Comparing Textual Descriptions to Process Models – The Automatic Detection of Inconsistencies

Han van der Aa[a,*], Henrik Leopold[a], Hajo A. Reijers[a,b]

[a] *Department of Computer Sciences, VU University Amsterdam, Faculty of Sciences, De Boelelaan 1081, 1081HV Amsterdam, The Netherlands*
[b] *Department of Mathematics and Computer Science, Eindhoven University of Technology, PO Box 513, 5600MB Eindhoven, The Netherlands*

Abstract

Many organizations maintain textual process descriptions alongside graphical process models. The purpose is to make process information accessible to various stakeholders, including those who are not familiar with reading and interpreting the complex execution logic of process models. Despite this merit, there is a clear risk that model and text become misaligned when changes are not applied to both descriptions consistently. For organizations with hundreds of different processes, the effort required to identify and clear up such conflicts is considerable. To support organizations in keeping their process descriptions consistent, we present an approach to automatically identify inconsistencies between a process model and a corresponding textual description. Our approach detects cases where the two process representations describe activities in different orders and detect process model activities not contained in the textual description. A quantitative evaluation with 53 real-life model-text pairs demonstrates that our approach accurately identifies inconsistencies between model and text.

Keywords: Inconsistency detection, Business process modeling, Natural language processing, Matching

[*] 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), `h.a.reijers@vu.nl` (Hajo A. Reijers)

1. Introduction

The documentation of business operations using process models has become a quintessential activity for many organizations [1]. However, organizations typically do not solely rely on process models for documenting business operations. Realizing that some stakeholders have difficulties with reading and interpreting process models [1, 2], organizations have recognized the value of maintaining text-based process descriptions alongside model-based ones [3]. While such textual descriptions may not be suitable to represent complex aspects of a process in a precise manner [4], they can be created, maintained, and understood by virtually everyone [5].

Despite these benefits, the usage of two representation formats for the same process can lead to considerable difficulties [6]. Most notably, there is a high risk of having to deal with inconsistencies between the two representation formats, in particular when different stakeholders develop or maintain the two representation formats independently from each other [7]. As a result of such inconsistencies, readers of the different representations may develop different expectations about what the process aims to establish or how it should be executed. Against the background of the potentially disastrous implications of inconsistencies, it is an important task of organizations to keep their process descriptions consistent. However, the associated effort to identify and clear up conflicts for an entire process repository is hardly manageable in a manual way.

To effectively deal with the problem of inconsistencies between model and text, we present a technique that automatically detects differences between textual and model-based process descriptions. Specifically, our technique identifies two types of inconsistencies. First, it identifies process model activities that are not contained in the accompanying textual description. Second, the technique detects cases where a process model and a textual description describe the process steps in a conflicting order. Our technique can be used to quickly identify the process models in a collection that are likely to diverge from their accompanying textual descriptions. This allows organizations to focus their efforts on the descriptions that can be expected to contain such inconsistencies. A quantitative evaluation demonstrates that the proposed technique is indeed able to effectively identify inconsistencies in a collection of model-text pairs obtained from practice.

The remainder of this paper is structured as follows. Section 2 illustrates the problem tackled by our approach and discusses the research gap that follows from a review of related work. Section 3 describes the proposed approach to detect inconsistencies. In Section 4, we present a quantitative evaluation of the approach. Finally, we discuss limitations in Section 5 and conclude the paper in Section 6.

2. Background

2.1. Problem Illustration

To illustrate the challenges that are associated with the detection of inconsistencies between textual and model-based process descriptions, consider the model-text pair shown in Figure 1. It includes a textual and a model-based description of a bicycle manufacturing process. On the left-hand side, we observe a textual description, which comprises eleven sentences. On the right-hand side, a corresponding model-based description can be seen, expressed in the Business Process Model and Notation (BPMN). The model contains nine activities, which are depicted using boxes with rounded edges. The diamond shapes that contain a plus symbol indicate concurrent streams of action; the diamond shapes containing a cross represent decision points. The gray shades suggest correspondences between the sentences and the activities of the process model.

A closer look at the example reveals that many connections between the two artifacts are evident. For example, there is little doubt that sentence (7) describes the "*reserve part*" activity or that sentence (8) describes the "*back-order part*" activity . In some cases, however, there is clearly an inconsistency between the two process representations. For instance, there is no sentence that is related to the "*ship bicycle to customer*" activity, i.e. that activity is missing from the textual description. Likewise, we can observe that sentences (4) and (5) occur in a different order than the corresponding activities in the model.

In other cases it is *less* straightforward to decide on the consistency – or lack thereof – between the representations. For example, the text of sentence (9) simply indicates that a part of the process must be repeated. By contrast, the model includes an activity, "*select unchecked part*", which associates an explicit action with this repetition. Whether or not sentence (9) actually describes an activity, and thus should be considered an inconsistency, seems to be open for debate.

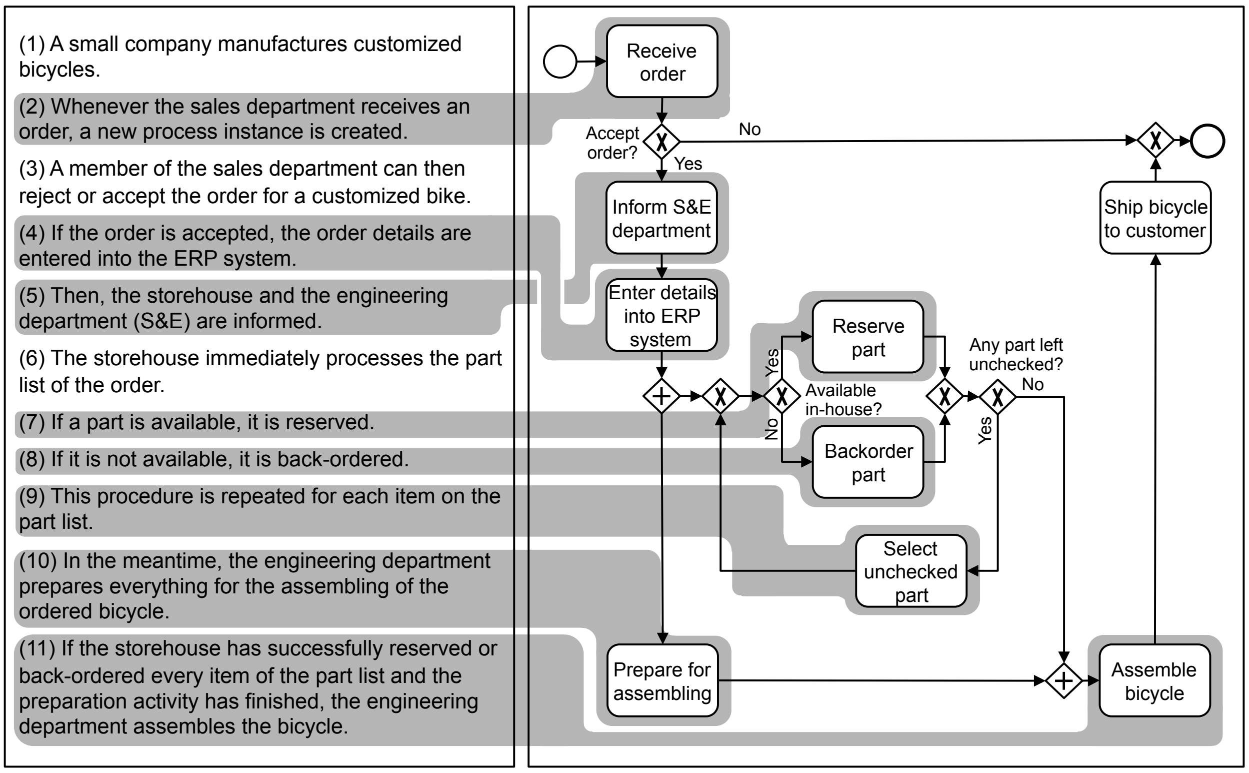
(1) A small company manufactures customized bicycles.
(2) Whenever the sales department receives an order, a new process instance is created.
(3) A member of the sales department can then reject or accept the order for a customized bike.
(4) If the order is accepted, the order details are entered into the ERP system.
(5) Then, the storehouse and the engineering department (S&E) are informed.
(6) The storehouse immediately processes the part list of the order.
(7) If a part is available, it is reserved.
(8) If it is not available, it is back-ordered.
(9) This procedure is repeated for each item on the part list.
(10) In the meantime, the engineering department prepares everything for the assembling of the ordered bicycle.
(11) If the storehouse has successfully reserved or back-ordered every item of the part list and the preparation activity has finished, the engineering department assembles the bicycle.

