Aligning Textual and Graphical Descriptions of Processes Through ILP Techniques

Josep Sànchez-Ferreres, Josep Carmona(✉), and Lluís Padró

Universitat Politècnica de Catalunya, Barcelona, Spain
{jsanchezf,jcarmona,padro}@cs.upc.edu

Abstract. With the aim of having individuals from different backgrounds and expertise levels examine the operations in an organization, different representations of business processes are maintained. To have these different representations aligned is not only a desired feature, but also a real challenge due to the contrasting nature of each process representation. In this paper we present an efficient technique for aligning a textual description and a graphical model of a process. The technique is grounded on using natural language processing techniques to extract linguistic features of each representation, and encode the search as a mathematical optimization encoded using Integer Linear Programming (ILP) whose resolution ensures an optimal alignment between both descriptions. The technique has been implemented and the experiments witness the significance of the approach with respect to the state-of-the-art technique for the same task.

Keywords: Process models · Natural language processing · Integer Linear Programming

1 Introduction

Nowadays organizations store processes descriptions in various representations. The reason for this is the different nature stakeholders have: while textual descriptions of processes are well-suited for non-technical users, they are less appropriate for describing precise aspects of the underlying process [1]. In contrast, formal and graphical process notations (e.g., BPMN) are unambiguous representations which can be the basis for automating the corresponding processes within the organization [2], but they are oriented to specialized users. In this context, due to the evolving nature of processes, there is a high risk of having deviations between the different representations, a problem that may have serious consequences for any organization [3].

In the last decade, the field of *Natural Language Processing* (NLP) has grown up to a mature enough level, where the algorithmic support to analyze any text is high. Currently, there are several powerful open-source libraries that can be integrated easily to any software project, thus making linguistic analysis a reality in many contexts [4–7]. In this paper we exploit state-of-the-art NLP algorithms

E. Dubois and K. Pohl (Eds.): CAiSE 2017, LNCS 10253, pp. 413–427, 2017.
DOI: 10.1007/978-3-319-59536-8_26

to extract advanced linguistic features for the text found in both representations, so that the corresponding linguistic footprint can be mapped to a canonical form. Several similarity metrics can be defined on top of this canonical form, including weighted versions which may favor particular characteristics of process descriptions such as the action performed.

Once the similarity metric is chosen, the problem is casted as an optimization, whose solution(s) represent an assignment between tasks and sentences such that the accumulated sum of similarity is maximum. In particular, we encode the problem as an Integer Linear Programming (ILP) model whose resolution provides the optimal alignment between the text and the model.

The work of this paper is inspired by and shares the motivation of the seminal work [8,9] (see Sect. 3 for an accurate comparison of both approaches). Remarkably, although the core algorithm for searching solutions of the techniques is very different from our approach's, the quality of both approaches is similar. However, due to the simplicity of the encoding proposed, the method proposed is much faster and can deal with model-text pairs of medium/large size in a feasible time, a crucial distinctive feature of our approach with respect to [8,9]. By mapping the problem as an ILP, we clearly separate problem encoding from computation, thus allowing to easily incorporate new dimensions to consider (as we have done in this paper by incorporating actors). Notably, the technique provides a result very fast, thus widenning the application scope from post mortem or batch analysis to real-time analysis.

The research method followed in this work is *Design Science* [10], which "creates and evaluates IT artifacts intended to solve identified organizational problems".

The remainder of the paper is organized as follows: we provide a motivating example in the next section. Then, in Sect. 3 a detailed comparison with related work is reported. Preliminaries are then provided in Sect. 4, and the main contribution of the paper is presented in Sect. 5. Experiments on reference benchmarks are presented in Sect. 6. Finally, Sect. 7 concludes the paper and provides future lines for research.

2 Motivating Example

To give some intuition let us consider the example represented by the textual description and its corresponding BPMN model in Fig. 1. The technique we present derives the correct alignment between sentences 1–13 and tasks A-P, except for task J, as this task is not mentioned in the text.

In the simpler cases, the correct alignment between a task and a sentence can be obtained just by comparing the words in the sentences and the task labels. In this work, we aim to expand on previous techniques by considering more information about the tasks in the form of features. To better illustrate this, let us consider sentences 6 and 8, which correspond to tasks G and L respectively. When performing the comparison only by looking at the task labels, there is no clear way to distinguish G from L since they have the same label. Because of

