed miners. Due to the noisy nature of the GPS sensors, some of the trajectories can often be problematic including points with inaccurate positions. After cleaning the trajectories we ended with a dataset of 531 trajectories. Using these trajectories we constructed a dataset of 28230 ETA queries with lengths that vary from 0.5km to 18km.

4.2 Detecting congestion in two heterogeneous cities

Below, we provide the implementation details of the software components of $\mathcal{L}_{cong}$ for our congestion mining use-case deployed in Dublin and Warsaw (cf., Fig. 3). The three layers with their data sources (s_0, s_1, s_2), encoders (e_0, e_1, e_2), and miner (m) are implemented as follows.

The miner, m, takes as input a data structure that lists all road segments in the city, alongside several attributes (e.g., position, and average speed). Further, it receives a continuous flow of detected speed events associated with every road segment. Using this information, the miner detects whenever a slowdown occuring in a given segment is to be classified as a congestion event.

The first encoder, e_0, does both off- and on-line processing. The off-line computation uses historical GPS position of the buses to approximate road segments and their attributes. The on-line processing translates the buses GPS position events into the speed events expected by the miner.

The second encoder, e_1, uses the road network to replace the approximated road segments and improves speed computation by taking into account the traveled distance instead of the Haversine distance between two GPS positions. In Fig. 7a, we observe that using an approximate road network, m overestimates the slowdown by cutting through the turn. With the real road network at level l_1, the miner obtains more accurate congestion estimation, as presented in Fig. 7b. Finally, the third encoder, e_3, adds to the attributes of road segments the presence of bus stops, which can than be taken into account by the miner. Knowing the position of bus stops using s_2, reduces the congestion severity for some of the segments (see Figs. 7b and c).

(a) Layer l_0 (b) Layer l_1 (c) Layer l_2

Fig. 7 Example of a congestion map for increasing information availability

4.2.1 Results

In our experiments, we aim at congestion mining in two scenarios: (**i**) we reduce the amount of information used to assess the impact of missing data on the results of congestion mining (GRADE), and (**ii**) we impute stop locations in Dublin to demonstrate DARE, and compare the result in terms of congestion analysis and bus stop location inference. Both scenario have been evaluated against the CONG dataset.

Scenario 1: GRADE —To evaluate the first scenario, which revolved around the GRADE alternative for REMI, we ran the congestion miner for layers 2, 1, and 0. For every layer, we used the corresponding miner, with less information for the lower levels. The controlled variables were accuracy measures for inferring congestion for the GRADE scenario. We regarded the congestion mining problem to be binary classification problems (i.e., congestion/no congestion). Subsequently, for the GRADE experiment, we present precision, recall, and the F-measure, which is the harmonic mean between precision and recall. Here, to demonstrate GRADE, we consider the congestion that we mine for l_2 as ground truth. The prior probability on congestion in Dublin was 0.6%. For Warsaw, congestion was inferred from the data, hence no ground truth was present.

Scenario 2: DARE —For the second scenario, we used a DARE approach to infer the missing stops. To this end, we applied a rule-based approach that decides whether a location is a stop. Specifically, we have tested whether the location is visited frequently in intersection with a stopping event. For the DARE experiment we use true congestion values that are readily available in the Dublin data. Here, we use classification rate as our accuracy measure for congestion mining. This is the result of numerous true negatives that appear when using the Dublin's data ground truth. As for stop locations, we present the three measures as in GRADE (precision, recall, F-measure), as we again consider the inference of stop locations to be a binary classification problem. The proportion of recordings of Dublin buses at stations was 21.5%. Hence, a classifier that would predict with accuracy above 21.5% would be considered improvement over a random choice.