Figure 1: A textual and a model-based description of a bicycle manufacturing process

Ambiguous cases that are already difficult to resolve for human readers pose even greater problems when texts are analyzed in an automatic manner.

The brief illustration of the model-text pair from Figure 1 shows that an appropriate technique for detecting inconsistencies (i) must be able to recognize structural as well behavioral process aspects in a natural language text and (ii) must be able to align this information with process model activities. In the following section, we review prior work in order to investigate to what extent existing techniques are capable of addressing these challenges.

2.2. Related Work

This paper is an extended version of an earlier conference paper [5]. Compared to [5], we extended our technique in order to detect two types of inconsistencies, missing activities and conflicting orders, separately. To achieve this, we introduce several different *predictors* that can detect inconsistencies at the process-level and the activity-level. As such, the presented technique can detect inconsistencies in a much more fine-granular manner than the technique of [5]. The previous work used a single, heuristic-based metric to detect inconsistencies at the process-level and did not distinguish between different types inconsistencies. Moreover, we revised the anaphora resolution technique so that it can deal with references to complex business objects, such as compounds (e.g., "*sales department*"). In addition, we improved the quantitative evaluation by extending the data collection and by presenting various new experimental results that reflect the more fine-granular nature of the improved inconsistency detection technique. Below, we review related work from the following streams of research: (i) research on the transformation between model and text and (ii) research on schema and process model matching.

2.2.1. Transformations between Model and Text

Research on the *transformation between model and text* is concerned with transforming a given representation into the other one. Prior work has addressed both directions, model-to-text as well as text-to-model. Transformations of conceptual models into textual descriptions typically aim at making the information captured by the model available to a wider audience. Among others, such techniques have been defined for UML diagrams [8], object models [9], and process models [3]. Techniques transforming text into models usually aim at providing support for model creation. Text-to-model transformation techniques also cover a variety of conceptual models, including UML class diagrams [10] and entity-relationship models [11]. Several approaches specifically focus on the elicitation of process models from different types of text documents, such as group stories [12], use case specifications [13], and process descriptions [14].

Despite the empirically demonstrated usefulness of the discussed transformation techniques, they do not help to address the challenges associated with detecting inconsistencies. Techniques for transforming models into texts focus on verbalizing the information from the model. Hence, they do not provide any means to align the information from a given model-text pair. Techniques for transforming process descriptions into models do address the problem of inferring structural and behavioral process information from a natural language text. However, these approaches have been found to produce incomplete or inaccurate models, which require extensive manual revision [15]. Hence, they are not suitable candidates to support the automatic detection of inconsistencies between textual and model-based process descriptions. These inaccuracies mainly manifest themselves during the extraction of activities from a process description and the derivation of relations between them. The approach presented in this paper overcomes these issues by taking process model activities and their inter-relations as the starting point for the detection of inconsistencies.

2.2.2. Matching

Research on *matching* is concerned with the task of identifying relations between concepts in different artifacts [16]. The result of a matching is typically referred to as an *alignment*. In the past, researchers have defined a plethora of matching approaches for various domains. The most widely-covered application areas include schema matching (see e.g. [7, 17, 18]) and ontology matching (e.g. [19, 20, 21]). In recent years, the potential of matching has also been recognized in the domain of process modeling [22]. Process model matchers are capable of automatically identifying correspondences between the activities of two process models. The application scenarios of these matchers range from harmonization of process model variants [23] to the detection of process model clones [24]. To accomplish these goals, matchers exploit different process model features, including natural language [25], model structure [26], and behavior [27].

The work on matching is closely related to the technique defined in this paper. In fact, prior research has already emphasized the importance of a proper alignment for the detection of inconsistencies between model and text [6]. However, to the best of our knowledge, there is no technique available that can compute an alignment between process models and textual process descriptions. So far, the required inference of structural and behavioral information from textual descriptions has not been sufficiently addressed.

The review of existing literature thus reveals that the challenges associated with detecting inconsistencies between model and text cannot be properly addressed by prior techniques. To address this research gap, we use the subsequent section to define a technique that is capable of automatically detecting inconsistencies between model and text-based process representations.

3. Conceptual Approach

This section describes our approach to identify inconsistencies in a model-text pair. Section 3.1 presents an overview of the approach and its main steps. Subsequently, sections 3.2 through 3.5

describe the individual steps of the approach in detail.

3.1. Overview

Figure 2 illustrates the four main steps of our approach. Our approach takes a process model and an accompanying textual description as inputs. It first subjects both inputs to a linguistic analysis. The goal of this analysis is to process sentences and activities in such a way that their similarity can be accurately assessed. Second, we compute similarity scores that quantify the semantic similarity between individual activities and sentences. Third, we construct an activity-sentence *alignment*. We do so by complementing the similarity scores with a consideration of the ordering relations that exist between the various process steps captured in model and text. In the fourth and final step, we detect inconsistencies between the model-based and textual process description. To achieve this, we use *predictors* that evaluate the quality of the obtained activity-sentence alignment. The final result of the approach is a set of predicted inconsistencies, both at a process-level, as well as at a more fine granular activity-level. We will now look at these steps in more detail.

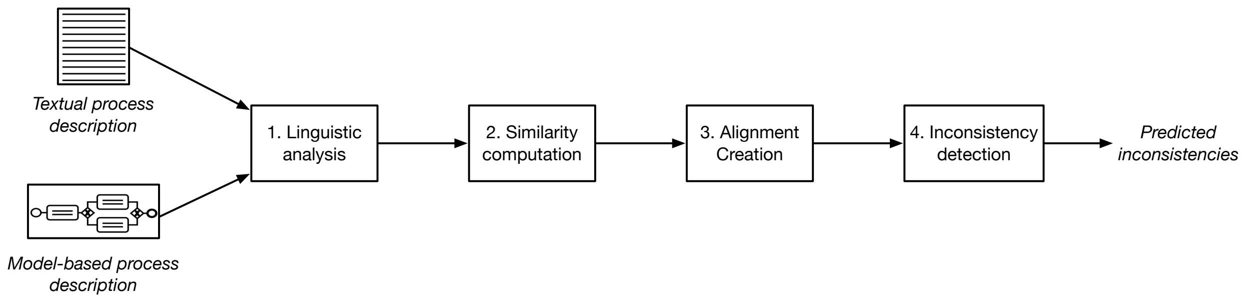
Textual process description
Model-based process description
1. Linguistic analysis
2. Similarity computation
3. Alignment Creation
4. Inconsistency detection
Predicted inconsistencies

Figure 2: Overview of the proposed approach.

3.2. Linguistic Analysis

In order to create an accurate activity-sentence alignment for a model-text pair, we first subject the textual process description and the activity labels to a linguistic analysis. In this step we make use of the *Stanford Parser*, a widely employed natural language processing tool [28]. Among other functions, the parser provides base forms of words (i.e. *lemmatization*), *part-of-speech* tags, and grammatical relations.

The linguistic analysis consists of three parts: (i) anaphora resolution, (ii) main clause extraction, and (iii) text sanitization. With these three sub-steps, we aim to obtain a representation that accurately reflects the important parts of a sentence, while omitting details that negatively impact an accurate determination of the similarity between an activity and a sentence. To illustrate the three sub-steps, we consider their impact on sentence (8) from the running example. We represent the original sentence as a bag-of-words[1]: s_8 = {"*if*", "*it*", "*is*", "*not*", "*available*", "*it*", "*is*", "*back-ordered*"}.

