Semantic-Guided Feature Selection for Industrial Automation Systems

Martin Ringsquandl[1](✉), Steffen Lamparter[2], Sebastian Brandt[2],
Thomas Hubauer[2], and Raffaello Lepratti[3]

[1] Ludwig-Maximilians University, Munich, Germany
martin.ringsquandl.ext@siemens.com
[2] Siemens AG Corporate Technology, Munich, Germany
{steffen.lamparter,sebastian-philipp.brandt,thomas.hubauer}@siemens.com
[3] Siemens AG Digital Factory, Nuremberg, Germany
raffaello.lepratti@siemens.com

Abstract. Modern industrial automation systems incorporate a variety of interconnected sensors and actuators that contribute to the generation of vast amounts of data. Although valuable insights for plant operators and engineers can be gained from such data sets, they often remain undiscovered due to the problem of applying machine learning algorithms in high-dimensional feature spaces. Feature selection is concerned with obtaining subsets of the original data, e.g. by eliminating highly correlated features, in order to speed up processing time and increase model performance with less inclination to overfitting. In terms of high-dimensional data produced by automation systems, lots of dependencies between sensor measurements are already known to domain experts. By providing access to semantic data models for industrial data acquisition systems, we enable the explicit incorporation of such domain knowledge. In contrast to conventional techniques, this semantic feature selection approach can be carried out without looking at the actual data and facilitates an intuitive understanding of the learned models. In this paper we introduce two semantic-guided feature selection approaches for different data scenarios in industrial automation systems. We evaluate both approaches in a manufacturing use case and show competitive or even superior performance compared to conventional techniques.

Keywords: Semantic data models · Feature selection · Automation systems · Machine learning

1 Introduction

Processing and mining of large data sets in modern industrial automation systems is a major challenge due to the vast amount of measurements generated by several types of field devices (e.g. sensors, controllers, actuators). Deployment of machine learning models requires upfront feature selection in order to obtain a reduced feature set, thereby speeding up processing time and preventing overfitting, while still preserving inherent characteristics of the original data. Even

M. Arenas et al. (Eds.): ISWC 2015, Part II, LNCS 9367, pp. 225–240, 2015.
DOI: 10.1007/978-3-319-25010-6_13

in the age of massively distributed data processing, feature selection remains one of the main problems in automation, since it is a highly domain expertise-intensive task [7]. On the other hand, data generated by engineered systems exhibits many structural dependencies that domain experts are well aware of. This holds especially for industrial automation systems that are systematically planned and simulated before going into production. For example, for a given electric motor, it is documented how torque, speed and power measurements relate to each other. Thus, there is no need to compute correlations between them for asserting statistical dependence.

These computations are common in most of today's feature selection techniques, therefore they exhibit some major disadvantages when applied in high-dimensional data as observed in today's automation systems [13]. By accessing huge proportions of the original data, they quickly become computationally expensive, plus they are prone to losing valuable information, especially when transforming the feature space to lower dimensions so that the remaining variables can no longer be intuitively interpreted. Motivated by the commonly faced difficulties of *a*) processing vast amounts of data and *b*) integrating domain knowledge into learning models, the semantic guidance approaches of this paper were developed in order to facilitate what remains the most expertise intensive task – feature selection.

In general, there are two different types of high-dimensional data that need to be considered in separation. The first case is present if we are given a huge number of both features and instances. In this scenario, we argue for an approach, in analogy to the notion of the usage of *OWL 2 QL* for ontology-based data access (OBDA), that makes it possible to perform feature selection on $\mathcal{T}$Box level rather than instance (data) level [10]. The other case is given when there are fewer instances than features, also called *sparse* data (e.g. rare events such as device failures). For this type of data, embedded feature selection techniques like Lasso have shown to be very effective [5]. Therefore, we also introduce an embedded feature selection approach that leverages from engineering background knowledge in semantic data models. The subsequent sections will describe both approaches and their application in more detail.

2 Application: Manufacturing Use Case

As an application scenario, consider multiple assembly lines that are part of a car door production facility. The core of each assembly line is responsible for welding the window frame and inner door panel, which happens in a semi-automated fashion, as the actual welding is done by a human worker. The overall system consists of an automated frame loading station that is responsible for putting window frames on a conveyor kit. These kits are then routed through the core assembly process by an electrically-operated conveyor. After the products have been assembled, they must go through a final quality control that verifies integrity of certain product characteristics. If quality conforms to specification, the product is sent to an outgoing packaging station.

