Aligning Textual and Model-Based Process Descriptions

Josep Sànchez-Ferreres[a,*], Han van der Aa[b], Josep Carmona[a], Lluís Padró[a]

[a] *Universitat Politècnica de Catalunya, Barcelona, Spain*
[b] *Humboldt University of Berlin, Berlin, Germany*

Abstract

Process model descriptions are an ubiquitous source of information that exists in any organization. To reach different types of stakeholders, distinct descriptions are often kept, so that process understandability is boosted with respect to individual capabilities. While the use of distinct representations allows more stakeholders to interpret process information, it also poses a considerable challenge: to keep different process descriptions aligned. In this paper, a novel technique to align process models and textual descriptions is proposed. The technique is grounded on projecting knowledge extracted from these two representations into a uniform representation that is amenable for comparison. It applies a tailored linguistic analysis of each description, so that the important information is considered when aligning description' elements. Compared to existing approaches that address this use case, our technique provides more comprehensive alignments, which encompass process model activities, events, and gateways. Furthermore, the technique, which has has been implemented into the platform `nlp4bpm.cs.upc.edu`, shows promising results based on experiments with real-world data.

Keywords:
Business process management; process models; natural language processing; alignments.

*Corresponding author. *Phone number:* 934137861
Email addresses: `jsanchezf@cs.upc.edu` (Josep Sànchez-Ferreres), `han.van.der.aa@hu-berlin.de` (Han van der Aa), `jcarmona@cs.upc.edu` (Josep Carmona), `padro@cs.upc.edu` (Lluís Padró)

1. Introduction

Organizational processes can be highly complex chains of inter-related steps, involving numerous stakeholders with various roles [52]. Due to this complexity, it is crucial that the coordination among process actors is well defined [30]. Therefore, having access to the right information on business processes is vital to their proper execution [5] and their compliance to rules and regulations [1]. To provide various stakeholders with the information that they need, organizations have recognized the value of capturing process descriptions in model-based as well as text-based representations [22, 33, 50]. The reason for maintaining both representation forms is that they each have their merits. Process models have been found to be better suited to express complex execution logic of a process in a more comprehensive manner than natural language [29]. By contrast, some stakeholders, especially workers who actually execute the process, have difficulties reading and interpreting process models and, therefore, prefer textual process descriptions over process models [7].

Despite these benefits, the usage of multiple descriptions of the same process can also lead to considerable difficulties. In particular, it is vital that the process information contained in different formats is correct, even when these formats are maintained independently from each other [50]. If users access inaccurate process descriptions, they can develop different expectations about what a process aims to establish or how it should be executed [48]. Such situations can have negative effects on the efficiency with which processes are executed and, furthermore, can lead to business process non-compliance [50]. A problem in this regard is, however, that processes are subject to continuous change [58] and, therefore, considerable manual effort is required to maintain process descriptions and clear up any conflicts. Given that organizations can have hundreds or even thousands of different different process models [36], this means that manually maintaining multiple representations for all processes is hardly manageable.

In this paper, we present an alignment approach that supports organizations in maintaining process information in both textual and model-based representa-

tions. Our approach aims to establish alignments between both representations by identifying *correspondences* between parts of a textual process description and elements in a process model. These alignments enable the identification of discrepancies between descriptions [48] and, furthermore, provide a starting point for the propagation of changes from one description to the other [55]. A quantitative evaluation demonstrates that our proposed approach outperforms alignment approaches previously developed by the authors [48, 37]. Furthermore, because our approach also identifies correspondences involving process model events and gateways, the alignments we obtain are also more comprehensive. Therefore, our approach provides an important foundation for the maintenance of process information in different representation formats.

The remainder of this paper is structured as follows. Section 2 provides necessary background information in the form of a problem illustration and discussion of related work. Section 3 presents our proposed alignment approach. Section 4 presents and discusses the results of a quantitative evaluation. Finally, Section 5 concludes the paper.

2. Background

This section discusses background information relevant to the alignment of textual and model-based process descriptions. In particular, Section 2.1 presents a running example and discusses the main challenges associated with the alignment task. Section 2.2 provides an overview of related work.

2.1. Problem Illustration

To illustrate the challenges that are associated with the alignment of textual and model-based process descriptions, consider the process model shown in Figure 1 and the textual process description contained in Table 1.

The process model is defined using the Business Process Model and Notation (BPMN), a standard notation for process models. The model contains nine *activities*, depicted as rounded rectangles. These activities denote the main tasks

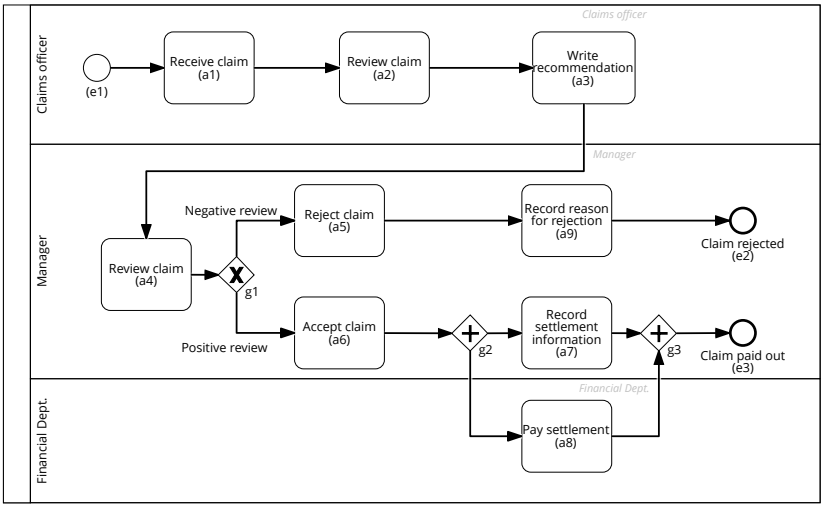

Figure 1: Exemplary process model of a claims handling process in BPMN.

performed in order to execute the process. The process model, furthermore, contains three *events*, illustrated by labeled circles, which describe the start and end points of the process. The directed edges connecting activities and events denote the control-flow of the process.The diamond shapes in the process model indicate special routing constructs in the control-flow, called *gateways*. Gateways with an X indicate a process choice, e.g. after $g1$, a claim can be either *rejected* (activity $a5$) or *accepted* ($a6$). By contrast, gateways with a + symbol indicate parallel or concurrent execution patterns. This means that the activities $a7$ and $a8$ can be executed at the same time. Finally, the horizontal lines denote so-called *swimlanes* in the process model. These swimlanes indicate which resource roles are involved in the execution of specific process steps.

The goal of aligning a process model and a textual process description is to identify correspondences between the elements of the process model and parts of the textual process description. In the context of this paper, we set out to align all elements that describe the flow in a process model, i.e. we align *activities*, *events*, and *gateways*. Our goal is here to identify the sentences of a textual process description that correspond to these process model elements,

Table 1: Textual process description of the claims handling process with correspondences to the model from Figure 1.

ID	Sentence	Corresp.
s_1	After receipt of a claim, a claims officer reviews the request.	$a1$, $a2$
s_2	Then, the claims officer writes a settlement recommendation and forwards it.	$a3$
s_3	A manager reviews the claim based on the written recommendation.	$a4$
s_4	If the review is negative, the claim is denied, otherwise it is accepted.	$g1$, $a5$, $a6$
s_5	In case of rejection, the process completes here.	$e2$
s_6	In case of acceptance, the manager records the settlement information, while the financial department pays the claimant.	$g2$, $a7$, $a8$

referred to as *correspondences*. The right-most column in Table 1 indicates these correspondences between the model and text.

To establish such alignments, several challenges must be overcome. These challenges are primarily caused by the flexibility of natural language, that allows the expression of the same concept via a large variety of words or phrases. In particular, we identified the following four main challenges during our earlier works on the development of alignment approaches [37, 48]:

1. *C1: Different grammatical structures:* Textual process descriptions can use a broad variety of grammatical structures to describe a process. As a result, there can exist considerable differences between the way in which a text and a model describe similar aspects of a process. Consider, for instance, the first phrase in sentence s_1, "*After receipt of a claim*", and the corresponding model activity "*receive claim*". The former uses a noun-based structure, whereas the later used a verb-based description of the same task. Therefore, an alignment technique must be able to detect such correspondences despite the presence of grammatical differences.

