International Journal of Software Engineering and Knowledge Engineering

Towards Optimizing Hospital Patient Transports by Automatically Identifying Interpretable Causes of Delays

Pieter Bonte, Femke Ongenae, Filip De Turck
IDLab, Department of Information Technology at Ghent University - imec,
Technologiepark 15
Ghent, 9052, Belgium,
Pieters.Bonte@ugent.be

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

The continuous financial pressure on hospitals forces them the rethink various workflows. We focus on optimizing hospital transports, within the hospital, as they count up to 30% of the overall hospital cost. In this paper, we discuss a self-learning platform that learns the causes of transport delays, in order to avoid these kinds of delays in the future. We pay special attention to the explainability of the self-learning system, such that management understands the learned causes and remains in control over the automated process. This is achieved by providing the learned causes as sentences that can be understood by non-technical personnel and allowing these causes to first be supervised before the system takes them into account. Once approved, the system will calculate how much more time should be assigned to these transports in order to avoid future delays. As a result, the scheduling of patient transportation can be automatically optimized, while management remains full control of the process.

Keywords: Rule Learning, Ontology, Hospital Transport

1. Introduction

1.1. *Background*

Due to the continuous financial pressure, hospitals struggle to balance budget while maintaining quality of care [1, 2]. Hospitals are forced to rethink and optimize various workflows in order to meet the financial constraints. In this paper, we focus on the transportation of patients within hospitals, as in-hospital transportation counts up to 30% of the total hospital cost [3]. Furthermore, not all transport tasks are performed by logistic personnel. It is estimated that nurses spent up to 10% of their time performing transportations of patients or goods [4], instead of taking care of patients. This leads to cost and care implications, but most importantly, due to the shortage of healthcare personnel, to social implications, such as stress-related diseases [4].

The increase in ICT infrastructure in hospitals can aid in the optimization of

hospital's workflows, as the better use of ICT infrastructure is essential to providing better care at lower cost [5, 6]. The advent of the Internet Of Things (IoT) allows the usage of non-intrusive sensors and devices to capture the environment through sensor readings [7]. These sensors and devices can be used to track the locations of various transports, to localize beds & wheelchairs, to easily notify staff members, etc.

Existing solutions have three major shortcomings. Firstly, the algorithms to schedule transports are based on fixed predefined parameters. For example, a transport from a patient room to a medical room always takes the same amount of time. They do not take the hospital's context into account. For example, the current occupancy level of the hospital is disregarded, thus regardless of the hospital's occupy, the same amount of time is scheduled to perform a transport. During visitor hours, the additional visitors might cause the transports to move more slowly. Secondly, the transport schedule is made in advance and last minute changes in the schedule are hard to achieve as the scheduler cannot be dynamically updated. Lastly, since these solutions lack the ability to model the context of what is happening inside the hospital, they are oblivious to why certain transports are late. As such, they cannot learn from past delays and keep making the same sub-optimal decisions.

The high cost and involvement of the nursing staff and the lack of automation make in-hospital patient transport an ideal candidate for optimization.

1.2. *Related Work*

Previous work has focused on learning delays in the health, financial and transport domain. Laskowski et al. [8] investigated how to reduce the patient wait time in the emergency department. The technique is specifically for the emergency department and does not provide interpretable explanations on the algorithmic suggestions.

Markovic et al. [9] proposed a system to learn the passenger train arrival delays, while Rebollo et al. [10] focused on predicting air traffic delays. Xu et al. [11] have focused on predicting traffic delays and Silva et al. [12] investigated the influence and delays on the public transports when certain stations or lines have been closed. However, these techniques predict the amount of time a certain transport will be late, but do not provide interpretable rules that can be supervised and provide an explanation of why the delays occur.

Lecue et al. [13] have shown the importance of explainability, as they allow flagged expenses to be explained to auditors in the financial sector. This allows the auditors to understand why certain expenses were flagged. Diagnosis of traffic congestion has also been investigated, allowing to explain and identify why certain roads are congested [14]. Even though these systems enable explanation, they provide the explanation in the form of rules, which still require domain experts to understand them.

Previous research has also focused on the scheduling of dynamic changing tasks in hospitals. Fiegl et al. [15] describe an online algorithm for dynamic scheduling

of pick-up and delivery tasks in hospitals. Hanne et al. [16], on the other hand, have focused on transport between hospital buildings. Beaudry et al. [17] provide a solution to scheduling dynamic hospital transport requests. They take both in-house transports, i.e. transports within the same hospital building, as campus-based transports, i.e. transports between hospital buildings that need to be provided by an ambulance. Kergosien et al. [18] take into account additional constraints, such as disinfection of a vehicle or type of vehicle needed. These algorithms are able to cope with the dynamic nature of the tasks requests, however, no insights are provided in why these transports are late.

1.3. *Objective*

In this paper, we present a solution, designed in collaboration with two Flemish hospitals, that models the hospital's context by integrating various sources of information: static information regarding the hospital layout, patient & staff information and dynamic data resulting from the sensor stream, such as sensor readings that capture the location of the transports. Based on a historical view of this context, we provide a self-learning module that learns the causes of transport delays. These causes are presented as human-interpretable sentences that can be supervised by management. Upon supervision, when one or more of these causes are accepted by management, the system takes these causes into account and when transports that adhere to the selected causes are requested in the future, the system will calculate the additional time needed to enable accurate scheduling and avoid future delays. Note that highly accurate scheduling is important. When too much time is assigned and a transport is finished too early, the next transport might not be ready for transportation yet. When too little time is assigned, the next transports need to wait until the transport is ready and the whole schedule gets turned over.

1.4. *Requirements*

To provide a system that can learn the causes of delayed hospital transports, provide them in a fashion that non-technical users can understand them and allow these causes to be taken into account such that future delays can be avoided, the following requirements should be adhered:

- **Extendability**: since the ICT infrastructure in hospitals keeps evolving, it should be possible to easily incorporate new sources of information, such that this new information can also be used to learn the causes of delays. These sources can be either sensors & devices providing dynamic data or databases providing descriptions regarding the hospital's static context.
- **Human-involvement**: since it is sensitive to allow an automated system to directly adapt the hospital work processes, it should, therefore, be possible to involve human decision making.
- **Explainability/Interpretability**: to involve human decision making, it

should be possible for non-technical users to interpret and understand which causes the self-learning algorithm is suggesting. It is important that users understand the decisions of an automated system [19].
- **Scalability**: to be applicable to different sizes of hospitals, the algorithms should scale adequately.
- **Usability**: non-technical users must be able to operate the system and evaluate the decisions from the self-learning module. The system should be easily accessible through a Graphical User Interface (GUI). Furthermore, the GUI should provide an easily interpretable overview of the findings of the module. Lastly, it should be intuitive to accept certain identified causes and update the system.

1.5. *Paper organization*

The remainder of the paper is organized as follows. Section 2 details the designed architecture to optimally schedule requested hospital transports, notify the staff, capture the data to learn upon and describes how it can be deployed as a whole in a hospital. In Section 3, we zoom in on the self-learning component and detail the devised algorithms. Section 4 describes how we enabled explainability of the learning component, such that non-technical personnel, can understand the outcome of the learning system. The evaluation of the learning module is described in Section 5 and the outcome is discussed in Section 6. In Section 7 we highlight the most important conclusions and describe opportunities for future work.

2. The AORTA platform

In this section, we describe how the data originating from various sources can be integrated and interpreted and detail the overall architecture of the designed Adaptive Optimization for Resource & Task Assignment in Hospitals (AORTA) platform.

2.1. *Ontologies & Reasoning*

To enable heterogeneous data integration and interpretation, an ontology [20] is composed that models the hospital's context. Ontologies are formal models that semantically describe a certain domain, in this case, the hospital domain. This description is made by modeling the different concepts within the domain and how they relate through the use of relations. Each concept, relation or individual (an instance of the former) is referenceable through a unique Uniform Resource Identifier (URI), e.g. http://aorta.intec.ugent.be/ontology/aorta.owl#PatientTransport. Ontologies are also an ideal tool for integrating IoT data [21], as it provides a uniform model for multiple heterogeneous data sources to adhere to. A part of the ontology designed to describe the hospital transports is depicted in Figure 1[a]. As

[a]The full ontology can be found on http://pbonte.github.io/Ontologies/aorta/aorta.owl

Figure 1 describes, there are two types of transports, i.e. *LogisticTransportTasks* and *PatientTransportTasks*, that each are executed by a certain *Person*, that has a certain *Role* and each *TransportTask* has a specific *Location* as destination, etc. Since the ontology is a uniform and formal model, different data sources can map their data onto the ontology allowing to integrate data from various sources. As such we get a complete, interpreted overview of the current context and status of the hospital transports.