(1) *When a visitor wants to become a member of Barcelona's ZooClub, the following steps must be taken.* (2) *First of all, the customer must decide whether he wants an individual or family membership.* (3) *If he wants an individual membership, he must prepare his personal information.* (4) *If he wants a family membership instead, he should prepare the information for its spouse and spawn as well.* (5) *The customer must then give this information to the ZooClub department.* (6) *The ZooClub enters the visitor's personal data into the system and takes the payment request to the Billing department.* (7) *The ZooClub department also forwards the visitor's information to the marketing department.* (8) *On receiving the request, the billing department also enters the visitor's personal data into their local database.* (9) *After that, the billing department sends the payment request to the bank.* (10) *The bank processes the payment information and, if everything is correct, charges the payment into user's account.* (11) *Once the payment is confirmed, the ZooClub department can print the card and deliver it to the visitor.* (12) *In the meantime, the Marketing department makes a request to mail the Zoo Club's magazine to the visitor's home.* (13) *Once the visitor receives the card, he can go home.*

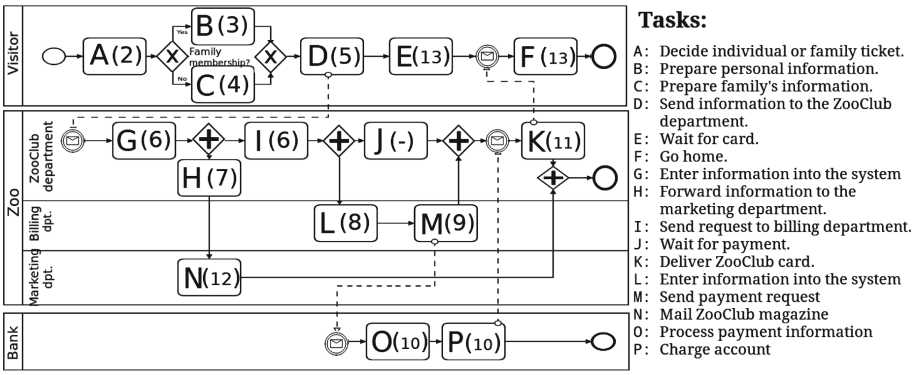

Fig. 1. Textual description and BPMN model of the Zoo business process. The correct alignment is displayed in parenthesis on the task labels.

that, there is nothing preventing both tasks from getting mapped to the same sentence. To correctly solve cases like this, semantic information is required, such as the fact that G and L are performed by different actors, so they should be assigned to sentences where the right actor is performing the task. Another helpful linguistic information can be obtained from gateways surrounding a task: Tasks following choice gateways are more likely to match sentences containing conditional statements.

3 Related Work

The contributions of this paper intersect with various works in the literature. In general, previous work can be categorized into transformations between models and text (e.g., [11,12] for UML diagrams, or [1,13] for BPMN), and schema [14] or process model [15] matching. Also, there has been work on generating process models from group stories [16] and from use-cases [17], which are less related

to this work since they restrict the form of the textual description used to describe the process. For the problem considered in this paper, the transformation approches can only be applied when the source process description is unambiguous, and the transformation used does not modify the underlying process. Hence, the rest of the section considers only the work that computes alignments without requiring a transformation between process descriptions.

The seminal work [8,9] proposed an algorithm for aligning textual descriptions and process models, with the particular aim of detecting inconsistencies between both representations. Their approach consists on using a linguistic analysis that derives a bag-of-words summary (i.e., resolving anaphoric references, extracting relevant clauses or removing prepositions) of the main elements in each representation. Then, a similarity computation between these elements is applied, and finally an optimal alignment which globally maximizes the similarity is computed, using a *best-first search* technique.

In our case, we extend the linguistic analysis with semantic role labeling, coreference resolution, and the computation of the semantic graph. Moreover, we encode the problem of computing an alignment as the resolution of an ILP model. As we will see in the experiments, this algebraic representation of the alignment problem represents a significant reduction (of several orders of magnitude) in the time requirements for computing an alignment. Finally, we map text sentences to feature vectors with a rich unbounded set of features, which do not depend on an apriori assumption on the importance of certain constructions. This rich feature representation allows to differentiate semantic roles such as *actor* or *object*, and also allows to include other process information besides the task labels.

Table 1 shows the derived alignment for the example in Sect. 2, by both our tool and the one introduced in [9]. We want to stress that in spite of our better performance for this particular example, the quality of our approach and the one in [9] is similar. We believe both contributions can be naturally combined to boost the quality of the alignments derived.

Table 1. Errors in task-to-sentence alignments produced by our approach and by [9]'s for the example in Sect. 2.

Task	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Groundtruth	2	3	4	5	13	13	6	7	6	–	11	8	9	12	10	10
[9]'s approach	✓	✓	3 ✗	6 ✗	✓	✓	✓	12 ✗	✓	6 ✗	✓	6 ✗	✓	✓	✓	✓
Our approach	✓	✓	✓	✓	✓	✓	✓	✓	✓	6 ✗	✓	✓	✓	✓	✓	✓

4 Preliminaries on Process Models and NLP

4.1 Graphical Process Notations

There exist a plethora of graphical notations to model processes. A full description of them is beyond the scope of this paper. In this paper we focus on BPMN, a notation that has become one of the most widely used to model business

processes. However, the techniques presented can be adapted to other notations like EPCs, Petri Nets, YAWL, among others.