2. *C2: Different terminology:* Next to differences in grammatical structures, there can exist considerable differences in the terminology used between model and text. Consider, for instance, sentence s_4, which refers to a claim being "*denied*" and activity $a4$, "*reject claim*". An alignment technique must be able to detect that the sentence and activity refer to the same step in a process. It is here important to note that terminological differences play an even bigger role when there are also differences in the level of detail used by the two types of descriptions [46].
3. *C3: Activities with identical labels:* A different challenge occurs when process models use activities with similar or even identical labels. This can, for instance, happen in larger processes with multiple actors or when the labels used in a model are fairly coarse-granular. Activities $a2$ and $a4$ provide an example of this, because they both have the label "*Review claim*". When establishing alignments, an alignment should be able to recognize that these identically labeled activities refer to process steps that are performed in different parts of the process, either by different actors (i.e., $a2$ by a *claims officer* and $a4$ by a *manager*), or even by the same actor but in different contexts.
4. *C4: Partial alignments:* Finally, it is important to recognize that a process model and textual description may not describe exactly the same steps that comprise a process, whether intentional or not [48]. For instance, activity $a9$, "*Record rejection reason*", does not have a corresponding sentence in the textual description and should, therefore, not be part of a correspondence. As a result, to produce a correct alignment, alignment techniques must also be able to detect when certain process model elements actually do not appear in the text. Those process model elements should, therefore, not be included in any correspondence.

These challenges illustrate the complexity associated with the alignment of textual process descriptions and process models. To overcome these challenges, we build on techniques from the areas of *natural language processing* and *match-*

ing, as discussed next.

2.2. Related Work

In this section we discuss how natural language processing (NLP) is applied in the context of Business Process Management (Section 2.2.1) and discuss various alignment approaches that exist in this context (Section 2.2.2).

2.2.1. Natural Language Processing in Business Process Management

NLP techniques are applied to address a variety of use cases in the context of business process management [43]. Several of these focus on the text contained in process models themselves. This includes a variety of works that focus on the quality of process model labels, for example by detecting violations of labeling conventions [3, 20, 53], inconsistent use of terminology [18], or common modeling errors [16]. Other approaches use NLP to augment process models with semantic or ontological information [21, 12, 4].

Other use cases involve texts that exist outside of process models. Several approaches extract process models from different kinds of text, such as from use cases [41], group stories [8], or methodological descriptions [11], while others take general textual process descriptions as input [15, 13]. However, these approaches have been found to produce inaccurate models, which require extensive manual revision [39]. Other use cases involving texts include a technique that considers *work instructions* when querying process repositories [25] for conformance checking against textual process descriptions [49].

2.2.2. Process Matching

The establishment of alignments between artifacts, often referred to as *matching*, has received considerable attention in the context of BPM. In particular the importance of *process model matching* has been recognized, in which alignments between two process models are established. This has resulted in the development of a considerable number of matching techniques. Nearly all process model matchers focus on the analysis of similarity between the labels of process model elements. To achieve this, techniques consider label similarity from a *syntactic*

perspective [57, 9, 27] as well as from a *semantic* perspective [23, 35, 38]. The former set focuses on how similar the characters used in labels are, whereas the later set focuses on similarity in the meaning of labels. Aside from the analysis of process model labels, existing matching techniques have also recognized the importance *behavioral* or *structural* characteristics [17]. By taking such characteristics into account, matchers, such as [9, 17, 26], are able to recognize if activities occur in the same parts of a process. Other techniques also focus on matching based on other information, such as a technique that take work instructions associated with models into account [57], as well as a technique that matches based on event-log information [44]. A new approach by Meilicke et al. [28] provides a means to create an *ensemble* of various process model matchers that can combine their different strengths.

Aside from techniques that establish alignments between different process models, focus has recently shifted towards the establishment of alignments among a broader range of process-related artifacts. For example, several techniques exist that establish correspondences between *event logs* and process models [2, 40], and a technique for the alignment of process performance indicators and process models [45].

There are several key differences that distinguish the technique proposed in this paper when compared to those existing works. First, our technique establishes more comprehensives alignments between model and text, because the alignments cover activities, events, and gateways, whereas the previous works only focus on the alignment of activities. Second, in order to address challenge C3 described in the previous section, which relates to activities with identical labels, our technique explicitly considers resource-related information. Third, we here encode the constraints into an *Integer Linear Problem (ILP) optimization*, rather than using a *best-first* search algorithm, which greatly improves the computational efficiency of our technique and provides more flexibility with respect to the inclusion of additional constraints and their weights. Finally, our approach applies predictors defined in [48] in a novel manner to detect and adapt to differences between process model and text. This allows us to also

address challenge C4, which implies that alignments between model and text are not always complete.

3. Alignment Approach

This section describes our proposed alignment approach. It takes as input a textual process description and a process model. Our approach is tailored towards graph-based process model notations like BPMN, Petri nets or Event-Driven Process Chains, and on the other hand imposes no restrictions on the structure of the textual process description." Furthermore, these two formats may have been defined and maintained independently from each other. Given this input, the approach aims to establish an optimal alignment between the sentences of the textual description and the elements, i.e., the activities, events, and gateways, of the process model. To achieve this, our approach consists of four main steps, as visualized in Figure 2.

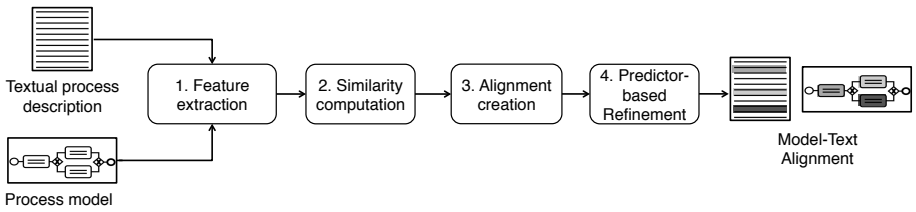

Figure 2: Overview of our alignment approach

For a model-text pair, our approach first aims to extract features that correspond to important process-related and linguistic information from the provided inputs. For example, we extract information about actions, actors, and business objects. Second, based on the features extracted, we set out to quantify the semantic similarity between process model elements and sentences. Third, we combine the semantic similarity scores with ordering information (i.e., information about the process flow) in order to establish an alignment between the textual description and process model. In the fourth and final step, our approach uses so-called predictors to detect if a provided model-text pair is likely

to contain inconsistencies. If this is the case, our approach uses this information to refine the previously established alignment in order to produce the final result.

In the remainder of this section, we describe the steps of our approach in detail.

3.1. Process Information Extraction

The goal of the first step in our approach is to extract process-related information from both a textual process description and a process model. Through this extraction step, we convert process information contained in the two heterogeneous sources into a format that enables their accurate comparison.

3.1.1. Extraction from Process Models

To describe the extraction approach, we first need to define the notion of a process model. Process models can be created using a variety of modeling languages, such as Petri nets, Event-Driven Process Chains (EPCs), and the Business Process Model and Notation (BPMN). The contributions of this paper are independent of the specific notation used to define a process model. Therefore, we define process models using the relevant parts of the generic definition provided in [19, p.13], given in Definition 1.

Definition 1 (Process Model) We define a process model as a tuple $M = (A, E, G, R, , L, N, F, t, \rho, \lambda)$, where:

- A is a finite set of activities,
- E is a finite set of events,
- G is a finite set of gateways,
- R is a finite set of resources,
- L is a finite set of labels,
- $N = A \cup E \cup G$ is a finite set of nodes,
- $P = A \cup E$ is a finite set of process steps,
- $F \subseteq N \times N$ is the flow relation, such that (N, F) is a connected graph,

- $t : G \rightarrow \{and, xor\}$ is a mapping that associates each gateway with a type,
- $\rho : R \rightarrow A \cup E$ is a surjective mapping that associates a resource r to an activity $a \in A$ or an event $e \in E$,
- $\lambda : N \cup R \rightarrow L$ is a surjective mapping that relates process model nodes and resources to labels.