The ontology can define implicit relations within the data, that can be inferred through the use of a reasoner. The reasoning process is comparable to the execution of rules but in a more formal environment. For example, as depicted in Figure 1, a *Nurse* is a subclass of *Staff*, which is a subclass of *Role*. The reasoner will infer that each *Nurse* is also a type of *Role*, when it is provided with an instance of the type *Nurse*. However, more complex constructions can be defined in the ontology. For example, we could define that a *PatientTransport* that has a relation *hasTransportType* to a *Bed* and a relation *hasErrorCode* to a *PatientNotReady* error code can be considered a late transport. This could be defined in a formal way as[b]:

$$LateTransport \equiv \exists hasTransportType.Bed \sqcap \exists hasErrorCode.PatientNotReady$$

When a transport is requested that adheres to this definition, the reasoner will infer that the transport is a *LateTransport*. Say the following fragment describes a newly requested transport:

$$PatientTransportTask(p1), hasTransportType(p1, t1), BedWithPatient(t1)$$
$$hasErrorCode(p1, e1), PatientIsEating(e1)$$

The reasoner will infer that the individual $p1$ is a $LateTransport$ because it knows that *BedWithPatient* is a subclass of *Bed* and *PatientIsEating* is a subclass of *PatientNotReady*. Note that $p1$, $t1$, $e1$ are data instances (individuals) that have a certain type and relations, e.g. $p1$ has the type *PatientTransportTask* and has a relation *hasTransportType* to $t1$.

The definition of these rules allows to incorporate the logic that is specific to a certain domain. Here, we will use the reasoning capabilities to identify transports that adhere to the previously learned causes that identify that a transport might be late. Note that the system assigns more time to these late transports, such that the scheduling can be defined more accurately.

2.2. *Architecture*

Now that we can model the context in our hospital and integrate the data from various data sources, we can describe the components in the AORTA architecture. Figure 2 provides a visual overview of the different components and how they interact.

[b]The $\exists$ denotes an existential quantifier and can be interpreted as 'there exists'.

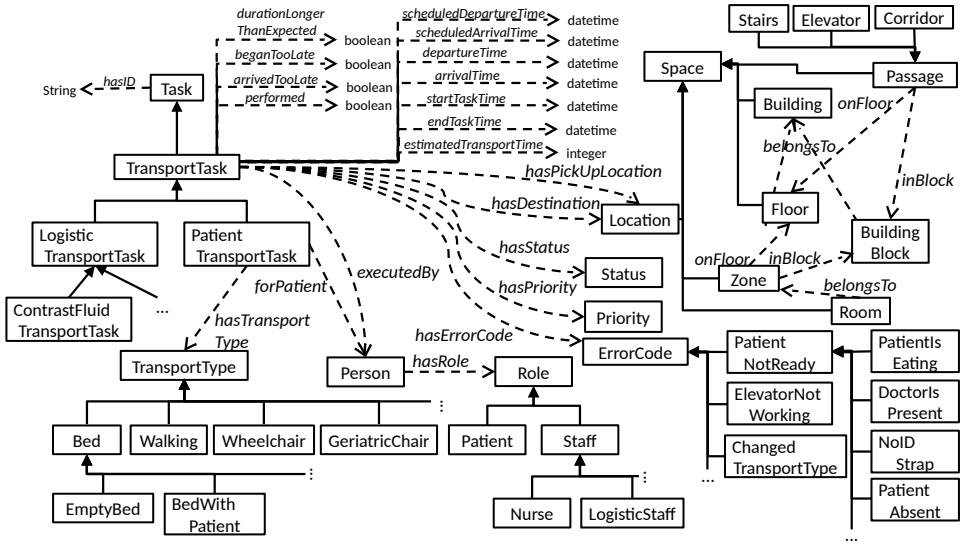

Figure 1. Overview of the most important concepts (visualized in the rectangles) and relationships between concepts (visualized through the dotted arrows) and the hierarchical interpretation of the concepts (interpreted as subconcepts and visualized through full arrows) of the designed hospital transport ontology.

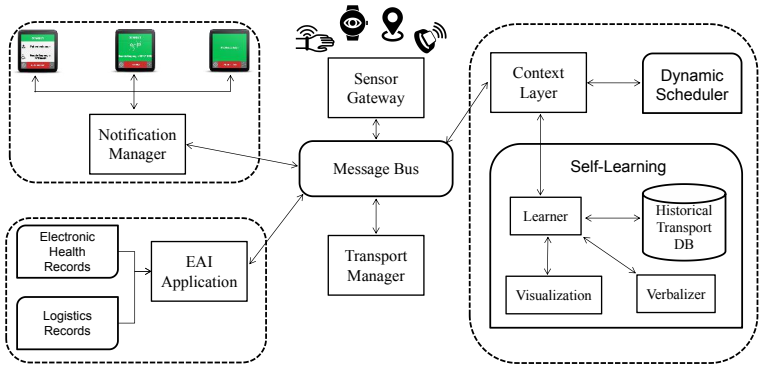

Figure 2. Architecture of AORTA project.

To capture the hospital's environment, **smart devices, wearables, and sensors** are introduced into the hospital's environment. These devices generate the dynamic data, e.g. data describing the location of the staff or the status of the transport. Each staff member is equipped with a smart wearable that allows to

receive, accept/decline transports notifications and transmit location updates. Furthermore, the smart wearable allows to scan, through the use of NFC or QR codes, each patient that needs to be transported, eliminating mix-ups. All this information is pushed on the **Message Bus** that routes the generated data to the interested services, which can subscribe to particular data on the bus. The **Notification Manager** communicates with the wearable devices and notifies the staff members about new tasks or updates. It captures whom of the staff are available on which devices, allowing the *Context Layer* to target the correct staff members when dispatching tasks.

The **EAI Application** module is responsible for integrating the existing hospital data tools, such as the electronic health records of patients and information regarding the staff members and the logistics. This module is responsible for extracting from these existing tools, the information relevant for scheduling and executing the transportation tasks and providing it to the *Context Layer*.

The **Transport Manager** allows the different hospital departments to request transport tasks, which are forwarded to the *Context Layer* for optimal scheduling.

The **Context Layer** captures the current context in the hospital by combining and integrating the data resulting from the *EAI Application* and *Transport Manager* with the dynamic sensor data. The ontology described above is exploited for this purpose. This contextual information is stored in a triple store, i.e. a database for ontological data. The Context Layer provides data to the *Dynamic Scheduler* that needs to know which tasks to be scheduled, which personnel is available and what are their locations, the achievable walking speed considering the current commotion of the hospital, etc. The *Context Layer* uses reasoning to infer missing and implicit data, based on the ontological definitions. The reasoning can indicate which transports need more time to be executed, e.g. by detecting delays and interruptions.

The **Dynamic Scheduler** receives the transportation requests from the *Context Layer* and uses its context to construct an optimal schedule such that all the requests can be handled in a timely manner with an optimal use of resources. To achieve this optimal rostering, the scheduler will request the dynamic context information from the *Context Layer*, e.g., the locations, availability, competences, work load & average walking speed of the staff, busy areas and possible causes of delay. It constantly maintains an overall optimal schedule and updates this schedules as new requests and status updates of on-going transports come in. When a staff member indicates that a transport has been finished, the *Context Layer* will communicate this to the Dynamic Scheduler, which will then assign a new task to this staff member based on this overall optimized schedule. Note that it is the *Context Layer* that indicates how much more time should be assigned to the transports that are expected to be late. The scheduler will try to optimally schedule the tasks based on the provided information from the *Context Layer*. To be able to dynamically update its schedule when new transports are requested, the scheduler uses a dynamic pick-up and delivery model [22].

The **Self-Learning** module consists of four components, the learner, the visualizer, the verbalizer and a historical database. The later keeps a historical view of the *Context Layer*. The learner requests the data from the historical database, such that it can learn from the historical data why certain transports were delayed. Based on this historical context, the learner identifies why transports in the past were delayed, such that these delays can be prevented in the future. For example, the module could learn that certain transports during the visiting hour on Friday are often late and more time should be reserved for them. Once the learner has learned the causes as ontological rules, the verbalizer can transform these rules to human readable sentences such that management can access the identified causes through the visualization and argue their validity while remaining control over the automated system. Once approved, these rules are added to the *Context Layer*. When similar transports are scheduled, they will be identified through the use of the reasoner and the platform will calculate how much more time will need to be incorporated to schedule these kinds of transports accurately. This module is further detailed in Section 3.

2.3. *Implementation*

We now explain some of the implementation details of the components that are not discussed in detail in the remainder of the paper.

The **Message Bus** is based on Mirth Connect[c], a message broker optimized for the transmission of healthcare messages.