BPMN models are composed by three types of nodes: events, activities and gateways. *Events* (represented as circles) denote something that happens (e.g., time, messages, ...), rather than *Activities* (rounded-corner rectangles) which are something that is done. Finally, *gateways* (diamond shapes) are used to describe the control flow. These elements can be partitioned into pools or lanes, to group activities performed by the same actor (person, department, institution, etc.). An example of BPMN is shown in Fig. 1.

4.2 Natural Language Processing

Natural Language Processing (NLP) is a wide research area inside Artificial Intelligence that includes any kind of technique or application related to the automatic processing of human language. NLP goals range from simple basic processing such as determining in which language a text is written, to high-level complex applications such as Machine Translation, Dialogue Systems, or Intelligent Assistants.

However, linguistic analysis tools can be used as a means to structure information contained in texts for its later processing in applications less related to language itself. This is our case, where we use NLP analyzers to convert a textual description of a BPM into a structured representation that can be compared, mapped, or analyzed using more conventional tools.

The NLP processing software used in this work is FreeLing[1] [5], an open–source library of language analyzers providing a variety of analysis modules for a wide range of languages. More specifically, the natural language processing layers used in this work are:

Tokenization and sentence splitting: Given a text, split the basic lexical terms (word, punctuation signs, numbers, ZIP codes, URLs, e-mail, etc.), and group these tokens into sentences.

Morphological analysis: For each word in the text, find out its possible parts-of-speech (PoS).

PoS-Tagging: Determine which is the right PoS for each word in a sentence. (e.g. the word *dance* is a verb in *I dance all Saturdays* but it is a noun in *I enjoyed our dance together.*)

Named Entity Recognition: Detect named entities in the text, which may be formed by one or more tokens, and classify them as *person*, *location*, *organization*, *time-expression*, *numeric-expression*, *currency-expression*, etc.

Word sense disambiguation: Determine the sense of each word in a text (e.g. the word *crane* may refer to an animal or to a weight-lifting machine). We use WordNet [18] as the sense catalogue and synset codes as concept identifiers.

Constituency/dependency parsing: Given a sentence, get its syntatic structure as a constituency/dependency parse tree.

[1] http://nlp.cs.upc.edu/freeling.

Semantic role labeling: Given a sentence identify its predicates and the main actors in each of them, regardless of the surface structure of the sentence (active/passive, main/subordinate, etc.)

Coreference resolution: Given a document, group mentions referring to the same entity (e.g. a person can be mentioned in the text as *Mr. Peterson*, *the director*, or *he*.)

Semantic graph generation: All the information extracted by the previous analyzers can be organized in a graph depicting events (mainly coming from predicates in the text), entities (coming from detected coreference groups), and relations between them (i.e. which entities participate in which events and with which role). This graph can be converted to triples and stored in an RDF database if needed.

5 Aligning Model and Text with ILP

5.1 Overview

A general description of the approach is shown in Fig. 2. The overall process can be separated into three categories. The modules handling the text in natural language (white), the ones treating the process model (light gray) and finally, those working on feature vectors (dark gray).

As a first stage, the model task labels and the textual process description are analyzed using FreeLing to obtain a structured representation of the text. After that, a phase of feature extraction follows where the model tasks and the text sentences are both converted into a canonical feature vector representation. These vectors can then be compared by means of standard distance metrics.

Parallel to that, a chronological partial order of both the sentences in the text and the tasks in the model is computed. To find an optimal alignment between model and text, these ingredients are encoded as an ILP model, whose solution denotes an optimal alignment between tasks and sentences. That assignment is used afterwards to both present the results to the end user and compute a numerical similarity score.

5.2 Linguistic Analysis of Text and Model

We perform a full NLP analysis on the text body corresponding to the input text, as listed in Sect. 4.2. This is a distinctive aspect of our approach with respect to [8,9], since we gather the linguistic information from the semantic graph, which contains a structured semantic representation of the text.

For the process model, we extract the natural language parts contained on it (from the activities, events, gateways, arcs, lanes and pools), and then a simple lingustic analysis (up to the word sense disambiguation step) is performed.[2]

[2] In order improve the performance of the word sense disambiguator on model sentences, the sentences from the text are provided as additional context to the analyzer. This greatly improves the disambiguation step when the model and the text are sharing a common semantic domain.