Note that this definition does not contain inclusive OR-gateways because of their marginal relevance in industrial process models [51] and since these are not generic to all graph-based modeling notations (e.g. Petri nets). From a given process model, we aim to extract the information depicted in Figure 3. In particular, we aim to extract information regarding activities, events, and gateways, their inter-relations, as well as the semantic components of their labels.

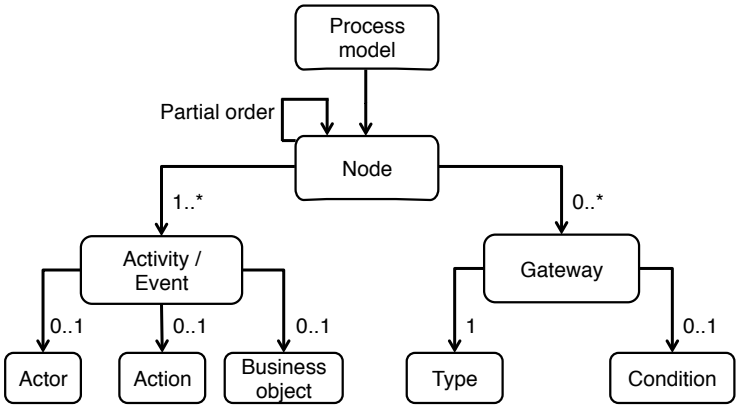

Figure 3: Process information extracted from process models

For process model activities and events, also referred to as *process steps*, we extract three semantic components: (i) an *actor* performing the step, (ii) an *action* that characterizes the step, usually described with a verb, and (iii) a *business object* on which the action is performed. We note that process steps are required to at least consist of an action, whereas the other components are optional, e.g. there can be steps without a defined actor or business object. Aside from process steps, we also extract information from the *gateways* in a process model, which denote routing aspects of a process, such as choices or

parallelism. A gateway is assigned a type, i.e., either *and* or *xor*. Furthermore, if an xor-gateway has an associated *execution condition*, such as illustrated in Figure 1 for $g1$, we augment the gateway with this information.

Process models have explicit constructs to denote activities, events, their actors, gateways, and the flow relation. Therefore, most information depicted in Figure 3 can be directly derived for process models abiding to Definition 1. However, the extraction of semantic components, namely actions and business objects, requires further processing of the natural language labels associated with process model elements:

Actions and business objects. Natural language labels associated with activities and events define their essential semantics [19]. In particular, labels generally convey the *action*, *business object*, and some additional information of a process step [31]. Therefore, the action and business objects used in the canonical format for process information need to be extracted from the labels associated with events and activities. A considerable problem here is that these labels often represent textual fragments, rather than proper sentences [20]. As a result, standard NLP techniques often fail to get accurate results for them [24]. To still be able to extract semantic components from labels, dedicated techniques have been developed that specifically aim to extract verbs, business objects, and *auxiliary* objects. For instance, Leopold [19] proposes a technique that uses knowledge about common label structures and an analysis of the model context to decompose activity and event labels. In our approach, we utilize a similar technique to extract the *action* and *business object* of process steps.

3.1.2. Extraction from Textual Descriptions

Unlike process models, textual process descriptions do not have explicit constructs that represent activities, events, or gateways. Therefore, all process-related information needs to be extracted from a textual process description using natural language processing. Our alignment approach aims to align process model elements to individual sentences. Therefore, our extraction step sets

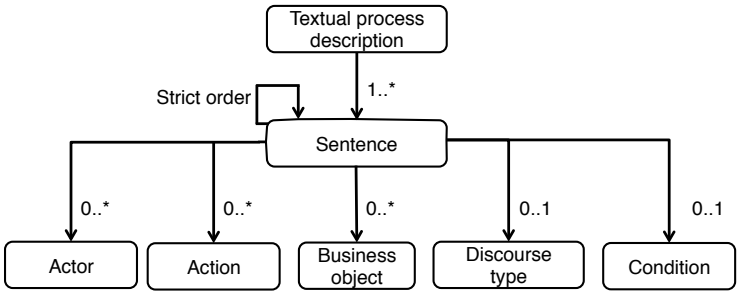

Figure 4: Process information extracted per sentence

out to identify process-related information at a sentence level. In particular, we aim to extract the components depicted in Figure 4.

As shown in the figure, we extract similar components from a sentence as we do from a process model element. The *actor*, *action*, and *business object* components correspond to the equally-named counterparts from Figure 3. These three describe the semantic components that characterize process steps. Furthermore, *discourse type* and *condition* are used to indicate if sentences contain information regarding the control-flow of a process, similar to the purpose of gateways in a process model. Finally, we also store the *strict order* relation that exists between sentences, which captures the order in which sentences appear in the text.

We extract the semantic components as follows:

Actions, actors, and business objects. In order to extract the desired process information from sentences in a textual process description, we can build on general-purpose NLP technology, such as techniques that analyze the grammatical and semantic structure of sentences and techniques for the resolution of anaphoric references (e.g., [34]) . These techniques have been widely applied in the context of textual process description for the extraction of activities and their actors, cf. [13, 47, 25]. We can employ such existing techniques in order to extract *actions*, *actors*, and *business objects* from a text.

For instance for the sentence 2 from the running example, "*Then, the claims officer writes a settlement recommendation and forwards it.*", the technique will

extract the following information: there are two *actions*: "*writes* and *forwards*, one actor, "*claims officer*", and a single business object, "*settlement recommendation*". Note due to *anaphora resolution* that is part of state-of-the-art techniques, our approach is also able to identify that the term "*it*" at the end of the sentence refers to the business object "*settlement recommendation*". This reference resolution is also possible if the business object is described in a separate sentence and can also be applied to actors that perform process steps.

Discourse types. To determine if a sentence describes a discourse marker, i.e., a choice or parallelism, we employ a technique similar to the method applied in the text-to-model generation technique from Friedrich et al. [13]. We use NLP techniques to detect discourse markers. A dictionary of multi-word expressions is applied to phrases like "in the meantime" to merge them into a single token *in_the_meantime*. As a result, our approach is able to detect if a sentence contains conditional statements (typically corresponding to process choices), such as seen for sentences $s4$, $s5$, and $s6$ of the running example, or describes steps that can be executed concurrently.

Conditions. Conditions are associated with the discourse markers used to identify conditional statements, as described above. For instance, statements such as "if" or "in case of" are followed by conditions, such as seen in the phrase "*If the review is negative*" in $s3$ of the running example. In those cases, the grammatical structure of the corresponding sentence is matched against several patterns in order to extract the clause containing the execution condition, e.g. to extract "*review is negative*" as a condition in sentence $s3$.

3.2. Similarity Computation

After extracting process information from a textual process description and process model, we encode the extracted information into feature vectors that can be used to accurately quantify the similarity between process model elements and sentences from a textual process description.

3.2.1. Feature Vectors

To quantify the similarity between process model nodes and sentences, we use *feature vectors* as a means to encode the information extracted from process models and sentences. These vectors represent a linearization of the extracted information that allows for an easy comparison. We define a number of different feature *types*, such as types related to *actions*, *actors*, or *business objects*. By doing so, we are able to assign different weight to the various types in order to tailor the quantification of similarity.

To quantify the similarity between a process model step p, i.e., an activity or an event, and a sentence s, we use the following feature types:

$contains\_action(a)$ This feature type denotes the actions contained in a step p or sentence s. For instance, when comparing the task "*write recommendation*" to sentence 2, we extract two actions: "*write*" and "*forward*", that produce two feature instances $contains\_action(write)$ and $contains\_action(forward)$. When considering these only two actions, on the task side this generates the vector $\langle 1, 0\rangle$, whereas for the sentence we obtain the vector $\langle 1, 1\rangle$.