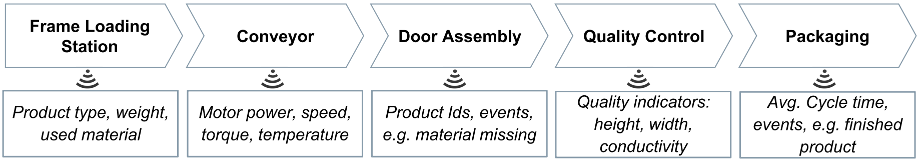

Frame Loading Station
Conveyor
Door Assembly
Quality Control
Packaging
Product type, weight, used material
Motor power, speed, torque, temperature
Product Ids, events, e.g. material missing
Quality indicators: height, width, conductivity
Avg. Cycle time, events, e.g. finished product

Fig. 1. Door assembly process and associated measurements

Plant operators are responsible for planning and scheduling of operations based on incoming orders. For this task, the operators have to assess uncertainties such as varying cycle times of each produced piece. Since production processes are of stochastic nature, it is non-trivial to get a solid estimate of the time when a certain product will be finished. In this case, decision support can be given by training advanced regression models that help to provide more robust and up-to-date time estimates. During production, all of the devices (e.g. light-barrier sensors, power meters, etc.) generate task-specific measurement data as shown in Figure 1. Today, data collection is agnostic of any machine learning tasks, as it is merely concerned with high-throughput historic data storage. As far as data analytics is concerned, any (sub-)set of the measurements taken could possibly be relevant for the time estimate regression task. A kind of brute-force approach would be to use all present data and try to train, for example, an ordinary least squares regression (OLS) model. However, this approach has some major shortcomings, since the OLS will include irrelevant and redundant information for fitting its coefficients and is very likely to overfit to particular patterns in the training data, therefore it is not going to generalize well in a live production scenario.

Consider the two regression models on the left-hand side of Figure 2 that try to predict cycle times. In this small example, employing five different predictor variables causes the model to overfit, while after p-value-based feature selection we obtain a more smoothed fit using only `Conveyor1Time`. On the right-hand side it can be seen how constraining the five predictors by a regularization penalty reduces their coefficients until they are effectively set to zero. For example, `LoadingProductWeight` quickly gets eliminated due to its small effect on the regression task. Since it is known that product weight is part of the overall weight feature, it could have been removed beforehand without any computation. Throughout this paper, we will relate to this learning problem of forecasting product-specific cycle times in an automated manufacturing system given a huge number of different sensor measurements as a running example.

3 Preliminaries

In this section, we first introduce the notion of semantic data models in the manufacturing domain in 3.1 and how their graph representation is used to measure structural similarity of feature entities. This is followed by a description of the general problem of embedded feature selection in linear models in 3.2.

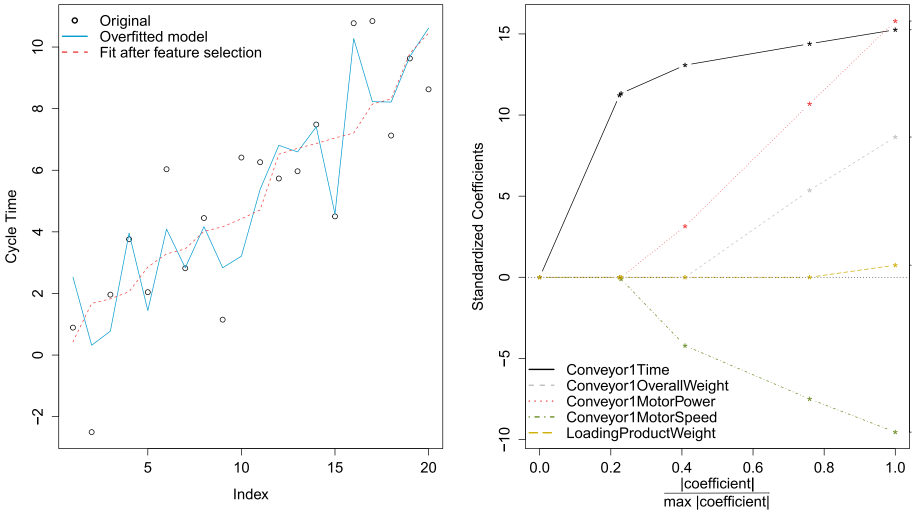


Fig. 2. Visualization of multiple linear regression model. Left: Overfitted model using all predictor variables vs. model after feature selection. Right: Coefficient values under decreasing regularization penalty

3.1 Semantic Representation of Manufacturing Data

Instead of having yet another information model language, we argue for the usage of the well-established Semantic Web standards to create, link, and share information models of automation systems in the manufacturing domain. For the purposes of feature selection, we augmented and tailored the Semantic Sensor Network (SSN) ontology[1] to meet the requirements of describing features in automation data, as can be seen in Figure 3. Here, the `Feature` concept is modeled as subclass of `ssn` : `InformationEntity`. Resorting to SSN is also beneficial for manufacturing systems, since devices and processes can naturally be integrated into its schema. Further details of this *feature ontology* are given in section 4.1. The graph representation of RDF-based[2] ontologies is a suitable property that we want to exploit for the description of dependencies between machine learning features. In formal terms, an RDF graph can be defined as a multi-graph.