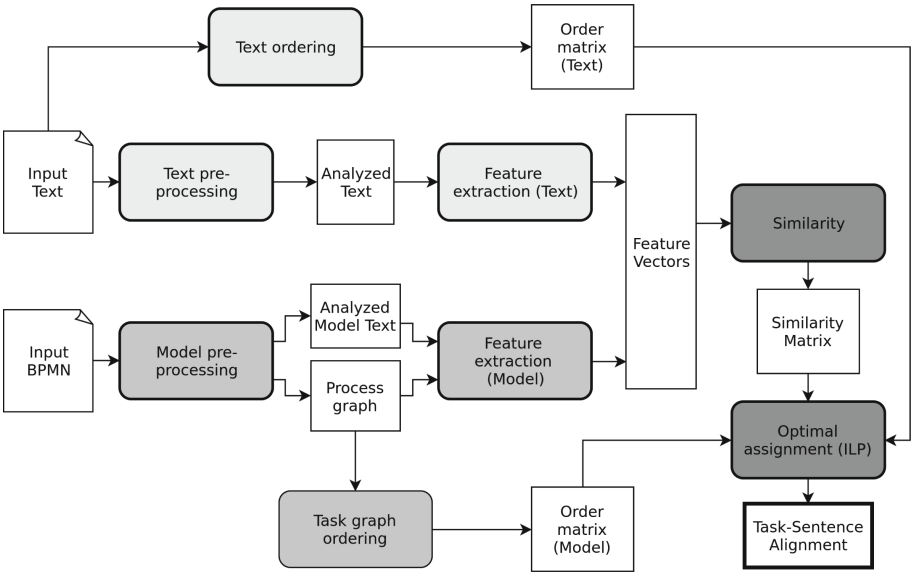

Fig. 2. Diagram illustrating the approach taken.

All the information gathered is then used in the feature extraction phase, explained in the next section.

5.3 Feature Extraction

Up to this point, the model and text are represented using different structures and handled separately. At this step they are both converted into an identical representation of feature vectors by the means of a feature extraction step. The purpose of this transformation is to enable the comparison by deriving a canonical representation. The motivation of using feature vectors is to aim for an *open* description of the text in both representations, i.e., to consider features as assignments to linguistic characteristics extracted from the text. We have considered the following linguistic characteristics in our approach (unless stated otherwise, *target text* in descriptions below refers to either a sentence in the textual description or to the label of a model task):

contains_lemma(l, pos) This feature is extracted from the target text if it contains a word with the lemma l and part-of-speech pos.

contains_action(a) This is extracted from a target text where the action a (typically a verb) is found.

agent_contains(l, v) This feature encodes who is performing the text or task action. It is extracted for text sentences whenever l is found as the agent of verb v, or for model tasks containing verb v and belonging to a swimlane/pool containing l.

contains_synset(s) This feature is extracted whenever the WordNet synset s appears in the target text.

contains_hypernym(s) This feature is extracted from a target text containing a word for which s is an hypernym[3] at distance HL or less. HL is a parameter of the algorithm.

object_contains(l, v) This feature is extracted when l is found as the direct object of verb v in the target text.

$follows\_conditional\_containing(l)$ This feature is extracted when l is found in a clause after a conditional statement (i.e. then, or else) in the text sentence, or when l is found in a task following an exclusive gateway with a question.

Table 2 shows some of the features extracted for sentence 3 in the example from Sect. 2, *"If he wants an individual membership, he must prepare his personal information"*. Note that the features include information such as lemma *customer* being mentioned, when it does not appear in the sentence. This is because the coreference resolution module detected that pronoun *he* in this sentence is referring to the actor *customer* mentioned somewhere else in the text. Also, output of the Semantic Role Labeler is also encoded in features stating that *customer* is the agent of action *want* and *prepare*, and that *information* is the object of *prepare*.

Table 2. Set of features (some omitted for brevity) for example sentence 3.

contains_lemma(customer, noun)	contains_lemma(want, verb)
contains_lemma(individual, adj.)	contains_action(want)
contains_action(prepare)	contains_synset(09984659-n::client)
contains_hypernym(10741590-n::user)	agent_contains(customer, wants)
agent_contains(customer, prepare)	object_contains(prepare, information)

Clearly, because the features are instantiated by words, this generates an open space of potentially infinite dimensions. For instance, the feature *contains_action* is instantiated twice for the sentence *The crane catched a fish and flew away*: *contains_action*$(catch)$ and *contains_action*(fly). In practice, this is handled by using a sparse representation of vectors.

The set of features proposed encodes high-level semantic information such as: who is the agent of the action, what is the action, or under what conditions is the task executed. That context is sometimes crucial in detecting whether a task is describing an action, referring to it or just using similar terms. The model and the text should generate similar feature vectors whenever the similarity between a sentence and a task is high, and vice-versa. This means the chosen features must represent properties that can be found both in the BPMN model and the textual description.

[3] 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.

