Transforming Unstructured Natural Language Descriptions into Measurable Process Performance Indicators Using Hidden Markov Models

Han van der Aa[a,*], Henrik Leopold[a], Adela del-Río-Ortega[b], Manuel Resinas[b], Hajo A. Reijers[a,c]

[a]*Department of Computer Sciences, VU University Amsterdam,*
, De Boelelaan 1081, 1081HV Amsterdam, The Netherlands
[b]*Departemento de Lenguajes y Sistemas Informticos, University of Seville,*
Avda. Reina Mercedes 41012, Sevilla, Spain
[c]*Department of Mathematics and Computer Science, Eindhoven University of Technology,*
PO Box 513, 5600MB Eindhoven, The Netherlands

Abstract

Monitoring process performance is an important means for organizations to identify opportunities to improve their operations. The definition of suitable Process Performance Indicators (PPIs) is a crucial task in this regard. Because PPIs need to be in line with strategic business objectives, the formulation of PPIs is a managerial concern. Managers typically start out to provide relevant indicators in the form of natural language PPI descriptions. Therefore, considerable time and effort have to be invested to transform these descriptions into PPI definitions that can actually be monitored. This work presents an approach that automates this task. The presented approach transforms an unstructured natural language PPI description into a structured notation that is aligned with the implementation underlying a business process. To do so, we combine Hidden Markov Models and semantic matching techniques. A quantitative evaluation on the basis of a data collection obtained from practice demonstrates that our approach works accurately. Therefore, it represents a viable automated alternative to an otherwise laborious manual endeavor.

Keywords: performance measurement, process performance indicators, natural language processing, hidden markov models, model alignment

[*]Corresponding author. *Phone number:* +31 20 59 87788
Email addresses: `j.h.vander.aa@vu.nl` (Han van der Aa), `h.leopold@vu.nl` (Henrik Leopold), `adeladelrio@us.es` (Adela del-Río-Ortega), `resinas@us.es` (Manuel Resinas), `h.a.reijers@vu.nl` (Hajo A. Reijers)

1. Introduction

Many organizations adopt a process-orientated view to identify opportunities for improving their operations [1]. Monitoring process-related Key Performance Indicators, so-called Process Performance Indicators (PPIs), represents an important prerequisite in this strive for continuous process optimization [2]. A key task for managers, therefore, is to define suitable PPIs, which are aligned with the strategic business objectives of the organization [3]. Typically, they achieve this by describing relevant PPIs using natural language descriptions [4, 5]. This has the great advantage that PPIs can be easily specified and understood by all stakeholders [6].

However, to successfully monitor PPIs, it must be clear how they relate to the technical implementation of a business process in a Workflow Management or Enterprise Resource Planning System [7]. For instance, the "*Average time to complete a received order*" PPI requires a system to a) infer that an average time measure has to be computed across all process instances and b) determine which events in the process indicate the receipt and the completion of an order, i.e. the start and end points of the time to be measured. One way to capture this information is through the use of a structured notation for specifying the contents of PPIs and their relation to business process elements (cf. [3, 4, 8]). The problem is that such structured notations are not at all similar to the unstructured natural language descriptions used and preferred by managers.

Currently, a manual transformation is the only way to obtain a structured specification of the contents of PPIs and their relations to business process elements [9]. Such a transformation requires considerable time and effort from a number of resources. To illustrate the causes of this, consider the situation we observed during our extensive research collaboration with the *Andalusian Health Service* [5, 10, 11]. There, the IT department received requests to measure PPIs for all organizational processes, as specified by other departments in natural language. The IT department had to manually establish SQL queries in order to compute the desired values for these PPIs. In many cases, the necessary interactions between business and IT led to miscommunication. For example, as a result of misinterpretations or incomplete specifications, the IT department often obtained incorrect PPI values. The time required to clear up these differences was considerable. In the general case, this necessary effort is even more problematic in light of the potentially vast size of process model repositories (possibly containing hundreds or even thousands of models [12]) and because processes and performance measures are subject to continuous change [13]. In sum, this makes the task of manually transforming PPIs into a structured notation hardly manageable in practice.

To overcome the problems associated with a manually performed transformation, this paper presents an approach to automatically translate natural language PPI descriptions into PPIs defined according to a structured notation. The proposed approach first transforms an unstructured natural language description into a structured format. In this step, we extract information on concepts relevant to the calculation of a PPI from the PPI description. For instance, we recognize "*average*" from the exemplary PPI as an *aggregation function* over the process instances. Secondly, our approach aligns the process concepts contained in the description with the corresponding elements of a process model. To this end, our approach establishes links between a PPI description and the implementation of the process, for instance, by identifying the system event that corresponds to the completion of an order. As a result, the approach delivers structured and aligned PPI descriptions that can be directly used for automated monitoring of process performance.

The remainder of this paper is structured as follows. First, Section 2 illustrates the problem addressed by our approach and describes streams of related work. Section 3 then defines templates that we use for the structured notation of PPIs. Section 4 describes our automated approach for the transformation of natural language PPI descriptions into this structured notation. In Section 5, we present a quantitative evaluation of our approach. The limitations of our transformation approach and the evaluation are discussed in Section 6. Finally, we conclude the paper and describe directions for future research in Section 7.

2. Background

2.1. Problem Illustration

To illustrate the challenges associated with the transformation of natural language PPI descriptions into a structured notation, consider the process model shown in Figure 1. This figure depicts a simplified order handling process using the Business Process Model and Notation (BPMN). Table 1 provides six exemplary PPI descriptions related to this process. These descriptions use linguistic patterns that are similar to those that have been observed in practice [5].

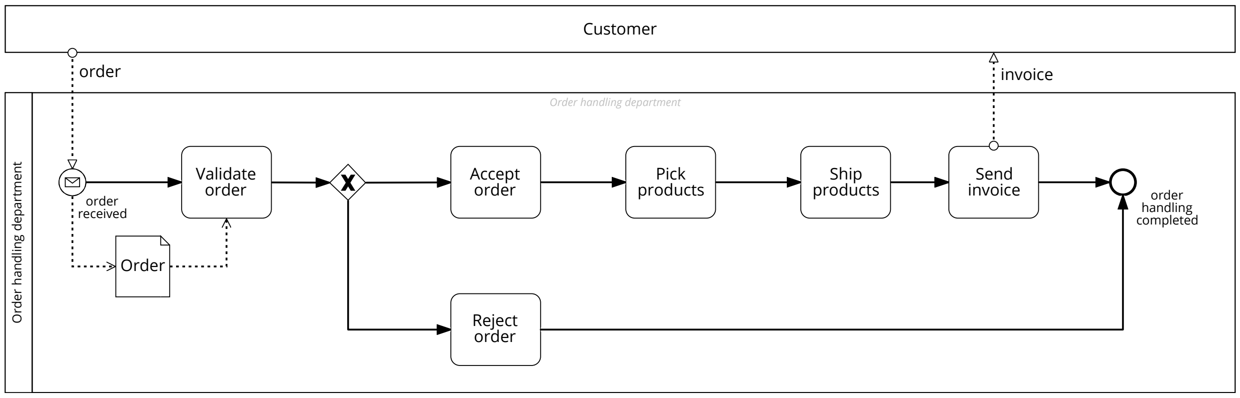

Figure 1: Process model for an order handling process

Table 1: PPIs for the order handling example

ID	Description
PPI1	Number of accepted orders.
PPI2	Average time between receipt and completion of an order.
PPI3	Average time to complete a received order.
PPI4	The percentage of rejected orders.
PPI5	The maximum time to transport an order.
PPI6	The total order amount per customer.

To actually compute values for these PPIs, the natural language descriptions must be transformed into a structured notation. In some cases, this transformation is relatively straightforward. For instance, it is clear that the description of $PPI1$ refers to the number of process instances for which the "*Accept order*" activity is executed. For $PPI2$, we are interested in the average time between the "*order received*" and "*order handling completed*" events. Table 2 provides an example of how this PPI can be captured in a structured manner. Here we use *measure type* and *aggregation* to specify that the PPI computes the average time over process instances, whereas we use *start event* and *end event* to establish the link between the PPI and the events of the process model depicted in Figure 1. However, in other cases, the automated transformation of a PPI description into a structured definition is associated with considerable challenges. These challenges mainly relate to the flexible and inherently ambiguous nature of natural language, as preferred by human users, but much less suitable for automated interpretation [14, 15]. We identify three main challenges in this regard.

Table 2: Example of a structured notation for $PPI2$

Slot	**Value**
ID	*PPI2*
Description	*Average time between receipt and completion of an order*
Measure type	*time*
Aggregation	*average*
Start event	*Order received*
End event	*Order completed*

The first challenge to overcome is that natural language descriptions do not follow a specific structure or notation. These descriptions can express the same PPI using a broad variety of syntactic patterns [16]. $PPI2$ and $PPI3$, for instance, both refer to the average time between the receipt and completion of an order. However, the two PPI descriptions use clearly distinct patterns to describe this measure. $PPI2$ refers explicitly to the start and end points of the measure in chronological order. By contrast, $PPI3$ describes these two points in a reverse order, i.e. the end point "*completed*" is described before the start point "*received.*"

The second challenge to overcome is that natural language PPI descriptions can depend on implicit knowledge for their proper interpretation. Consider, for example, $PPI4$, "*The percentage of rejected orders*". This PPI refers to some fraction, where the numerator refers to the number of process instances in which the "*Reject order*" activity is executed. However, the PPI description does not specify the denominator for this fraction, i.e. it is not clear from the description by what number this numerator should be divided. Instead, the description depends on the implicit assumption that the denominator, most likely, refers to the total number of received orders.