GRADE Experiments —In the GRADE experiment, we first tested the ability of the miner m to run under various information availabilities. First, we ran it at l_2 to collect the ground truth for congestion. Then, we removed data sources, going from $\widetilde{S} = \{s_0, s_1, s_2\}$ to $\{s_0, s_1\}$ and then to $\{s_0\}$ only. The results of the experiment are summarized in Fig. 8. As expected, a reduction in the amount of available information has an impact on precision. However, we see that recall is less sensitive to information reduction. We

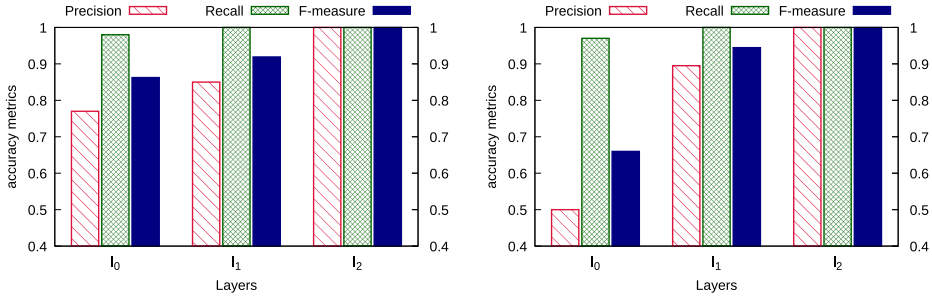


Fig. 8 Precision, Recall and F-Measure for L_{cong} in Dublin and Warsaw

observe that the decline in precision due to GRADE is more severe in Warsaw than in Dublin.

DARE Experiments —As an alternative to feeding the miner m with less information when some of the data sources are not available, we now choose to DARE data source s_2 (bus stop locations) using a threshold-based approach. Specifically, we assume that a stop exists in a location that is visited multiple times by the same vehicle, and where a stop is longer than a fixed threshold τ. In the final results we are using a cross-validated τ, which yielded the greatest accuracy in terms of the classification rate. Unsurprisingly, the results of the DARE experiment, as presented in Fig. 9, show that having more information results in higher accuracy. Furthermore, using DARE to impute missing stops improves accuracy over layer l_1 by approximately 0.47 in terms of classification rate. Lastly, we measured precision and recall for the task of stop location inference, arriving at values 0.64 and 0.66, respectively.

4.3 Estimating bus arrival time in Dublin

In this section, we provide the implementation details of the software components for the second use-case ($\mathcal{L}_{eta}$), where the arrival time of buses in Dublin is estimated. It is worth noting that, the architecture differs from Fig. 3 only in the set of miners, which do not detect congestion, but rather aim at estimating arrival times.

The miner (m) is an implementation of a state-of-the-art solution (Gal et al. 2017) to estimate arrival times. In particular, we use the snapshot principle method due to its simplicity and due to its low computational complexity. The method works by segmenting the journey of a specific line. Then whenever a vehicle traverses a segment a report is generated regarding the travel time required in order to traverse it. In order to answer an ETA query the method initially identifies the journey segments included in the query and then it

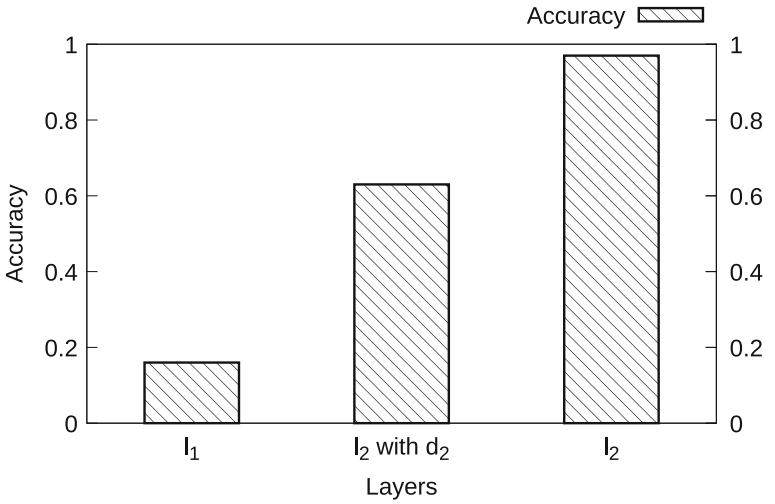


Fig. 9 Accuracy for L_{cong} using the DARE approach in Dublin

calculates the ETA as the sum of the last travel time reported for each of the segments. Thus, the method is able to dynamically adapt to sudden changes such as congestion in specific areas.