$contains\_actor\_word(w)$ & $actor\_main\_word(w)$ These features denote the actors that execute steps in a process. The former feature, $contains\_actor\_word$, is extracted for each word w that is part of an actor. For instance, the "*claims officer*" actor comprises the two words "*claims*" and "*officer*". By contrast, we use the $actor\_main\_word$ feature to denote the main word of the actor, typically represented as the main noun, e.g., to explicitly capture the word "*officer*". This separation allows us to give different importance to the main word with respect to the others. See Section 3.2.2 for more details.

$contains\_object\_word(w)$ & $object\_main\_word(w)$ These features encode the same information for *business objects* as the previously described features do for *actors*.

contains_lemma(l, pos) This feature is extracted from the target text if it contains a word[1] with the lemma l and part-of-speech *pos*. This feature has a lower abstraction level than the previous ones. This feature is included as a fall-back solution whenever the actor, role or action cannot be determined due to natural language ambiguity. In those cases the algorithm works at the word level. For example, for the event "Claim paid out", the features *contains_lemma(claim, noun)* and *contains_lemma(pay-out, verb)* will be extracted.

contains_synset(s) This feature is extracted whenever the WordNet [32] synset s appears in the text sentence. It captures the semantics of words to help identify similarity when synonyms are used. For example, the WordNet synset *05747582-n* recognizes that "*review*" is a synonym of "*evaluation*", such as used in the running example.

contains_hypernym(s) This feature is extracted from a target text containing a word for which s is an hypernym[2] at distance *HL* or less. *HL* is a parameter of the algorithm. In the running example, a hypernym of "*review*" is "*assessment*" (*05733583-n*)

3.2.2. Vector Similarity

Instead of treating features as binary values, we opted for associating weights to the features individually. This way, the different importance each feature has can be considered in the comparison. The weight of each feature is the product of two magnitudes:

The feature instance weight is different for each instance of a feature type, and solves the problem of all instantiations not being equally important in terms of information provided. This is used in all lemma-based features,

[1] Note that in all feature types stopwords are not considered.

[2] A word w_1 is a *hypernym* of w_2 iff w_1 describes a superclass of w_2 (e.g. *mammal* is a hypernym of *cat*, and *document* is a hypernym of *letter*). Hypernymy is obtained from WordNet.

where the extracted lemma has an instance weight equal to the *tf-idf* score of the word[3]. It is also used in synset-based features, where the instance weight of the feature is K^{-l}, where l is the length of the hipernymy chain with respect to the original synset and K is a parameter of the algorithm.

The feature family weight is a weight defined for each feature type that accounts for the problem that not all feature types are equally likely and some of them provide more information than others. For example, a process step label sharing a word with a sentence is less important than having the same main action. The second reason for family weights is to adapt the scale of the instance weights, since they measure different magnitudes depending on the feature family.

Finally, the real-valued feature vectors are compared by using a standard similarity metric. In particular, we apply the *Weighted overlapping index*, defined as follows:

$$WeightedOverlapping(A, B) = \frac{\sum_{f \in A \cap B} w_f}{\sum_{g \in smallest(A,B)} w_g} \quad (1)$$

Where w_x is the weight, i.e. the product between the family weight and the instance weight, of a feature x.

The rationale for using this metric arises from the nature of the alignment problem we address. In general, textual process descriptions are more verbose than process model. Therefore, textual descriptions have larger feature vectors. Other metrics like the *Jaccard Index* or the *Cosine Similarity* [42], produce overall lower values when the compared elements differ in size. By contrast, the *Overlapping index* is able to deal with such size differences and, hence, provides more intuitive results.

[3]We define the *tf-idf* of a token t as the product of tf := (Number of appearances of t in its sentence / Number of tokens in that sentence) and idf := log_e(Total number of sentences / Sentences containing t)

3.3. Alignment Creation

The next step in our approach is the computation of the alignment between process model elements and the sentences of a textual description. The alignment contains the corresponding sentence for each activity, gateway and event of the process model (see the rightmost column in Table 1 for an example of alignment).

Formally, we define an alignment σ to be a set of correspondences of the form $n_k \sim s_i$ for some $n_k \in N,\ s_k \in S$, where N denotes the set of nodes of a process model and S is the set of sentences that comprise a text.

3.3.1. Alignment Constraints

In order to establish an alignment between process model elements and sentences, we impose two types of constraints on the alignment: *cardinality* constraints and *ordering* constraints.

Process step-to-sentence cardinality. For process model activities and events, which we shall jointly refer to as *process steps* in a set $P = A \cup E$, we enforce that each of them is aligned to exactly one sentence, whereas we allow multiple steps to be aligned to the same sentence. This constraint imposes the assumptions that each step is described in the text and that steps are not described repeatedly. Furthermore, it enables the proper alignment of sentences that describe multiple process steps, such as seen for sentence 5. This sentence corresponds to both the "*record settlement information*" and "*pay settlement*" activities.

Gateway-to-sentence cardinality. Gateways require different cardinality constraints. In particular, we allow gateways from the set G to be aligned to one sentence or to none at all. Furthermore, we allow at most one gateway to be aligned to a single sentence, given that sentences typically describe at most one control-flow structure.

Process step ordering. If we, for the purposes of this step in our approach, assume that a textual process description and a process model do not contradict each other, we can impose ordering constraints on the alignments that our

approach establishes. To define these ordering restrictions, we use the following notations. First, given two nodes $n, n' \in N$, we use $n \succ n'$ to denote that there is a path from node n to n' according to the flow relation $F \subseteq N \times N$, as given in Definition 1. Second, given two sentences $s, s' \in S$, we use $s \rightsquigarrow s'$ to denote that sentence s occurs before s' in a textual process description.

We require that process steps that precede each other in a process model cannot be aligned to sentences that occur in the reverse order. Therefore, we impose the following restriction on the order of process steps from the set $P = A \cup E$: if a process step $p \in P$ precedes the execution of a step $p' \in P$, then node p cannot be aligned to a sentence s that occurs after the sentence s' to which the node p' is aligned. Formally, this means that if it holds that $p \succ p'$ and $s' \rightsquigarrow s$, an alignment cannot contain both $p \sim s$ and $p' \sim s'$. Note that, in situations where two activities p and p' respectively follow each other in a loop, we only consider the order relation $p \sim p'$ and not $p' \sim p$. This enables the approach to appropriately apply the ordering constraints to loops.

Gateway ordering. The ordering constraints required for the alignment of gateways should account for several challenges. To illustrate these, consider the fragment of a textual process description and process model depicted in Figure 5. To align text and model, the following correspondences are required: $\{s7 \sim a7,\ s8 \sim g2,\ s8 \sim a8,\ s9 \sim e3\}$. These correspondences indicate that, unlike for process steps-to-sentences, the order in which gateways and process steps are described in a text can be reversed. In particular, the model contains relation $g2 \rightsquigarrow a7$, whereas $a7$ occurs *before* the description of $g2$ in the text. Furthermore, none of the sentences explicitly denotes when the parallel construct is closed, i.e., none of the sentences describes gateway $g3$.

Because of these complications, we can only impose fairly weak constraints. Specifically, we can state that:

- Process steps appearing before the opening gateway in the model cannot be described after the corresponding discourse marker in the text. E.g. if $g2 \sim s_i$ holds, any activity occurring before $g2$ cannot be aligned to a

s7: The manager records the settlement information.

s8: In the meantime, the financial department takes care of the payment.

s9: The process ends when the claim is paid out

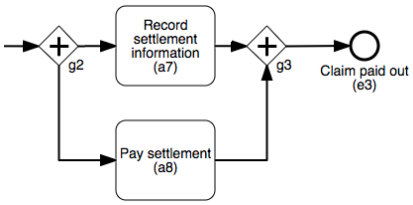

Figure 5: Description of a parallel construct in text

sentence s_j for which $s_i \rightsquigarrow s_j$ holds.

- Process steps appearing after the closing gateway in the model cannot be described before the corresponding discourse marker in the text. E.g. if $g2 \sim s_i$, then event $e3$ cannot be aligned to a sentence s_k for which $s_k \rightsquigarrow s_i$ holds.

3.3.2. Optimal Alignment

For an alignment σ to be considered optimal $\hat{\sigma}$, the following three properties must hold:

Cardinality consistency The alignment follows the cardinality constraints described in the previous section.