Lastly, the third challenge that a transformation approach must address are problems caused by differences in terminology between PPI descriptions and process models. For example, the description of $PPI5$ refers to the time it takes to *transport orders*, whereas the process model refers to this action as *product shipment.* Such differences occur in particular because PPIs and process models are generally defined by organizational stakeholders that have different perspectives on a process [17].

In summary, an automated transformation approach must be able to deal with (i) a variety of syntactic patterns, (ii) partially implicit PPI descriptions, and (iii) differences in terminology between description and model. In Section 2.2 we review existing work that relates to these challenges.

2.2. Related Work

This paper presents the first work on transforming natural language PPI descriptions into a structured notation that allows for the automatic computation of the respective PPI values. However, there are several approaches that relate to our work or partially address similar challenges. To properly reflect on this, we review three main streams of research that are related to the goal of our approach: (i) structured notations of PPIs, (ii) information extraction from natural language, and (iii) research on model matching.

2.2.1. Structured PPI Notations

Performance measurement is an active research field in management science, which has gained interest in both academia and business [8]. Much work has been performed on the identification and classification of Key Performance Indicators in general settings [18] and those relevant for specific domains such as logistics, production, and supply chains (cf. [19, 20, 21, 22]). Within the context of *Process Performance Measurement*, great effort has been put on the formalization of PPI definitions. This has resulted in a number of notations and frameworks for the description and monitoring of PPIs [4, 5, 8, 23, 24, 25, 26, 27, 28]. The main differences among them are found in their expressiveness, i.e. the different types of PPIs that can be defined, and their features to directly support monitoring.

Most of the aforementioned frameworks can be used to capture PPIs in a structured notation and automatically monitor their values. In this work, we describe an approach that works independent of a specific framework. Our approach takes as input an unstructured natural language PPI description and transforms it into a structured PPI definition. The structured notation that we define for this purpose is generic; its components can be mapped to concepts used in specific PPI frameworks, such as PPINOT [5] or notations used in [4, 24]. Since our approach can transform an unstructured PPI description into a structured notation, it addresses the second challenge we described in Section 2.1. To achieve this, we make use of information extraction techniques.

2.2.2. Information Extraction from Natural Language

To transform a natural language PPI description into one in a structured notation, we need to identify the parts of a PPI description that correspond to concepts of the structured notation. This task resembles a so-called *template-filling problem* [29]. In this context, a template consists of fixed sets of *slots*, which take as values *slot-fillers* belonging to particular classes. The task of template filling is then to fill the slots with information extracted from a text. By performing such an information extraction, we can deal with the variable structure of natural language PPI descriptions. Therefore, template filling addresses the first challenge denoted in Section 2.1.

Template filling, which is also referred to as slot filling [30] or semantic-based understanding [31], has been extensively studied and applied in a variety of contexts. A major application area for these techniques is *spoken language understanding*, where information is extracted from unstructured natural language text in the context of a dialog system [32]. To achieve this, many approaches employ probabilistic models, such as (variations on) Hidden Markov Models (HMMs) [33, 34]. These and other existing approaches have been shown to work well in specification application contexts. However, Jurafsky [29] notes that their good performance is partially due to certain constraints. Namely, in the evaluated application scenarios, all texts are known to be relevant for the specific task and are relatively small. Furthermore, the slots of the templates are to be filled with snippets from the text itself. The former two constraints are also applicable in the context of this work. Yet, the latter is not applicable here. Instead of filling slots with information directly extracted from textual descriptions, we need to establish links between the concepts contained in these descriptions and the actual implementation of a business process. In the context of slot-filling problems, this task is referred to as *entity linking* [35]. To achieve this, we build on techniques from the area of model matching, as we describe in the next sub section.

2.2.3. *Model Matching*

The definition of automatically computable PPIs also depends on the establishment of links between concepts described in a PPI description and the elements of the process model that capture the actual process implementation. The task of identifying such relations between concepts in different artifacts is generally referred to as *semantic matching* [36]. This data integration task has received wide attention in the form of (database) schema matching [37, 38, 39] and ontology matching [40, 41]. In the context of business processes and process modeling, process model matchers have been established, which automatically identify correspondences between activities and events of process models [42]. Process model matchers exploit a variety of process model features, including natural language [43], model structure [44], and behavior [45]. Other approaches also consider the establishment of links between process models and other artifacts, such as event logs [46] and textual process descriptions [47, 48]. All of these approaches deal with the third challenge identified for the approach presented here: dealing with differences in terminology between the concepts of various artifacts. Therefore, we will use techniques applied in approaches that consider the semantic similarity between terms and the structural relations between process model elements.

The contribution of this paper is a transformation approach that deals with the challenges associated with the automated transformation of unstructured PPI descriptions into a structured notation. To achieve this, we build on the three research areas that we reflected upon in this section. We combine information extraction and model matching techniques in a novel manner. Furthermore, this work is the first to apply these techniques in the context of process performance measurement.

3. PPI Templates

In this paper we set out to transform a natural language PPI description into a structured notation that allows to automatically compute the PPI value. This requires us to first define the structured notation into which we aim to transform the unstructured descriptions. We define this structured notation in the form of *PPI templates*. Each PPI template captures the different concepts of a PPI definition that are required for its automated computation. We refer to the placeholder of each concept in the template as a *slot*, to which a value, a so-called *slot filler*, must be assigned when defining a PPI. For the definition of these templates, we build on the PPINOT metamodel [5]. We have selected this metamodel because of its high degree of expressiveness and because it explicitly establishes links to process model elements. This metamodel can be used to define structured PPIs whose values can be computed in an automated manner. Specifically, we base the templates for our approach on the templates and associated linguistic structures defined in [10]. In Section 3.1, we introduce the slots that correspond to the different semantic concepts that constitute the PPI templates. Section 3.2 describes the value domains with which each of these slots can be filled.

3.1. Template Slots

We define four different PPI templates, each corresponding to a different type of measure that may underly a PPI: *numerical*, *temporal*, *data-based*, and *fractional* PPIs. These templates can capture the vast majority of PPIs that have been observed in empirical studies (cf. [5, 9]). Table 3 presents the four templates and provides an example for each of them.

Table 3: PPI templates and examples

Slot	**Value**	**Slot**	**Value**
ID	*PPI1*	ID	*PPI2*
Description	*Number of accepted orders*	Description	*Average time between receipt and completion of an order*
Measure type	*count*	Measure type	*time*
Aggregation	*sum*	Aggregation	*average*
Counted event	*Accept order <completed>*	Start event	*Order received*
Group-by	—	End event	*Order completed*
		Group-by	—
ID	*PPI4*	ID	*PPI6*
Description	*The percentage of rejected orders*	Description	*The total order amount per customer.*
Measure type	*fraction*	Measure type	*data*
Aggregation	*sum*	Aggregation	*sum*
Numerator	*Reject order <completed>*	Measured attr.	*Order [amount]*
Denominator	*Order received*	Group-by	*customer*
Group-by	—		

Aside from an identifier and the natural language description, all of the templates capture four distinct semantic concepts, as illustrated by Figure 2: (i) measure type, (ii) aggregation function, (iii) group-by property, and (iv) one or more process concepts. The first three semantic concepts together characterize the measure of a PPI. The fourth concept links components of the PPI to the actual implementation of a business process. We now elaborate on these semantic concepts.

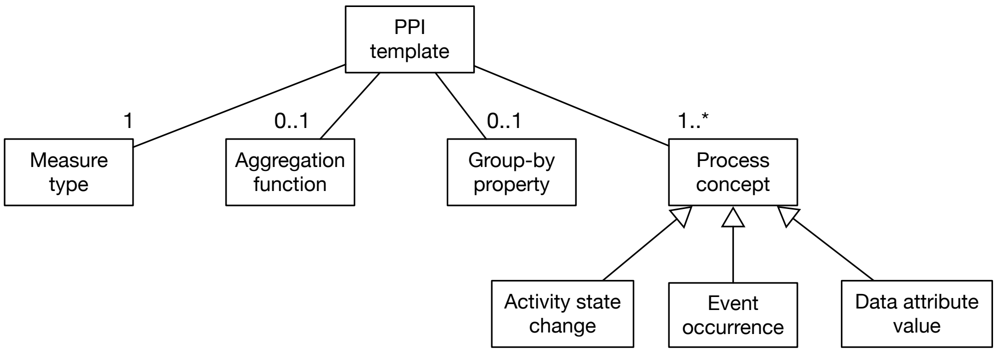

Figure 2: Semantic concepts in a PPI definition

Measure type. The measure type classifies the nature of the measure underlying a PPI. In this work, we focus on the four most common measure types according to [10]: count, time, data, and fraction measures. A *count* PPI measures the number of times something happens. For instance, $PPI1$ from the order handling example measures the number of orders that are accepted. *Time* measures consider the duration between two instants, i.e. the start or completion of an activity or the execution of an event. $PPI2$ and $PPI3$ represent examples of such time measures. *Data* measures consider the attribute values of data objects. For example, $PPI6$ sums the amounts associated with "*order*" data objects. Finally, *fraction* measures divide the value of one measure by another. For example, $PPI4$ divides the number of *rejected orders* by the number of *received orders.*

Aggregation function. Aggregation functions are used to aggregate the values of multiple process instances. The most common functions are *sum*, *maximum*, *minimum*, and *average* [10]. For example, $PPI2$ and $PPI3$ consider the *average* time of individual order handling instances, whereas $PPI5$ considers the *maximum* time it takes to transport an order. Note that the aggregation function is an optional slot in the templates, since it is not necessary to specify it for every measure.