The **Notification Manager** receives the scheduled tasks from the *Message Bus*, which are already targeted for a given user. The *Notification Manager* chooses the best way to notify the user, i.e. choose the most convenient device that the user is carrying at a given moment. Then, the *Notification Manager* transforms the tasks into notifications that are tailored to be presented on the selected device.

The **Context Layer** utilizes RDFox [23] to store the contextual data and perform the reasoning on it. RDFox is the fastest reasoning-enabled triplestore, i.e. a database to store ontological data, currently available. The *Context Layer* also needs to map the healthcare messages from the *Message Bus* to the ontological data, this is done through the use of RML [24], that allows to map raw data to the ontology model.

2.4. *Workflow Self-Learning component*

When a new transport is requested, it is captured by the *Context Layer* that models the current view of the hospital's context. After its execution, the details of the transport are communicated with the *Self-Learning module*, such that the module can store a history of past transports. Figure 2 shows the components of the

[c]https://www.nextgen.com/products-and-services/integration-engine

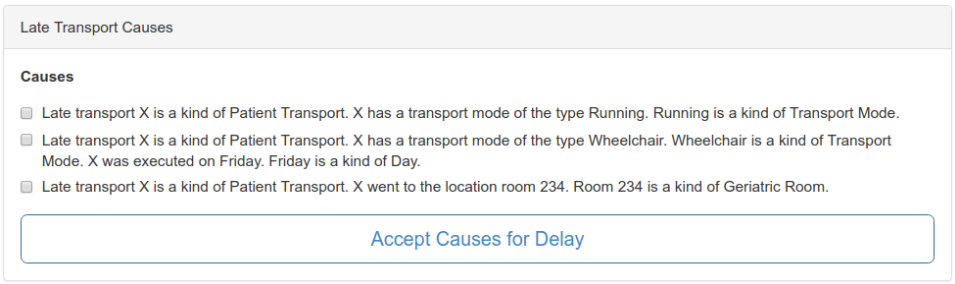

Figure 3. Example of the verbalized rules.

learning module. The new transport arrives through the *Message Bus*, the *Learner* component first captures the transport and stores it in the historical database. It also does some quick preprocessing such that a real-time overview of the transports can be shown in the *Visualization*.

When a more in-depth description of the causes of delays is necessary, through the visualization one can request the *Learning* component to start learning the delays of the transports in the selected time range, e.g. the last month. It will load the data from the historical database and start one of the learning algorithms detailed in Section 3. The *Learner* passes the learned rules to the *Verbalizer* and sends them to the *Visualization* so they can be shown in the visualization dashboard. Figure 3 shows an example of the verbalized rules in the dashboard.

When one of the verbalized causes get accepted by management, the learning module calculates the average delay that was caused by these transports. The additional time is added as part of the rule. The *Context Layer* is then updated by adding the new cause as a rule that will be invoked by the reasoner when a similar transport is being scheduled. Since the extra time necessary to execute the transport within accepted time is part of the rule, the additional time is automatically added to the new transport.

Example 1. (Adding time to late transports) Say the following rule has been accepted:

$$hasTransportType.Bed \wedge hasPeriod.visitingHour \rightarrow LateTransport$$

When adding this rule to the reasoner and a transport is requested with the transport type bed and requested during visiting hours, the reasoner will know that the transport will be late. We can now calculate how much more time, on average, is necessary to provide the scheduler with as accurate data as possible, to prevent future delays. This can be done by calculating how much more time is necessary to finish this task within time. This calculated time can then be added as part of the rule. If 10 additional minutes are required, we can update the rule, to automatically

add this time to the transport:

$hasTransportType.Bed \wedge hasPeriod.visitingHour \rightarrow LateTransport, addTime(10min)$

When the *Context Layer* adds this rule to the reasoner, and a new transport is requested that adheres to this rule, it will know that the transport will be late and that 10minutes additional time should be reserved to perform the transport.

3. Rule Learner module

The *Self Learning module* and more specifically the *Learner*, learns from historical information regarding the transports why certain transports are delayed. By indicating the causes of these delays, future delays can be prevented. We investigated and optimized two techniques to learn the transport causes: an Assocation Rule Mining (ARM) [25] and an Inductive Logic Programming (ILP) [26] technique. We detail each technique and how we combined them for the best performance.

3.1. *Association Rule Mining*

ARM was originally developed to discover hidden knowledge from transactional data, such as relational databases. A transaction is an observation of the co-occurrence of a set of items.

$I = \{i_1, i_2, \ldots, i_m\}$ is defined as a set of m *items* describing the different elements the database could contain and $D = \{t_1, t_2, \ldots, t_n\}$ as a *database of n transactions*, where each transaction in D is a subset of I and can be seen as a database entry. We name a subset of items an *itemset*. $supp(X)$ is the *support* of an itemset X, i.e. the percentage of transactions in the database D that contain X.

An *association rule* r can then be defined as a rule of the form $X \Rightarrow Y$ where X and Y are non-empty subsets of I, and $X \cap Y = \emptyset$. X is called the *antecedent* of r and Y is the *consequent* of r. The support and confidence of a rule are respectively denoted as

$$supp(r) = \frac{|\{t \in D \wedge X \subseteq t\}|}{|D|} \tag{1}$$

and

$$conf(r) = \frac{supp(X \cup Y)}{supp(X)} \tag{2}$$

where de confidence describes the how confident one can be that the antecedent is related to the consequent of the rule.

Mining association rules is the process of finding all association rules with a support and confidence greater than a predefined threshold. This mining process can be divided into two phases. First, frequent itemsets of the transactions have to be computed according to the minimum support threshold. Second, rules are generated from these frequent itemsets with respect to the minimum confidence threshold.

Table 1. Example transactions

transactionID	Walking	WheelChair	VisitingHour	Late
t_1	1	0	1	0
t_2	0	1	0	0
t_3	0	1	1	1
t_4	1	0	0	0
t_5	0	1	1	1

Example 2. (Association Rule Mining) Table 1 shows a database with four items: $I = \{Walking, WheelChair, VisitingHour, Late\}$ and five transactions. If we want to calculate if the association rule $r = \{WheelChair, VisitingHour\} => \{Late\}$ holds, we calculate the support as $supp(r) = \frac{2}{5}$, as only two transaction (t_3, t_5) consist of the itemset $\{WheelChair, VisitingHour\}$ and the database consists out of five transactions. The confidence is calculated as $conf(r) = \frac{supp(X \cup Y)}{supp(X)} = \frac{supp(\{WheelChair, VisitingHour, Late\})}{supp(\{WheelChair, VisitingHour\})} = \frac{2/5}{2/5} = 1$.

Since ARM works specifically on items and transactions, it needs to be adapted to work with semantic data. Our previous work [27] describes in detail how the ontological data can be converted to items and transactions and how various optimizations can be executed. This conversion is necessary as the semantic data described by the ontology can be seen as a graph rather than a set of transactions. The conversion consists of the following steps:

(1) We identify a concept in the ontology we want to get insights from and retrieve all individuals from that concept, e.g. all patient transports in a certain hospital.
(2) We follow all the relations the selected individuals have to other individuals, store them as so-called features, and follow the relations from the new concepts until we reach a certain threshold that indicates how many concepts to follow.
(3) We also store the types of each followed individual, i.e. the ontology concept they have been assigned to and look up the hierarchy of these concepts in the ontology and store the hierarchy as well.
(4) The stored features can now be used as items for the transactions database.

Example 3. (Converting ontological data to transactions)
Say we have the following five (simplified) transports in our ontology:

- PatientTransport(t1), hasTransportType(t1,walking), Walking(walking), duringPeriod(t1,visitorHours),
- PatientTransport(t2), hasTransportType(t2,wheelchair), Wheelchair(wheelchair), duringPeriod(t2,morning),
- PatientTransport(t3), hasTransportType(t3,wheelchair), Wheelchair(wheelchair), duringPeriod(t3,visitorHours),

Table 2. Example ontology conversion to transactions

t	hasTransportType	hasTransportType .wheelchair	hasTransportType .walking	hasTransportType .Wheelchair	hasTransportType .Walking
1	1	0	1	0	1
2	1	1	0	1	0
3	1	1	0	1	0
4	1	0	1	0	1
5	1	1	0	1	0

- PatientTransport(t4), hasTransportType(t4,walking), Walking(walking), duringPeriod(t4,morning),
- PatientTransport(t5), hasTransportType(t5,wheelchair), Wheelchair(wheelchair), duringPeriod(t5,visitorHours)

When we follow the above described steps, we can convert the ontological data into transactions:

(1) The individuals we want to learn about are t1, t2, t3, t4, and t5, which are all PatientTransports.
(2) When following their relations we obtain for each individual the feature hasTransportType and when coupling the relation to the linked individual we obtain the features: hasTranspor Type.walking and hasTransport-Type.wheelchair. Following the next relations, we obtain the features: during-Period,duringPeriod.visitingHours and duringPeriod.morning.
(3) When taking the types into account we obtain the features: hasTransport-Type.Walking, hasTransportType.Wheelchair, duringPeriod.TimePeriod. Taking the hierarchy of the ontology into account we obtain the feature: hasTransportType.TransportType.
(4) We can now convert the selected features to items and use each patient transport individual as a transaction. Table 2 shows a part of these transactions. Note that some items should be filtered out as they do not provide any information gain, e.g. all transaction have the item hasTransportType.