The three encoders differ in how they segment the raw trajectories in order to be used by the snapshot principle algorithm.

At the lowest level of information availability, the first encoder (e_0) casts a fine grained grid on the relevant geographical area, where grid entries are the segments. At layer l_1, the bus stops for the trip and their positions are known. The segments provided by the second encoder (e_1) match the original definition (Gal et al. 2017). However, in this layer the road network structure is unknown. Finally at layer l_2, the road network is available including detailed information regarding the ways traversed by the bus line considered. The third encoder (e_1) defines a segment as fixed length portions of the ways traveled by a bus.

Concerning the DARE approach, the missing data are either the bus stops (s_1) or the road network (s_2). To approximate the road network given the GPS trajectories (i.e., DARE s_2), we used the *corridor approach* (Zygouras and Gunopulos 2017).

A corridor can be thought as a route that is commonly traversed by a considerable number of moving objects. Consider, for instance, the trips of 5000 buses in the city of Dublin illustrated in the left part of Fig. 10, while the right part of Fig. 10 shows the 50 most frequently accessed corridors at the city of Dublin. The corridors can be used in order to abstract the main moving patterns from a given set of trajectories. Each path can be written as a sequence of the observed corridors.

In this work we first detected the set of corridors from the set of bus trips. Then we detected the sequence of corridors that reconstructed a particular bus line. In Fig. 11, we illustrate the 8 corridors that were used in order to reconstruct the path of the examined bus line. Then for each bus that crossed a corridor we extracted the travel time. In Fig. 12 we present the travel time of *Corridor 3* for 4 weekdays, showing a daily periodicity, where travel time value do not deviate significantly from its previously reported value.

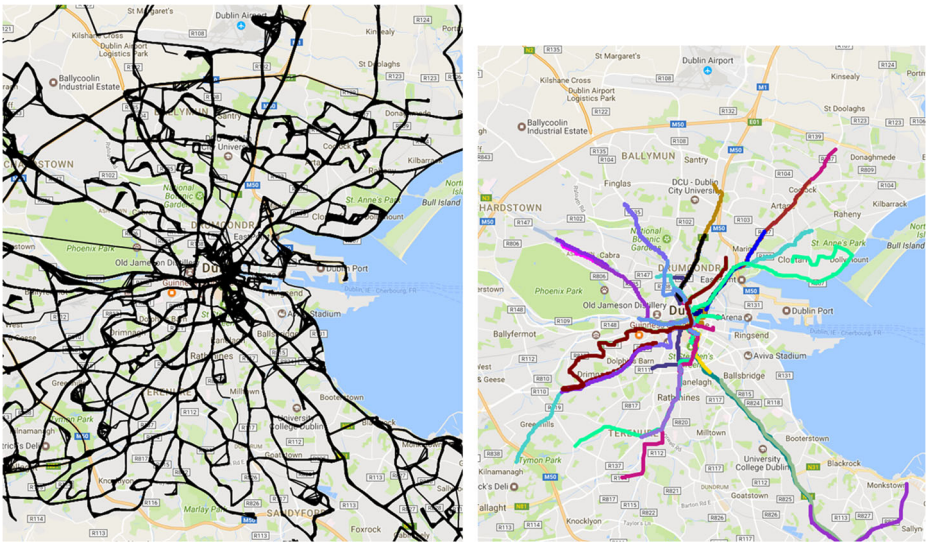


Fig. 10 Bus trips at Dublin (left) and the 50 most frequent corridors (right)