Group-by property. Group-by properties can be applied on measures with an aggregation function in order to compute a value aggregated over those process instances that have a certain property. These properties can be based on data attributes or on resources that are involved in the execution of a process instance. For example, $PPI6$ uses a group-by property in order to determine the total order amounts *per customer.* A group-by property is an optional slot in a PPI template. If no such property is defined, the aggregated measure is simply applied on all process instances.

Process concepts. A process concept refers to a value or an occurrence of something during the execution of a business process relevant to the computation of a PPI. As shown in Figure 2, process concepts, in the context of a PPI template, can refer to different things. For instance, an activity is started, a process model event occurs, or the value of a data attribute changes. Each PPI template has one or more slots associated with process concepts. The exact number and the semantics of these slots differ per measure type. In particular, we define the following semantic roles for the different measure types:

- count measure: *counted event*;
- time measure: *start event*, *end event*;
- data measure: *measured attribute*;
- fraction measure: *numerator*, *denominator.*

Each of the semantic roles associated with a particular measure type should be filled with at least one process concept. For example, a time measure should always have a *start* and an *end* event. Furthermore, a slot may be filled with multiple events. For example, a count measure can be used to measure the total number of occurrences of multiple events, such as the number of accepted *and* the number of rejected orders.

3.2. Slot Domains

As illustrated in the previous section, PPI templates contain four different types of slots: measure types, aggregation functions, group-by properties, and process concepts. In a structured PPI definition, each of these slot types can only be filled with values from a closed class, i.e. from a specific *domain.* By filling slots with values from known domains, we can ensure that values for the defined PPIs can actually be computed in an automated manner. Consider, for example, the textual description of $PPI2$. This description refers to the "*completion of an order*". Without incorporating domain knowledge, it is not clear to which process concept this statement exactly refers. Therefore, it is not possible to compute a value for this PPI based on just this information. If we, instead, fill this slot with a value from the appropriate slot domain, i.e. by associating the slot with the "Order handling completed" process model event, we overcome this issue. Then, we know that the computation of a value for this PPI requires us to stop the measurement when this specific event occurs. Table 4 summarizes the domains for the four different slot types.

Table 4: Domains associated with template slots

Slot type	**Domain values**
Measure type	*count*, *time*, *data*, *fraction*
Aggregation function	*minimum*, *maximum*, *average*, *sum*
Group-by property	Process model resource roles & data attributes
Process concepts	Process model activities, states, events, & data attributes

Measure type & Aggregation. The domains for measure type and aggregation function slots are independent of the process to which the PPI relates. A measure type slot always receives a value from one of the four measure types considered by our approach: count, time, data, and fraction. Similarly, the domain for aggregation functions covers the most common functions identified in [10]: minimum, maximum, average, and sum. The semantics of these aggregation functions differ per measure type. For instance, for time measures an *average* function will yield the average time between two events. By contrast, when this same function is applied to a data measure related to an "*Order [amount]*" attribute, the function yields the average order amount.

Group-by property. The domain for group-by properties depends on the contents of the process model to which the PPI relates. The group-by property describes a data attribute or resource role that is used to aggregate process instances. Therefore, the domain for slots of this type depends on the set of resource roles R and the attributes of the data objects D included in the process model. For instance, the group-by property of $PPI6$ corresponds to the resource role *customer* $\in R$.

Process concepts. Process concept slots are filled with references to parts of a process implementation. Therefore, the domain associated with these slots differs per process model. This domain is based on the activities, events, and data attributes of a process model. Given these elements, a process concept slot can be filled in three different ways, as previously indicated in Figure 2.

First, process concepts in a PPI definition can refer to the state change of a process model activity. We consider state changes of activities, rather than activities as a whole, because the calculation of PPIs often requires the consideration of an exact time instant. For example, in order to compute the time it takes to ship products, we are interested in the difference between the time instants of the *start* and *completion* of the activity execution. These two, start and completion, represent the most common state change of activities. However, PPIs can also refer to other states, such as *pausing* or *cancellation* of activities. Second, process concept slots can correspond to the triggering of *process model events.* Since such event occurrences are always instantaneous, this is not associated with a state change. As an example, consider $PPI2$. There, both the start and end times refer to process model events, i.e. to the "*order received*" and "*order handling completed*" events. Third, process concept slots can be associated with data attribute values. For example, to compute $PPI6$, "*The total order amount per customer*", we have to consider the value of the data attribute *amount* of the *order* data object.

In the next section, we present our approach, which automatically fills the defined templates for natural language PPI descriptions.

4. Transformation Approach

To transform an unstructured natural language PPI description into a structured notation, we define the transformation task as a *template-filling* problem. In this section, we describe a transformation approach that addresses this problem. Given a natural language PPI description, our approach aims to obtain the information necessary to fill the slots of a PPI template. To achieve this, the transformation approach performs two subsequent steps, as depicted in Figure 3.

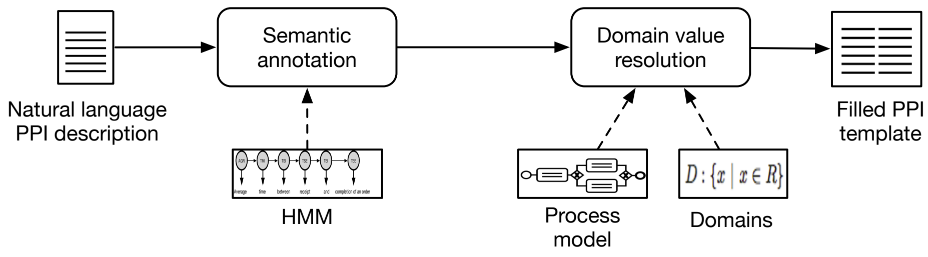

Figure 3: Overview of the proposed transformation approach

First, the approach parses a natural language description in order to identify the parts of the description that correspond to slots of the PPI template. For this parsing step we employ HMMs. We refer to this parsing task as *semantic annotation*. Second, the approach determines the appropriate slot-fillers by matching the semantically annotated parts to values in the relevant domain, e.g. by matching concepts from the natural language description to activities in a process model. We refer to this task as *domain value resolution*. In the remainder of this section, Sections 4.1 and 4.2 respectively describe the semantic annotation and domain value resolution tasks in detail. Afterwards, Section 4.3 considers the operationalization of our approach and its extensibility.

4.1. Semantic Annotation

The first step of our approach identifies those parts of a natural language PPI description that correspond to the semantic concepts contained in the PPI templates. For example, for $PPI5$: "*the maximum time to transport an order*", we aim to identify that "*maximum*" corresponds to the concept of an aggregation function, "*time*" corresponds to the measure type, and "*transport an order*" describes the events to be measured. For this semantic annotation task, we define a tag set $\mathcal{M}$ that can be used to annotate relevant semantic concepts in natural language PPI descriptions. Section 4.1.1 describes this tag set and provides exemplary annotations. In Section 4.1.2, we then illustrate how we employ HMMs to automatically annotate unstructured PPI descriptions.

4.1.1. Semantic Tags

Table 5 presents the tag set we use for the semantic-annotation step. We establish this tag set $\mathcal{M}$ by defining two sorts of tags. First, $\mathcal{M}$ includes tags that correspond to each of the semantic concepts from the PPI templates. For example, we use AGR to refer to an aggregation function and

FNE to denote a *numerator event* of a fraction PPI. Second, $\mathcal{M}$ contains tags used to annotate textual indicators that signal the transition of the description to a new semantic concept. For example, we use the tag TEI, i.e. a *time end indicator*, to show that a time end event will be described next. This can be seen for $PPI2$, "*Average time between receipt and completion of an order*", where the term *and* denotes a transition from the description of the start event (*receipt*) to the end event (*completion of an order*). Here, it is important to note that the term *and* only denotes this transition in this specific context, i.e. when it occurs in the description of a time measure and between the description of two events. In other contexts, this term likely has a considerably different meaning. Being able to deal with such context-dependent meanings of terms is a particular strength of the HMMs that we employ for the annotation.