3.2. *Inductive Logic Programming*

ILP is a machine learning technique that combines inductive machine learning and logic programming. ILP is able to learn rules as ontology concepts and fully exploits the semantics describing the data. Thus, ILP can work directly with the semantic

data and generates very accurate rules, however, it is less scalable than statistical approaches such as ARM. Statistical relational learning [28] is an extension of ILP that incorporates probabilistic data and can handle observations that may be missing, partially observed, or noisy. However, since our observations are not possible missing or partially observed, we do not consider it here.

ILP starts from the idea of positive and negative examples and a background describing the domain. ILP tries to learn a hypothesis such that the positive examples follow from the hypothesis but the negative examples do not. Finding the hypothesis is, of course, the difficult part. The Class Expression Learning for Ontology Engenieering (CELOE) algorithm [29] from DL-learner[d] takes a generate and test approach where it appends ontology concepts and relations to the learned hypothesis in order to achieve the highest possible accuracy. This is possible by making the hypothesis more generic or more specific. The latter is calculated on the fact that more positive examples are contained by the hypothesis compared to negative examples.

Example 4. (Inductive Logic Programming) We reuse the transports form Example 3 where both transport t3 and t5 are positive examples. The algorithm will take the following steps:

(1) Create a new concept, e.g. LateTransport;
(2) Add a new concept or relation to the concept, e.g. adding the relation hasTransportType. We thus generate a new concept that says that a LateTransport has a relation hasTransportType.
(3) The algorithm tests the coverage of the new concept and sees that both positive and negative examples have the relation hasTransportType. The coverage is tested by adding the new concept to the ontology and ask the reasoner for all the individuals that have the new concept as a type. The accuracy is calculated as the percentage of individuals that have the type LateTransport and were in fact in the positive examples.
(4) The algorithm can decide to make the generated class more specific by changing any hasTransportType relation specifically for one type of transport, i.e. for wheelchairs. After specifying this relation, all the positive examples are contained, however, one negative example (transport t2) is also contained.
(5) Therefore, the algorithm tries to add another relation, i.e. the duringPeriod.visitorHours relation. Now all the positive examples are contained and none of the negative examples.

Eventually the algorithm generates a new class:
$LateTransport \equiv PatientTransport \wedge \exists hasTransportType.\{wheelchair\}$
$\wedge \exists duringPeriod.\{visitorHours\}$.

[d]http://dl-learner.org/

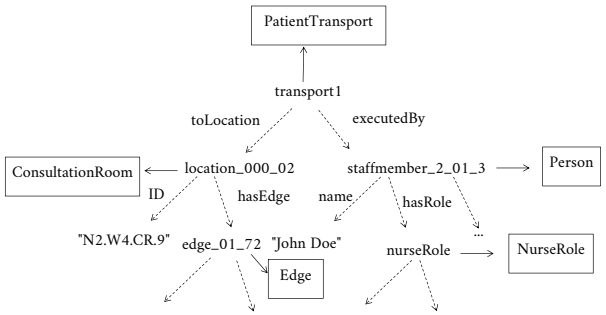

Figure 4. Example of a fragment of a transports described by the ontology schema.

In a realistic dataset, there are many reasons that might cause transport delays. This means that multiple rules need to be identified. In previous work, we coupled the ILP technique with an ontological clustering technique to split the dataset into some more manageable clusters, such that the algorithm can easier find the various delays [30].

3.3. *Combining ARM & ILP for optimal identification of causes of delays*

We have combined the two approaches such that we can benefit from the scalable statistical analysis from ARM and the correctness of ILP. Our technique is based on a statical evaluated generate and test method. Thus the statistical evaluation from ARM combined with the generate and test methodology from CELOE. ARM generates rules that are applicable to the whole dataset, however, since we are only interested in rules detailing the lateness of transports, many rules need to be filtered. Furthermore, since we have positive and negative examples, more rules need to be filtered that occur both in the positive and negative examples. Thus, there are many unnecessary computations as many rules need to be filtered to make the technique applicable. CELOE has the advantage of testing each addition it generates but requires for each addition a call to the reasoner, that does not scale very well, to compute the coverage.

In this approach, we directly compare the support, see Equation 1, of each item in the rule in both the positive and negative examples. We take a Breadth-First Search (BFS) through the graph to compose the rules. Each edge we transverse, we test if the information is adding value to the generation of the rule. For each edge, we take four steps which are visualized in Figure 5 that executes the algorithm on

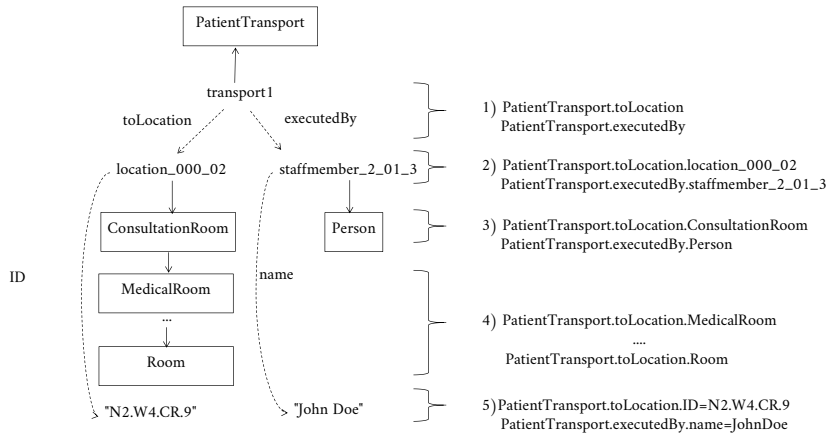

Figure 5. The algorithm traverses the graph in BFS mode and first adds the relations as items, then the individual names, followed by the types and super types and finally the data properties.

an example detailed in Figure 4. These four steps are:

(1) We add the edge (relations) to the path and use it as a feature.
(2) We add the URIs of the individuals to the path and use it as a features.
(3) Instead of the URI we add the type of the individual to the path and use it as a feature.
(4) Instead of the type we generate for each super type (defined in the ontology hierarchy) of the type a new feature.
(5) Instead of the super types, we add the data properties with their values to the path and use them as features.

We test which one of the four steps is the best feature candidate, both in terms of support and interpretability. The interpretability hierarchy is configurable, standard the most specific type has priority over the concepts in the type hierarchy, then the data properties, the relations and lastly the individual names. Note that this is only considered if multiple of these produce the same results. Once all the potential candidates have been generated we test which conjunctions enable a significantly higher drop in the support of the negative dataset, compared to the positive set.

Example 5. (Conjunction example) Let us consider the transports from Example 3 where t3 and t5 were late. We will calculate the conjunction between the transports with wheelchair transport type and the transports scheduled during visiting hours. Table 3 shows the transactions that allow us to calculate the support of the conjunction. Table 4 shows the support calculation for each of the positive (t3, t5) and negative (t1, t2, t4) examples. The table shows the support for the transports scheduled with the transport type wheelchair, i.e. a support of 2/5 for

Table 3. Example of conjunction.

t	hasTransportType .Wheelchair	hasPeriod .visitingHours	isLate
1	0	1	0
2	1	0	0
3	1	1	1
4	0	0	0
5	1	1	1

Table 4. Support calculation of the conjunction between transports with the transport type wheelchair and transports scheduled during visiting hours.

	Positive	Negative
support(hasTransportType.Wheelchair)	$\frac{\lvert\{t3,t5\}\rvert}{5} = 2/5$	$\frac{\lvert\{t2\}\rvert}{5} = 1/5$
support(hasPeriod.visitingHours)	$\frac{\lvert\{t3,t5\}\rvert}{5} = 2/5$	$\frac{\lvert\{t1\}\rvert}{5} = 1/5$
support(hasTransportType.Wheelchair $\wedge$ hasPeriod.visitingHours)	$\frac{\lvert\{t3,t5\}\rvert}{5} = 2/5$	$\frac{\lvert\{\}\rvert}{5} = 0/5$

the positive set and 1/5 for the negative. The support for the transports scheduled during visiting hours are 2/5 for the positive and 1/5 for the negative. The table also shows that the support drops in the negative set when calculating the conjunction between the rules, while the support in the positive set remains. This means that this conjunction should be considered as a candidate result or temporarily result, e.g. when additional conjunctions are required to find significant difference in support.