Order consistency The alignment is consistent with the order restrictions described in the previous section.

Optimality The value of $\sum_{n_k \sim s_i \in \hat{\sigma}} sim(n_k, s_i)$ is the maximum value such that the two other properties hold.

To define the order restrictions in a formal way we first introduce the following definitions:

$S_{disc} \subseteq S$: the set of sentences containing a discourse marker.

$G_{split} \subseteq G$: the set of gateways with one input flow and more than one output flow[4].

$\nu : G \to G \cup \{\bot\}$: a function that given a gateway returns the corresponding gateway such that the two delimit a single-entry single-exit region in the model [54], or $\bot$ if there is no such gateway.

In order to obtain a solution, the aforementioned properties are encoded in the following ILP:

maximize:

$$\sum_{s \in S} \sum_{n \in N} \sigma_{n,s} \cdot sim(n, s)$$

subject to:

$$\begin{aligned}
&\textstyle\sum_{s \in S} \sigma_{p,s} = 1 && \forall p \in P \\
&\textstyle\sum_{s_d \in S_{disc}} \sigma_{g,s_d} \leq 1 && \forall g \in G \\
&\textstyle\sum_{g \in G} \sigma_{g,s_d} \leq 1 && \forall s_d \in S_{disc} \\
&\sigma_{p,s'} + \sigma_{p',s} \leq 1 && \forall (s, s') \in S \times S,\ (p, p') \in P \times P, p \preceq p' \wedge s \rightsquigarrow s' \\
&\sigma_{g,s_d} + \sigma_{s,p} \leq 1 && \forall s \in S, s_d \in S_{disc}, g \in G_{op}, p \in P, p \preceq g, s_d \rightsquigarrow s \\
&\sigma_{g,s_d} + \sigma_{s,p} \leq 1 && \forall s \in S, s_d \in S_{disc}, g \in G_{op}, p \in P, \\
& && \nu(g) \neq \bot, \nu(g) \preceq p, s \rightsquigarrow s_d
\end{aligned}$$

variables:

$$a_{n,s} \in \{0, 1\} \qquad \forall s \in S, n \in N$$

The variables $\sigma_{n,s}$ can be interpreted as binary variables meaning: "Model node n corresponds to sentence s", i.e: $\sigma_{s,n} = 1 \iff s \sim n$. The first set of three constraints enforce the *Cardinality consistency* property[5]. This is

[4]Note that we purposefully avoid considering anomalous cases such as multiple inputs and outputs or single-input single-output gateways and restrict ourselves to the subset of well-formed BPMN models.

[5]Note that these equations can also be encoded using the *Special Ordered Sets* (*SOS*) constraint of the form: $a_{t,1}, \cdots, a_{t,|S|}$ for all model elements t, which denotes exactly the same constraint, and has better performance on the ILP solvers that implement it.

naturally encoded in ILP by restricting the sum of a subset of binary variables to either exactly one, or at least one depending on the desired cardinality. The second set of three constraints encodes the *Order consistency* property. This is done by explicitly restricting pairs of $s \sim p$ correspondences, i.e. they cannot be both true at the same time, for those pairs that constitute an order violation.

3.4. Predictor-based Refinement

The alignments that results from Section 3.3 are established under the assumption that a process model and a textual process description describe the same process, i.e., that the two representations do not contain inconsistencies. By operating under this assumption, we are able to impose constraints that have considerable positive effects on the quality of the resulting alignments, most notably resulting from the application of the ordering constraints described in Section 3.3.1. Nevertheless, it is important to acknowledge that varying descriptions of a process can contradict each other in practical settings [50]. Therefore, this section presents the final step of our approach in which we set out to detect the presence of inconsistencies and, if so, adapt obtained alignments accordingly.

To perform this refinement step, we first apply so-called *predictors* that quantify the likelihood that an obtained alignment contains inconsistencies. This notion of a predictor is inspired by according notions used to analyze alignments in the context of schema and process model matching [14, 56] and were originally applied in Van der Aa et al. [48]. The core premise underlying predictors is that alignments associated with consistent and inconsistent model-text pairs differ. For a consistent model-text pair, all process model elements should be aligned to a sentence with a high similarity score. By contrast, the similarity scores of the correspondences in the alignment of an inconsistent model-text pair will have different characteristics. Predictors quantify these characteristics and, as such, quantify the likelihood that an alignment contains a particular kind of inconsistency. If these predictors indeed detect likely inconsistencies, we subsequently weaken the alignment constraints accordingly in order to establish a refined alignment that takes the presence of likely inconsistencies into account.

s : "A manager must review the claim before he can accept it"

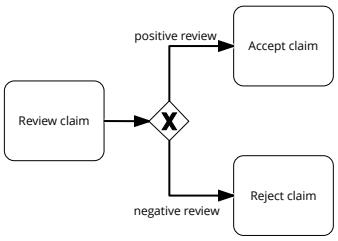

Figure 6: Example of a missing activity

We perform this refinement step for two types of inconsistencies: (i) process model elements that are missing from a textual description and (ii) ordering conflicts.

3.4.1. Missing Process Steps

The alignment technique described in Section 3.3 aligns each activity and event to a single sentence. However, it can happen that not all model elements are actually described in the text. Consider for instance the example depicted in Figure 6. In this fragment of a model-text pair, the textual description only focuses on the positive outcome of a review step and, therefore, does not describe the possibility that a claim can also be rejected. By contrast, the process model shows both possibilities.

The similarity scores of correspondences included in an alignment represent highly valuable indicators to identify missing elements. Process model elements that are contained in a textual description are expected to be aligned to sentences with a high similarity score. By contrast, if a process model element is not described in a textual description, it cannot be aligned to a sentence with a high similarity score, because none of the sentences in the text describe the same process step as the model elements. For instance, none of the sentences related to the example from Figure 6 contain terms related to the rejection of a claim, which results in considerably lower similarity scores for the "*reject claim*" activity. Given these lower similarity scores of missing process model elements,

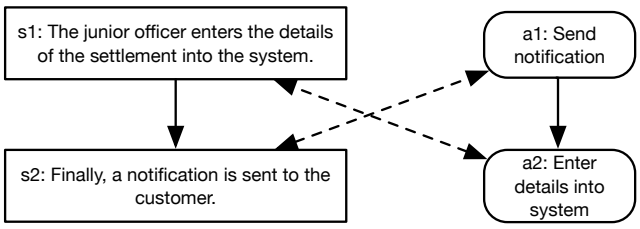

Figure 7: Example of an ordering conflict

we use predictors that recognize such instances.

To distinguish between *high* and *low* similarity scores, we evaluate similarity scores in an optimal alignment according to the following predictors:

- *p-sim*(p, s): the likelihood that a correspondence $p \sim s$ relates to a missing step, given as the similarity score between the step p and the sentence s i.e. $sim(p, s)$.

- *p-rel-S*(p, s): the value of the previous predictor, normalized by the maximum similarity between s and any other process step, i.e. $\frac{sim(p,s)}{max\ \{sim(p',s)\ \mid\ p' \in P\}}$

Subsequently, we remove correspondences from the obtained alignment that are likely to be missing from the textual description, i.e. the correspondences for which the predictor values are below a certain threshold. In this manner we can improve the quality of obtained alignments in terms of their *precision*, because incorrect correspondences will be excluded.

3.4.2. Ordering Conflicts

The ordering constraints imposed on alignments, described in Section 3.3.1, assume that a process model and a corresponding textual description describe the various steps of a process in the same order. However, when a model-text pair is inconsistent, it can happen that text and model contain ordering conflicts. Figure 7 presents an example of this, in which the two artifacts denote a different order of the "*send notification*" and "*enter details into system*" activities.

Due to the applied ordering constraints, alignments cannot contain both correspondences $a1 \sim s2$ and $a2 \sim s1$, even though these denote actual correspondences between model and text. To refine such alignments, we apply a predictor that aims to identify ordering conflicts, which allows us to remove the ordering constraints that impede the ability to establish alignments. The existence of conflicting orders between model and text can manifest itself in the form of large differences between the similarity scores contained in an optimal alignment and potential similarity scores that could have been achieved without these ordering constraints. For instance, in the above example, the total similarity scores of having the true correspondences $a1 \sim s2$ and $a2 \sim s1$ will be much higher than the similarity scores of the correspondences that abide to the ordering constraints.