When annotating a PPI description, our approach assigns tags to parts, i.e. *chunks*, of a PPI description rather than to individual words. Each word in the description is part of exactly one chunk and each chunk is assigned exactly one tag. We use $\pi =< \varphi_1, \ldots, \varphi_n >$ to denote the chunks of a partitioned description and $M =< m_1, \ldots, m_n >$ to describe the sequence of tags assigned to the chunks, where $m_i \in M$ is the tag that corresponds to the chunk $\varphi_i \in \pi$.

Table 5: Tag set used for semantic annotation

Tag	**Description**	**Examples**
AGR	Aggregator function	average, sum, maximum
GBI	Group by indicator	per, for each
GBC	Group by concept	category, customer, department
CMI	Count measure indicator	number of, volume of, number of times
CE	Counted event	accepted orders, incidents
TMI	Time measure indicator	time, duration, throughput time
TSI	Time start indicator	from, between
TSE	Time start event	order received, product picking
TEI	Time end indicator	to, until
TEE	Time end event	order completion, product shipment
TBE	Time start and end event	service interruptions, product packaging
FMI	Fraction measure indicator	percentage of, ratio of, proportion for
FNE	Fraction numerator event	rejected orders
FDI	Fraction division indicator	divided by, relative to, as a percentage of
FDE	Fraction denominator event	received orders

To illustrate the usage of the tag set, consider the following annotation of $PPI2$ from the order handling process:

$\pi \backslash M$ = `average\AGR, time\TMI, between\TSI, receipt\TSE, and\TEI, completion of an order\TEE`.

From this annotation, we can learn that "*average*" corresponds to the aggregation function slot of a PPI template, "*receipt [of an order]*" to the start event and "*completion of an order*" to the end event.

In the tag set $\mathcal{M}$, we purposefully do not define separate tags for data measures, because their semantic structure generally follows a similar syntactic pattern as other measures. Consider, for instance, $PPI6$: "*the total order amount per customer*". This description of a data measure follows an identical structure as a count measure would. However, in the context of this process model *amount*, it refers to a data attribute of an *order* rather than describing a regular count measure. Since such distinctions solely depend on the contents of the process model, i.e. on its data objects, we leave the differentiation between data and other measure types for the second step of our approach.

4.1.2. Semantic Annotation using HMMs

To automatically annotate a PPI description, we use an HMM, a so-called probabilistic sequence classifier. A *sequence classifier* is a model that assigns a label or class to each unit in a sequence, thus mapping a sequence of observations to a sequence of labels. In the context of our approach, it assigns a tag to chunks of natural language PPI descriptions. HMMs are *probabilistic* in the sense that they compute a probability distribution over all possible sequences of tags and choose the best tag sequence [29]. Formally, given a word sequence W that constitutes a PPI description, the goal of the HMM is to find the semantic representation of the meaning M that has the maximum *a posteriori* probability $P(M|W)$ [33]. This probability is given by the following equation:

$$\hat{M} = \arg\max_{M} P(M|W) = \arg\max_{M} P(W|M)P(M) \tag{1}$$

Equation 1 shows that, to compute $\hat{M}$, the HMM combines two separate models: a semantic prior $P(M)$ and a lexicalization model $P(W|M)$. We now briefly describe how these models work

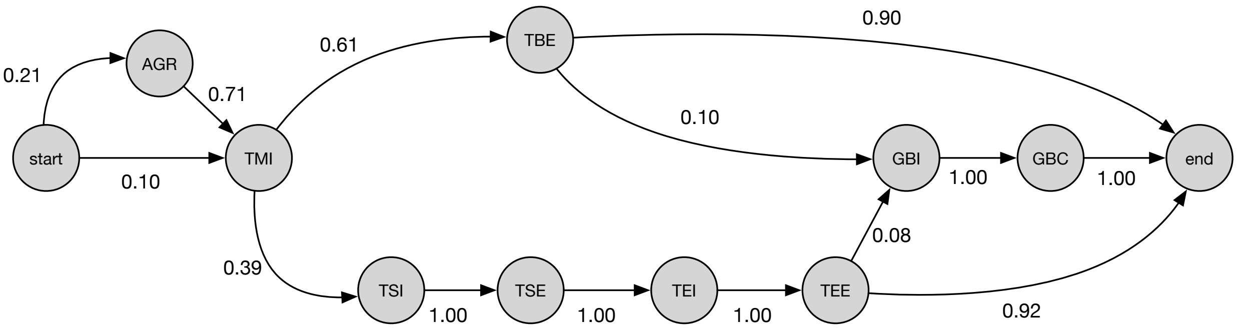

Figure 4: Fragment of a semantic prior $P(M)$

in the context of our transformation approach. For a more detailed explanation of the technical aspects underlying HMMs, we refer the interested reader to [49].

Semantic prior. The semantic prior $P(M)$ assigns a probability to an underlying semantic structure M. Intuitively, it models the probability that a PPI description follows a certain semantic structure, i.e. a sequence of tags from $\mathcal{M}$. The model can be represented as a probabilistic finite automaton. The states of this automaton correspond to labels in the set $\mathcal{M}$. The transition probabilities between states denote the probabilities that these states follow each other in a semantic structure. These probabilities can be learned by training an HMM on a collection of (partially) annotated PPI descriptions. Section 4.3 explains the training of an HMM in detail.

The semantic prior that results from a training phase can be used to determine the likelihood that PPI descriptions follow a certain semantic structure. As an example, consider the fragment of a semantic prior depicted in Figure 4. This figure shows the semantic prior relevant to the annotation of time measures[1]. From this semantic prior, we can observe that, despite the huge variability that natural language PPI descriptions can use to describe time measures, there is much less diversity in the semantic structure that they follow. For example, the prior shows that if a time measure (in this training set) has an *aggregation function*, this is the first semantic concept that is described, whereas an optional *group-by property* occurs at the end of a description. Furthermore, the prior shows that a *time measure indicator* (TMI) is followed by a specification of an event that describes both the beginning and end of the measure (TBE) with a probability of 0.61, i.e.

[1]Note that we only visualize non-zero transition probabilities.

$TMI \rightarrow TBE$ has a probability of 0.61. In the other cases, TMI is followed by a *time start indicator* (TSI). By contrast, a TSI is always followed by a *time start event* (TSE) (probability of 1.00), because there are no other semantic structures in which a TSI occurs. This means that a TSI will never be directly followed by, for instance, a time end event or a group-by indicator.

Lexicalization Model. The lexicalization model $P(W|M)$ quantifies the probability that a given word sequence W is used to model a certain semantic structure M. Intuitively, it quantifies the probability that W is used to convey a meaning M. In particular, it models the transition probabilities between words given a certain context, i.e. a certain semantic concept. Probabilities in the lexicalization model have the form $P(word_n|word_{n-1}, context)$, which is the probability of taking a transition from one word to another given a particular context [34]. For example, $P(time \mid throughput, TMI)$ is the probability that the word *time* follows the word *throughput* in the context of a TMI (time measure indicator). As for the semantic prior, these probabilities are learned from a (partially) annotated training set. By training a lexicalization model, we can, for instance, learn that the word *for* is much more likely to be followed by the word *each* in the context of a *group by indicator* (GBI) than in the context of a *time start event* (TSE). Therefore, the HMM will assign a higher likelihood to the possibility that "*for each*" indicates a *group-by property* than that it is part of the description of a start event.

PPI Annotation. Once the HMM has been trained, it can be used to assign tags to a previously unseen PPI description. To find the best tag sequence $\hat{M}$, the optimization problem denoted by Equation 1 considers the product of the probabilities from the semantic prior $P(M)$ and lexicalization $P(W|M)$ models. The combination of these two models follows intuitively, because the probability that a word sequence W corresponds to a certain semantic structure W, depends on both the likelihood of this semantic structure occurring $P(M)$ and the likelihood that the individual words in W are associated with a certain semantic meaning $P(W|M)$. Because this optimization problem refers to the most likely tag sequence for an entire PPI description, HMMs are able to assign the correct tag to terms that have context-specific semantics, such as described in Section 4.1.1.

The task of finding the optimal sequence of labels $\hat{M}$ is also referred to as *decoding.* The most common decoding algorithm for HMMs is the *viterbi algorithm* [29]. The viterbi algorithm is a form of dynamic programming that is applied in a wide range of natural language processing contexts, including machine translation [50], part-of-speech tagging [51], and spoken language processing [52]. Because of the common application of the algorithm and the fact that it only affects the computational efficiency of the optimization problem and not the final outcome, we abstract from its details and refer the reader to [29, 53] for a more detailed explanation. Section 4.1.1 already showed the semantic annotation obtained in this manner for PPI2. For the remaining three PPI descriptions considered in Section 3, we obtain the following semantic annotations:

$PPI1 : \pi_1\backslash M_1 =$ `number\CMI of accepted orders \CE`

$PPI4 : \pi_4\backslash M_4 =$ `the percentage\FMI of rejected orders \FNE`

$PPI6 : \pi_6\backslash M_6 =$ `the total\AGR order amount\CE per\GBI customer\GBC`

We take these semantic annotations as input for the next step of our approach: the domain value resolution step.