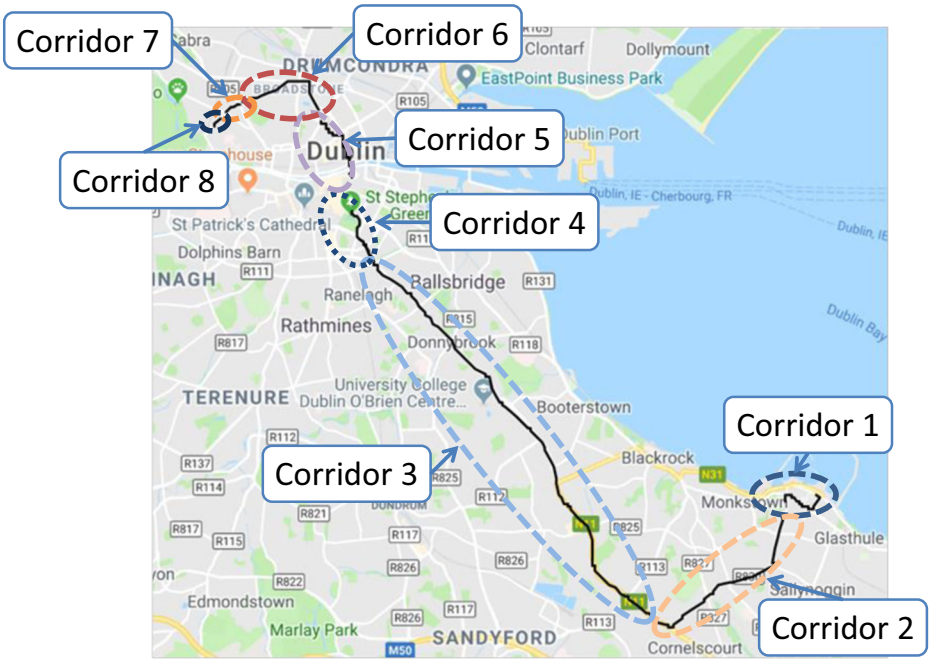


Fig. 11 The 8 corridors that were used in order to reconstruct the examined bus line

Finally, for each query path we estimated the travel time applying the snapshot principle, considering the travel time of the last bus that travelled the corridor. If the query path concerned a part of the corridor and not the whole corridor then our method returned the corresponding weighted last travel time, considering the total length of the corridor and the length of the query path.

To infer the bus stops (i.e., DARE s_1) we used the DBSCAN density clustering algorithm on the locations where a vehicle is potentially stopped. Using a sample of the trips for the bus line under consideration we initially identified all the data points that correspond to stopped positions. Then we run the DBSCAN algorithm on these locations and we assume that each of the clusters formed corresponds to a bus stop.

Similarly to the congestion mining use case, our experiments aim at evaluating the impact of missing data (GRADE) and/or imputed data (DARE) on the results of estimating the arrival time at bus stops.

4.3.1 Results

For the ETA use-case we use the Mean Absolute Error (MAE) and the Mean Absolute Percentage Error (MAPE) as evaluation metrics, given the actual travel time for a route are known.

Figure 13 shows the MAE obtained against the ETA dataset. Figure 13a shows the average MAE for non DARE-ed layers l_0, l_1 and l_2, clearly demonstrating that introducing relevant additional data improves the performance. On average, the first layer of information l_0 achieves an MAE of 271 while using the additional information of layer l_2 results to