5.4 Similarity Metrics

After the feature vector transformation defined in Sect. 5.3 it suffices to compare similarities between feature vectors in order to compute the similarity between a task and a sentence. In order to adjust the relevance of a feature f, we associate a scalar weight w_f to it as the product of two values: a constant value w_f^c that is particular for each feature class (e.g.: *contains_lemma*), and a variable value w_f^v whose magnitude depends on certain conditions. The set of constant weights of each feature class is a parameter of the algorithm. As an example, the particular weight of a feature such as $contains\_lemma(customer, noun)$ can be defined as the product of a constant factor for all instances of *contains_lemma* and the *tf-idf*[4] score of this lemma in the sentence.

Three similarity metrics are available as parameters: The *Cosine similarity* and the weighted versions of the *Jaccard index* and the *Overlapping index*. We have evaluated all three metrics and have chosen the last one as the default for our tool, since it gives more intuitive numerical values and the performance between all three does not differ significantly:

Weighted overlapping index This metric expands the Overlapping index by considering weighted elements in the set, such as in our case:

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

This metric returns a bounded value between 0 and 1.

5.5 Text and Model Ordering

When only considering the similarity metrics defined in the previous section, a task and a sentence might be very similar but may appear at very different parts of the corresponding representations [8]. For example, the action described by the sentence might occur in the last part of the text, while the task could be amongst the first tasks to execute in the process model. This means the chronological order of the events must be taken into account when trying to determine whether a task and a sentence refer to the same action.

Consequently, we seek to find the partial order relation $\preceq$ between the elements of both representations, text and model, such that $e \preceq e'$ means: "Element e happens before, or at the same time as e'". This allows us to define the strict order relation $\rightsquigarrow$ as

$$e \rightsquigarrow e' \iff e \preceq e' \wedge e' \not\preceq e$$

In the process model, the computation of the strict ordering relation goes beyond the mere structure of the model, and instead should be computed from

[4] The *tf-idf* of a token t is 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).

the underlying behavior. Fortunately, there are efficient techniques to determine the strict order relation [19,20] of a process model. In this paper, the relation $\rightsquigarrow$ corresponds to that same relation in the *behavioral profile* of the model as explained in [19].

Using the example in Sect. 2, a full *behavioral profile* would be extracted containing relations such as $G \rightsquigarrow I$ (happens before), $B + C$ (exclusive) or $H||L$ (parallel).

For the case of the text, it has been shown that the ambiguities present in textual descriptions make it impossible to determine the order of the tasks in them with total certainty [21]. This makes it hard to precisely extract the order unless techniques for extracting temporal relations are applied [22,23]. In our case, we have chosen to simplify the problem assuming a sequential order of the events depicted in the text. This assumption fails whenever the text deliberately reports events in reverse order such as in: "Task A is performed. But before A, Task B must have been executed.". In practice we hardly found such reverse ordering constructions in the texts describing process models.

5.6 Optimal Alignment Computation

This final step aims to find the optimal alignment between sentences in the textual description and tasks in the model. That information is then used in order to compute the global similarity between the model and the text as a numeric score. The information found in the optimal alignment can also aid in finding the actual inconsistencies between both representations as seen in [8]: (i) Tasks describing actions not appearing in the text, (ii) Sentences describing actions which are not in the model, and (iii) Different orderings of tasks.

For a formal definition of the problem, let the task set be T, the sentence set be S and $sim(s,t)$ be the computed similarity between $s \in S$ and $t \in T$[5] (cf Sect. 5.4). We assume for all pairs of elements both in S and T the order relation $\rightsquigarrow$ has been computed.

We define an alignment as a partial function $f_A : T \nrightarrow S$ of tasks to sentences such that $f_A(t) = s$, meaning that task t is describing the same actions as sentence s. In a fashion similar to that of [8], we define the optimal alignment f_A^* to be the alignment fulfilling the following properties:

Partial assignment The domain of f_A^*, denoted by $Dom(f_A^*)$ is a subset of the whole set of tasks, i.e. all $t \in T'$, for $T' \subseteq T$.

Order consistency Let $s = f_A^*(t)$ and $s' = f_A^*(t')$ for some pair of different tasks (t, t'). Then, the following restriction must hold: $t \rightsquigarrow t' \implies s \preceq s'$

Optimality The value of $\sum_{t \in Dom(f_A^*)} sim(t, f_A^*(t))$ is the maximum value such that the two other properties hold.

[5] In this case, $sim(s,t)$ corresponds to $WeightedOverlapping(v_s, v_t)$ where v_s and v_t are the feature vectors of s and t respectively.