Definition 1 (RDF Graph). *An RDF graph is a multi-graph* $G = \langle V, E\rangle$ *where each edge* $e_i \in E$ *is defined as triple (s,p,o):* $s, o \in V$ *and p is the edge's label.*

This formal definition allows us to specify the degree of similarity between feature entities in the graph by means of common structural patterns. Graph kernel

[1] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[2] http://www.w3.org/RDF/

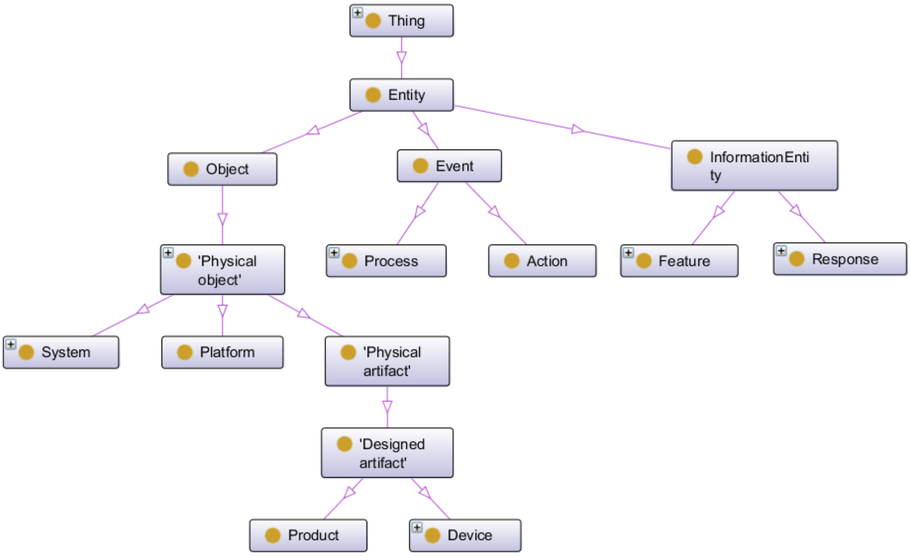


Fig. 3. Taxonomy of manufacturing feature ontology

functions have been shown to work well for capturing such patterns. The emerging field of machine learning in Linked Data has brought up a number of graph kernel functions particularly designed for RDF graph data.

Definition 2 (RDF Graph Kernel). *A graph kernel function is any function* $\kappa : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$ *s.t. for all* $G_i, G_j \in \mathcal{G}$ *satisfies* $\kappa(G_i, G_j) = \langle \phi(G_i), \phi(G_j) \rangle$ *is a valid kernel, where* $\mathcal{G}$ *is the space of RDF graphs and* ϕ *is a mapping to some inner product space.*

Given n entities, the graph kernel can be denoted as kernel matrix:

$$\kappa = \begin{bmatrix} \kappa(G_1, G_1) & \kappa(G_1, G_2) & \dots & \kappa(G_1, G_n) \\ \kappa(G_2, G_1) & \kappa(G_2, G_2) & \dots & \kappa(G_2, G_n) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(G_n, G_1) & \kappa(G_n, G_2) & \dots & \kappa(G_n, G_n) \end{bmatrix}$$

In order to get pairwise similarities between all feature entities in our feature ontology, we can resort to one of the state-of-the-art graph kernels [8]. The idea of graph kernels is that every entity can be represented as the graph that is spanned by its adjacent entities up to a certain depth d. Then, similarity between two graphs is given by some metric, e.g. size of the intersection graph or pairwise isomorphisms like in the popular family of Weisfeiler-Lehman graph kernels [3]. In Figure 4 a simplified example of the graph spanned by feature `Conveyor1Speed` is shown at different levels of depth d. Data properties of entities like RDF literals (formatted in *italic* style) are usually considered to belong to their respective entity and therefore they do not span a new depth level. Clearly, quality of similarity calculations depends on the amount of knowledge put into the ontology creation process. Nevertheless, we expect that already a small number of annotations can support feature selection.

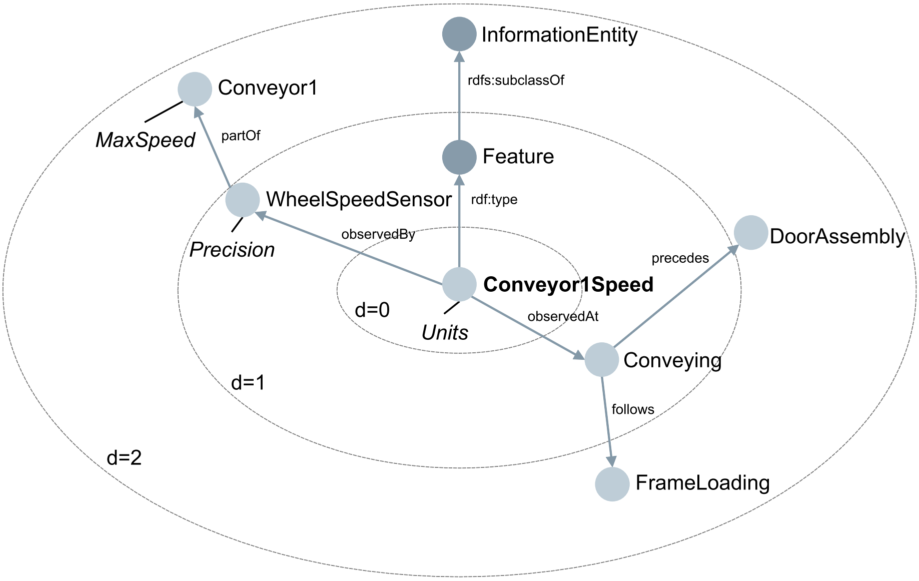


Fig. 4. Neighborhood graph of feature `Conveyor1Speed` at different depth values

Before describing how to exploit this notion of similarity between features in the training procedure of the manufacturing machine learning models, a general introduction to learning linear models is given.