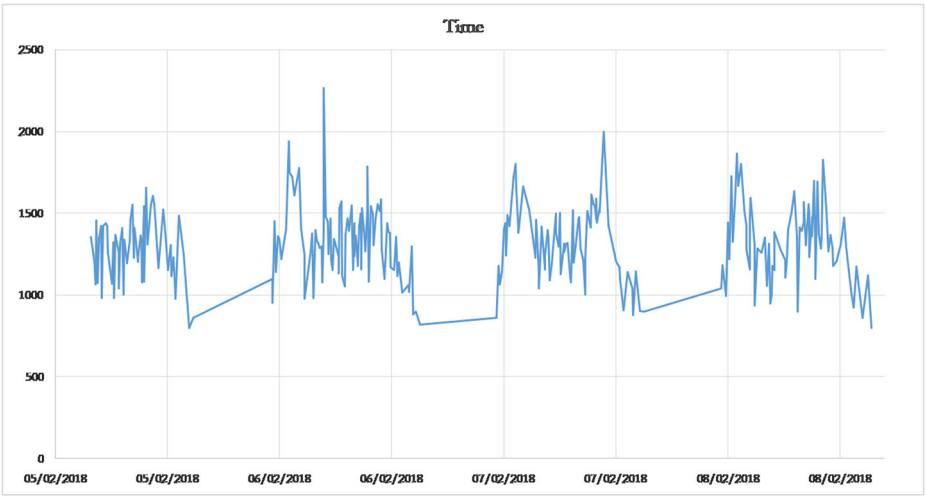
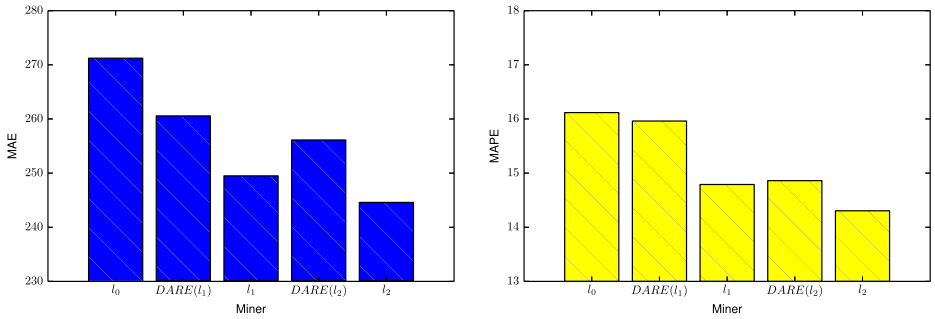


Fig. 12 The travel times for *Corridor 3* for 4 weekdays

an MAE score of 244 an approximately 10% improvement in comparison to layer l_0. The MAPE is 16.1 and 14.3 for the layers l_0 and l_2, respectively.

It is noteworthy that through the DARE approach, we are able to recover most of the gap between to layers. Figure 13 demonstrates that when using DARE to infer the information available on layer l_2 we achieve an MAE score of 256. The improvement is less significant than using the actual data of the layer l_2 but we still achieve a 6% improvement over layer l_0. The MAPE obtained using the inferred layer l_2 information is 14.8 suggesting that the inferred road network leads to 5% improvement in MAPE in comparison to using the actual road network.

Figure 14 shows the MAE as a function of the query distance. As expected, MAE is increased with distance. For long trip queries the baseline layer l_0 scores an MAE of 624 while the l_2 miner scores an MAE of 455, approximately 27% error reduction, suggesting that more layers of information are highly useful in longer and more difficult to estimate queries.



(a) Dublin ETA average MAE (b) Dublin ETA average MAPE

Fig. 13 Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) for the Dublin ETA