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

$$\textbf{maximize: } \sum_{s \in S} \sum_{t \in T} a_{t,s} \cdot sim(t,s)$$

subject to:

$$\forall t \in T : \qquad \sum_{s \in S} a_{t,s} = 1$$

$$\forall (s,s') \in S \times S, (t,t') \in T \times T, t \rightsquigarrow t' \wedge s' \rightsquigarrow s : a_{t,s} + a_{t',s'} \leq 1$$

variables:

$$\forall s \in S, t \in T : \qquad a_{t,s} \in \{0,1\}$$

The variables $a_{t,s}$ can be interpreted as: "Task t is assigned to sentence s", i.e.: $a_{s,t} \iff s = f_A^*(t)$. The first family of constraints limits the number of sentences per task to exactly one[6]; this contradicts the requirement for function f_A^* to be partial, since a solution to the ILP model will has domain T. In practice, however, a threshold is used on the value $sim(t,s)$, and hence, assignments between tasks and sentences below this threshold are discarded. Finally, the second family of constraints encodes the *Order consistency* property by discarding the cases in which the order restriction would be violated.

Theorem 1. *The ILP model for aligning textual descriptions and process models is feasible and computes an optimal alignment f_A^* for a similarity metric sim.*

6 Experiments and Tool Support

The techniques of this paper have been implemented and are available as a web application[7]. The tool uses FreeLing for linguistic analysis, and Gurobi [24] as ILP solver. As similarity metric, we used the *weighted overlapping index.* Below we provide the two main experiments performed, devoted to analyze the positioning of the tool with respect to the state-of-the-art tool for the same task, (Sect. 6.1) and to test the tool capabilities in handling large instances (Sect. 6.2). The experiments for both tools have been performed on the same machine.

[6] Note that these equations can also be encoded using the *Special Ordered Sets* (*SOS*) constraint $\forall t : a_{t,1}, \cdots, a_{t,|S|}$, which denotes exactly the same constraint, and yields better performance in the ILP solvers that implement it.

[7] The web application is available at: http://xorrai.cs.upc.edu:8080/bpmninterface/. The tool we present in this paper corresponds to the *BPMN vs Text* tab.

6.1 Comparison with the Technique from [9]

To validate the quality of the results provided by our tool, we compare them with the ones generated by the approach in [9], on a gold standard from [13] that was later extended by the authors of [9]. We also expanded the gold standard with the last group of models, taken from [13]. The models in this benchmark were manually analyzed in [9,13] to obtain the correct assignment between tasks and sentences, so that the quality of a tool can be assessed.

Table 3 reports the results. For each model, we provide the number of tasks and sentences. Moreover, for each approach we report the accuracy (ratio of tasks correctly assigned to its matching sentence) and the execution time (average time per task) for each tool. To obtain a global perspective of the results, we provide a micro-average (total computation time over total number of tasks in all models), a macro-average (total computation time over number of models) and a median. The hughe differences between both methods are caused by a small subset of models that are more difficult to solve than the rest. Although our approach produces slightly better accuracies than [9], the difference is not significant. However, our approach can obtain the same accuracy in the alignment with a remarkable reduction of computation time.

Table 3. Accuracy and solving time of our proposal and the one in [9].

Model	$\|T\|$	$\|S\|$	[9] approach		Our proposal	
			Acc.	$^{ms}/_{task}$	Acc.	$^{ms}/_{task}$
Model1-2	8	6	100.0%	98	100.0%	29
Model1-4	7	11	100.0%	256	100.0%	45
Model10-1	4	3	75.0%	94	75.0%	27
Model10-10	10	8	70.0%	92	80.0%	26
Model10-11	9	7	77.8%	53	66.7%	23
Model10-12	5	4	80.0%	14	80.0%	23
Model10-13	4	3	100.0%	15	100.0%	25
Model10-14	10	5	50.0%	595	60.0%	29
Model10-3	12	11	91.7%	807	75.0%	28
Model10-4	11	9	90.9%	1,221	90.9%	28
Model10-5	4	4	100.0%	338	100.0%	24
Model10-6	4	3	75.0%	89	75.0%	20
Model10-7	8	7	100.0%	555	100.0%	19
Model10-8	5	7	80.0%	374	60.0%	27
Model10-9	8	5	100.0%	388	75.0%	23
Model2-1	26	38	76.9%	7,532	76.9%	134
Model2-2	19	30	63.2%	7,706	73.7%	84
Model3-1	6	7	100.0%	97	83.3%	28
Model3-2	6	4	100.0%	72	100.0%	20
Model3-3	4	5	100.0%	228	100.0%	29
Model3-4	2	4	50.0%	153	50.0%	56
Model3-5	11	9	81.8%	214	72.7%	31
Model3-6	6	8	83.3%	515	83.3%	28
Model4-1	18	40	33.3%	30,757	55.6%	173
Model5-1	2	6	0.0%	341	0.0%	73
Model5-2	5	5	60.0%	572	80.0%	30
Model5-3	9	10	55.6%	1,015	55.6%	34
Model6-2	4	5	75.0%	255	75.0%	33
Model6-3	5	9	80.0%	985	80.0%	1,880
Model6-4	9	14	66.7%	1,204	44.4%	47
Model7-1	4	7	100.0%	442	100.0%	30
Model8-1	5	3	100.0%	167	80.0%	18
Model8-2	5	6	40.0%	461	60.0%	26
Model8-3	5	5	100.0%	445	80.0%	26
Model9-1	7	8	71.4%	151	85.7%	33
Model9-3	6	4	100.0%	83	66.7%	25
Model9-4	7	5	28.6%	89	71.4%	26
Model9-5	8	7	62.5%	579	62.5%	24
Model9-6	8	13	37.5%	1,275	25.0%	45
BicycleManuf	9	12	100.0%	772	66.7%	41
ClaimsCreation	6	5	83.3%	467	100.0%	30
HotelService	12	11	91.7%	196	83.3%	33
Dispatch-of-g	7	7	71.4%	709	100.0%	35
Hospital	14	14	28.6%	25,109	71.4%	35
Hotel	12	11	83.3%	198	83.3%	30
Self-service	18	13	83.3%	1,002	88.9%	38
Underwriter	7	11	85.7%	274	100.0%	43
Zoo	15	12	46.7%	214	73.3%	31
Micro average			**73.7%**	**3,517**	**76.3%**	**76**
Macro average			**75.6%**	**1,860**	**76.4%**	**79**
Median			**80.0%**	**357.5**	**80.0%**	**29.5**