4.2. Domain Value Resolution

In the second step of our approach, we obtain a PPI definition by filling the slots of a PPI template with values from the appropriate domains. The semantic annotations obtained in the previous step tell us which chunks of text correspond to which slots in a PPI template. For example, from the annotation of $PPI2$, we know that the chunk φ_6 : "*completion of an order*" corresponds to the *end event* slot in a template. We illustrate the full connection between semantic annotation and template on the left-hand side of Table 6. However, this only presents an intermediate result of our approach. To be able to actually measure the value of this PPI definition, links have to be established to the events that occur during the execution of the process. For example, to compute a value for $PPI2$ we need to determine to which process model event the chunk φ_6 corresponds. Therefore, in this second step of our approach, we fill the template slots with values from the appropriate domains in order to obtain a template as depicted in Table 6b.

To perform this *domain value resolution* for aggregation functions and group-by properties, we consider the *semantic similarity* between a chunk φ and the respective domain values. For

Table 6: Domain value resolution for *PPI*2

(a) Fillers from semantic annotation

Slot	**Value**
Measure type	*time*
Aggregation	*"average"*
Start event	*"receipt"*
End event	*"completion of an order"*
Group-by	—

(b) Fillers after domain value resolution

Slot	**Value**
Measure type	*time*
Aggregation	*average*
Start event	*Order received*
End event	*Order completed*
Group-by	—

process concepts, we combine semantic similarity with *constraints*, which we impose based on the semantics of the various measure types. In the remainder, Section 4.2.1 and Section 4.2.2 describe semantic similarity and the additional constraints in detail.

4.2.1. Semantic Similarity

We use semantic similarity to determine for a given chunk $\varphi \in \pi$ which value in its associated domain D_φ has the most similar meaning. We consider *semantic* rather than (just) *syntactic* similarity for this task because it allows us to deal with semantically related terms, such as synonyms. For example, by considering semantic similarity, we can learn that the chunk "*transport an order*" is most closely related to the "*product shipping*" activity.

Tokenization. To determine the semantic similarity, we first apply a *tokenization* function on a chunk φ and on the domain values D_φ. This function performs three tasks. First, it splits the textual contents of a chunk or a domain value into individual terms. Second, the function filters out stop words like "*the*", "*an*", and "*from*", since they have been found to be of little use when considering similarity between texts [54]. Finally, the tokenization function *lemmatizes* the remaining terms. This means that all remaining terms are transformed into their grammatical base form, which is also referred to as *lemma.* For instance "*is*" and "*been*" are both transformed into "*be*". To implement these steps we use the *Stanford Parser* and the associated toolkit [55]. We denote the resulting bag-of-words for a chunk φ with ω_φ. As an example, the tokenization function yields the following bag-of-words for the chunk $\varphi =$"*completion of an order*": $\omega_\varphi = \{\texttt{completion, order}\}$.

Semantic similarity computation. Given two bags-of-words ω_φ and ω_d, we quantify the semantic similarity using a similarity measure proposed by Mihalcea et al. [56]. This measure combines the semantic similarity of individual terms with word specificity scores. Equation 2 presents this measure for a chunk φ and a domain value d.

$$sim(\varphi, d) = \frac{1}{2}\left(\frac{\sum\limits_{t\in\omega_\varphi} maxSim(t,\omega_d) \times idf(t)}{\sum\limits_{t\in\omega_\varphi} idf(t)} + \frac{\sum\limits_{t\in\omega_d} maxSim(t,\omega_\varphi) \times idf(t)}{\sum\limits_{t\in\omega_d} idf(t)}\right) \tag{2}$$

In Equation 2, $maxSim(t, \omega)$ determines the maximum semantic similarity between a single term t and a bag-of-words ω. Formally, this is computed as follows:

$$maxSim(t, \omega) = \max\{sim(t, t') \mid t' \in \omega\} \tag{3}$$

In this equation, $sim(t, t')$ captures the semantic similarity between two terms. We compute this value using a so-called *second order similarity* method [57]. Second order similarity is based on the statistical analysis of co-occurrences of terms in large text collections. These methods have the great advantage that they can deal with context-specific terms, such as typically found in the business settings relevant to our presented approach [58]. The function $sim(t, t')$ returns a value in the range $[0, 1]$, where 1.0 indicates perfect semantic similarity, i.e. the terms are equal.

Lastly, Equation 2 incorporates word specificity scores in the form of the well-established *inverse document frequency* (idf). The idf gives a low weight to common terms, because they have relatively low discriminating power, and a high weight to relatively uncommon terms.

The calculation of semantic similarity suffices for aggregation function and group-by property slots, since the appropriate values for these slots solely depend on the textual similarity between chunk and domain value. By contrast, for the filling of process concept slots, we also have to consider constraints related to the semantics of a measure type.

4.2.2. Constraints on event alignment

When filling the slots of a PPI template, it is important that the end result makes sense from a semantic perspective. For instance, the start event of a time measure should always occur before its end event. Otherwise, the resulting metric will be semantically invalid. For this reason, we impose certain constraints on the inter-relations of the process model elements assigned to process concept slots. To achieve this, we formulate the slot-filling task for event slots as an alignment problem. *Alignments* capture relations that exist between concepts from different artifacts. In the context of this work, an alignment σ consists of a number of pair-wise *correspondences*, each between a tagged chunk from a PPI description $\varphi \in \pi_e$ and an element of a process model $m \in M$, denoted as $\varphi \sim m$. We impose constraints on an alignment σ through a constraint function Γ. In particular, we use Γ to impose three semantic constraints. The first constraint applies to time measures, the other two relate to fraction measures. Because the constraints are applicable to the inter-relations that exist between the correspondences assigned to different slots, we do not define constraints for count measures, which only have a single slot to be filled through the correspondences.

Time measures. For time measures, we impose an ordering constraint on the events and activities associated with the *start event* and *end event* slots. From a semantic viewpoint, a PPI in which the start event occurs after the end event does not make sense. Therefore, we ensure that the model element m_i aligned to the start event slot π_i occurs in the process model before the element m_j associated with the end event slot π_j. We achieve this by considering the *strict order relation* $\rightsquigarrow$ that exists between the events and activities in a process model, where $a \rightsquigarrow b$ denotes that an element a never occurs after the element b [59]. We thus impose the constraint that given, an alignment for a time measure $\sigma = \{\pi_i \sim m_i, \pi_j \sim m_j\}$, it must hold that $m_i \rightsquigarrow m_j$.

Non-trivial fractions. For fraction measures, we make sure that the obtained result is not a trivial measure, i.e. that the defined measure does not always yield 1.0. For this reason, we ensure that the numerator slot π_n and denominator slot π_d are not aligned to the same model element. Therefore, given an alignment $\sigma = \{\pi_n \sim m_n, \pi_d \sim m_d\}$, it must hold that $m_n \neq m_d$.

Computable fractions. Lastly, we ensure that a fraction measure can actually be computed, instead of resulting in a divide-by-zero error. Therefore, we must ensure that the denominator slot is filled, even for those cases where a PPI description does not explicitly mention a denominator. This can, for example, be seen for $PPI4$, which simply specifies "*the percentage of rejected orders*", without specifying a denominator. In these cases, we align the slot π_d by default to a process concept that represents a more coarse-granular version of the process concept aligned to the numerator slot π_n. There are two possibilities for such default numerators. If pi_n is aligned to a data object with a specific status, e.g. "*order [updated]*", we align pi_d to the data object without the status specification, e.g. the "*order*" data object, if any. As such, we obtain a fraction measure that divides the number orders that have been *updated* by the *total* number of orders. For all other alignments of pi_n, we align the denominator by default to the start event(s) of a process. In this way, the denominator corresponds to the total number of process instances. For example, $PPI4$ will then divide the number of *rejected orders* by the total number of *received orders.*

We combine these constraints with semantic similarity scores. Therefore, we set out to obtain an alignment that has the highest possible sum of semantic similarity scores for the correspondences in σ, as long as σ abides to the alignment constraints Γ. The alignments obtained in this way, in combination with the domain value resolution of the other slot types, then provide the final result of our transformation approach: a filled-in PPI template.

4.3. Operationalization and Extensibility

The transformation approach described in this section can be regarded as an extensible framework for the transformation of natural language PPI descriptions into measurable indicators. In this paper, we have so far described an instantiation of this framework that covers the most common types of PPIs according to insights obtained from practice [3, 5, 10]. A prototypical implementation of the approach, which we also employ in the quantitative evaluation of Section 5, is made publicly available.[2] This approach comes with a pre-trained HMM and is, therefore, ready to use to transform PPIs that suit the currently used set of templates. However, should users desire to extend our approach, for example, by incorporating different measure types, this can be achieved in a straightforward manner.

An extension requires the specification of new (or adapted) PPI templates and a re-training of the HMM. By designing new PPI templates in accordance to a structured PPI notation, such as

[2]Download from: www.hanvanderaa.com/downloads/ppi-transformation

the PPINOT metamodel [3], it can be ensured that also the values of these newly covered measure types can be computed in an automated manner. Once a new template has been defined, the HMM must be adapted to be able to generate PPI definitions in accordance to this template. To this end, tags must first be defined to cover newly introduced semantic concepts. As previously explained in Section 4.1.1, two tags must be defined for each new semantic concept: (i) a tag for the semantic concept itself and (ii) a tag for an indicator of the concept. Once the adapted tag set has been defined in accordance with these guidelines, several additional PPI descriptions must be annotated with these tags in order to retrain the HMM to incorporate the newly defined measure type.