3.2.1. Anaphora Resolution

A problem that must be tackled when analyzing textual process descriptions is the resolution of anaphoric references or *anaphors*. Anaphors are usually pronouns ("*he*", "*her*", "*it*") or determiners ("*this*", "*that*") that refer to a previously introduced textual unit. These references represents an important challenge when creating a model-text alignment, because, if unresolved, they obscure information relevant to the similarity between activities and sentences. As an example, consider sentence s_8 from the running example and activity a_5, "*back-order part*". s_8 and a_5 both describe the act of back-ordering a "*part*", which represents the business object of the process step. However, s_8 does not explicitly describe this business object; the sentence rather uses the pronoun "*it*" to refer to the term "*part*" contained in its preceding sentence. The reference thus obscures information that affects the similarity between s_8 and a_5, namely that both describe actions applied to the same business object. To overcome this problem, we introduce an anaphora resolution technique that resolves these backward references.

The anaphora resolution technique in our approach sets out to identify the objects contained in a sentence. We identify objects by considering *Stanford Dependencies*, which reflect grammatical relations between words [29]. To identify objects in a sentence, the most important relations to consider include *direct objects* and *nominal subjects*. The direct object relation (dobj) denotes the business object in active sentences. For instance, in sentence (2), the relation *dobj(receives, order)* indicates that "*order*" is the object being received. Nominal subjects similarly identify business objects in passive sentences, as seen in sentence (7). There, *nsubj(reserved, part)* shows that the

[1]In the interest of readability, we preserve the original order of the words.

object "*part*" is reserved. The *dobj* and *nsubj* relations identify the noun, i.e. the main term, of a business object. Business objects, however, can comprise multiple terms, e.g. "*customized bicycles*" and "*the part list*". To complete the identification of a business object, we therefore also take word specifiers related to the identified object or nominal subject into consideration. Common types of specifiers include adjectival modifiers (e.g. "*small*", "*customized*", "*electronic*"), possession modifiers (e.g. "*customer's*", "*his*"), and compounds (e.g. "*sales department*"). The most important grammatical relations for these specifiers include *amod* (for adjectives), *nn* (compounds), and *poss* (possession modifiers).

Once the business objects of a sentence are identified, we check if all objects in the sentence are anaphoric references. This is the case when the sentence includes only pronouns and determiners as business objects. For these sentences, we resolve the references by replacing pronouns with objects of the preceding sentence. For instance, in sentence s_8, we replace the two occurrences of the pronoun "*it*" with "*part*", which is the business object of the preceding sentence s_7. As a result, we successfully resolve the anaphoric references and obtain the following bag-of-words representation for s_8: {"*if*", "*part*", "*is*", "*not*", "*available*", "*part*", "*is*", "*back-ordered*"}.

3.2.2. Relevant Clause Extraction

Sentences in a textual description describe actions that are performed in a process, its flow, and additional information. To accurately align process model activities, it is important to identify (parts of) sentences related to actions, while excluding parts unrelated to these actions from consideration. The most problematic cases are sentences with a *dependent clause* specifying a condition that contains terms similar or equal to those used in the activity labels. The dependent clause of sentence s_{11} represents an example of such a case: "*If the storehouse has successfully reserved or back-ordered every item of the part list and the preparation activity has finished* [...]" This clause has a high term similarity with the activities "*reserve part*" and "*back-order part*". However, it is clear that these activities are actually described earlier in the textual description. The conditional statement merely describes the requirement that these activities must have been completed. To nullify the impact that conditional statements have on similarity scores, we exclude such dependent clauses by extracting the main clause from the sentence.

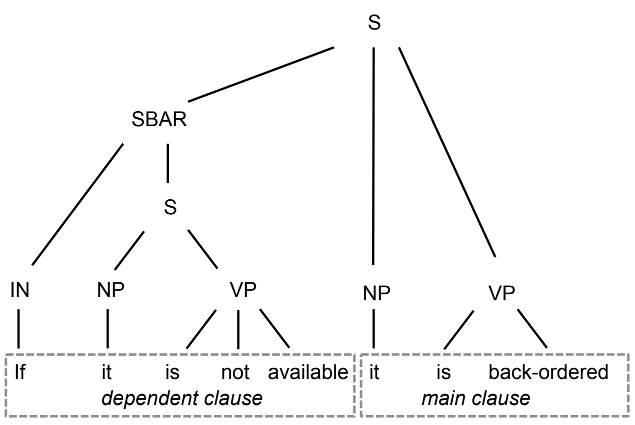

Figure 3: Simplified parse tree for sentence s_8.

In order to differentiate between conditional statements and main clauses, we use *parse trees* generated by the Stanford Dependency Parser. In these trees, conditional expressions are represented as subordinate clauses (SBARs), as seen for the parse tree of sentence s_8, shown in Figure 3. The subordinate clauses of interest start with a term that signals that the clause describes a condition, e.g. "*if*", "*in case*", or "*once*". For these sentences, we extract the remaining clauses from the parse tree. As a result, the following bag-of-words remains for s_8: {"*part*", "*is*", "*back-ordered*"}.

3.2.3. Text Sanitization

The final linguistic analysis sub-step involves text sanitization on both (previously processed) sentences and activity labels. Text sanitization sets out to create a similar and comparable representation of activity labels and sentences. The sanitization applied here comprises the removal of stop words and word lemmatization.

First, we remove *stop words* from each activity label and sentence. Stop words are common words that are of little value when considering similarity between texts [30]. We remove *closed class* determiners, prepositions, and conjunctions (e.g. "*the*", "*in*", "*to*", "*for*") from the activity labels and sentences. This procedure is in line with many approaches from the domain of process model matching (see e.g. [31, 32, 33]). Second, we lemmatize the remaining words using the Stanford Parser. The resulting *lemmas* represent grammatical base forms of words. By considering lemmas, it is straightforward to determine whether two words have a similar root. E.g. "*sing*", "*sang*", and "*sung*" are all mapped to the common lemma "*sing*" [34].

Text sanitization concludes the linguistic analysis, which is the first main step in our approach. For sentence s_8, this results in the final bag-of-words representation: {"*part*", "*be*", "*back-order*"}. The next step takes the processed activity labels and sentences as input, to quantify their semantic similarity.

3.3. Similarity Computation

The ability to judge the similarity between a sentence and an activity is critical to the performance of our approach. A sentence s and an activity a are considered to be similar if they refer to the same stream of action. To accurately assess this, we have to take the variability of natural language expressions from the sentences into account [35]. Hence, we consider two key factors in the context of our similarity computation: word specifity and semantic word similarity.

Word specificity refers to the discriminatory power of terms in a given context. It is an important factor for the similarity computation, since the occurrence of a relatively rare term in a and s is a much better indicator of the similarity between a and s than the occurrence of two rather common terms. Consider, for instance, the term "*part*" from the bicycle manufacturing example. A brief look at the model and text reveals that this term occurs in a large number of sentences and activities and, hence, has only little discriminatory power. The term "*receive*", by contrast, only occurs in sentence s_2 and in activity a_1. Thus, this term is likely to indicate that s_2 and a_1 describe the same process step, whereas this is not the case for the occurrence of "*part*" in, e.g. sentence s_6 and the activity "*back-order part*". To account for word specificity in the similarity calculation, we use the *inverse document frequency* (idf), which assigns a low score to common terms, and a high score to relatively rare terms. The idf for a term t in the collection of process model activities A and sentences S, i.e. $\mathcal{L} = A \cup S$, is given by Equation 1.

$$idf(t, \mathcal{L}) = \log \frac{|\mathcal{L}|}{|l \in \mathcal{L} \; : \; t \in l|} \tag{1}$$

Semantic similarity represents the degree to which terms have a similar meaning. Considering the semantic similarity of terms allows us to recognize when syntactically different terms are likely to refer to the same unit. Typically, terms with a high semantic similarity occur as synonyms, e.g. "*cancel*" and "*abort*", or hypernyms, e.g. "*vehicle*" and "*car*". A number of different metrics and implementations exist that quantify the semantic similarity between terms. In this paper, we employ the semantic similarity measure proposed by Lin, because it has proven to correlate well with human judgments of semantic similarity [36].