3.2 Linear Model Embedded Feature Selection

Linear models are still one of the most popular machine learning models, especially in domains, where the emphasis lies on insights gained from looking at the model's coefficients. For example, the coefficients of the cycle time estimator regression can be interpreted for decision support in order to take action and reduce their influence on the overall cycle time. In case of sparse data, embedded feature selection techniques have shown to be very effective compared to other conventional feature selection. In the subsequent, we will introduce some standard formal notation of generalized linear models and their sparsity-inducing feature selection ability.

Given a training set $\{x_i, y_i\}_{i=1}^n$ where $x_i \in \mathcal{R}^p$ is a p-dimensional feature vector and $y_i \in \mathcal{R}$ is the response, i.e. for regression or classification. We consider learning a linear model $h : R^p \rightarrow R$ with $h(\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x}$, where $\boldsymbol{w}$ is a parameter vector. The general form of the regularized optimization problem is:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}}\, l(y, h(\boldsymbol{w})) + \lambda \varOmega(\boldsymbol{w}) \tag{1}$$

Here, $l(\cdot)$ denotes the loss function and $\varOmega(\cdot)$ is the regularization term, also called penalty. The value of λ controls how much weight is given to the penalty, which is used to prevent overfitting of large parameter values. Setting $l(\cdot)$ to the square loss and $\varOmega(\cdot)$ to the ℓ_1-regularization results in the standard Lasso model:

$$\hat{w}_{Lasso} = \underset{\boldsymbol{w}}{\operatorname{argmin}}(y - h(\boldsymbol{w}))^2 + \lambda\|\boldsymbol{w}\|_1 \tag{2}$$

The ℓ_1-norm can be used for embedded feature selection by increasing the amount of shrinkage (λ) in the Lasso model, which effectively sets non-influencing components of $\boldsymbol{w}$ to zero.

Due to their embedded feature selection ability, Lasso models have gained increasing attention for learning in sparse data sets, where the number of features is high, but many of them are irrelevant to the learning task [12]. Furthermore, in some applications, we want to include prior domain knowledge about relationships between features, for example if we know that motor speed and torque are depending on each other, they should also have similar influence (i.e. parameter weight) on the response variable. When features are represented in a graph structure, this quality is often called the smoothness property. The notion is as follows: If we specify relationships between features as *undirected* graph $G = \langle V, E\rangle$, the graph Lasso (Glasso) can be defined as

$$\begin{aligned}\hat{w}_{GLasso} &= \underset{\boldsymbol{w}}{\operatorname{argmin}}\, l(y, h(\boldsymbol{w})) + \lambda(\alpha\|\boldsymbol{w}\|_1 + (1-\alpha)\sum_{i,j\in E}(w_i - w_j)^2) \\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}}\, l(y, h(\boldsymbol{w})) + \lambda(\alpha\|\boldsymbol{w}\|_1 + (1-\alpha)\boldsymbol{w}^T \boldsymbol{L}\boldsymbol{w}),\end{aligned} \tag{3}$$

where the second regularization term encourages connected features in the graph to have similar weights (smoothness). It can be preferably weighted against ℓ_1 by decreasing the α parameter. A more convenient formulation of the sum over squared weight differences is given by $\boldsymbol{w}^T\boldsymbol{L}\boldsymbol{w}$, where L is the Laplacian matrix of the graph.

4 Approach

This section presents the main technical discussion of the developed semantic-guided feature selection approach. First, an introduction to concepts and axioms in the use case feature ontology is given, followed by a description of the semantic feature selection procedure. Ultimately, we present a custom linear model designed for embedded feature selection in RDF graphs.

4.1 Feature Ontology

There is a wide variety of modeling standards for manufacturing data that are used to facilitate interoperability of different systems, for example the exchange of material information between warehouse management and manufacturing execution systems. Recent developments of the OPC UA[3] standard are concerned with a unified information model of field device descriptions, PLC programs, interfaces to enterprise levels such as ERP systems, and many more. Although these legacy data models describe several facets (e.g. device topologies, sensor measurements) of manufacturing systems, they are almost solely used for data exchange without taking advantage of their contained semantics.

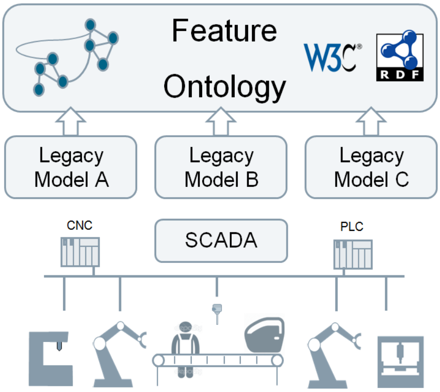


Fig. 5. Feature ontology on top of automation system legacy models

Instead of having another custom information model, our approach makes use of Semantic Web technologies that integrate existing semantics of legacy models into a unified ontology as shown in Figure 5. On top of the automation system and its supervisory control and data acquisition (SCADA), a feature ontology is deployed that represents domain concepts and relations between devices, events and information entities, such as taken sensor measurements. In this context feature means any piece of information that could be used as input to a learning algorithm. From an ontology engineering perspective, this is rather an ad-hoc modeling approach without strong axiomatization.