We capture this characteristic difference between consistent and inconsistent model-text pairs in the predictor *max-constrained*. This predictor quantifies the maximum difference that exists between the aligned and potential score for a single process step in a model-text pair. It thus captures the largest similarity difference caused by imposing ordering restrictions on the optimal alignment. We operationalize this as follows:

- *max-constrained*($\hat{\sim}$): the maximal difference between the potential and aligned similarity scores for a process step $p \in P$.

For the cases where the predictor detects a likely ordering conflict, i.e. where the value of *max-constrained* is above a certain threshold, our approach recomputes a renewed alignment without ordering constraints. For instance, for the provided example, the refined alignment will contain both correspondences $a1 \sim s2$ and $a2 \sim s1$.

4. Evaluation

To demonstrate the capabilities of our approach, we conduct a quantitative evaluation by comparing automatically generated model-text alignments to a

manually created *gold standard*. The goal of this evaluation is to assess the quality of the automatically generated alignments. Both the data collection and the implementation of our approach used to conduct the evaluation are publicly available [6].

4.1. Test Collection

To perform the evaluation, we use a set of 74 model-text pairs obtained from various sources. Given the different nature of these sources, we partition the set into two data collections.

The first collection consists of 49 model-text pairs, corresponding to the original dataset used for the evaluation of existing model-text alignment approaches [48, 37]. In this test collection, we detected 30 models (61%) containing at least one missing activity and an average of 1.32 nodes per sentence. The model-text pairs in this dataset have been obtained from 11 different industrial and academic sources, including *inubit AG*, the *Federal Network Agency* of Germany, and various universities. [48] provides an in-depth overview of these sources. The gold standard alignments of this collection were built on the already established ones used in [37]. However, since the existing gold standard only included activity-to-sentence correspondences, we augmented it with correspondences involving events and gateways.

The second collection consists of 25 additional model-text pairs. For this second test collection, we observed 12 models (48%) with at least one missing activity and an average of 1.54 nodes per each sentence. The process models in this collection have been obtained from the repository of the *BPM Academic Initiative*[10]. The texts accompanying these models were authored by 8 experienced modelers. In order to obtain textual descriptions covering a variety of styles, the experts were asked to perform the following three steps: *(i)* Study the process model diagram. *(ii)* Write the textual description without looking at the source model. *(iii)* Compare the textual description with the source model to

[6] `https://github.com/setzer22/alignment_model_text`

make sure the text accurately describes the process model. This final step was introduced to reduce the amount of inconsistencies between texts and models. Furthermore, the authors of the text were not involved in the development of the approach of this paper nor aware of the exact purpose of their task. For the 25 model-text pairs in this new collection, the gold standard alignments were annotated and subsequently verified by the authors of this paper.

Table 2 provides an overview of the characteristics of the model-text pairs included in the evaluation set.

Table 2: Characteristics of model-text pairs in the test collections

Collection	**P**	**P_{ma}**	**N**	**A**	**G**	**E**	**S**	**N/S**
Original	49	30	11.99	8.12	1.66	2.21	9.13	1.32
New	25	12	12.40	7.44	2.80	2.16	7.72	1.54
Total	**74**	**42**	**12.13**	**7.89**	**2.05**	**2.19**	**8.65**	**1.39**

Legend: P = Model-text pairs, P_{ma} = Amount of model-text pairs with missing elements, N = Nodes per model (avg.), A = Activities per model (avg.), G = Gateways per model (avg.), E = Events per model (avg.) S = Sentences per text (avg.), N/S = Nodes per sentence (avg.)

4.2. Setup

To conduct the evaluation, we have implemented our proposed alignment approach in the form of a Java prototype. For this implementation we used Freeling [34] for the natural language processing and Gurobi as the ILP solver to compute optimal alignments.

To quantify the quality of a generated alignment for a given model-text pair we compute the widely-employed *accuracy* metric. This metric quantifies the number of correct correspondences with respect to the total number of correspondences. Let σ be the alignment generated by our approach over a set of sentences S and a set of process model nodes N. furthermore, let σ^* be

the gold standard alignment. We then define the *accuracy* of our approach as follows:

$$\text{accuracy} = \frac{|\sigma \cap \sigma^*|}{|N|}$$

In order to operationalize our implementation, the parameters of the approach have been set in the following way: The feature family weights used were tuned using a genetic algorithm exploration. The predictor used for missing elements is *p-rel-S*, with a threshold value of 0.1. The predictor used for order conflicts was *max-constrained*, with a threshold value of 0.8. In Section 4.4 we explore the impact of alternate parameter settings on the performance of our approach. Finally, as a benchmark, we compare the performance of our approach to the two earlier proposed alignment techniques from [37, 48].

4.3. Results

Table 3 shows an overview of the evaluation results for our approach with and without predictors, as well as for the two benchmark approaches. In particular, we depict the accuracy our approach achieves on the alignment of activities, events, gateways, and the overall accuracy.

The result shows that our approach achieves an overall high accuracy of 0.71 for the total test collection. When considering the different types of nodes, we observe that the approach performs best for the alignment of activities, achieving an accuracy of 0.79. For events (0.56) and gateways (0.50), the approach is less accurate. This difference in performance could be explained by the fact that our approach is more informed with respect to activities than the other two element types. Particularly, the *agent*, *action* and *business object* features are not extracted for gateways and do not always fit the writing style of event descriptions. Furthermore, we can observe that the use of predictor-based refinement has a positive impact on the accuracy of the approach, increasing the accuracy from an overall of 0.69 to 0.71. The benefits of using predictors are most apparent for the alignment of gateways, where the accuracy increases from 0.40 to 0.50.

Table 3: Overview of the evaluation results

		Accuracy			
Collection	**Configuration**	**Activities**	**Events**	**Gateways**	**Overall**
Original	Approach from [37]	0.76	n/a	n/a	n/a
	Approach from [48]	**0.78**	n/a	n/a	n/a
	Without predictors	**0.78**	0.42	0.32	0.66
	With predictors	**0.78**	**0.44**	**0.43**	**0.68**
New	Approach from [37]	0.79	n/a	n/a	n/a
	Approach from [48]	0.77	n/a	n/a	n/a
	Without predictors	**0.81**	**0.80**	0.55	0.75
	With predictors	0.80	**0.80**	**0.63**	**0.76**
Total	Approach from [37]	0.78	n/a	n/a	n/a
	Approach from [48]	0.77	n/a	n/a	n/a
	Without predictors	0.79	0.55	0.40	0.69
	With predictors	**0.79**	**0.56**	**0.50**	**0.71**

The results show that our approach slightly improves upon the accuracy achieved by the existing approaches from [37, 48]. However, the primary difference, as clearly shown in Table 3 is that our proposed approach provides much more complete alignments in terms of the process model elements that it considers.

When comparing the performance of the approach across the two test collections, we observe that the performance is comparable with respect to the alignment of activities (0.78 versus 0.80). However, for events (0.44 versus 0.80) and gateways (0.43 versus 0.63), the approach performs considerably better on the model-text pairs in the new collection. There are several factors that may have influenced this performance gap. On one hand, the new dataset has a substantially higher amount of labelled gateways, which help create more informed alignments. On the other hand, generating textual descriptions from models may favor a more literal style when describing events when compared

to the some independently developed descriptions of the original dataset. Despite these differences, the evaluation results show that the approach achieves promising results for model-text pairs from a wide range of sources.

By taking a more in-depth look at the results, we can furthermore make the following observations:

- Our approach is able to establish alignments even when a model contains multiple activities with identical or near-identical labels. This is achieved because the approach considers additional semantic information obtained from features like actors and the structural information provided by the ordering constraints.