We combine the *idf* and *Lin* similarity scores to compute a semantic similarity score for an activity a and sentence s. in particular, we use the measure proposed by Mihalcea et al. [37], given by Equation 2.

$$sim(a,s) = \frac{1}{2}\left(\frac{\sum_{t\in\omega_a} maxSim(t,\omega_s)\times idf(t)}{\sum_{t\in\omega_a} idf(t)} + \frac{\sum_{t\in\omega_s} maxSim(t,\omega_a)\times idf(t)}{\sum_{t\in\omega_s} idf(t)}\right) \tag{2}$$

$$maxSim(t,\omega) = \max\{Lin(t,t_2) \mid t_2 \in \omega\} \tag{3}$$

Here, $maxSim(t,\omega)$ denotes the maximum semantic similarity between a term t and any term t_2 contained in a bag of words ω, as given by Equation 3. This maximum similarity is multiplied by the idf for each of the terms in ω_a and ω_s. Ultimately, $sim(a,s)$ yields a score in the range [0..1], where 1 indicates perfect similarity between the terms of a and s.

The similarity between a sentence and an activity plays an important role in the creation of an alignment between a process model and a textual description. To improve the correctness of the generated alignment, our approach also considers the order in which activities and sentences appear. Section 3.4 describes the details of the alignment creation, which is the third step in our overall approach.

3.4. Alignment Creation

This section describes how we obtain an optimal alignment between the activities of a process model and the sentences of a textual description. An alignment σ consists of a number of *pair-wise correspondences* between an activity set A and a set of sentences S. Each correspondence relates a single activity to a sentence, denoted with $a \sim s$. Our approach aligns each activity to exactly one sentence such that it maximizes the sum of similarity scores of the individual correspondences, while respecting the ordering constraint we describe below. As a result, multiple activities can be aligned to the same sentence. We refer to the alignment obtained in this manner as the optimal alignment $\hat{\sigma}$.

The ordering constraint we impose on the optimal alignment aims at excluding individual correspondences that violate the nature of textual descriptions. Recognizing that descriptions generally describe process steps in a chronological order [38], the rationale is to only allow for correspondences that comply with this logic. To this end, we introduce the strict order relation $\rightsquigarrow \subset S \times S$, which captures the order of the sentences in S. The expression $s_i \rightsquigarrow s_j$ means that s_i precedes s_j in the textual description. To describe the order between activities in a process model, we introduce the partial order relation $\leq \subset A \times A$. This partial order relation $a_k \leq a_l$ expresses that activity a_k must occur before a_l in case both activities are executed. Hence, it takes into account that process models may contain *alternative* or *concurrent* execution paths. To avoid conflicting correspondences as described above, we require that two correspondences $a_k \sim s_j$ and $a_l \sim s_i$ cannot both be included in an alignment σ if $a_k \leq a_l$ and $s_i \rightsquigarrow s_j$.

The consideration of these ordering constraints improves the ability of our approach to identify the correct correspondences between activities and sentences. With these constraints, the approach considers the semantic similarity of an activity a and a sentence s, but also their position relative to other correspondences included in an alignment. As such, the approach identifies correspondences more accurately, which leads to a better detection of inconsistencies.

Because process models are not purely sequential, finding the optimal alignment is not straightforward. Different combinations of activity execution orders must be considered as a possible solution in which the activities are contained in the textual process description. This can result in a huge number of possible correspondences. Hence, this optimization problem calls for an efficient solving approach. Therefore, to find the optimal alignment $\hat{\sigma}$, we adopt a *best-first search* algorithm similar to those used in machine translation problems [34]. Instead of aligning one language to another, we here align the activities of A with sentences of S. Intuitively, the best-first search algorithm traverses a search space of partial hypotheses, which consists of activity-sentence alignments between A and S. The algorithm explores the search space by expanding the partial hypothesis with the highest possible score. Because the approach exempts unpromising hypotheses from expansion, the explored search space is greatly reduced. Since the algorithm merely affects computational efficiency – not the resulting optimal alignment $\hat{\sigma}$ – we abstract from further details and refer the interested reader to [34, 39] for a detailed description. We use the obtained optimal alignment $\hat{\sigma}$ as input for the final step of the approach: the inconsistency detection step.

3.5. Inconsistency Detection

This section describes the detection of inconsistencies by analyzing the optimal alignments generated in the previous step of our approach. For this analysis, we make use of so-called *predictors.* These predictors are metrics, designed to correlate with characteristics of the generated alignments that differ between consistent and inconsistent model-text pairs.

Section 3.5.1 introduces the general notion and desired properties of predictors for the detection of inconsistencies. Then, Section 3.5.2 describes the predictors proposed to detect missing activities and Section 3.5.3 the predictors that detect conflicting orders between model and text.

3.5.1. Detection using Predictors

Given an optimal alignment $\hat{\sigma}$ between an activity set A and a sentence set S, a predictor quantifies the probability that $\hat{\sigma}$ does not contain correct correspondences. This notion of a predictor is inspired by according notions used to analyze alignments in the context of schema and process model matching [17, 40]. The core premise underlying the predictors is that the similarity scores in the optimal alignments have different characteristics for consistent and inconsistent model-text pairs. In a consistent pair, every activity is aligned to a sentence with a high similarity score in the optimal alignment, while this is not the case for inconsistent model-text pairs. The proposed predictors adhere to the desired structural properties of matching predictors: generalization and tunability [41]. *Generalization* reflects the applicability of predictors to tasks of different granularity levels. In the context of this work, we achieve this by defining predictors that can detect inconsistencies at two levels of granularity: the activity-level and the process-level. *Tunability* refers to the ability to tune predictors such that they put more emphasis on different quality aspects of the results. The proposed predictors meet this requirement by allowing users to alternate between higher precision or higher recall.

We introduce two sets of predictors, each designed to detect a specific kind of inconsistency between model and text, i.e. *missing activities* and *conflicting orders*. Each set builds on a different characteristic that distinguishes consistent model-text pairs from model-text pairs with a particular inconsistency. In particular, to detect missing activities, predictors focus on absolute and relative similarity scores. For the detection of conflicting orders, predictors instead quantify the impact of ordering constraints on the correspondences included in an optimal alignment. The following sections describe these characteristics and the related predictors in detail.

3.5.2. Missing Activities

A model-text pair contains missing activities if the textual process description does not describe all activities contained in its related process model. In a model-text pair where all the activities in A are also described by the set of sentences S, then we expect that each activity $a \in A$ is aligned to a sentence $s \in S$ with a high similarity score $sim(a, s)$. By contrast, if an activity a is not described in a textual description, i.e. a is a *missing activity*, then a is not aligned to a sentence with a high similarity score. To distinguish between high and low similarity scores, a single score can be evaluated according to three dimensions: (i) by its absolute value, (ii) by its value relative to the similarity scores of the activity with other sentences, i.e. a horizontal comparison, and (iii) by its value relative to the similarity scores of other activities, i.e. a vertical comparison. We introduce predictors to operationalize each of these perspectives. Each predictor presented here can be applied on an *activity-level* to detect individual missing activities and on a *process-level* to identify model-text pairs that contain one or more missing activities. The activity-level predictors can be used to detect inconsistencies at a fine-granular level of detail, whereas process-level predictors can be used to identify inconsistent model-text pairs with a higher accuracy. This higher accuracy stems from the fact that it is inherently easier to predict that *something* is wrong than to predict *what* exactly is wrong.

First, we consider the absolute similarity score of a correspondence $a \sim s$ contained in an optimal alignment $\hat{\sigma}$. A similarity score quantifies the likelihood that activity a and sentence s describe the same part of a process. So, a higher value straightforwardly implies that activity a is less likely to be missing. By contrast, a low similarity score indicates that the sentence is not so similar to an activity. Such an activity is, therefore, more likely to be missing from the textual description. To capture this property, we introduce a predictor *p-sim*. We apply this predictor at activity- and process-levels as follows:

- *p-sim*(a): the likelihood that activity a represents a missing activity, given as the similarity score $sim(a, s)$ of the correspondence $a \sim s \in \hat{\sigma}$;
- *p-sim*$(\hat{\sigma})$: the likelihood that a model-text pair contains one or more missing activities, given by the lowest similarity score $sim(a, s)$ contained in the optimal alignment $\hat{\sigma}$.