The amount of new training data required depends on the (expected) variety of linguistic patterns that can be used to describe the measures. As the evaluation in the next section illustrates for fraction measures, sometimes only a handful of annotated PPI descriptions suffices. Two approaches are possible to train the HMM. In case an available training set $\mathcal{T}$ is fully annotated, i.e. all PPI description in the training set are annotated, the models $P(M)$ and $P(W|M)$ can be learned by simply counting [33]. For instance, we can learn the probability that a TEE tag follows a TMI tag by taking the number of descriptions in $\mathcal{T}$ that contains the subsequence $< TMI, TEE >$ divided by the total number of descriptions in $\mathcal{T}$ containing $< TMI >$. In case $\mathcal{T}$ is only partially annotated, estimation algorithms can be used to compute the probability distributions. The most commonly used are *forward-backward* algorithms, such as the Baum-Welch method (see e.g. [49]). Due to the ability of these methods to work with partially annotated data, it is possible to add large amounts of additional training data, without the need to annotate them all.

Through similar adaptations, the approach can be extended in other ways. For example, new aggregation functions can be included, the functionality of group-by properties can be extended, or the HMM can be trained to work on specific application domains.

5. Evaluation

To demonstrate the capabilities of our transformation approach, we conduct a quantitative evaluation by comparing automatically generated structured PPI definitions to a manually created *gold standard.* The goal of this evaluation is to learn how well the automated approach approximates manual transformations of PPI descriptions. If our transformation approach can automatically generate high quality PPI definitions, i.e. definitions that closely resemble those created manually by experts, our approach can be regarded as a viable and efficient alternative to an otherwise time-consuming manual endeavor. To assess this potential, we use a test collection consisting of 129 PPIs obtained from industry. Both the data collection and prototypical implementation used in this evaluation are made publicly available.[3]

5.1. Test Collection

To evaluate our approach, we use a collection of process models and accompanying natural language PPI descriptions from practice. Part of the test collection consists of an industrial data set stemming from prior research on the formalization of PPI definitions and service level agreements [5, 9]. In particular, this set consists of processes from three different organizations: (i) a healthcare institute, (ii) a university, and (iii) a telecommunications provider. This data collection was originally provided in Spanish, but translated to English in collaboration with the respective industrial partners for the, aforementioned, earlier research projects. We augmented the industrial data set with a number of process models and PPIs from the SCOR (Supply Chain Operations Reference) reference framework. From the SCOR framework, we selected processes with a high number of associated performance indicators per process and a considerable complexity of the associated process model. The selected processes cover different aspects of the logistics domain: procurement, production, and delivery. The PPI descriptions obtained in this manner were not altered with respect to their contents. The set of descriptions varies greatly with regard to their language and structure. Due to these varying characteristics in combination with the different nature of the organizations from which they were obtained, we believe that the collection is well-suited to achieve a high external validity of the results.

We had to exclude nine PPI descriptions from the original data collection. These PPI descriptions could not be manually transformed into a structured notation based on the contents of the available process models, i.e. their computation required information that is not conveyed in the

[3]Download from: www.hanvanderaa.com/downloads/ppi-transformation

Table 7: Overview of the test collection

Source	P	E	PPIs	Count	Time	Data	Frac.	Aggr.	Group-by
Industry	10	12.5	47	25	20	1	7	12	8
SCOR	3	8.3	76	27	21	24	4	39	1
Total	13	11.5	129	52	41	25	11	51	9

model. The resulting test collection consists of 12 process models with a total of 129 associated PPIs. Table 7 presents an overview of the further characteristics of the collection. This table shows the average number of elements (activities and events) per process model E, the number of PPIs per type, and the number of PPIs that are associated with an aggregation function (aggr.) or group-by property.

5.2. Setup

To conduct the evaluation, we implemented the presented transformation approach in the form of a Java prototype. The prototype uses the Stanford CoreNLP toolkit [55] for the tokenization of PPI descriptions and the semantic similarity implementation DISCO [57] to calculate second order similarity scores. We use this prototype to generate structured definitions for the PPI descriptions included in the test collection.

We compare the generated definitions to a manually created *gold standard*. For the creation of this gold standard, we mostly built on formalized definitions for the industrial PPIs that were created in the context of earlier work [5, 11]. For the remaining processes, we let two researchers independently establish a gold standard. This yielded an inter-annotator agreement of 0.97. The five discrepancies between the two gold standards were resolved through a discussion.

To perform the comparison between the generated definitions and the gold standard, we computed the well-known *precision* and *recall* metrics. We used $\mathcal{A}$ to denote the set of slots filled by our approach and $\mathcal{R}$ for the slots filled in the gold standard.[4] *Precision* then reflects the fraction of slots that our automated approach filled correctly according to the gold standard. *Recall* represents the fraction of slots filled in the gold standard that were also correctly filled by our

[4]Note that we exclude *null* values assigned to the optional aggregation function and group-by property slots from consideration.

approach. Equations 4 and 5 provide the formal definitions of the metrics. Furthermore, we report the f_1-score as the harmonic mean of precision and recall.

$$prec(\mathcal{A}, \mathcal{M}) = \frac{|\mathcal{A} \cap \mathcal{R}|}{|\mathcal{A}|} \quad (4) \qquad rec(\mathcal{A}, \mathcal{M}) = \frac{|\mathcal{A} \cap \mathcal{R}|}{|\mathcal{R}|} \quad (5)$$

Since the presented work provides the first transformation approach for PPI descriptions, there is no established benchmark against we can compare its performance. To be able to still provide some benchmark, we compare the performance of our approach to earlier work presented in [11]. That work uses a heuristic-based three-step approach to establish alignments between a PPI description and the elements of a process model. Although our transformation approach addresses a wider problem, both approaches establish links between PPI descriptions and process model elements. Therefore, the work from [11] provides a suitable benchmark for one of the most important parts of our approach. We refer to the results obtained by the existing approach, on the same data collection, as the *baseline*.

To compute the evaluation results, we train our approach on a part of the PPI collection, referred to as the *training set*, and apply it on the remainder of the data collection, the *test set*. To avoid any bias in the result due to sampling, we perform a repeated *k-fold cross-validation*. In k-fold cross-validation, a data set $\mathcal{D}$ is randomly split into k mutually exclusive subsets (i.e. folds), $\mathcal{D}_1, \ldots, \mathcal{D}_k$ of approximately equal size [60]. In each experiment run, our approach is then tested k times. Each time $t \in \{1, 2, \ldots, k\}$ we trained the HMM on $\mathcal{D} \backslash \mathcal{D}_t$ and tested it on $\mathcal{D}_t$. By repeating the cross-validation with different random splits of the data set, we can learn how much the results are affected by a particular partitioning of the data collection.

The repeated cross-validation furthermore enables the use of statistical tests to compare the distribution of the results achieved by our transformation approach to the baseline. We employ the well-known *Kolmogorov-Smirnov* test (see e.g. [61]) for this purpose. Let $F_{t,n}$ and $F_{b,n'}$ be the distribution functions of the results obtained by the cross-validation for, respectively, our approach and the baseline. The Kolmogorov-Smirnov test evaluates the null hypothesis that $F_{t,n}$ and $F_{b,n'}$ are equal, i.e. that there is no statistical significant difference between the results obtained using our approach and the baseline.

5.3. Results

In this section, we present the results of our evaluation experiments. Section 5.3.1 first gives an overview of the results. Section 5.3.2 then provides details about how our approach compares to the baseline. Section 5.3.3 finally discusses the challenges our approach faces in the context of a post-hoc analysis.

5.3.1. Overview

We used our prototype to conduct a k-fold cross-validation with $k = 10$, which we repeated 30 times. We performed this cross-validation for the *industry* and *SCOR* data collections separately, as well as for the combined collection. Table 8 summarizes these results for our approach and the baseline. It shows that our transformation approach performs well, obtaining an average F_1-score of 0.85. The approach achieves an average precision of 0.89, ranging between 0.78 for group-by properties and 0.93 for aggregation functions. The average recall obtained by the approach is 0.82, ranging from 0.72 for the alignment of process concepts and 0.96 for the identification of aggregation functions. From the low observed standard deviations (0.01 for the entire set of slots), we can learn that the performance of the approach is stable in the context of this data set. Furthermore, we observe that the alignment of process concepts differs considerably between the two data collections. For the other slot types, the average performance is comparable.

Lastly, it is interesting to consider that our approach achieved a fully correct transformation for 67% of the PPIs in the data collection. Furthermore, the transformation approach returns at most one incorrect slot filler for a total of 86% of the PPIs.