The result of what we call *Semantic Feature Selection*, i.e. inference about feature dependencies on a semantic level (rather than data level), is represented as RDF graph that contains a task-specific, reduced feature set that is tailored for consumption by machine learning models. These models are then able to perform preprocessing, training, and evaluation on the reduced feature set. One of the goals in development of the feature ontology was to keep the complexity of reasoning as small as possible so that modeling can be done with one of the *OWL 2* profiles that allow scalable reasoning as the number of features in

[3] https://opcfoundation.org/about/opc-technologies/opc-ua/

the automation system grows. As discussed in section 3.1, the feature ontology references some concepts defined within the SSN ontology. In addition to that, we introduce some further relations concerning the connection between processes, devices and measurements in the manufacturing domain. Most importantly, we allow generic relations `dependsOn` and `independentOf` between features that subsume specific relations in existing engineering models, in case no further information is given.

Table 1. Main relations of the feature ontology

Relation	Description
`derivedFrom` $\sqsubseteq$ `directlyDependsOn`	*Connects an event or measurement that is derived from an original source, e.g. threshold overshoot events like 'temperature too high'*
`follows` $\equiv$ `precedes`$^{-1}$	*Processes or Events that happen in a time-dependent order, e.g. packaging follows the assembly process*
`partOf`	*Partonomy describing device topologies, e.g. temperature sensor is part of a motor*
`directlyDependsOn` $\sqsubseteq$ `dependsOn`	*A measurement is directly influencing another without further information, e.g. cycle time directly depends on failure events*
`physicalRelation` $\sqsubseteq$ `directlyDependsOn`	*Measurements are connected by inherent physical laws, e.g. current and voltage*
`apartFrom`	*Events that happen at different locations, e.g. two assembly lines operating at separated shop floors*
`independentOf`	*A measurement is known to have no (direct) influence on the other, e.g. product ID does not influence conveyor motor temperature*
`observedBy`	*Measurement is sampled by a specific sensing device*
`observedAt`	*A measurement sampled during a specific process*

Table 1 gives an overview of important relations that bear semantics for feature selection purposes. The independence statement is of statistic nature, therefore it is also a symmetric relation based on the fundamental probability theorem $P(X|Y) = P(X) \Leftrightarrow P(Y|X) = P(Y)$.

The $\mathcal{R}$Box of the feature ontology further specifies some axioms that propagate dependencies through the feature space, as described in Table 2.

4.2 Feature Selection Procedure

Consider that we are given a learning problem of the form $y = f(x)$, where y is part of the semantic model, such that $\mathcal{O} \models \texttt{Response}(\texttt{y})$, then the feature space of x can be reduced by excluding everything that is known to be independent of y. Furthermore, a good choice of features has to include anything that is known

Table 2. $\mathcal{R}$Box axioms of the feature ontology

Axiom	Description
symmetric(`independentOf`), symmetric(`apartFrom`)	*Independence and separation of processes are defined to be symmetric*
$\texttt{dependsOn} \cdot \texttt{observedBy} \cdot \texttt{observedBy}^{-1} \sqsubseteq \texttt{dependsOn}$	*Dependencies propagate between measurements observed by the same sensing device*
$\texttt{independentOf} \cdot \texttt{dependsOn} \sqsubseteq \texttt{independentOf}$	*If x is independent of y it is also independent of anything that y depends on*
$\texttt{observedAt} \cdot \texttt{apartFrom} \cdot \texttt{observedAt}^{-1} \sqsubseteq \texttt{independentOf}$	*Assert independence of measurements taken at physically separated processes*
transitive(`partOf`), transitive(`follows`)	*Process flows and device topologies are transitive by nature*

to directly influence the behavior of the response variable. Hence, the $\mathcal{T}$Box is accordingly augmented with the following axioms:

$$\mathcal{T} =$$

$$\begin{aligned} \exists\texttt{independentOf.Response} &\sqsubseteq \texttt{ExcludedFeature} \\ \exists\texttt{directlyDependsOn}^{-1}\texttt{.Response} &\sqsubseteq \texttt{MandatoryFeature} \\ \texttt{ExcludedFeature} \sqcap \texttt{MandatoryFeature} &\sqsubseteq \bot \end{aligned}$$

Overlap in excluded and mandatory features results in an inconsistent feature ontology that is most likely due to a feature modeling mistake, since there should not be independence and direct dependence for two information entities at the same time.