6.2 Experiments on Large Instances

In the previous experiment validated the quality of the results provided by our technique. In this second experiment we focus on the time performance, using models of increasing size. This will allow to extrapolate the capabilities of our approach for larger instances. For the sake of comparison, we also include the execution times for the current implementation of the tool described in [9].

Due to the small number of available model–text pairs, and to reduced range of model sizes in existing data, we opted for generating a synthetic dataset of model-text pairs. The model generation consists of two steps: first we use the PGL2 tool [25] to generate the structure of a BPMN model. The second step consists of enriching the generated model by replacing model labels with randomly generated task descriptions. Once a process model is generated, a text is also generated with a random number of sentences $|S| = |T| \pm k$, where $|T|$ is the number of tasks in the model, and k was set to three in the experiments. Both the text sentences and the task descriptions in the model are generated with a simple word–bigram Markov model built using [26], trained with all the textual descriptions from the benchmark in Sect. 6.1. The generated synthetic benchmark has 400 model–text pairs ranging from 1 to 115 tasks.

Figure 3 shows the execution time of both tools for all model sizes[8]. The plots show that our approach has an asymptotic behavior with a complexity much lower than the methods in [9]. Remarkably, there is a correlation between the variance in the execution time and the input size: from size 50 upwards in the plot of the right of Fig. 3, one can see that the execution time for models of similar size varies significantly. This suggests that other factors, apart from the model size, influence the execution time.

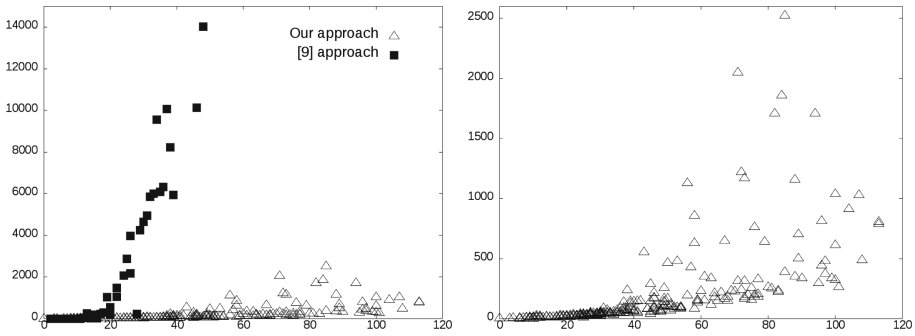

Fig. 3. Left: Execution times (in seconds) for [9] and our approach. Right: Zoom-in for the execution times of our approach.

[8] We could not include all the executions for the approach from [9] since instances bigger than 46 tasks hit the imposed 4 h time limit.

7 Conclusions and Future Work

In this paper we have proposed a novel approach for aligning textual descriptions and graphical models of processes. By applying a full linguistic analysis that results in an extensive set of features, and casting the problem as a mathematical optimization, we were able to align instances of unprecedented size. Moreover, in terms of quality the technique performs similar to the state of the art approach.

As a future work, we plan to expand the capabilities of the tool in different dimensions. First, we plan to incorporate the analysis of temporal relations in the text so that the control flow is better described. Second, a full exploration of the parameters of the technique (e.g., the weights for the similarity metric) will be done to boost the quality of the results. Finally, we plan to evaluate the tool in more realistic scenarios.

Acknowledgements. We would like to thank Han van der Aa and Henrik Leopold for their help and support to this work, and for sharing their software and part of the data used in the experiments of the paper. This work is funded by the Spanish Ministry for Economy and Competitiveness (MINECO), the European Union (FEDER funds) under grants COMMAS and Graph-Med (ref. TIN2013-46181-C2-1-R, TIN2016-77820-C3-3-R).