5.3.2. Baseline Comparison

Comparing the performance of our approach against the baseline, the results show that our transformation outperforms the baseline approach from [11]. The Kolmogorov-Smirnov test reveals that these results are significant ($p < 0.01$) for all metrics and across both applicable slot types. With respect to the identification of *measure types*, our approach achieved precision and recall scores of 0.92, whereas the baseline obtained scores of 0.87 for both metrics. This difference can be attributed to the more error-prone, keyword-based classification method used by the baseline approach. This classification method can make mistakes, because it looks at terms in isolation and does not consider the position of terms in the PPI description. By contrast, our approach considers the semantics of the entire PPI description, i.e. all terms and their positions, rather than focusing on individual words for the classification. Similar to the identification of measure types, our approach outperforms the baseline with respect to slots containing *process concepts*. Our approach achieves an F_1-score of 0.79, whereas the baseline approach achieves an F_1-score of 0.69. This difference can mainly be attributed to the usage of an HMM for the semantic annotation step. The HMM represents a parser that is specifically tailored to deal with PPI descriptions. Therefore, it can identify these chunks of a PPI description that describe process concepts with high accuracy. Since the baseline approach uses a more generic parser (i.e. a general parser for grammatical structures), it fails to achieve the same level of accuracy.

Table 8: Evaluation results

		Baseline			**Approach**		
Data source	**Slot type**	**prec.**	**rec.**	**F_1**	**prec.**	**rec.**	**F_1**
Industry	Measure type	.82	.82	.82	.90	.90	.90
	Process concepts	.63	.59	.61	.67	.63	.65
	Aggregators	-	-	-	.92	1.00	.97
	Group-by	-	-	-	.74	.74	.74
	Total	.71	.68	.69	.78	.76	.77
SCOR	Measure type	.87	.87	.87	.94	.94	.94
	Process concepts	.85	.67	.75	1.00	.80	.89
	Aggregators	-	-	-	1.00	1.00	1.00
	Group-by	-	-	-	–	–	–
	Total	.88	.77	.82	.98	.88	.93
Full collection	Measure type	.87	.87	.87	.92	.92	.92
	Process concepts	.75	.63	.69	.87	.72	.79
	Aggregators	-	-	-	.93	.96	.94
	Group-by	-	-	-	.78	.78	.78
	Total	.79	.71	.74	.89	.82	.85

5.3.3. Post-hoc Analysis

The quantitative evaluation shows that our transformation approach achieves a high result accuracy. A post-hoc analysis of these results reveals that the approach faces two main types of challenges. One challenge corresponds to the semantic annotation step and the other to the domain value resolution step of our approach.

The usage of HMMs for *semantic annotation* enables our approach to deal with a broad variety of linguistic patterns, even if a PPI description contains previously unseen terms or syntactic constructs. However, the parser can produce incorrect annotations if unseen terms play a prominent role in a PPI description. For instance, the collection of PPI descriptions from the SCOR framework contains a single PPI description that uses the term "*leadtime*" in a time measure. Because this is a unique occurrence in the used data collection, the HMM parser does not recognize this important term. Therefore, it incorrectly annotates this PPI description as a count measure, which results in an incorrect measure type prediction. Furthermore, due to the semantic difference between count and time measures, it also results in an incorrect alignment of process concepts. A count measure refers to only a single process event, whereas a time measure requires both a start and an end event. Therefore, the misclassification will also lead to at least one event not being correctly aligned. Still, such cases are rare and their occurrences can be mitigated by extending the data collection used to train the HMM.

The *domain value resolution* step of our approach has to deal with the highly complex task of identifying the event that a chunk of text describes. The main challenge here is that certain correspondences depend on context-specific information for their identification. As an illustration, consider a PPI description from the industrial collection: "*the elapsed time between the technician arrival to headquarters and the closure of the intervention*". The start event of this description corresponds to the chunk "*the technician arrival to headquarters*", as is correctly identified in the parsing step. However, identifying the correct process concept for this chunk is far from trivial. The process model accompanying this PPI does not contain any activity or event that describes an "*arrival*" or "*headquarters*". Instead, the event corresponds to the start of an activity labeled "*Perform field intervention*". To identify this correspondence correctly, background knowledge is required to establish the connection between the "*arrival of a technician*" and the start of a "*field intervention*". Our automated approach does not take such domain knowledge into account and, in some cases, identifies incorrect alignments between the PPI description and a process model. From the results described in Table 8, it becomes clear that such cases are particularly present in the industrial data collection.

Despite these challenges, the evaluation results demonstrate that the PPI definitions generated by our automated approach closely approximate PPI definitions manually created by experts. Therefore, we can conclude that our automated approach presents an efficient alternative for an otherwise highly tedious, manual task.

6. Limitations

Our quantitative evaluation demonstrates that our approach achieves promising results in practical settings. However, these results need to be considered against the background of certain limitations. In particular, we identify limitations related to the *transformation approach* and limitations related to the *evaluation* of the approach.

Regarding the transformation approach itself, we identify two limitations. First, our transformation approach provides an automated alternative to a highly complex task. As a result, the PPI definitions the approach generates are accurate, but not one hundred percent perfect. If desired, inaccuracies can be manually resolved by experts. This arguably requires less manual effort than it takes to manually define all PPIs from scratch. As such, our transformation approach allows users to trade-off between invested time and effort versus result quality. Second, it has to be considered that our transformation currently is not able to detect when it is dealing with a PPI description that it cannot transform based on the available information. Such a situation can occur if the PPI describes a measure type that is currently not included in the set of templates or if the PPI description requires information that is not captured in a process model. The latter case was observed for the nine PPI descriptions that we had to exclude in the evaluation.

The presented quantitative evaluation results are bound to the specifics of the employed data collection. This data collection is not representative in a statistical sense. In fact, the creation of a statistically representative sample is hardly feasible due to the variability of PPI descriptions. This variability manifests itself in two manners. First, the utilized data collection is not guaranteed to provide a statistically representative coverage of the types of measures that exist in practical scenarios. As a result, the included templates do not cover all imaginable PPIs, but rather the ones most observed in our empirical studies. As stated earlier, the impact of this limitation is diminished by the extensibility of our approach. Second, within each measure type, the data collection presents a sample of the natural language patterns that can be used to specify measures of this type. It is possible that in other organizations, measures are described in different ways.

These factors should be taken into account when interpreting the specified results. Still, we aimed to compose a data collection that is as heterogeneous as possible by obtaining data from various sources. Therefore, we are confident that our evaluation indeed shows a realistic picture of the performance of our approach in practice.

7. Conclusion

In this paper, we presented an approach for the automated transformation and alignment of natural language descriptions of PPIs. Our approach takes as input a natural language PPI description and produces a structured, template-based notation of a PPI of which the value can be computed automatically. To achieve this, the approach builds on HMMs as a linguistic parser to identify the parts of a PPI description that correspond to slots in a PPI template. Then, we use semantic similarity measures and semantic constraints to fill the slots with the appropriate values belonging to particular domains. We evaluated the performance of our approach with a set of real-world PPI descriptions and accompanying process models obtained from various industrial sources. The evaluation revealed that the structured PPI definitions generated by our approach are a good approximation of those created by manual experts. Furthermore, the evaluation demonstrated that our transformation approach significantly outperforms an existing approach, which also tackled a problem with a more limited scope. Therefore, our approach represents a viable, automated alternative to an otherwise highly laborious and time-intensive, manual task.

In future work, we aim to pursue two main directions. First, we intend to improve the accuracy and coverage of our transformation by addressing the aforementioned limitations. For this we can exploit the extensibility of our approach. If data on other measure types is available, the approach can be extended by simply introducing additional semantic tags and, optionally, alignment constraints. A second promising direction is to find the process model related to a PPI description or vice versa. This could, for instance, be used to develop a querying technique that automatically identifies the PPIs that are relevant for a certain business process.

Bibliography

[1] A. Grosskopf, G. Decker, M. Weske, The process: business process modeling using BPMN, Meghan Kiffer Press, 2009.

[2] A. Kronz, Managing of process key performance indicators as part of the ARIS methodology, in: Corporate performance management, Springer, Heidelberg, Germany, 2006, pp. 31–44.

[3] A. del Río-Ortega, C. Cabanillas, M. Resinas, A. Ruiz-Cortés, Ppinot tool suite: a performance management solution for process-oriented organisations, in: International Conference on Service-Oriented Computing, Springer, Heidelberg, Germany, 2013, pp. 675–678.

[4] B. Wetzstein, Z. Ma, F. Leymann, Towards measuring key performance indicators of semantic business processes, in: International Conference on Business Information Systems, Springer, 2008, pp. 227–238.

[5] A. del Río-Ortega, M. Resinas, C. Cabanillas, A. Ruiz-Cortés, On the definition and design-time analysis of process performance indicators, Information Systems 38 (4) (2013) 470–490.

[6] G. Lami, S. Gnesi, F. Fabbrini, M. Fusani, G. Trentanni, An automatic tool for the analysis of natural language requirements, Tech. rep., Informe técnico, CNR Information Science and Technology Institute (2004).

[7] F. Franceschini, M. Galetto, D. Maisano, Management by measurement: Designing key indicators and performance measurement systems, Springer, Heidelberg, Germany, 2007.

[8] V. Popova, A. Sharpanskykh, Modeling organizational performance indicators, Information Systems 35 (4) (2010) 505–527.