Algorithm 1 summarizes this schema-level feature selection procedure. First, the assertion of y as an individual of the class `Response` must be given. Further classification of individuals is done by a standard *OWL 2* reasoner (e.g. HermiT[4]). In case of an inconsistency, i.e. disjointness of mandatory and excluded features can not be satisfied, the procedure exits. Otherwise, the sets of excluded features and mandatory features are collected, respectively. Finally, the algorithm returns the set of mandatory features and the set of optional features for the learning task. Note that this can be done without looking at the data, but by relying on engineering domain knowledge. After applying *Semantic Feature Selection* learning tasks such as cycle time forecasting can be employed with the reduced set of mandatory and optional features.

The main advantages of our approach in comparison with common feature selection procedures are summarized in Table 3.

[4] http://hermit-reasoner.com/

Algorithm 1. Semantic Feature Selection

```
Input : Learning problem $y$, feature ontology $\mathcal{O}$, feature space $\mathcal{F}$
Output : Mandatory features $\mathcal{S}_m$, optional subset $\mathcal{S}_o$
$\mathcal{A} \leftarrow \texttt{Response(y)}$                    ▷ Instantiate response variable
$\mathcal{S}_o \leftarrow \mathcal{F}$
$\mathcal{S}_m \leftarrow \emptyset$
if $\mathcal{O} \models Dis(\texttt{ExcludedFeature}, \texttt{MandatoryFeature}) \sqsubseteq \perp$ then
    return                    ▷ Unsatisfiable disjointness, inconsistent ontology
end if
for $x_i \in \{x \mid \mathcal{O} \models \texttt{ExcludedFeature(x)}\}$ do
    $\mathcal{S}_o \leftarrow \mathcal{S}_o \setminus x_i$
end for
for $x_i \in \{x \mid \mathcal{O} \models \texttt{MandatoryFeature(x)}\}$ do
    $\mathcal{S}_m \leftarrow x_i$
end for
return $\mathcal{S}_o, \mathcal{S}_m$
```

Table 3. Advantages of semantic feature selection

Criterion	Today	Our approach
Complexity	Grows with dimensionality and size of data sets	Grows only with dimensionality (i.e. new feature entities)
Re-usage	Need to be re-executed for every incoming instance	Needs only re-execution if new features are added
Intepretability	Reducing feature spaces often loses intuitive interpretability	Explicitly focuses on facilitated human interpretation

4.3 Graph Kernel Lasso

In some cases, data sets of automation systems are sparse. For example, if we want to forecast cycle times of rarely produced products, there will only be very few instances available for training. For better learning model performance it would be beneficial to further consider the semantics of the feature space during model training. In order to tackle this problem, we present a technique that integrates semantic dependencies into linear model learning and simultaneous feature selection.

In contrast to the standard graph Lasso defined in (3), where a relation between two features is either present or not, i.e. the graph's adjacency matrix $A_{i,j} \in \{0, 1\}$, we want to use a more enhanced notion of dependencies that also takes semantics into account. In our application, we use RDF-graph kernels to capture similarities between entities in the manufacturing feature ontology. In reference to (3), we therefore define a graph kernel-weighted regularization term that encourages smoothness between similar entities in the RDF graph of the feature ontology.

$$\Omega(\boldsymbol{w}) = \alpha\|\boldsymbol{w}\|_1 + (1-\alpha)\sum_{i,j\in V}\kappa(G_i,G_j)(w_i - w_j)^2 \tag{4}$$

where G_i and G_j are the spanned graphs of entities i, j in the vertex set of the whole RDF graph V and κ is some RDF graph kernel. It is easy to see that the second regularization term can be expressed as weighted Laplacian L_κ.

$$\sum_{i,j\in V}\kappa(G_i,G_j)(w_i - w_j)^2 = \boldsymbol{w}^T L_\kappa \boldsymbol{w} \tag{5}$$

with $L_\kappa = diag(r_1, r_2, ..., r_p) - \kappa$, and r_i denoting the ith row sum of the kernel matrix κ. We refer to this model as the graph kernel Lasso (*GraKeLasso*). Note that if the kernel matrix is set to the identity matrix, this model is equivalent to the ElasticNet [14].

As mentioned above, the regularization penalty induces a smoothing, or grouping in a sense, that features that are similar with respect to the feature ontology graph have similar parameter values. Intuitively, if two features are closely related, e.g. torque and speed measurements of a conveyor motor, they should have a similar influence (i.e. signal) on the response variable. Additionally, if one feature turns out to be irrelevant, all its closely related features are also very likely to be irrelevant. The graph kernel Lasso model enforces both of these properties.

For our use case application, we can resort to a wide variety of graph kernels, such as the implementations of the *mustard* framework[5].