- The size of a process models, in terms of the notes to be aligned, has no significant impact on the performance of our approach. The average alignment accuracy for models with $N > 15$ is 0.70 and 0.65 for $N > 20$, which does not differ significantly from the average performance of 0.75 for al modes (with average N = 12.13). This observation is confirmed through a Pearson correlation test [6], which yielded a correlation coefficient of -0.11 and a p-value of 0.33, indicating no significant correlation between model size and alignment accuracy.

- Using the *tf-idf* as a multiplicative factor for word-based features helps automatically regulate their importance: A word that is used throughout the process does not contribute as much to the similarity as one that is only mentioned in a subset of the description. This is important in the case of gateways, where the high-level information –i.e. agent, action and business object– cannot be obtained and the word-level features alone determine the similarity.

- Our approach currently treats events in the same way as activities. While some events are typically described like activities, some other event types are more naturally described in a different style. One such example is timer events, where sentences like "After X days have passed" are more

common. Parsing errors that result from trying to find activity structure in those events result in less informed alignments.

- Most gateway alignments missed in the evaluation were implicitly described in the textual description. For example, a process model described different alternatives for sorting invoices with an exclusive gateway, but the textual description simply stated: "The invoices can be sorted in two ways: by amount and by vendor." Such implicit descriptions cannot be detected by our alignment tool since no discourse marker is present.

4.4. Influence of the Parameters

The parametrization of our proposed approach can have a considerable impact on the quality of the obtained results. In order to understand how this occurs, we study the effect of the following parameters:

Feature family weights (Section 3.2.2) define the importance of features with respect to each other.

Predictors, and their thresholds (Section 3.4) define the strategy and likelyhood of detecting a *missing* node or an *order inconsistency* in the process.

In the next two sections we report the exploration done which additionally lead us to the final parameters parameters used to obtain the results shown in Table 3.

4.4.1. Effect of Feature Weights

To assess the influence of feature weights, we considered several exploration techniques. Using an exhaustive exploration technique, or an experimental design requires discreticising the weights. Due to the high sensibility of the parameters and the computation time required to evaluate each individual combination, we avoided such an exhaustive analysis. Instead, we opted for an heuristic search based on *genetic algorithms*, which are well known for metaparameter

Figure 8: Evolution of the genetic algorithm optimization showing the population maximum, average and minimum fitness values.

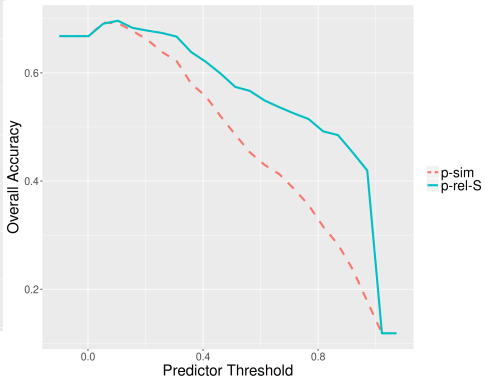

Figure 9: Accuracy of the tool when varying the threshold for different predictors.

optimization. In order to guide the search, the fitness function was defined as the overall accuracy obtained by the tool using the original dataset.

Figure 8 shows the evolution of the optimization for one of the executions. The maximum and average values for the population fitness have a tendendy to increase, which shows the positive effect of a good parametrization. On the other hand, the minimum values fluctuate uniformly because of random individuals being added at each generation. This shows the impact of the weight parameters in our technique. While a bad set of weights affects the results negatively, the algorithm can still perform with a reasonable accuracy even with such bad parametrization.

Some common characteristics were observed in the fittest individuals of the last generation: The most important features were actions, business objects and discourse markers. On the other hand, word-based features and actors had substantially lower weights. Finally, for features that distinguish the main word, such as *contains_actor_word* and *actor_main_word*, the latter type was found to consistently have a higher weight.

4.4.2. Effect of Predictors and Thresholds

In order to study the effect of predictors, we conducted an experiment using only the first test collection. The goal was to observe the effect of the predictor type (*p-sim* or *p-rel-S*), as well as the threshold value used to detect a missing step, on the overall accuracy.

Figure 9 shows the performance of the tool when using both predictors and varying their thresholds from 0 to 1, with a step length of 0.05. The value at 0 indicates the performance of the tool when not using predictors, while the value at 1 represents the maximum achievable improvement of using predictors for the detection of missing elements.

As shown, all predictor types have an initial peak region where performing the predictor-based refinement offers some benefit. After some point, the approach becomes too strict and considers too many elements as missing, resulting in a drop in the overall accuracy. This peak region is similar for the three approaches, but *p-rel-S* is more stable since it offers a less steep curve.

This exploration shows us the potential benefits of predictor-based refinement in our technique. We conclude a good range of values for the similarity threshold lays in the interval $(0.05, 0.2)$. Finally, the increased stability of the *p-rel-S* predictor makes it more suited for a generic approach.

5. Conclusions

This paper presented a fully automated approach to align textual descriptions to process models. The proposed approach combines tailored NLP processing techniques, semantic matching, and predictors in order to establish optimal alignments between the nodes of a process model and the sentences of a textual description. Unlike existing approaches that address this task, our approach aligns a broad range of process model elements, including events and gateways, rather than just focusing on the alignment of activities. A quantitative evaluation performed on a collection 74 model-text pairs demonstrates that our approach achieves satisfactory results.

We foresee several research directions that can be followed. On the one hand, incorporating support to more element types and constructs may lead to a more precise analysis; among others, we may consider *subprocesses*. Another interesting direction is to improve the characterization and computation of the order for the sentences in the text, for instance learning a classifier tailored towards describing control flow in textual descriptions.

Acknowledgements

This work has been supported by funds from the Spanish Ministry for Economy and Competitiveness (MINECO), the European Union (FEDER funds) via grants GRAMM (TIN2017-86727-C2-1-R) and Graph-Med (ref. TIN2013-46181-C2-1-R, TIN2016-77820-C3-3-R), and the Alexander von Humboldt Foundation.

References

[1] E. Andrade, H. van der Aa, H. Leopold, S. Alter, H. A. Reijers, Factors Leading to Business Process Noncompliance and its Positive and Negative Effects: Empirical Insights from a Case Study, in: 22nd Americas Conference on Information Systems, AMCIS, 2016.

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

[3] J. Becker, P. Delfmann, S. Herwig, L. Lis, A. Stein, Formalizing Linguistic Conventions for Conceptual Models, in: Conceptual Modeling - ER 2009, LNCS, Springer Berlin Heidelberg, 70–83, 2009.

[4] M. Born, F. Dörr, I. Weber, User-Friendly Semantic Annotation in Business Process Modeling, in: WISE 2007 Workshops, vol. 4832 of *LNCS*, Springer, 260–271, 2007.

[5] T. R. Browning, On the alignment of the purposes and views of process models in project management, Journal of Operations Management 28 (4) (2010) 316–332.

[6] J. L. Bruning, B. L. Kintz, Computational handbook of statistics, Scott, Foresman & Co, 1987.

[7] S. Chakraborty, S. Sarker, S. Sarker, An exploration into the process of requirements elicitation: A grounded approach, Journal of the association for information systems 11 (4) (2010) 212–249.

[8] J. C. de Gonçalves, F. M. Santoro, F. A. Baiao, Business process mining from group stories, in: Computer Supported Cooperative Work in Design, 2009. CSCWD 2009. 13th International Conference on, IEEE, 161–166, 2009.

[9] R. M. Dijkman, M. Dumas, L. García-Bañuelos, Graph matching algorithms for business process model similarity search, in: International Conference on Business Process Management, Springer, 48–63, 2009.

[10] R.-H. Eid-Sabbagh, M. Kunze, A. Meyer, M. Weske, A Platform for Research on Process Model Collections, in: J. Mendling, M. Weidlich (Eds.), Business Process Model and Notation, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 978-3-642-33155-8, 8–22, 2012.

[11] E. V. Epure, P. Martín-Rodilla, C. Hug, R. Deneckère, C. Salinesi, Automatic process model discovery from textual methodologies, in: Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on, IEEE, 19–30, 2015.

[12] C. Francescomarino, P. Tonella, Supporting Ontology-Based Semantic Annotation of Business Processes with Automated Suggestions, in: International Conference on Enterprise, Business-Process and Information Systems Modeling, vol. 29 of *LNBIP*, Springer, 211–223, 2009.