References

1. Leopold, H., Mendling, J., Polyvyanyy, A.: Supporting process model validation through natural language generation. IEEE Trans. Softw. Eng. **40**(8), 818–840 (2014)
2. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Heidelberg (2013)
3. van der Aa, H., Leopold, H., Mannhardt, F., Reijers, H.A.: On the fragmentation of process information: challenges, solutions, and outlook. In: Gaaloul, K., Schmidt, R., Nurcan, S., Guerreiro, S., Ma, Q. (eds.) CAISE 2015. LNBIP, vol. 214, pp. 3–18. Springer, Cham (2015). doi:10.1007/978-3-319-19237-6_1
4. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations, pp. 55–60 (2014)
5. Padró, L., Stanilovsky, E.: Freeling 3.0: towards wider multilinguality. In: Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC, Istanbul, Turkey, pp. 2473–2479, May 2012
6. Bird, S., Loper, E., Ewan, K.: Natural Language Processing with Python. O'Reilly Media Inc., Sebastopol (2009)
7. Apache Software Foundation: Apache OpenNLP (2010). http://opennlp.apache.org/
8. van der Aa, H., Leopold, H., Reijers, H.A.: Detecting inconsistencies between process models and textual descriptions. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 90–105. Springer, Cham (2015). doi:10.1007/978-3-319-23063-4_6
9. van der Aa, H., Leopold, H., Reijers, H.A.: Comparing textual descriptions to process models - the automatic detection of inconsistencies. Inf. Syst. **64**, 447–460 (2016)

10. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
11. Meziane, F., Athanasakis, N., Ananiadou, S.: Generating natural language specifications from UML class diagrams. Requir. Eng. **13**(1), 1–18 (2008)
12. Bajwa, I.S., Choudhary, M.A.: From natural language software specifications to UML class models. In: Zhang, R., Zhang, J., Zhang, Z., Filipe, J., Cordeiro, J. (eds.) ICEIS 2011. LNBIP, vol. 102, pp. 224–237. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29958-2_15
13. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 482–496. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21640-4_36
14. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001)
15. Cayoglu, U., et al.: Report: the process model matching contest 2013. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 442–463. Springer, Cham (2014). doi:10.1007/978-3-319-06257-0_35
16. de A.R. Gonçalves, J.C., Santoro, F.M., Baião, F.A.: Business process mining from group stories. In: Proceedings of the 13th International Conference on Computers Supported Cooperative Work in Design, CSCWD, Santiago, Chile, pp. 161–166, April 2009
17. Sinha, A., Paradkar, A.M.: Use cases to process specifications in business process modeling notation. In: IEEE International Conference on Web Services, ICWS, Miami, Florida, pp. 473–480, July 2010
18. Fellbaum, C.: WordNet. An Electronic Lexical Database. Language, Speech, and Communication. MIT Press, Cambridge (1998)
19. Weidlich, M.: Behavioural profiles: a relational approach to behaviour consistency. Ph.D. thesis, University of Potsdam (2011)
20. Polyvyanyy, A., Weidlich, M., Conforti, R., Rosa, M., ter Hofstede, A.H.M.: The 4C spectrum of fundamental behavioral relations for concurrent systems. In: Ciardo, G., Kindler, E. (eds.) PETRI NETS 2014. LNCS, vol. 8489, pp. 210–232. Springer, Cham (2014). doi:10.1007/978-3-319-07734-5_12
21. van der Aa, H., Leopold, H., Reijers, H.A.: Dealing with behavioral ambiguity in textual process descriptions. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 271–288. Springer, Cham (2016). doi:10.1007/978-3-319-45348-4_16
22. Mirza, P.: Extracting temporal and causal relations between events. Ph.D. thesis, International Doctorate School in Information and Communication Technologies, University of Trento, Italy (2016)
23. UzZaman, N., Llorens, H., Derczynski, L., Allen, J., Verhagen, M., Pustejovsky, J.: SemEval-2013 Task 1: TEMPEVAL-3: evaluating time expressions, events, and temporal relations. In: Second Joint Conference on Lexical and Computational Semantics (*SEM), vol. 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), Atlanta, Georgia, USA, pp. 1–9. Association for Computational Linguistics, June 2013
24. Gurobi Optimization Inc: Gurobi optimizer reference manual (2016). https://www.gurobi.com/documentation/6.5/refman.pdf
25. Burattin, A.: PLG2: multiperspective process randomization with online and offline simulations. In: Online Proceedings of the BPM Demo Track 2016, Rio de Janeiro, Brasil, September 2016
26. Schwartz, H.R.: Markov sentence generator (2010). https://github.com/hrs/markov-sentence-generator