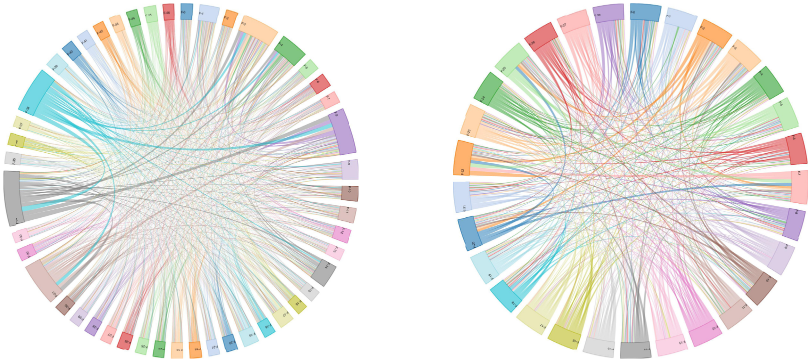
A visual representation of the feature ontology graph kernels from our use case data set is given in Figure 6(a) for the full feature space on the left-hand side and (b) for the reduced feature space after semantic feature selection on the right-hand side. Every feature is represented by a segment on a circle the width of the chords connecting two features depicts the strength of similarity between the two. It can be seen that the full feature space contains some very dominant feature similarities, while the reduced feature space exhibits a more uniform distribution.

5 Evaluation

To show the value of semantic guidance in feature selection and the custom Lasso model, we evaluated the performance of forecasting cycle times, as sketched in the manufacturing use case scenario. The regression models are trained on different data sets generated by a discrete-event simulation model that conforms to the manufacturing process in section 2.

We compare five different regression models for the cycle time estimation task: Lasso, ElasticNet, Graph Lasso, *GraKeLasso*, and OLS. For *GraKeLasso*, we used a subtree-based variant of the Weisfeiler-Lehman graph kernel, whereas Graph Lasso incorporates only information about feature individuals connected via `dependsOn`, i.e. a simple dependency graph.

[5] https://github.com/Data2Semantics/mustard



(a) Full feature space **(b)** Reduced feature space

Fig. 6. Visualization of pairwise graph kernel weights between features. (a) Full feature space, (b) Reduced feature space after semantic feature selection

Set Up. Our simulation set up comprises of a source that generates two different product types at a specified time interval, each of which has a different distribution for weight and size. These products are sent to two separated assembly lines, where first a loading station measures product qualities using a balance and a barcode scanner. For the conveyor we monitor its electric motor (power, torque, speed, temperature) and some induced failure events. The simulated quality control again measures sizes of the products. Finally, the packaging station samples the cycle times we want to forecast. Each station further observes its current workload and operating timestamp (seen as soft sensors).

Overall, the final feature ontology consists of 6 processes (one additional for the separated assembly), 18 sensing devices that sample 47 different measurements, i.e. feature individuals. In addition to that there are 13 concrete instantiations of relations between two features.

Table 4. Original and reduced variants of product cycle time data set

Data Set	**n**	**p**	**Reduction**	**OLS CV RMSE**
Cycle time full	2000	47	-	$\approx 1.36 \times 10^{11}$
Cycle time semantic reduced	2000	29	38.3 %	0.08
Cycle time p-value reduced	2000	18	**61.7** %	**0.06**
Cycle time sparse	40	47	-	9.49

Data Sets & Results. Starting from the original cycle time data set, we obtain three additional variants for evaluation purposes. Table 4 depicts their individual

Table 5. Embedded model performances on 10-fold cross validation

Data Set	**Model**	**Reduction**	**CV RMSE**
Cycle time full	Lasso	**47.4** %	0.42
	ElasticNet	34.4 %	0.57
	Graph Lasso	4.5 %	**0.32**
	GraKeLasso	4.9 %	**0.32**
Cycle time sparse	Lasso	**51.3** %	0.48
	ElasticNet	8.7 %	0.46
	Graph Lasso	8.9 %	0.54
	GraKeLasso	6.8 %	**0.43**

characteristics. The full data set consists of 47 dimensions and 2000 instances, while in the sparse case, the number of instances is kept to 40 so sparseness is preserved. After applying the reasoning procedure presented in Algorithm 1, the number of features p reduces to 29 – approximately 38 % reduction. On the other hand, a common p-value based selection reduces dimensionality to 18 (at 0.05 significance threshold). This means that there are many linear dependencies in the original data set which can be eliminated by pairwise correlation. The semantic approach does not eliminate dependencies by correlation, but excludes features that are inferred to be independent of the response variable by means of the feature ontology. For each of the data sets an OLS model is trained and evaluated with respect to the coefficient of variation of the root-mean-square error (CV RMSE) in cycle time seconds. It can be seen that best performance is given for the p-value reduced data set, however, the semantic reduced data set shows competitive results.

The embedded feature selection models are evaluated in a similar setting. Model performances shown in Table 5 correspond to the overall best value determined by a grid search over λ, whereas $\alpha \in]0, 1]$ was set to the best of an inner cross validation, respectively. Final performance results are again averaged over 10-fold cross validation. The reduction column also reports on each of the embedded model's average feature selection capabilities, i.e. number of zero valued coefficients.

Discussion. Overall, our results indicate two main insights. First, compared to p-value based selection, which does a better job at reducing dimensionality, semantic feature selection shows competitive performance in a sense that it keeps the needed features in its original form without any data-intensive computations. Interestingly, after upfront feature selection the ordinary least squares model outperforms all the other approaches for this setting and yields the overall lowest error. Second, our *GraKeLasso* shows best performance on the full and the sparse data set, because it can take similarities of the whole feature space into account. In summary, it can be seen that both of our approaches effectively decrease prediction errors and show competitive or even superior performance compared to conventional techniques. Due to this limited simulation scenario, we could not show that a combination of both approaches is beneficial. Further evaluations on large-scale systems, when upfront feature selection alone does not suffice, are necessary to investigate this.