3.4. *Related approaches*

Nebot et al. [31] proposed an ARM technique for ontological data. The concept and the features to learn about are defined through a SPARQL query , i.e. a query language for ontological data, and translated to transactions for the ARM algorithm. Our ARM approach build upon their proposed technique in the sense that no explicit indication of the learning features is necessary and various optimizations are proposed to prune the learned rules.

AMIE[32] and its successor AMIE+[33] provide an algorithm for mining rules in large knowledge bases where there are no negative examples. The technique is more scalable than standard ILP techniques, however, it does not enable reasoning during the learning phase. This means that the algorithm cannot make a rule more specific or more generic to match the examples. Furthermore, it is not optimized to cope with positive and negative examples.

4. Interpretable Results

In Section 3, we have shown how we can learn rules that describe why transports are late. However, as can be seen in Example 4 and 5, these rules are not very interpretable or intuitive for non-technical end-users, e.g. management of a hospital.

4.1. *Verbalizing Rules*

To make the learned rules more interpretable, we can convert the rules to human readable sentences. Since we make use of an ontological model, the model describing the rules is fixed. Therefore, we can make use of a verbalizer such as NaturalOWL [34] that can convert ontology concepts to readable sentences. By defining how the classes and properties in the ontology should be verbalized, NaturalOWL can generate fluent human-readable text. This makes it easier for the management to interpret the learned rules. In practice, to enable this, the ontology needs to be annotated and indicated which concepts should be interpreted as adjectives, nouns or verbs and how they construct readable sentences when combined. However, this typically needs to be done only once, since the ontology itself does not change (often).

Example 6. (Verbalization) The concepts in the ontology are annotated with verbalization information and various sentence plans are defined to be able to construct human-readable sentences. The class assertion definitions can be verbalized through the following sentence plan:

$[OWNER_{OWNER}][is_{verb}][a\ kind_{string}][of_{prop}][FILLER_{FILLER}]$

This means that the assertion $PatientTransport(trans_1)$ will be verbalized as *"$trans_1$ is a kind of PatientTransport"* . Where $OWNER$ is the individual assigned to the class, here $trans_1$ and $FILLER$ is the class itself, here the class $PatientTransport$.

The relation transportMode can be verbalized through the sentence plan:

$$[OWNER_{OWNER}][has_{verb}][a_{string}][transportmode_{noun}][of\ the\ type_{string}][FILLER_{FILLER}]$$

This means that the relation $hasTransportMode(trans_1, bed)$ will be verbalized as *"$trans_1$ has a transport mode of the type bed"*. Here $OWNER$ is the individual from where the relation starts, here $trans_1$ and $FILLER$ is the individual that is linked by the relation, here the individual *bed*.

4.2. *Dashboard*

We are now able to learn the causes of the delays and verbalize them such that management can interpret them. However, they are still not usable as distinct tools. Therefore, we provide a visualization through a dashboard that enables insights into the transports and allows to activate the learning-verbalization-chain. Figure 6 visualizes the dashboard. It provides some graphical analytics such as:

- An overview of the number of tasks that were on-time versus the ones that were late.
- An overview of all the transport modes and how they influence the arrival times.
- An overview of both the location the transport came from/is going to and how they relate to the arrival times.

The exact metric for what is shown in the overview can be easily configured through the use of a query. For example, in Figure 6 two queries are defined, one for selecting the transport types and one for selecting the destination location of the transports. When a more in-depth analysis is necessary, the learning module can be activated from the dashboard to inspect the causes of transports delays over a specific time range. Figure 3 shows an example of how the verbalized rules in the dashboard are shown. Each of the learned rules is translated into readable sentences and can, after inspection of management, be incorporated into the system to avoid future delays.

5. Evaluation

This section elaborates on the evaluation of the *Self-Learning module* and more specifically on its learning capabilities. We make a comparison between the learning capabilities of the different algorithms discussed in Section 3.

5.1. *Dataset*

As the IoT system described in Section 2.2 can only be deployed in a real hospital setting after thorough evaluation and proof that the system functions correctly, we do not have enough real-time data to be used in the evaluation. We note that the IoT platform has been evaluated in a controlled hospital environment, to prove its feasibility. To enable the learning phase, data of many transports is necessary and since the platform could only be deployed in a smaller controlled environment, capturing enough data to enable the learning phase was not possible. However, the hospitals currently have a static scheduling system, describing the various dispatched tasks. Even though the static does not contain all the context as it would in the IoT case, it is still a good starting point to show the feasibility of the learning component. As this static data is maintained in a relational database, we extract the data, map the data to the semantic model through the use of RML [24], which allows non-semantic data to be mapped to a semantic model, such that it can be used by the learning algorithms. We note that in the IoT deployment, when more data is available, more accurate rules can be learned.

We received static transport data from two Flemish hospitals describing over three months worth of patient transports details, based on 40 variables. On average, around 10000 transports are scheduled each month and about 26% of these transports are late. For the first two evaluations, i.e. Section 5.2 and Section 5.3, we adapted the hospital dataset so we can manipulate its distributions in order to illustrate the underlying mechanics of the learning algorithms. We selected one

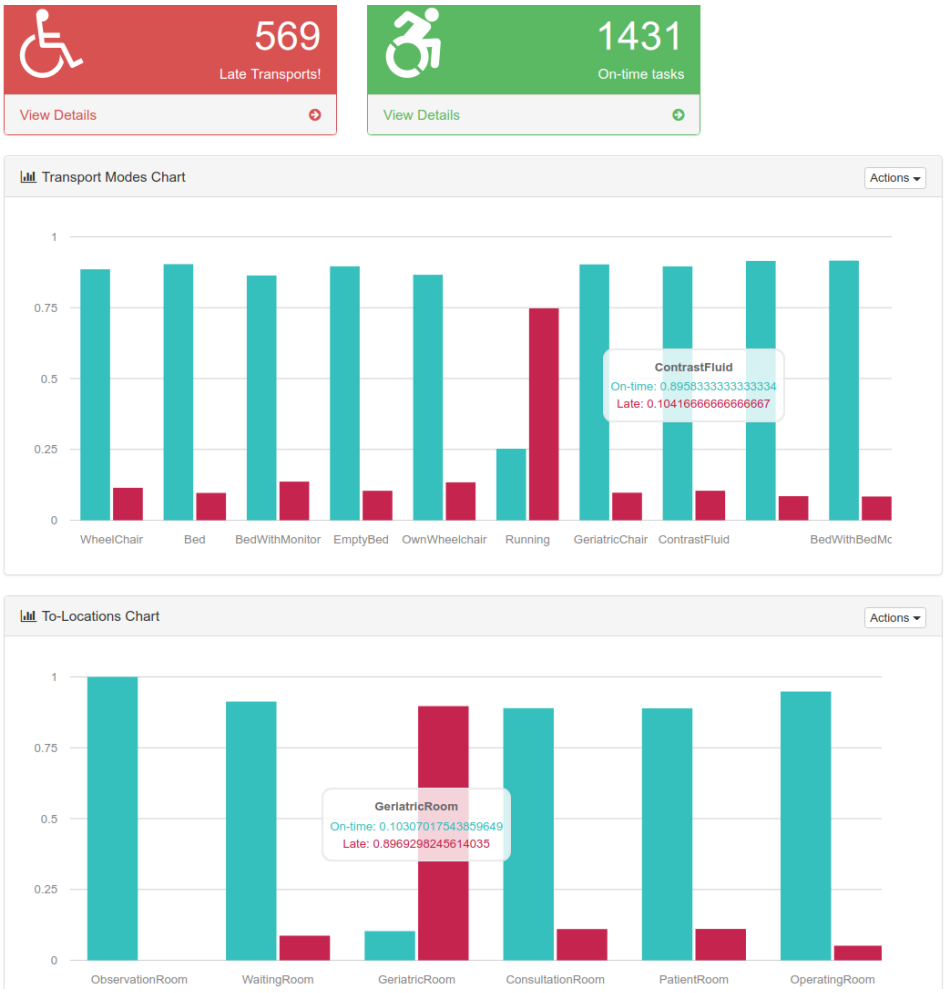

Figure 6. Overview of the AORTA Self-Learning Dashboard.

month worth of data and removed all transports that were late and added on time transports from the next months to obtain a total of 10000 transports. To evaluate the learning capabilities of the algorithm, we injected several causes of delays in the dataset, allowing to evaluate accurately if the algorithm is capable of detecting these causes. Among these causes are 1) transports from a patient room to a consultation room where the patient had to walk, 2) transports on Friday in the evening and 3) transports in the afternoon towards the operating room. For the last evaluation, Section 5.4, we used the received dataset to explain why certain transports were late.