A second property that influences the confidence to be placed in a correspondence $a \sim s$ relates to the difference between $sim(a, s)$ and the similarity between a and other sentences in S. The similarity between an activity and a sentence is influenced by factors such as terminology and the amount of additional details that a sentence provides for a given process step. Such factors can lead to considerable differences in the similarity scores between correct activity-sentence correspondences. Therefore, we also define predictors that consider a similarity score relative to other scores in a similarity matrix between A and S. The underlying notion is commonly applied in schema matching in the form of a *dominates* property (see e.g. [17, 42]). These predictors build on the premise that a sentence is more likely to describe an activity a if it is more similar to a than other sentences are. The left-hand similarity matrix presented in Table 1 illustrates this. Both activities a_2 and a_3 are aligned to sentences with a score of 0.5. For a_2 it is clear that its corresponding sentence s_2 is the most similar in the textual description. For a_3, this clarity is missing. Two others sentences are as similar to a_3 as its corresponding sentence s_3. This implies that s_3 is not more similar to a than these other sentences, which reduces the likelihood that the sentence actually describes the same process step as a. Despite their equal similarity scores, a_3 is more likely to be a missing activity than a_2. To capture this, we define the predictor *diff-S*:

- *diff-S*(a): the difference between $sim(a, s)$ for $a \sim s \in \hat{\sigma}$ and the average similarity score of a to sentences in S;
- *diff-S*$(\hat{\sigma})$: the lowest value *diff-S*(a) for any activity in A, i.e. $\min\{$*diff-S*$(a) \mid a \in A\}$.

Table 1: Examplary similarity matrices with bolded correspondences

	s_1	s_2	s_3	s_4
a_1	**0.6**	0.3	0.2	0.0
a_2	0.2	**0.5**	0.1	0.1
a_3	0.2	0.5	**0.5**	0.5

	s_1	s_2	s_3	s_4
a_4	**0.9**	0.3	0.2	0.0
a_5	0.2	0.4	**0.9**	0.1
a_6	0.2	0.5	**0.5**	0.5

The third and final property to consider when evaluating a similarity score compares its value to other similarity scores contained in an optimal alignment. Like *diff-S*, this property also considers $sim(a, s)$ relative to other scores in the similarity matrix in order to cope with the differences that can exist among the similarity scores of correct correspondences. However, we here perform a vertical rather than a horizontal comparison. The premise underlying this property is that a correspondence $a \sim s$ is less likely to represent a correct correspondence if its similarity score is much lower than other similarity scores contained in the optimal alignment $\hat{\sigma}$. This is illustrated in the two similarity matrices presented in Table 1. The three correspondences of the left-hand matrix have an average similarity score of 0.53. For the right-hand matrix, this average score is 0.77. A score of 0.50 is thus close to the average score of the left-hand matrix. While, by contrast, this same score is much further below the average score of the right-hand matrix. The correspondence $a_6 \sim s_3$ is thus much less similar than the average similarity of the correspondences. Therefore, a_6 is more likely to be inconsistent than a_3, despite their equal similarity scores. We operationalize this factor with the predictor *diff-A* as follows:

- *diff-A*(a): the difference between $sim(a, s)$ for $a \sim s \in \hat{\sigma}$ and the average similarity scores of the correspondences in $\hat{\sigma}$;
- *diff-A*$(\hat{\sigma})$: the lowest value *diff-A*(a) for any activity in A, i.e. $\min\{$*diff-A*$(a) \mid a \in A\}$.

The predictors *p-sim*, *diff-S*, and *diff-A* each predict inconsistencies in the form of missing activities by quantifying a property that allows us to distinguish between high and low similarity scores. Section 3.5.3 describes a predictor designed to detect model-text pairs that have conflicting orders.

3.5.3. Conflicting Orders

The second type of inconsistency we consider occurs when a process model and an accompanying textual description describe the steps of a process in different orders. For example, the process model of the running example shows that the "*Inform S & E departments*" activity precedes "*Enter details into ERP system*", whereas the textual description states that these activities occur in the reverse order. The ordering constraints we impose on the creation of alignments bars our approach from including correspondences between activities and sentences when their orders do not match. The existence of conflicting orders between model and text, therefore, manifests 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.

Table 2: Fragment of the similarity matrix for the running example

	s_2	s_3	s_4	s_5	s_6
a_2	0.1	**0.2**	0.2	0.7	0.2
a_3	0.1	0.1	**0.9**	0.2	0.1

To illustrate this, consider the fragment of the similarity matrix in Table 2, where a_2 refers to "*Inform S & E departments* and a_3 to "*Enter details into ERP system*". The similarity score $sim(a_2, s_5)$ is high (0.7), because both a_2 and s_4 describe the same process step. The similarity score for $sim(a_3, s_4)$ is even higher (0.9), for the same reason. However, since a_2 precedes a_3 in the model ($a_2 \leq a_3$) and s_4 precedes s_5 in the textual description ($s_4 < s_5$), $a_2 \sim s_5$ and $a_3 \sim s_4$ cannot both be contained in $\hat{\sigma}$ without violating the imposed constraints. As a result, the approach instead aligns a_2 to s_3, despite its considerably lower similarity score.

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 activity 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*($\hat{\sigma}$): the maximal difference between the potential and aligned similarity scores for an activity $a \in A$;

The predictors defined for the detection of missing activities and conflicting orders each quantify conceptual notions that differ between the similarity scores of consistent versus inconsistent model-text pairs. In Section 4, we present a quantitative evaluation that demonstrates how well our approach, which incorporates these predictors, is able to detect inconsistencies in practice.

4. Evaluation

This section presents a quantitative evaluation that demonstrates how well the proposed approach is able to identify inconsistencies in model-text pairs. We have manually annotated the inconsistencies in a collection of 53 model-text pairs obtained from practice. This annotation is referred to as the *gold standard* against which we compare the results of our approach. Section 4.1 describes this test collection in detail. Subsequently, sections 4.2 through 4.4 respectively describe the set-up of the evaluation, its results, and a discussion of the strength and weaknesses of our approach.

4.1. Test Collection

To evaluate our approach, we use a collection of 53 model-text pairs that originate from different sources, including academia, textbooks, industry, and public sector organizations. The majority of the model-text pairs were introduced in [14], while we extended this collection with 7 processes obtained from a new industrial source. Table 3 presents the main characteristics of the model-text pairs contained in the test collection. The included process models are heterogeneous with regard to several dimensions, such as size and complexity. Also, the corresponding textual descriptions vary in several aspects. For instance, they describe the processes from different perspectives (first and third person) and differ in terms of how explicitly and unambiguously they refer to the process model content. Finally, it is important to note that the ratio of sentences per activity (S/A) differs significantly throughout the collection. Some sources use a single sentence on average per activity, while other sources use up to 7 sentences. This increased ratio follows from the presence of more informational sentences and details on how certain process steps must be executed (i.e. work instructions). Due to these varying characteristics, we believe that the collection is well-suited to achieve a high external validity of the results.

Table 3: Overview of the test collection

ID	Source	Type	P	P_m	P_{ma}	P_o	A	S	S/A
1	HU Berlin	Academic	4	1	1	0	9.0	10.3	1.1
2	TU Berlin	Academic	2	2	2	0	22.5	34.0	1.5
3	QUT	Academic	8	0	0	0	6.1	7.1	1.2
4	TU Eindhoven	Academic	1	1	2	0	18.0	40.0	2.2
5	VU Amsterdam	Industry	7	1	2	0	4.3	30.1	7.0
6	Vendor Tutorials	Industry	3	2	3	2	5.3	7.0	1.3
7	inubit AG	Industry	4	1	1	1	9.0	11.5	1.3
8	BPM Practitioners	Industry	1	0	0	0	4.0	7.0	1.8
9	BPMN Practice Handbook	Textbook	3	2	2	1	5.0	4.7	0.9
10	BPMN Guide	Textbook	6	5	7	0	7.0	7.0	2.2
11	Federal Network Agency	Public Sector	14	4	6	0	8.0	6.4	0.8
	Total	-	**53**	**19**	**26**	**4**	**7.6**	**12.0**	**1.6**

Legend: P = Model-text pairs per source, P_m = Model-text pairs with missing activities, P_{ma} = Missing activities, P_o = Model-text pairs with conflicting orders, A = Activities per model (avg.), S = Sentences per text (avg.), S/A = Sentences per activity (avg.)