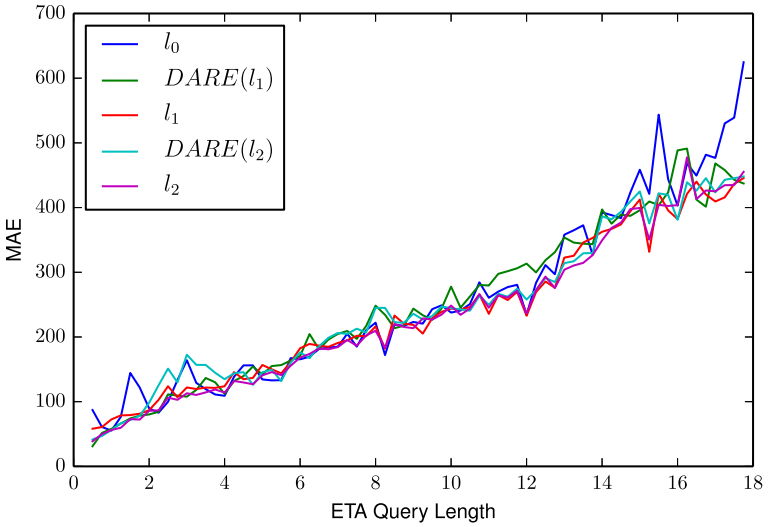


Fig. 14 Dublin ETA MAE for the different miners with respect to the length of the ETA query

5 Conclusion

In this paper, we presented a reusable elements framework (REMI) for mining complex event patterns from data. We have shown how REMI handles missing data by either *data enrichment* (DARE), or via *graceful degradation* (GRADE). Further, we demonstrated multiple scenarios, where REMI's reusability can be utilized. These include fault tolerance, incremental development and inter- and intra-usability. In addition, we have shown a mapping of the REMI architecture into an optimization problem (OTP) formulated via a Linear Programming Problem (LPP). The LPP balanced between the three decisions that managers must make when facing missing data sources: gathering, imputation, and graceful degradation of the mining algorithms. Finally, we have demonstrated an implementation and deployment of REMI for two smart city use-cases for the cities of Dublin and Warsaw.

To evaluate the implementation and demonstrate the usefulness of REMI, we performed multiple observational experiments using real-world data. A first experiment has demonstrated the benefit of collecting additional information, as well as demonstrated the accuracy reduction due to GRADE. We used the second experiment to show that imputing missing data DARE can be useful when information is unavailable. A third experiment support those findings through an alternative use-case.

In future work, we would like to extend our experimental evaluation of REMI, to include additional domains (e.g., the hospital data that we have mentioned in Section 2.3). Furthermore, we aim at implementing additional components of the REMI architecture, using state-of-the-art algorithms for the encoders, miners, and DARE (e.g., Pinelli et al. (2009)). Another relevant research direction is to apply multi-view learning techniques to a subset of the REMI layered stack.

Acknowledgements This project received funding from the European Union Horizon 2020 Programme (Horizon2020/2014-2020), under grant agreement 688380.

References

Artikis, A., Weidlich, M., Schnitzler, F., Boutsis, I., Liebig, T., Piatkowski, N., Bockermann, C., Morik, K., Kalogeraki, V., Marecek, J., Gal, A., Mannor, S., Kinane, D., Gunopulos, D. (2014). Heterogeneous stream processing and crowdsourcing for urban traffic management. *EDBT*, *14*, 712–723.

Bockermann, C., & Blom, H. (2012). The streams framework. Technical Report 5. TU Dortmund University, 12.

Cao, X., Cong, G., Jensen, C.S. (2010). Mining significant semantic locations from GPS data. *Proceedings of the VLDB Endowment*, *3*(1-2), 1009–1020.

Chen, C., Lu, C., Huang, Q., Yang, Q., Gunopulos, D., Guibas, L.J. (2016). City-scale map creation and updating using GPS collections. In KDD, pages 1465–1474. ACM.

Chen, M., Mao, S., Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, *19*(2), 171–209.

Cole, T.A., Wanik, D.W., Molthan, A.L., Roman, M.O., Griffin, R.E. (2017). Synergistic use of nighttime satellite data, electric utility infrastructure, and ambient population to improve power outage detections in urban areas. *Remote Sensing*, *9*(3), 286.

Cuzzocrea, A., Folino, F., Guarascio, M., Pontieri, L. (2015). A multi-view learning approach to the discovery of deviant process instances, pp. 146–165.