The dataset itself contains 474 unique locations, each mapped to the hospital layout and specific function of the location, 8 transport modes (e.g. bed, wheelchair,

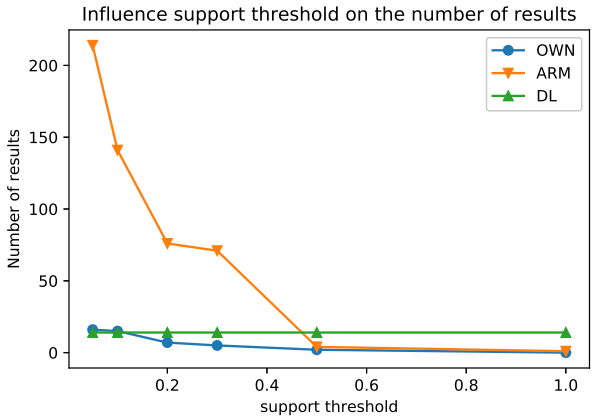

Figure 7. Influence of the support parameter on the number of rules.

walking, etc.), the period of the day (i.e. morning, afternoon, evening), the exact time, etc. Since we started from the received dataset, the data distributions are realistic.

5.2. *Minimum number of late transports*

The cause of the late transports can only be detected if a sufficient number of these transports are contained in the positive examples. The question remains, how frequent should they occur to be detected? This is defined by the *support parameter* that defines the minimum frequency a candidate item should occur before it can be considered. This filters out very low occurring items and reduces the number of generated rules. Thus, the lower the *support parameter* the higher the chances it is detected by the algorithm. However, since lower support parameters imply more generated rules, more noise will be produced and complicates the interpretability by management. Figure 7 shows the influence of the *support parameter* on the dataset, OWN indicates our combined algorithm, ARM our ARM approach and DL the CELOE approach provided by DL-learner. It is worthing noting that the CELOE does not have a support parameter and needs to be configured in the function of the number of results it may generate and the amount of time it may execute. We fixed the number of results to the same number as the expected number of results, i.e. the number of rules contained in the data and the correct execution time was obtained by iterating over various execution times until the causes were detected by the algorithm. The figure shows that for ARM and OWN, the lower the parameter, the more rules are generated. This makes sense as none of the injected rules are contained in more than 30% of the late transports. Therefore, the causes are only starting to be detected as the support threshold decreases below 0.3. It is clear that the ARM approach generates many more rules, but more rules do not necessarily

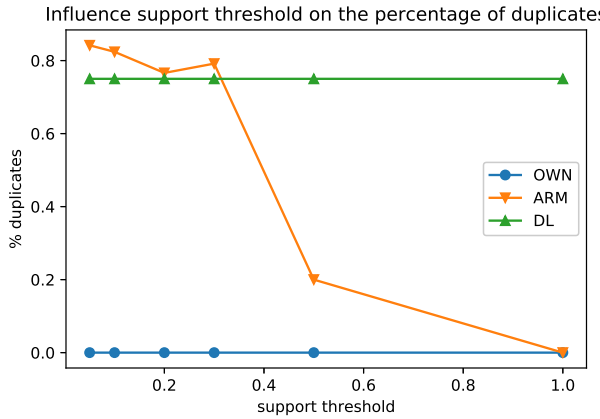

Figure 8. Influence of the support parameter on the duplicates.

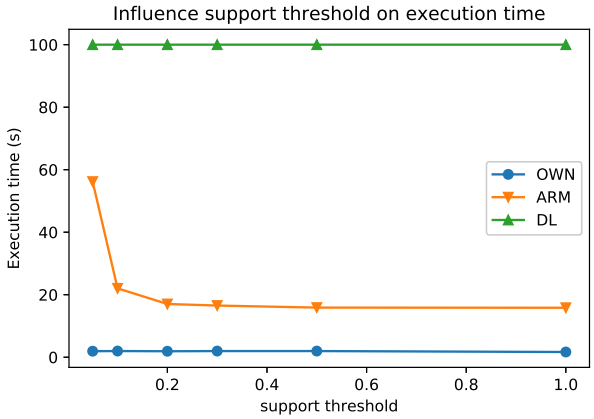

Figure 9. Influence of the support parameter on the execution time.

mean better results.

Figure 8 shows the percentage of duplicates contained in the results. These duplicates are unique rules that have the same meaning, e.g., each transport mode has a certain ID, resulting in two rules, one stating that the transport mode 'Running' causes transports to be late or one stating that the transport mode with id '131' causes transports to be late. These are different rules but have the same meaning. As explaining in Section 3.3 our own algorithm is tailored to only generate the most meaningful features and rules. This is reflected in the results of Figure 8 as the percentage of duplicates is low for our algorithm.

In Figure 9 the execution time of each algorithm is plotted. The ARM and

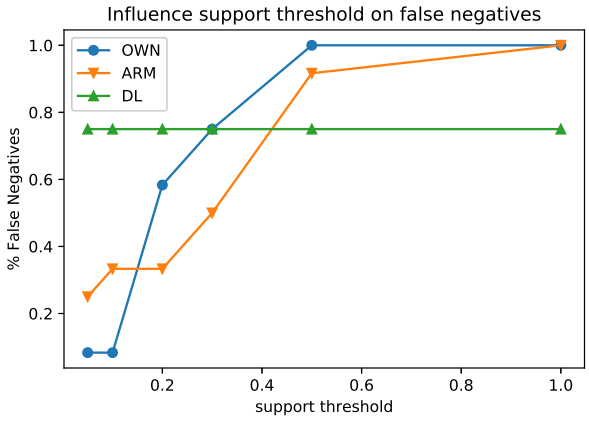

Figure 10. Influence of the support parameter on the false negatives.

OWN algorithms are faster than the CELOE algorithm as they take a more scalable statistical approach. For low support thresholds the execution time increases, this is because more features are selected which results in more combinations that need to be tested to detect the rules. Our own algorithm is less prone to this, as it uses a generate and test approach and only considers combinations of features if they are improving the accuracy. The ARM approach generates more combinations and is thus slower.

Figure 10 shows the percentage of false negatives, i.e. the percentage of rules that should have been detected but were not. It is clear that as the support parameter decreases, more rules are found for the ARM and OWN approach and the percentage of false negatives drops. As the support parameter has no influence on the DL approach, it remains constant. However, it fails to find most rules even after long execution times. The false negatives decrease faster in the ARM approach, however, only our OWN approach finds all rules. The reason for this slower decrease is because our OWN approach is very selective in which rules to generate.

5.3. *Noise in the dataset*

A second important aspect is the ability to cope with is noise. Many transports are late for no reason and are thus adding noise to the positive examples as there is not a straightforward explanation.

Figure 11 shows the number of found rules in function of the percentage of noise in the dataset for a support threshold of 0.2, 0.1 and 0.05 for the ARM and OWN approach. We did not further include the DL approach, as it has troubles to deal with noisy data. We artificially added additional late transports to the dataset, which were selected from the set of transports that were on time and thus do not contain any real causes for their delays and can be considered as noise.

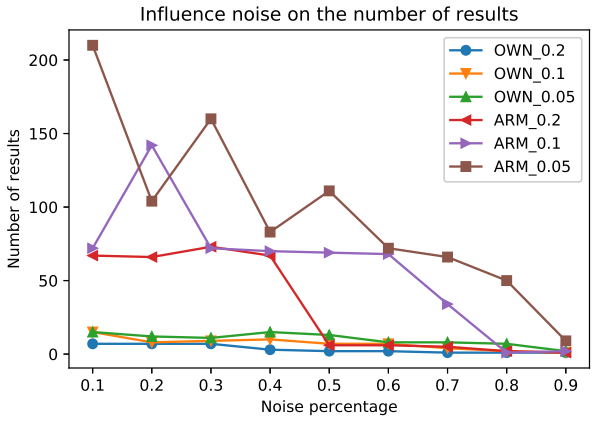

Figure 11. Influence of the noise on the number of results.

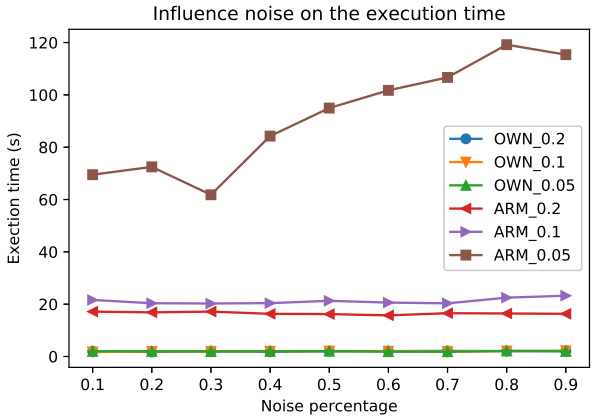

Figure 12. Influence of the noise has the execution time.