To provide a basis for comparison, we manually identified the inconsistencies in the 53 model-text pairs. We involved three researchers in the creation of this gold standard. Two of them independently identified inconsistencies for each model-text pair. The inter-annotator agreement was high, having only 4 initial disagreements on whether or not an activity was missing. The cause for discussion involved implicitly described actions, such as seen for the "*select unchecked part*" activity in the bicycle manufacturing example. The differences were resolved in a discussion, involving the third researcher to settle ties. Out of the 406 activities contained in the process models, 26 are considered to be missing in the textual description. These activities occurred in 19 different model-text pairs. Furthermore, 4 model-text pairs were found to contain conflicting orders.

4.2. Setup

We evaluate the performance of our approach by comparing its predictions to the manually created gold standard. We compare the performance in practice of three predictors for the detection of missing activities, *p-sim*, *diff-S*, and *diff-A*, both at the level of activity and process. For the detection of model-text pairs with conflicting orders, we evaluate the performance of the *max-*

constrained predictor. To demonstrate the usefulness of this predictor, we compare its performance to a baseline. For this baseline, we create an alignment without imposing ordering constraints and afterwards check if this alignment violates any ordering constraints. We refer to the heuristic that identifies these cases as *base-check*.

Aside from the choice for a certain predictor, the performance of the approach depends for a major part on the alignments it generates. Our approach to create these alignments consists of several components, including the linguistic analysis and the inclusion of ordering constraints. To demonstrate the added value of the individual components, we test the performance of our approach using different configurations to generate the alignments. They include the following:

- **Baseline (BL):** As a baseline configuration, we aligned every activity a to the sentence s with the highest value for $sim(a, s)$;
- **Linguistic analysis (LA):** For this configuration, prior to the computation of similarity scores, we applied all linguistic analysis activities described in Section 3.2. We thus subjected the textual description to (i) text sanitization, (ii) resolved anaphoric references, (iii) and extracted relevant clauses;
- **Ordering constraints (OC):** This configuration computes an alignment between activity set A and sentence set S that achieves a maximal similarity score, while respecting the ordering constraints implied by the partial order of the process model and the strict order of the textual description;
- **Linguistic analysis + ordering constraints (LA + OC):** For the full configuration, we performed all linguistic analysis activities *and* imposed ordering constraints on the optimal alignment $\hat{\sigma}$.

We assess the performance of our approach with standard information retrieval metrics. More specifically, we calculate precision, recall, and F_1 score by comparing the generated results against the manually created gold standard. *Precision* describes the fraction of predictions that are correct. *Recall* represents the fraction of all inconsistencies that are identified by our approach. We define these metrics in the context of this work as given by Equations 4 and 5.

$$\text{precision} = \frac{|C_I \cap C_\rho|}{|C_\rho|} \quad (4) \qquad\qquad \text{recall} = \frac{|C_I \cap C_\rho|}{|C_I|} \quad (5)$$

We use C_ρ to denote the set of model-text pairs or activities that are predicted to be inconsistent with a prediction score in the range $[0.00, \rho]$. A higher value for ρ thus increases the number of activities or model-text pairs that are included in C_ρ, i.e. that are predicted to be inconsistent. C_I denotes the set of activities or model-text pairs that contain a certain type of inconsistency. Finally, we also report the F_1 *score*, which provides the harmonic mean between precision and recall.

4.3. Results

This section presents the results of the quantitative evaluation. Section 4.3.1 assesses the ability of the predictors to detect inconsistencies. Then, Section 4.3.2 illustrates how the individual components of the alignment approach contribute to the quality of the obtained results.

4.3.1. Predictor Performance

We computed precision and recall scores for increased values of predictor score ρ for each of the predictors for the different types of inconsistencies. The precision-recall graphs of Figure 4 and Figure 5 depict the performance of the approach when detecting missing activities at the activity level and process level. Both graphs illustrate the trade-off between precision and recall. The reason for a trade-off to exist is that increasing the recall requires the consideration of lower confidence values. As a result, some activities are erroneously classified as missing, which negatively affects precision. The maximum results achieved by the predictors are presented in Table 4. This table also displays the predictor value that achieves the maximum F_1 score, which we here refer to as the optimal predictor value.

Missing Activities The precision-recall graph of Figure 4 shows that our approach is able to detect missing activities with a good accuracy. The approach reaches a maximum F_1 score of 0.44 for the *p-sim* predictor. At this point, 17 missing activities have been detected (recall = 0.58) with a precision of 0.36. These results should be considered in light of the relatively low fraction

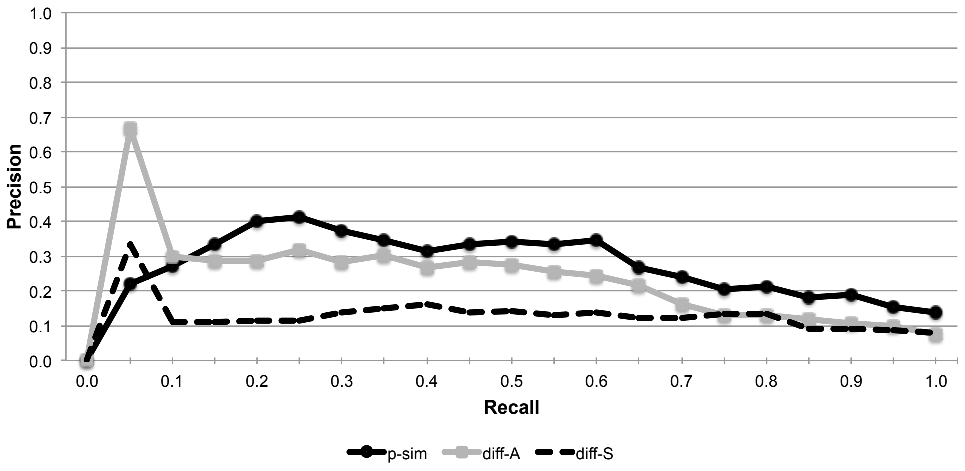

Figure 4: Precision-recall graph for the detection of missing activities (activity-level)

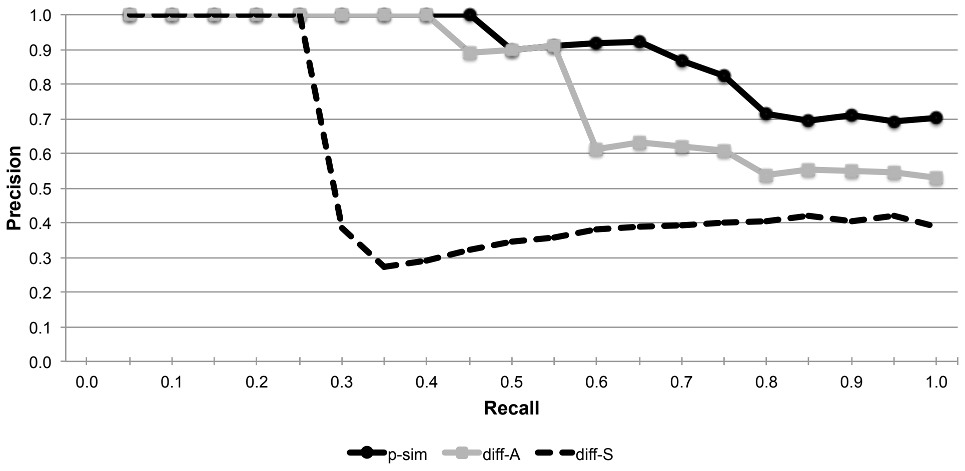

Figure 5: Precision-recall graph for the detection of model-text pairs with missing activities (process-level)

of the total activities that are missing (0.07). The graph shows that beyond a recall of 0.15, *p-sim* consistently outperforms the other predictors. The *diff-A* predictor reaches a maximum precision of 0.67 and a maximum F_1 score of 0.37. The predictor *diff-S* has the lowest performance, with a maximum precision of 0.33 and F_1 score of 0.21.