Deb, K. (2014). Multi-objective optimization. In: Search methodologies, pages 403–449. Springer.

Docker. Inc. Docker. https://www.docker.com/.

Gal, A., Mandelbaum, A., Schnitzler, F., Senderovich, A., Weidlich, M. (2017). Traveling time prediction in scheduled transportation with journey segments. *Information Systems*, *64*, 266–280.

Lee, C.-H., Birch, D., Wu, C., Silva, D., Tsinalis, O., Li, Y., Yan, S., Ghanem, M., Guo, Y. (2013). Building a generic platform for big sensor data application. In: BigData Conference, pages 94–102. IEEE.

Mihalkova, L., Huynh, T., Mooney, R.J. (2007). Mapping and revising markov logic networks for transfer learning. In: Proceedings of the 22nd national conference on artificial intelligence. AAAI'07, pp. 608–614.

OpenStreetMap Foundation. OpenStreetMap. https://www.openstreetmap.org/copyright.

Pinelli, F., Hou, A., Calabrese, F., Nanni, M., Zegras, C., Ratti, C. (2009). Space and time-dependant bus accessibility: A case study in rome. In: 2009 12th international IEEE conference on intelligent transportation systems, pp. 1–6.

Pratt, L.Y. (1993). Discriminability-based transfer between neural networks. In: Advances in Neural Information Processing Systems 5, [NIPS Conference], pp. 204–211.

Rogers, S., Langley, P., Wilson, C. (1999). Mining GPS data to augment road models. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 104–113. ACM.

Schieferdecker, I., Tcholtchev, N., Lämmel, P. (2016). Urban data platforms: An overview. In *Proceedings of the 12th international symposium on open collaboration companion, OpenSym '16, pages 14:1–14:4*. New York: ACM.

Schnitzler, F., Liebig, T., Marmor, S., Souto, G., Bothe, S., Stange, H. (2014). Heterogeneous stream processing for disaster detection and alarming. In: BigData Conference, pages 914–923. IEEE.

Thakur, G.S., Bhaduri, B.L., Piburn, J.O., Sims, K.M., Stewart, R.N., Urban, M.L. (2015). PlanetSense: a real-time streaming and spatio-temporal analytics platform for gathering geo-spatial intelligence from open source data. In: SIGSPATIAL/GIS, pages 11:1–11:4. ACM.

The Apache Software Foundation. Apache Flink. https://flink.apache.org/.

Xu, C., Tao, D., Xu, C. (2013). A survey on multi-view learning. CoRR.

Zhang, D., Zhao, J., Zhang, F., He, T., Lee, H., Son, S.H. (2016). Heterogeneous model integration for multi-source urban infrastructure data. *ACM Trans. Cyber-Phys. Syst.*, *1*(1), 4,1–4,26.

Zheng, Y., Zhang, L., Xie, X., Ma, W.-Y. (2009). Mining interesting locations and travel sequences from GPS trajectories. In: Proceedings of the 18th international conference on World wide web, pages 791–800. ACM.

Zygouras, N., & Gunopulos, D. (2017). Discovering corridors from gps trajectories. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL'17, pages 61:1–61:4*. New York: ACM.

Affiliations

Avigdor Gal[1] · Dimitrios Gunopulos[2] · Nikolaos Panagiotou[2] · Nicolo Rivetti[1] · Arik Senderovich[3] · Nikolas Zygouras[2]

Nikolaos Panagiotou
n.panagiotou@di.uoa.gr

Nicolo Rivetti
nrivetti@technion.ac.il

Arik Senderovich
sariks@mie.utoronto.ca

Nikolas Zygouras
nzygouras@di.uoa.gr

1 Technion – Israel Institute of Technology, Technion City, Haifa, Israel

2 University of Athens, Athens, Greece

3 University of Toronto, Toronto, Canada