The figure shows that as the noise increases, the number of found rules decreases. This is because the percentage of the late transports in the dataset that should be detected decreases and random causes increase. When further decreasing the support parameter, the rules can be detected again. However, when decreasing too much, random rules will start populating the results. Table 5 shows this trend for a dataset containing 80% noise. The table shows for both OWN and ARM that as the support parameter decreases, more rules are generated. However, as the support parameter decrease, more random rules are considered as well. This can be expected, as with a support parameter of 0.001, a rule only needs to occur in 0.01% of the examples. For a dataset of one month, this means that only two occurrences should

Table 5. The influence of the decreasing support parameters for a dataset with high number of noise (80%). Both the number of generated results (#res) are compared to the number of correct results (#correct).

Support	OWN		ARM	
	#res	#correct	#res	#correct
0.2	1	1	2	1
0.1	2	2	2	1
0.05	7	7	50	16 (56% duplicates)
0.01	16	14	-	-
0.005	35	14	-	-
0.001	198	14	-	-

be present in the data. This leads to the production of many random results. The ARM approach is not able to produce rules for a support parameter of 0.01 and lower. This is due to the fact that too many conjunctions need to be tested and therefore the execution time explodes.

5.4. *Evaluating the unmodified dataset*

We also executed the algorithms on the datasets received from the hospitals, i.e. the dataset described in Section 5.1 without the artificially injected late transports. Since we did not inject the transports, there is no objective metric to evaluate the correctness of the learned rules. Therefore, we provide a discussion of our findings. Figure 13 depicts the dashboard of the received dataset and shows some of the dataset characteristics.

While executing the learning algorithms, we found that some of the learned causes are rather trivial, such as the fact that if the task started on time or the priority of the task is low, then transports are often late. Other causes are however less trivial. In one of the hospitals, transports in the morning that need a wheelchair are often late. Transports on Friday or Saturday or to the consultation room share the same fate. However, transports that need a bed with a bed mover or transports planned on Wednesday/Thursday afternoon are mostly on time. However, the algorithm also reveals more sensitive data, such as certain persons or teams that cause more delays than other. Figure 14 provides an example of the verbalization of some of the learned rules on the static dataset.

In Section 6 we discuss how to deal with sensitive data and the advantages of having a system that provides explanations in these situations.

5.5. *Comparison*

Compared to ARM our technique scales very well as it does not need to generate rules regarding the whole dataset that later on needs to be filtered out. Furthermore, since we pick the features very carefully, the number of elements that are used

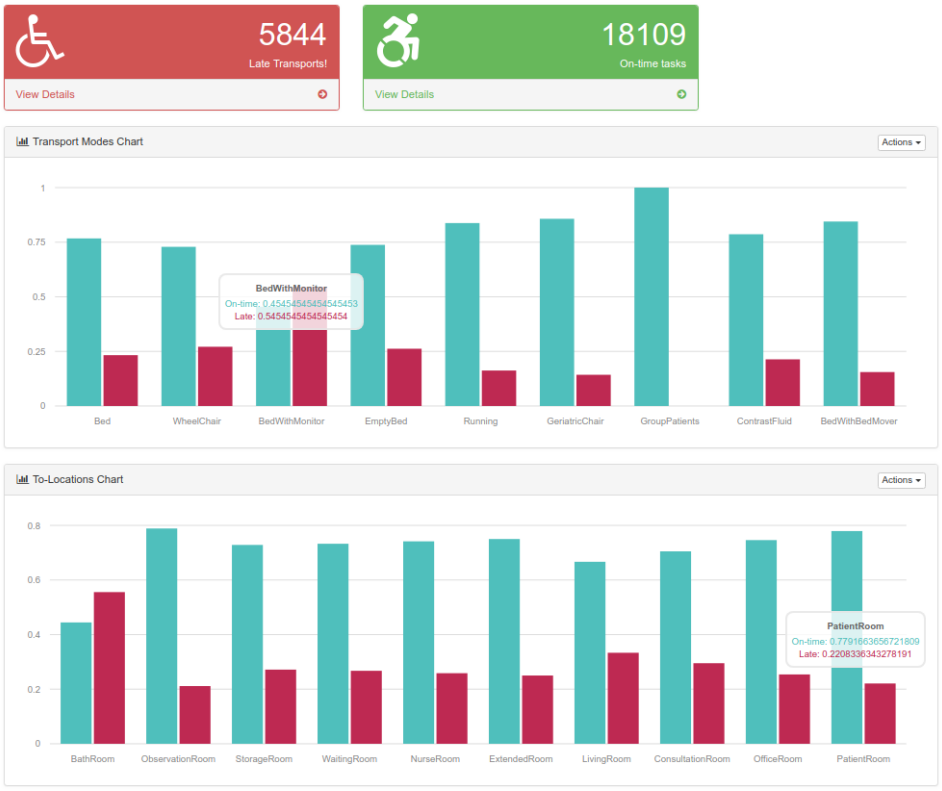

Figure 13. The dashboard visualizing some of the characteristics of the dataset received from the hospitals.

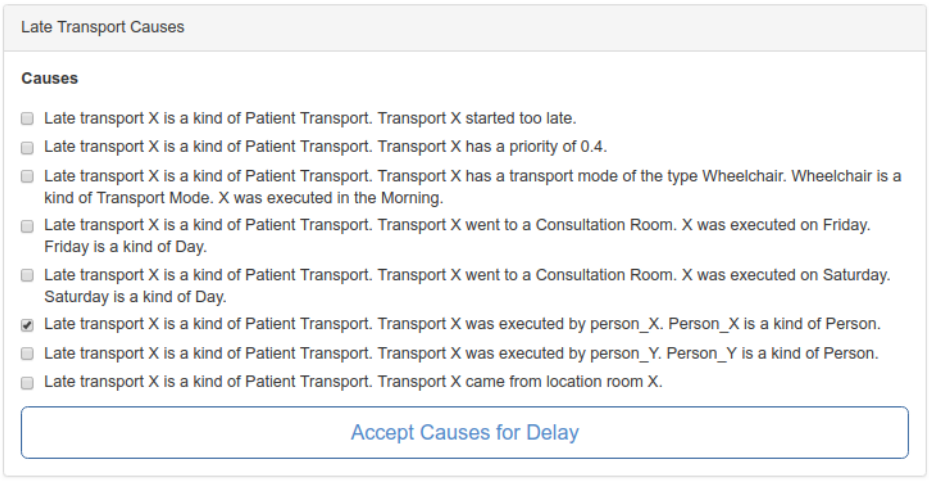

Figure 14. Example of the verbalized rules for the hospital datasets.

to calculate the conjunctions is limited and thus faster. We also have a different algorithm for creating conjunctions of rules. ARM has multiple algorithms to achieve this but essentially checks if the support of the conjunction is above a certain threshold. We take a different approach where we only consider conjunctions that enable a greater drop in the support of the negative dataset, compared to the positive dataset. This means that the conjunction occurs more frequently in the positive dataset compared to the negative dataset and it also appears relatively more as a conjunction. Furthermore, because we filter the features early on by selecting only the most interpretable features, fewer conjunctions need to be calculated and less duplicate results are produced.

Compared to ILP (DL/CELOE) we take a statistical approach to test the coverage of the rule and not a logical one. This is possible by converting the graph to features, by the generation of the paths. This is more scalable than the ILP approach as the coverage is easier to compute. The CELOE algorithm generates possible rules from the ontology concepts and checks if it matches the dataset. We take another approach by starting from the dataset and generalize the found rules by incorporating the knowledge in the ontology. The ILP algorithm has also troubles to find rules in a noisy dataset. The ILP technique is more suited when one specific and possibly complex rule needs to be found. In our case, because multiple causes for delay exist in the dataset and noise can be present, the technique is not ideal. Furthermore, the configuration of our algorithm is easier than to configure the CELOE algorithm, as it requires to indicate the execution time.

It is clear that our approach is the fastest in execution time and also produces the most correct results. By incorporating the filtering techniques during the generation of the features, the number of duplicate rules is minimized. Also, the detection of the correct rules is higher than in the other algorithms. This allows to provide only the essential rules and give a clear overview and insights into the data.

6. Discussion

The proposed system is able to learn rules that identify possible causes of why transports in hospitals are late. By identifying the context in which transports are often late, we can predict which transports will be late in the future and more importantly avoid future delays. Both patients, staff members and hospital management benefit from more accurate scheduling. Currently, patients often have to wait before being picked up before or after a medical intervention, which is often uncomfortable. For staff members, it is stressful to see their tasks pile up as the assigned transports take longer than expected.

Furthermore, by explaining the cause as human-readable sentences, management gets understandable insights into their underlying hospital's mechanics. The fact that these causes can be understood by non-experts, allows management to be involved in the automated process and provides them with the final judgment.