6 Related Work

Due to the plethora of research concerned with feature selection, we will only present related works that are closely connected to the one in this paper. For a general overview of the field, we refer to the survey paper [4].

Coming from a semantic perspective on feature selection, the technique introduced by [6] describes a fuzzy approach to capture implicit semantics of data sets in order to reduce their dimensionality. They apply fuzzyfication on the degree of which features are dependent based on their co-occurring consistency with the decision variable. However, this technique does not rely on any explicit semantics defined in the data model and also needs preferably access to the full data set. In the field of biomedical machine learning applications, considering domain knowledge in feature selection has been studied and shown that feature spaces for association rules can be greatly reduced when medical domain knowledge about concepts is introduced, see [1]. The authors introduce dependencies between medical concepts, such as diseases and treatments, in order to learn more compact association rules. Another kind of related work that has been studied with increasing interest is the family of Lasso regularization models. Algorithms like the GOSCAR have been applied to consider dependency knowledge between genes in DNA sequences as penalty for group regularization in graph Lasso models. These models have shown to increase classification accuracy in several studies, e.g. [11]. Similarly, for image classification semantic dependencies between labeled images have been integrated into a Lasso approach [2].

In summary, including semantics into feature selection has been the concern of very few research studies outside of text and image processing domains, where semantics are mostly given through natural language. However, there are certain knowledge-based approaches that argue for the dynamic adaption of features to account for changes in the data generation process [9].

7 Conclusion and Future Work

In this work, we introduced the application of semantic feature selection for machine learning models in modern industrial automation systems. Only few works have been concerned with the usage of explicit semantics, in order to facilitate feature selection, which still remains one of the main issues. Especially in the manufacturing domain, there are many known dependencies between measurements that are cut out to guide this process. In this paper, we show how a small amount of semantic relations can be used to significantly reduce the size of feature spaces for exemplary learning problems in manufacturing systems and still yield good performance. Furthermore, we presented an embedded feature selection for linear models that captures feature similarities within RDF graphs and outperforms conventional approaches when applied to sparse data sets.

In future work, we plan to implement the developed approach in a real-life automation system. A particular promising direction seems to be the conjunction of this approach within OBDA systems, where ontologies are used to retrieve instance data. By deploying machine learning on top of OBDA, a coupling of our approach and ontology-based queries could reveal some synergy effects.

References

1. Blake, C., Pratt, W.: Better rules, fewer features: a semantic approach to selecting features from text. In: Proc. of IEEE Int. Conf. on Data Mining, pp. 1–8 (2001)
2. Chen, X., Yuan, X., Yan, S., Tang, J., Rui, Y., Chua, T.S.: Towards multi-semantic image annotation with graph regularized exclusive group lasso. In: Proc. of 19th ACM Int. Conf. on Multimedia - MM 2011, pp. 263–272 (2011)
3. de Vries, G.K.D.: A fast approximation of the Weisfeiler-Lehman graph kernel for RDF data. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part I. LNCS, vol. 8188, pp. 606–621. Springer, Heidelberg (2013)
4. Guyon, I.: An Introduction to Variable and Feature Selection. J. Mach. Learn. Res. (JMLR) **3**, 1157–1182 (2003)
5. Jenatton, R., Audibert, J.Y., Bach, F.: Structured Variable Selection with Sparsity-Inducing Norms. J. Mach. Learn. Res. (JMLR) **12**, 2777–2824 (2011)
6. Jensen, R., Shen, Q.: Semantics-preserving dimensionality reduction: Rough and fuzzy-rough-based approaches. IEEE Transactions on Knowledge and Data Engineering **16**(12), 1457–1471 (2004)
7. Jirkovsky, V., Obitko, M., Novak, P., Kadera, P.: Big data analysis for sensor time-series in automation. In: Proc. of Emerging Technology and Factory Automation (ETFA), pp. 1–8 (2014)
8. Lösch, U., Bloehdorn, S., Rettinger, A.: Graph kernels for RDF data. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 134–148. Springer, Heidelberg (2012)
9. Ringsquandl, M., Lamparter, S., Lepratti, R.: Context-aware analytics in MOM applications. In: Workshop Notes of the 6th International Workshop on Acquisition, Representation and Reasoning about Context with Logic (2014)
10. Rodríguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: ontop of databases. In: Proc. of the 12th Int. Sem. Web Conf. (2013)
11. Yang, S., Yuan, L., Lai, Y.c., Shen, X., Wonka, P., Ye, J.: Feature grouping and selection over an undirected graph. In: Proc. of the Int. Conf. on Knowledge Discovery and Data Mining (KDD), pp. 922–930 (2012)
12. Ye, J., Liu, J.: Sparse Methods for Biomedical Data. SIGKDD explorations **14**(1), 4–15 (2012)
13. Yu, L., Liu, H.: Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Poc. of the 20th Int. Conf. on Mach. Learn., pp. 1–8 (2003)
14. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society **67**, 301–320 (2005)