[9] A. del Río-Ortega, A. M. Gutiérrez, A. Durán, M. Resinas, A. Ruiz-Cortés, Modelling service level agreements for business process outsourcing services, in: Advanced Information Systems Engineering, Springer, 2015, pp. 485–500.

[10] A. del Río-Ortega, M. Resinas, A. Durán, A. Ruiz-Cortés, Using templates and linguistic patterns to define process performance indicators, Enterprise Information Systems 10 (2) (2016) 159–192.

[11] H. van der Aa, A. del Río-Ortega, M. Resinas, H. Leopold, A. Ruiz-Cortés, J. Mendling, H. A. Reijers, Narrowing the business-IT gap in process performance measurement, in: Advanced Information Systems Engineering, Springer, Heidelberg, Germany, 2016, pp. 543–557.

[12] M. Rosemann, Potential Pitfalls of Process Modeling: Part A, Business Process Management Journal 12 (2) (2006) 249–254.

[13] M. Weidlich, J. Mendling, M. Weske, Propagating changes between aligned process models, Journal of Systems and Software 85 (8) (2012) 1885–1898.

[14] B. Marshall, H. Chen, T. Madhusudan, Matching knowledge elements in concept maps using a similarity flooding algorithm, Decision Support Systems 42 (3) (2006) 1290–1306.

[15] M. Selway, G. Grossmann, W. Mayer, M. Stumptner, Formalising natural language specifications using a cognitive linguistic/configuration based approach, Information Systems 54 (2015) 191–208.

[16] C. B. Achour, Guiding scenario authoring, Information Modelling and Knowledge Bases X 51 (1999) 152–171.

[17] H. van der Aa, H. Leopold, F. Mannhardt, H. A. Reijers, On the fragmentation of process information: Challenges, solutions, and outlook, in: Enterprise, Business-Process and Information Systems Modeling, Springer, Heidelberg, Germany, 2015, pp. 3–18.

[18] R. S. Kaplan, D. P. Norton, The balanced scorecard: measures that drive performance, Harvard Business Review 83 (7) (2005) 172.

[19] P. Brewer, T. Speh, Using the balance scorecard to measure supply chain performance, Journal of Business Logistics 21 (2000) 75–93.

[20] F. Chan, Performance measurement in a supply chain, International Journal of Advanced Manufacturing Technology 21 (2003) 534–548.

[21] E. Krauth, H. Moonen, V. Popova, M. C. Schut, Performance measurement and control in logistics service providing, in: International Conference on Enterprise Information Systems, 2005, pp. 239–247.

[22] G. Vaidyanathan, A framework for evaluating third-party logistics, Communications of the ACM 48 (1) (2005) 89–94.

[23] B. Korherr, B. List, Extending the EPC and the BPMN with business process goals and performance measures, in: International Conference on Enterprise Information Systems, 2007, pp. 287–294.

[24] C. Momm, R. Malec, S. Abeck, Towards a model-driven development of monitored processes, in: Wirtschaftsinformatik (2), 2007, pp. 319–336.

[25] C. Pedrinaci et al., Sentinel: a semantic business process monitoring tool, in: International Workshop on Ontology-Supported Business Intelligence, 2008, pp. 26–30.

[26] C. Costello, O. Molloy, Building a process performance model for business activity monitoring, in: Information Systems Development, Springer, Heidelberg, Germany, 2009, pp. 237–248.

[27] O. González, R. Casallas, D. Deridder, MMC-BPM: A Domain-Specific Language for Business Processes Analysis, Business Information Systems 21 (2009) 157–168.

[28] J. Saldivar, C. Vairetti, C. Rodrguez, F. Daniel, F. Casati, R. Alarcn, Analysis and improvement of business process models using spreadsheets, Information Systems 57 (2016) 1 – 19.

[29] D. Jurafsky, J. H. Martin, Speech & language processing, Pearson Education India, 2000.

[30] G. Tur, D. Hakkani-Tür, L. Heck, What is left to be understood in ATIS?, in: Spoken Language Technology Workshop, IEEE, 2010, pp. 19–24.

[31] Y. Wang, A. Acero, Combination of CFG and n-gram modeling in semantic grammar learning, in: Eurospeech, International Speech Communication Association, 2003, pp. 2809–2812.

[32] A. L. Gorin, G. Riccardi, J. H. Wright, How may I help you?, Speech communication 23 (1) (1997) 113–127.

[33] Y. Wang, L. Deng, A. Acero, Spoken language understanding, Signal Processing Magazine 22 (5) (2005) 16–31.

[34] S. Miller, R. Bobrow, R. Ingria, R. Schwartz, Hidden understanding models of natural language, in: Proceedings of the 32nd annual meeting on Association for Computational Linguistics, Association for Computational Linguistics, 1994, pp. 25–32.

[35] D. Bikel, V. Castelli, R. Florian, D.-j. Han, Entity linking and slot filling through statistical processing and inference rules, in: Proceedings of the Text Analysis Conference, 2009.

[36] F. Giunchiglia, P. Shvaiko, M. Yatskevich, Semantic matching, in: Encyclopedia of Database Systems, Springer, Heidelberg, Germany, 2009, pp. 2561–2566.

[37] A. Gal, Uncertain schema matching, Synthesis Lectures on Data Management 3 (1) (2011) 1–97.

[38] H.-H. Do, E. Rahm, Matching large schemas: Approaches and evaluation, Information Systems 32 (6) (2007) 857–885.

[39] L. Po, S. Sorrentino, Automatic generation of probabilistic relationships for improving schema matching, Information Systems 36 (2) (2011) 192–208.

[40] J. Euzenat, P. Shvaiko, et al., Ontology matching, Springer, Heidelberg, Germany, 2007.

[41] S. Raunich, E. Rahm, Target-driven merging of taxonomies with Atom, Information Systems 42 (2014) 1–14.

[42] U. Cayoglu, R. M. Dijkman, M. Dumas, P. Fettke, L. Garcıa-Banuelos, P. Hake, C. Klinkmüller, H. Leopold, A. Ludwig, P. Loos, et al., The process model matching contest 2013, in: 4th International Workshop on Process Model Collections: Management and Reuse, Springer, Heidelberg, Germany, 2013, pp. 442–462.

[43] M. Weidlich, E. Sheetrit, M. C. Branco, A. Gal, Matching business process models using positional passage-based language models, in: International Conference on Conceptual Modeling, Springer, 2013, pp. 130–137.

[44] R. M. Dijkman, M. Dumas, B. F. Van Dongen, R. Käärik, J. Mendling, Similarity of business process models: Metrics and evaluation, Information Systems 36 (2) (2011) 498–516.

[45] M. Künze, M. Weidlich, M. Weske, Behavioral similarity–a proper metric, in: International Conference on Business Process Management, Springer, Heidelberg, Germany, 2011, pp. 166–181.

[46] T. Baier, J. Mendling, M. Weske, Bridging abstraction layers in process mining, Information Systems 46 (2014) 123–139.

[47] H. van der Aa, H. Leopold, H. A. Reijers, Detecting inconsistencies between process models and textual descriptions, in: International Conference on Business Process Management, Springer, Heidelberg, Germany, 2015, pp. 90–105.

[48] H. van der Aa, H. Leopold, H. A. Reijers, Comparing textual descriptions to process models: The automatic

detection of inconsistencies, Information Systems 64 (2017) 447–460.

[49] L. R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.

[50] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, R. L. Mercer, The mathematics of statistical machine translation: Parameter estimation, Computational linguistics 19 (2) (1993) 263–311.

[51] J. Kupiec, Robust part-of-speech tagging using a hidden markov model, Computer Speech & Language 6 (3) (1992) 225–242.

[52] M. Mohri, Finite-state transducers in language and speech processing, Computational linguistics 23 (2) (1997) 269–311.

[53] G. D. Forney Jr, The viterbi algorithm, Proceedings of the IEEE 61 (3) (1973) 268–278.

[54] C. D. Manning, P. Raghavan, H. Schütze, Introduction to information retrieval, Vol. 1, Cambridge university press Cambridge, 2008.

[55] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, D. McClosky, The Stanford CoreNLP natural language processing toolkit, in: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014, pp. 55–60.

[56] R. Mihalcea, C. Corley, C. Strapparava, Corpus-based and knowledge-based measures of text semantic similarity, in: Proceedings of the 21st National Conference on Artificial Intelligence, Vol. 6, 2006, pp. 775–780.

[57] P. Kolb, DISCO: A multilingual database of distributionally similar words, in: Proceedings of KONVENS, 2008.

[58] A. Islam, D. Inkpen, Second order co-occurrence PMI for determining the semantic similarity of words, in: International Conference on Language Resources and Evaluation, 2006, pp. 1033–1038.

[59] M. Weidlich, J. Mendling, M. Weske, Efficient consistency measurement based on behavioral profiles of process models, IEEE Transactions on Software Engineering 37 (3) (2011) 410–429.

[60] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, pp. 1137–1145.

[61] N. M. Razali, Y. B. Wah, Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests, Journal of Statistical Modeling and Analytics 2 (1) (2011) 21–33.