The use of the semantic model allows to easily extend the platform and integrate

additional data that could offer more accurate insights. For example, a new sensor could be added that captures the exact route a transport takes. This would allow to detect various bottlenecks in the transport routes, such as taking a specific elevator that is slow during certain times of the day (maybe visitors tend to use the elevator as well) or routes that pass a certain corridor in the hospital that is often very busy.

The learning phase can also detect sensitive data, such as certain staff members or teams that are underperforming. These are scenarios where management should open a discussion with their employees to find out the real cause of the problem. By first providing the causes to management for verification, management is provided with the opportunity to have this discussion and staff members are not rewarded for executing their tasks more slowly. In a fully automated system, the system would detect that a certain staff member takes more time and automatically assign more time to the tasks this staff member has to execute. Other insights can also provide opportunities for optimizations, e.g. transport towards certain specific locations that are always late could indicate that there is a structural problem in the department and perhaps a reorganization of the department would be beneficial.

The dashboard provides an easy access to the learning tool. By providing some graphical overviews of the transport distributions between timely and late transports, management can have a quick visual overview. By making the overviews adaptable through queries, the content can be easily adapted. However, the construction of these queries might not be trivial for non-technical persons. Therefore, we provide some basic queries and allow the option to monitor the transports that adhere to the previously selected causes of delays. This allows to quickly validate if the system is now assigning the required time to execute these transports more accurately.

A disadvantage of learning from past transports is that data about past transports need to be available. If management restructures the execution of transports, it takes time to see the influence in the learned causes. One solution to solve this is to only take the transports into account that were conducted from the time the restructuring took place. However, data about the transports is still necessary.

Besides late transports, there might also be cases where the transports are assigned too much time, i.e the transports arrive too early compared to the assigned delivery time. These transports can be identified in the same way we identified late transports. By alternating both identifying the early and late tasks, the system will converge to an optimal setting.

7. Conclusion & Future Work

In this paper, we propose a learning system that can indicate the causes of why certain hospital transports are late. Special precautions are taken to make sure that the learned causes can be explained to management, enabling management to remain in full control of the automated system. We have shown that our platform is capable of learning said causes and verbalize them in interpretable sentences for

further inspection by the hospital management.

In future work, we wish to incorporate additional sensors to allow the detection of more accurate and complex rules. We also wish to further extend the usability of the dashboard. For example, allow management to easily construct the overview queries in a natural and interpretable manner for non-technical users.

Acknowledgment. This research was funded by the iMinds ICON AORTA project, co-funded by VLAIO, iMinds, Xperthis, Televic Healthcare, AZ Maria Middelares and Ziekenhuis Netwerk Antwerpen.

Bibliography

[1] Mladovsky P, Srivastava D, Cylus J, Karanikolos M, Evetovits T, Thomson S, et al. Health policy responses to the financial crisis in Europe edited by Philipa Mladovsky et al. 2012;.

[2] Karanikolos M, Mladovsky P, Cylus J, Thomson S, Basu S, Stuckler D, et al. Financial crisis, austerity, and health in Europe. The Lancet. 2013;381(9874):1323–1331.

[3] Hastreiter S, Buck M, Jehle F, Wrobel H. Benchmarking logistics services in German hospitals: a research status quo. In: Service Systems and Service Management (ICSSSM), 2013 10th International Conference on. IEEE; 2013. p. 803–808.

[4] Landry S, Philippe R. How logistics can service healthcare. In: Supply Chain Forum: An International Journal. vol. 5. Taylor & Francis; 2004. p. 24–30.

[5] Bates DW, Ebell M, Gotlieb E, Zapp J, Mullins H. A proposal for electronic medical records in US primary care. Journal of the American Medical Informatics Association. 2003;10(1):1–10.

[6] Marschollek M. Recent progress in sensor-enhanced health information systems slowly but sustainably. Informatics for Health and Social Care. 2009;34(4):225–230.

[7] Atzori L, Iera A, Morabito G. The internet of things: A survey. Computer networks. 2010;54(15):2787–2805.

[8] Laskowski M, McLeod RD, Friesen MR, Podaima BW, Alfa AS. Models of emergency departments for reducing patient waiting times. PloS one. 2009;4(7):e6127.

[9] Marković N, Milinković S, Tikhonov KS, Schonfeld P. Analyzing passenger train arrival delays with support vector regression. Transportation Research Part C: Emerging Technologies. 2015;56:251–262.

[10] Rebollo JJ, Balakrishnan H. Characterization and prediction of air traffic delays. Transportation research part C: Emerging technologies. 2014;44:231–241.

[11] Xu J, Deng D, Demiryurek U, Shahabi C, Van Der Schaar M. Mining the situation: Spatiotemporal traffic prediction with big data. IEEE Journal of Selected Topics in Signal Processing. 2015;9(4):702–715.

[12] Silva R, Kang SM, Airoldi EM. Predicting traffic volumes and estimating the effects of shocks in massive transportation systems. Proceedings of the National Academy of Sciences. 2015;p. 201412908.

[13] Lecue F, Wu J. Explaining and predicting abnormal expenses at large scale using knowledge graph based reasoning. Web Semantics: Science, Services and Agents on the World Wide Web. 2017;44:89–103.

[14] Lecue F. Applying Machine Reasoning and Learning in Real World Applications. In: Reasoning Web International Summer School. Springer; 2016. p. 241–257.

[15] Fiegl C, Pontow C. Online scheduling of pick-up and delivery tasks in hospitals. Journal of Biomedical Informatics. 2009;42(4):624–632.

[16] Hanne T, Melo T, Nickel S. Bringing robustness to patient flow management through

optimized patient transports in hospitals. Interfaces. 2009;39(3):241–255.
[17] Beaudry A, Laporte G, Melo T, Nickel S. Dynamic transportation of patients in hospitals. OR spectrum. 2010;32(1):77–107.
[18] Kergosien Y, Lente C, Piton D, Billaut JC. A tabu search heuristic for the dynamic transportation of patients between care units. European Journal of Operational Research. 2011;214(2):442–452.
[19] Huh J, Pollack M, Katebi H, Sakallah K, Kirsch N. Incorporating user control in automated interactive scheduling systems. In: Proceedings of the 8th ACM Conference on Designing Interactive Systems. ACM; 2010. p. 306–309.
[20] Gruber T. Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies. 1995;43(5-6):907–928.
[21] STRANG T. A context modeling survey. In: 1st International Workshop on Advanced Context Modelling, Reasoning and Management, 2004; 2004. p. 34–41.
[22] Vancroonenburg W, Esprit E, Smet P, Vanden Berghe G. Optimizing internal logistic flows in hospitals by dynamic pick-up and delivery models. In: Proceedings of the 11th international conference on the practice and theory of automated timetabling; 2016. p. 371–383.
[23] Nenov Y, Piro R, Motik B, Horrocks I, Wu Z, Banerjee J. RDFox: A highly-scalable RDF store. In: International Semantic Web Conference. Springer; 2015. p. 3–20.
[24] Dimou A, Vander Sande M, Colpaert P, Verborgh R, Mannens E, Van de Walle R. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In: LDOW; 2014. .
[25] Agrawal R, Imieliński T, Swami A. Mining association rules between sets of items in large databases. In: Acm sigmod record. vol. 22. ACM; 1993. p. 207–216.
[26] Lavrac N, Dzeroski S. Inductive Logic Programming. In: International Conference on Inductive Logic Programming. Springer; 1994. p. 146–160.
[27] Bonte P, Ongenae F, Hoogstoel E, De Turck F. Mining semantic rules for optimizing transport assignments in hospitals. In: ISWC2016, the 15th International Semantic Web Conference; 2016. p. 1–6.
[28] De Raedt L, Kersting K. Statistical Relational Learning. Encyclopedia of Machine Learning. 2011;p. 916–924.
[29] Lehmann J, Auer S, Bühmann L, Tramp S. Class expression learning for ontology engineering. Web Semantics: Science, Services and Agents on the World Wide Web. 2011;9(1):71–81.
[30] Bonte P, et al. Learning Semantic Rules for Intelligent Transport Scheduling in Hospitals. In: European Semantic Web Conference (ESWC2016); 2016. .
[31] Nebot V, Berlanga R. Finding association rules in semantic web data. Knowledge-Based Systems. 2012;25(1):51–62.
[32] Galárraga LA, Teflioudi C, Hose K, Suchanek F. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In: Proceedings of the 22nd international conference on World Wide Web. ACM; 2013. p. 413–422.
[33] Galárraga L, Teflioudi C, Hose K, Suchanek FM. Fast rule mining in ontological knowledge bases with AMIE + +. The VLDB Journal. 2015;24(6):707–730.
[34] Androutsopoulos I, Lampouras G, Galanis D. Generating natural language descriptions from OWL ontologies: the NaturalOWL system. Journal of Artificial Intelligence Research. 2013;48:671–715.