The precision-recall graph of Figure 5 shows that the approach performs even better when detecting model-text pairs with missing activities, particularly using the *p-sim* predictor. This predictor correctly identifies more than 40% of the model-text pairs with missing activities with perfect precision. For increasing recall values, the approach maintains a high precision. The highest obtained F_1 score of 0.83 is reached when all incorrect model-text pairs have been detected (i.e. recall = 1.00), with a precision of 0.70. The *diff-A* and *diff-S* predictors display lower prediction accuracy, they reach maximal F_1 scores of 0.69 and 0.58, respectively.

Table 4: Predictor performance evaluation results

Predictor	**Maximum precision**	**Maximum F_1 score**	**Optimal predictor value**
p-sim(a)	0.39	0.44	0.74
diff-A(a)	0.67	0.37	1.18
diff-S(a)	0.33	0.21	0.92
p-sim$(\hat{\sigma})$	1.00	0.83	0.74
diff-A$(\hat{\sigma})$	1.00	0.69	0.52
diff-S$(\hat{\sigma})$	1.00	0.58	0.47
base-check(σ)	0.16	0.28	n/a
max-constrained$(\hat{\sigma})$	1.00	0.73	0.18

Conflicting Orders Finally, we evaluated the performance of the *max-constrained* metric for the detection of model-text pairs with conflicting orders and compared it to the baseline metric *base-check*. We omit the depiction of a precision-recall graph because there are only four cases with inconsistent orders in the test collection. The *max-constrained* predictor identifies these cases with high accuracy. The predictor reaches a maximum precision of 1.00 and successfully identifies all inconsistent model-text pairs with a precision score of 0.57. This results in a maximum F_1-score of 0.73. Furthermore, we can observe that the *max-constrained* predictor greatly outperforms the baseline metric. The reason for this is that *base-check* yields a considerable amount of false positives, resulting in a precision of 0.16 and a maximum F_1-score of 0.28. These false positives occur because a correct alignment does not just depend on the term similarity between activities and sentences, but also on the order in which they are described. As a result, the gold standard contains a number of model-text pairs in which one or more activities are not aligned to the sentences with the highest similarity score. In these cases, instead, the activity is generally aligned to a sentence with a slightly lower score (e.g. a score of 0.58 versus a score of 0.60). The *base-check* predictor is not able to differentiate between such cases and cases that truly contain conflicting orders, which are characterized by much larger differences in similarity scores. By contrast, the *max-constrained* predictor successfully differentiates between small and large differences in similarity scores, which results in a considerably better performance.

4.3.2. Alignment Configurations

The results presented in the previous sections demonstrate that our approach is able to detect inconsistencies with high accuracy by using the full configuration for the creation of alignments. Table 5 presents the performance of the approach for different configurations of the alignment approach. For the sake of brevity, we depict the highest F_1 score achieved by each predictor for a given configuration.[2] The results clearly illustrate that each of the individual components of the alignment approach contribute positively to the prediction accuracy, since the scores of LA and OC are higher than the baseline scores reached by BL.

Table 5: Highest F_1-measures for different configurations and predictors.

Predictor	**BL**	**LA**	**OC**	**LA + OC**
p-sim(a)	0.22	0.32	0.23	0.44
diff-A(a)	0.20	0.29	0.21	0.35
diff-S(a)	0.15	0.18	0.15	0.23
p-sim$(\hat{\sigma})$	0.62	0.72	0.66	0.83
diff-A$(\hat{\sigma})$	0.61	0.64	0.61	0.69
diff-S$(\hat{\sigma})$	0.54	0.58	0.55	0.58
max-constrained$(\hat{\sigma})$	n/a	n/a	0.50	0.73

The linguistic analysis improves the performance for all use cases and predictors. Notably, it increases the maximum performance for the detection of missing activities with 45%, from 0.22 to 0.32. The performance for the detection of model-text pairs with missing activities is similarly increased from a maximum of 0.63 to 0.72. Though the inclusion of ordering constraints consistently improves the results in comparison to the baseline configuration, it does not yield gains as large as the linguistic analysis. For instance, the detection of model-text pairs with missing activities is improved from 0.63 to 0.66. However, the usefulness of ordering considerations is particularly apparent when the constraints are incorporated in combination with the linguistic analysis. Then,

[2]Note that the BL and LA configurations cannot detect inconsistent orders, since these configurations do not incorporate the ordering restrictions required to identify these.

the results are improved from 0.32 to 0.44 and from 0.72 to 0.83 for the detection of missing activities at the activity level and process level, respectively.

4.4. Discussion

The evaluation shows that the approach successfully identifies inconsistencies between model-text pairs. Especially at the process-level, the approach detects erroneous model-text pairs with a high prediction accuracy. The lower performance scores for detection at the activity-level can be attributed to two causes. First, it is inherently easier to detect that inconsistencies exist in a model-text pair than to identify exactly *where* these occur. Second, the difference occurs because the approach creates an alignment optimized for the entire process. For inconsistent model-text pairs, it can therefore happen that a missing activity also impacts the alignments of other activities in the process. As a result of such a shift, sometimes the wrong activities are predicted to be missing. This leads to lower precision at the activity-level, while the prediction accuracy at the process-level remains unaffected.

The results for the different predictors demonstrate that predictors considering the absolute similarity values outperforms others both at the the activity- and process-levels. This implies that the absolute similarity value of an activity-sentence pair has the strongest correlation to inconsistencies from the considered factors. However, the performance of the other predictors indicates that the consideration of similarity relative to other scores also has its merits. This applies in particular to the *diff-A* metric, which compares similarity scores relative to other scores included in an optimal alignment. This predictor still achieves considerable prediction accuracy, despite achieving lower performance scores than the *p-sim* predictor. The results furthermore demonstrate that the approach can identify model-text pairs with conflicting orders between the process steps described in the process model and those in the textual description. The *max-constrained* predictor, which compares the maximum difference between a similarity scored in the optimal alignment and the the potential similarity score for an activity, greatly outperforms the baseline metric. It is important to note that the conclusions to be drawn from these results are impaired by the low number of model-text pairs containing such inconsistencies. Nevertheless, the approach succeeds in detecting these few cases with a high precision.

The obtained evaluation results have several implications for the application of the proposed approach in practice. The results reveal that both the linguistic analysis and ordering constraints consistently improve the accuracy with which inconsistencies are detected. This implies that the full configuration (BL + LA) should be used to generate alignments between model and text. The comparison of the different predictors furthermore demonstrates that the *p-sim* and *max-constrained* predictors achieve the best results on our heterogeneous data collection. As such, the evaluation shows that these predictors should be selected for application in practical settings.

Finally, it is important to note the trade-off between activity-level and process-level predictors for the detection of missing activities. The former provide more fine-granular results, while the latter predictors achieve a higher predictive accuracy. Since clearing up inconsistencies will always involve manual effort, it is, therefore, worthwhile to consider the usage of process-level predictors, to identify those model-text pairs in a collection that are most likely to contain inconsistencies.

5. Limitations

Our evaluation demonstrated that our approach achieves promising results. However, these results need to be reflected against the background of some limitations. In particular, we identify limitations related to the approach and limitations related to the evaluation.

Limitations related to the *approach* concern three aspects. First, the employed NLP techniques, such as the Stanford Parser and the anaphora resolution technique, are not fully accurate. Second, the employed Lin similarity may not be able to identify semantic relationships between synonymous words from highly specific domains. Third, our approach does not take discourse statements such as "*after*" or "*before*" into account. Instead, we use a conservative approach that only considers the order in the text, because existing techniques that analyze discourse statements have been shown to produce inaccuracies [15] As a result of these limitations, our approach may identify wrong alignments. Therefore, the proposed approach remains a prediction approach intended as a means to support users. Nevertheless, as such it greatly helps organizations to quickly detect inconsistencies between textual and model-based descriptions of their processes. Hence, the approach reduces the effort required to identify and resolve these differences in large process model repositories.