[13] F. Friedrich, J. Mendling, F. Puhlmann, Process model generation from natural language text, in: International Conference on Advanced Information Systems Engineering, Springer, 482–496, 2011.

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

[15] A. Ghose, G. Koliadis, A. Chueng, Process discovery from model and text artefacts, in: Services, 2007 IEEE Congress on, IEEE, 167–174, 2007.

[16] V. Gruhn, R. Laue, Detecting Common Errors in Event-Driven Process Chains by Label Analysis, Enterprise Modelling and Information Systems Architectures 6 (1) (2011) 3–15.

[17] J.-Y. Jung, J. Bae, Workflow clustering method based on process similarity, in: International Conference on Computational Science and Its Applications, Springer, 379–389, 2006.

[18] A. Koschmider, E. Blanchard, User Assistance for Business Process Model Decomposition, in: Proceedings of the 1st IEEE International Conference on Research Challenges in Information Science, 445–454, 2007.

[19] H. Leopold, Natural language in business process models, Springer, 2013.

[20] H. Leopold, R.-H. Eid-Sabbagh, J. Mendling, L. G. Azevedo, F. A. Baião, Detection of naming convention violations in process models for different languages, Decision Support Systems 56 (2013) 310–325.

[21] H. Leopold, C. Meilicke, M. Fellmann, F. Pittke, H. Stuckenschmidt, J. Mendling, Towards the automated annotation of process models, in: International Conference on Advanced Information Systems Engineering, Springer, 401–416, 2015.

[22] H. Leopold, J. Mendling, A. Polyvyanyy, Supporting Process Model Validation through Natural Language Generation, IEEE Transactions on Software Engineering 40 (8) (2014) 818–840.

[23] H. Leopold, M. Niepert, M. Weidlich, J. Mendling, R. M. Dijkman, H. Stuckenschmidt, Probabilistic optimization of semantic process model matching, in: International Conference on Business Process Management, Springer, 319–334, 2012.

[24] H. Leopold, S. Smirnov, J. Mendling, Refactoring of process model activity labels, in: Natural Language Processing and Information Systems, Springer, 268–276, 2010.

[25] H. Leopold, H. van der Aa, F. Pittke, M. Raffel, J. Mendling, H. A. Reijers, Searching textual and model-based process descriptions based on a unified data format, Software & Systems Modeling (2017) 1–16.

[26] J. Ling, L. Zhang, Q. Feng, An Improved Structure-based Approach to Measure Similarity of Business Process Models., in: SEKE, 377–380, 2014.

[27] K. Liu, Z. Yan, Y. Wang, L. Wen, J. Wang, Efficient syntactic process difference detection using flexible feature matching, in: Asia-Pacific Conference on Business Process Management, Springer, 103–116, 2014.

[28] C. Meilicke, H. Leopold, E. Kuss, H. Stuckenschmidt, H. A. Reijers, Overcoming individual process model matcher weaknesses using ensemble matching, Decision Support Systems 100 (2017) 15–26.

[29] J. Mendling, Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness, vol. 6, Springer Science & Business Media, 2008.

[30] J. Mendling, B. Baesens, A. Bernstein, M. Fellmann, Challenges of smart business process management: An introduction to the special issue, Decision Support Systems 100 (2017) 1–5, URL `https://doi.org/10.1016/j.dss.2017.06.009`.

[31] J. Mendling, H. A. Reijers, J. Recker, Activity labeling in process modeling: Empirical insights and recommendations, Information Systems 35 (4) (2010) 467–482.

[32] G. A. Miller, WordNet: a lexical database for English, Communications of the ACM 38 (11) (1995) 39–41.

[33] A. Ottensooser, A. Fekete, H. A. Reijers, J. Mendling, C. Menictas, Making sense of business process descriptions: An experimental comparison of graphical and textual notations, Journal of Systems and Software 85 (3) (2012) 596–606.

[34] L. Padró, E. Stanilovsky, FreeLing 3.0: Towards Wider Multilinguality, in: Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012, 2473–2479, URL `http://www.lrec-conf.org/proceedings/lrec2012/summaries/430.html`, 2012.

[35] F. Pittke, H. Leopold, J. Mendling, G. Tamm, Enabling reuse of process models through the detection of similar process parts, in: International Conference on Business Process Management, Springer, 586–597, 2012.

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

[37] J. Sànchez-Ferreres, J. Carmona, L. Padró, Aligning Textual and Graphical Descriptions of Processes Through ILP Techniques, in: International Conference on Advanced Information Systems Engineering (In press), Springer, 2017.

[38] M. L. Sebu, H. Ciocârlie, Similarity of business process models in a modular design, in: Applied Computational Intelligence and Informatics (SACI), 2016 IEEE 11th International Symposium on, IEEE, 31–36, 2016.

[39] 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.

[40] A. Senderovich, A. Rogge-Solti, A. Gal, J. Mendling, A. Mandelbaum, The ROAD from Sensor Data to Process Instances via Interaction Mining, in:

International Conference on Advanced Information Systems Engineering, Springer, 257–273, 2016.

[41] A. Sinha, A. Paradkar, Use Cases to Process Specifications in Business Process Modeling Notation, in: IEEE International Conference on Web Services, 473–480, 2010.

[42] P.-N. Tan, et al., Introduction to data mining, Pearson Education India, 2006.

[43] H. van der Aa, J. Carmona, H. Leopold, J. Mendling, L. Padró, Challenges and Opportunities of Applying Natural Language Processing in Business Process Management, in: International Conference on Computational Linguistics, 2018.

[44] H. Van der Aa, A. Gal, H. Leopold, H. A. Reijers, T. Sagi, R. Shraga, Instance-Based Process Matching using Event-Log Information, in: International Conference on Advanced Information Systems Engineering (In press), Springer, 2017.

[45] H. Van der Aa, H. Leopold, A. del Rio-Ortega, M. Resinas, H. A. Reijers, Transforming Unstructured Natural Language Descriptions into Measurable Process Performance Indicators Using Hidden Markov Models, Information Systems 71 (2017) 27–39.

[46] H. Van der Aa, H. Leopold, F. Mannhardt, H. A. Reijers, On the Fragmentation of Process Information: Challenges, Solutions, and Outlook, in: International Conference on Enterprise, Business-Process and Information Systems Modeling, Springer, 3–18, 2015.

[47] H. Van der Aa, H. Leopold, H. A. Reijers, Checking Process Compliance on the Basis of Uncertain Event-to-Activity Mappings, in: International Conference on Advanced Information Systems Engineering (In press), Springer, 2017.

[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] H. van der Aa, H. Leopold, H. A. Reijers, Checking Process Compliance against Natural Language Specifications using Behavioral Spaces, Information Systems .

[50] H. Van der Aa, H. Leopold, I. van de Weerd, H. A. Reijers, Causes and Consequences of Fragmented Process Information: Insights from a Case Study, in: 23rd Americas Conference on Information Systems, AMCIS, 2017.

[51] W. M. Van der Aalst, The application of Petri nets to workflow management, Journal of circuits, systems, and computers 8 (01) (1998) 21–66.

[52] W. M. P. Van der Aalst, A. H. Ter Hofstede, M. Weske, Business process management: A survey, in: International conference on business process management, Springer, 1–12, 2003.

[53] B. Van der Vos, J. A. Gulla, R. van de Riet, Verification of conceptual models based on linguistic knowledge, Data & Knowledge Engineering 21 (2) (1997) 147 – 163.

[54] J. Vanhatalo, H. Völzer, J. Koehler, The refined process structure tree, Data Knowl. Eng. 68 (9) (2009) 793–818.

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

[56] M. Weidlich, T. Sagi, H. Leopold, A. Gal, J. Mendling, Predicting the quality of process model matching, in: International Conference on Business Process Management, Springer, 203–210, 2013.

[57] 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, 130–137, 2013.

[58] M. Weidlich, M. Weske, J. Mendling, Change propagation in process models using behavioural profiles, in: Services Computing, 2009. SCC'09. IEEE International Conference on, IEEE, 33–40, 2009.