As for limitations related to the *evaluation*, we would like to point out that the presented quantitative results are bound to the specifics of the model-text pairs used in the evaluation. The employed data set is not representative in a statistical sense. In fact, the creation of a statistically representative sample is hardly feasible, since natural language offers such a high degree of freedom. Still, we tried to compose a data set that is as heterogeneous as possible by collecting model-text pairs from a broad variety of external sources. This set included only four model-text pairs with conflicting orders. This should be taken into account when interpreting the results related to this type of inconsistency. Inconsistencies in the form of missing activities were better represented in the data collection, with 26 occurrences. Thus, for this type of inconsistency we are confident that our evaluation indeed shows a realistic picture of the performance of our approach in practice.

6. Conclusions

In this paper, we presented an approach to automatically detect inconsistencies between textual and model-based process descriptions. Our approach combines linguistic analysis, semantic similarity measures, and ordering relations to obtain an alignment between the activities of a process model and the sentences of a textual process description. By building on a set of predictors, our approach detects missing activities as well as conflicting orders between model and text. A quantitative evaluation with a set of real-world processes against a manually created gold standard showed that our approach can successfully identify inconsistent model-text pairs. The evaluation revealed which predictors are particularly useful in real-life settings. It furthermore demonstrated that tailored natural language processing techniques as well as ordering restrictions positively contribute to the predictive accuracy of the approach.

With respect to future work, we identify four main directions. First, we intend to improve the predictive accuracy of our approach by addressing the limitations we have identified. One promising course for this is to make use of domain-specific information in order to overcome problems caused by the usage of specialized terms from, for example, the health-care domain. Second, we aim at extending the current approach with respect to its coverage of process dimensions beyond the control-flow perspective. For instance, we would like to check the consistency of execution conditions and other data restrictions. Third, we set out to use the alignments we obtain between model and text to address use cases beyond the detection of inconsistencies. For example, these alignments can be used to moderate the evolution of model and text over different versions of a process. Ultimately, this can lead to the direct propagation of one-sided process updates to the other mode of representation. In this way, the consistency between multiple process representations cannot only be checked, but also automatically maintained. This direction relates closely to work that exists on change propagation and version tracking for process models (cf. [43, 44]). Fourth, we recognize that organizations also capture process information in formats other than the textual and model-based descriptions considered in this paper. Common examples include checklists, rules and regulations, and spreadsheets. Hence, we aim to adapt and apply the techniques developed in this paper to this broader spectrum of process representation formats.

Bibliography

[1] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, 2013.

[2] S. Chakraborty, S. Sarker, S. Sarker, An exploration into the process of requirements elicitation: A grounded approach, J. AIS 11 (4).

[3] H. Leopold, J. Mendling, A. Polyvyanyy, Supporting process model validation through natural language generation, IEEE Transactions on Software Engineering 40 (8) (2014) 818–840.

[4] T. Allweyer, BPMN 2.0: introduction to the standard for business process modeling, BoD–Books on Demand, 2010.

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

[6] 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, 2015, pp. 3–18.

[7] E. Rahm, P. A. Bernstein, A survey of approaches to automatic schema matching, the VLDB Journal 10 (4) (2001) 334–350.

[8] F. Meziane, N. Athanasakis, S. Ananiadou, Generating natural language specifications from UML class diagrams, Requirements Engineering 13 (1) (2008) 1–18.

[9] B. Lavoie, O. Rambow, E. Reiter, The modelexplainer, in: Eighth International Workshop on Natural Language Generation, Herstmonceux, Sussex, 1996.

[10] I. S. Bajwa, M. A. Choudhary, From natural language software specifications to UML class models, in: Enterprise Information Systems, Springer, 2012, pp. 224–237.

[11] F. Gomez, C. Segami, C. Delaune, A system for the semiautomatic generation of ER models from natural language specifications, Data & Knowledge Engineering 29 (1) (1999) 57–81.

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

[13] A. Sinha, A. Paradkar, Use cases to process specifications in Business Process Modeling Notation, in: Web Services (ICWS), 2010 IEEE International Conference on, 2010, pp. 473–480.

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

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

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

[18] A. Doan, A. Y. Halevy, Semantic integration research in the database community: A brief survey, AI magazine 26 (1) (2005) 83.

[19] N. F. Noy, Semantic integration: a survey of ontology-based approaches, ACM Sigmod Record 33 (4) (2004) 65–70.

[20] P. Shvaiko, J. Euzenat, Ontology matching: state of the art and future challenges, Knowledge and Data Engineering, IEEE Transactions on 25 (1) (2013) 158–176.

[21] D. Aumueller, H.-H. Do, S. Massmann, E. Rahm, Schema and ontology matching with coma++, in: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, 2005, pp. 906–908.

[22] U. Cayoglu, R. Dijkman, M. Dumas, P. Fettke, L. García-Bañuelos, P. Hake, C. Klinkmüller, H. Leopold, A. Ludwig, P. Loos, et al., Report: The process model matching contest 2013, in: Business Process Management Workshops, Springer, 2013, pp. 442–463.

[23] M. La Rosa, M. Dumas, R. Uba, R. M. Dijkman, Business process model merging: An approach to business process consolidation, ACM Transactions on Software Engineering and Methodology (TOSEM) 22 (2) (2013) 11.

[24] R. Uba, M. Dumas, L. García-Bañuelos, M. La Rosa, Clone detection in repositories of business process models, in: Business Process Management, Springer, 2011, pp. 248–264.

[25] M. Ehrig, A. Koschmider, A. Oberweis, Measuring similarity between semantic business process models, in: Proceedings of the fourth Asia-Pacific conference on Comceptual modelling-Volume 67, 2007, pp. 71–80.

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

[27] M. Künze, M. Weidlich, M. Weske, Behavioral similarity–a proper metric, in: Business Process Management, Springer, 2011, pp. 166–181.

[28] 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.

[29] M.-C. De Marneffe, C. D. Manning, The stanford typed dependencies representation, in: Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation, 2008, pp. 1–8.

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

[31] U. Cayoglu, A. Oberweis, A. Schoknecht, M. Ullrich, Triple-s: A matching approach for petri nets on syntactic, semantic and structural level, Tech. rep., Technical report (2013).

[32] C. Klinkmüller, I. Weber, J. Mendling, H. Leopold, A. Ludwig, Increasing recall of process model matching by improved activity label matching, in: Business Process Management, Springer, 2013, pp. 211–218.

[33] M. Weidlich, R. M. Dijkman, J. Mendling, The ICoP framework: Identification of correspondences between process models, in: Advanced Information Systems Engineering, Springer, 2010, pp. 483–498.

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

[35] P. Achananuparp, X. Hu, X. Shen, The evaluation of sentence similarity measures, in: Data Warehousing and Knowledge Discovery, Springer, 2008, pp. 305–316.

[36] D. Lin, An information-theoretic definition of similarity., in: ICML, Vol. 98, 1998, pp. 296–304.

[37] R. Mihalcea, C. Corley, C. Strapparava, Corpus-based and knowledge-based measures of text semantic similarity, in: AAAI, Vol. 6, 2006, pp. 775–780.

[38] P. Schumacher, M. Minor, E. Schulte-Zurhausen, Extracting and enriching workflows from text, in: Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on, IEEE, 2013, pp. 285–292.

[39] Y.-Y. Wang, A. Waibel, Decoding algorithm in statistical machine translation, in: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, 1997, pp. 366–372.

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

[41] T. Sagi, A. Gal, Schema matching prediction with applications to data source discovery and dynamic ensembling, The International Journal on Very Large Data Bases 22 (5) (2013) 689–710.

[42] J. Wang, J.-R. Wen, F. Lochovsky, W.-Y. Ma, Instance-based schema matching for web databases by domain-specific query probing, in: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, 2004, pp. 408–419.

[43] B. Benatallah, H. R. Motahari-Nezhad, et al., Enabling the analysis of cross-cutting aspects in ad-hoc processes, in: Advanced Information Systems Engineering, Springer, 2013, pp. 51–67.

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