Improving the Extraction of Process Annotations from Text with Inter-Sentence Analysis

Luis Quishpi, Josep Carmona, and Lluís Padró

Computer Science Department
Universitat Politècnica de Catalunya
Barcelona, Spain.
{quishpi,jcarmona,padro}@cs.upc.edu

Abstract. The automatic extraction of formal process information from textual descriptions of processes is a challenging problem, but worth exploring, since it enables organizations to align complementary information that talks about processes. In this paper we continue our previous work on this area, based on defining hierarchical/tree patterns on the dependency trees that arise from the linguistic analysis. We now incorporate a new abstraction layer on these patterns, that consider relationships between nearby sentences. The aim of this extension is to capture inter-sentence relationships that typically arise in textual descriptions of processes. The experiments done on publicly available benchmarks corroborate this intuition, showing a significant rise in the ability to capture all the important control-flow relationships defined in the text.

1 Introduction

As it has been recently acknowledged, there are quite important challenges on applying Natural Language Processing (NLP) techniques in the field of Business Process Management (BPM) [12]. Among the important ones, the extraction of process models from textual process descriptions is a very attractive use case, since the creation of process models consumes up to 60% of the time spent on process management projects. This paper focuses on this challenging task.

Although different approaches have been considered in the last years (see section 2), a number of open challenges remain for reaching a maturity level enabling its widespread adoption. For instance, techniques must be able to identify sentences that provide contextual information, rather than describe process steps. Furthermore, the inherent ambiguity of natural language can lead to different interpretations regarding the process that is described [14].

In this paper we significantly expand the techniques and results recently presented in [9], where we described robust tree-based patterns to be queried over the dependency trees arising from the NLP analysis of the textual descriptions. Patterns in [9] where only applicable in the context of a single sentence, which made our approach unable to extract inter-sentence relationships. The contribution of this paper is therefore the extension of our previous contribution with a more general set of patterns, resulting in a significant boost in the recall of the original framework (see experiment results in section 5).

The paper is organized as follows: next section shortly describes the work related to this contribution. Section 3 overviews the main components of our proposal, presented in Section 4. Experiments and tool support are reported in Section 5, whilst Section 6 concludes the paper and outlines future work.

2 Related work

For the sake of space, we only report here the related work that focuses on the extraction of process knowledge from textual descriptions [1,4,13], or the work that considers textual annotations in the scope of BPM [7,10].

For the former, the work by Gonçalves et al. [1] adopts important steps to extract the different BPMN elements and the work by Friedrich et al. [4] is acknowledged as the state-of-the-art for extracting process representations from textual descriptions, so we focus our comparison on this approach. As we will see in the evaluation section, our approach is significantly more accurate with respect to the state-of-the-art in the extraction of the main process elements. Likewise, we have incorporated as well the patterns from [13], and a similar outcome is reported in the experiments. The main reason is that approach relies on a deep NLP analysis and patterns on the syntactic structure of the sentence, instead of a shallow analysis and flat patterns.

For the later type of techniques ([7,10]), we see these frameworks as the principal application for our techniques. In particular, we have already demonstrated in the platform `https://modeljudge.cs.upc.edu` an application of the use of annotations in the scope of teaching and learning process modeling[1].

3 Preliminaries

The core of our proposal is the use of deep NLP analyzers to convert a textual description of a process into a syntax-semantic structure. Then, this structure is mined using tree-shaped patterns to obtain a conceptual representation of the process. Although other tools could be used, we resort to FreeLing as a NLP analyzer, TRregex as a tree-oriented pattern matching tool, and ATDP as a conceptual representation support. We describe each of them below.

3.1 Natural Language Processing

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: we use NLP analyzers to convert a textual description of a process model into a structured representation.

[1] The reader can see a tutorial for annotating process modeling exercises in the `ModelJudge` platform at `https://modeljudge.cs.upc.edu/modeljudge_tutorial/`.

The NLP processing software used in this work is FreeLing[2] [8], 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 & sentence splitting, morphological analysis, PoS-Tagging, Named Entity Recognition, Word sense disambiguation, Dependency parsing, Semantic role labeling and Coreference resolution. The three last steps are of special relevance since they allow the top-level predicate construction, and the identification of actors throughout the whole text: dependency parsing identifies syntactic subjects and objects (which may vary depending, e.g., on whether the sentence is active or passive), while semantic role labeling identifies semantic relations (the *agent* of an action is the same regardless of whether the sentence is active or passive). Coreference resolution links several mentions of an actor as referring to the same entity.

3.2 Annotated Textual Descriptions of Processes (`ATDP`)

`ATDP` is a formalism proposed in [10], aiming to represent process models on top of textual descriptions. This formalism naturally enables the representation of a wide range of behaviors, ranging from procedural to completely declarative, but also hybrid ones. Different from classical conceptual modeling principles, this highlight ambiguities that can arise from a textual description of a process, so that a specification can have more than one possible interpretation[3].

`ATDP` specifications can be translated into linear temporal logic over finite traces [5,2], opening the door to formal reasoning, automatic construction of formal models (e.g. in BPMN) from text, and other interesting applications such as simulation: to generate end-to-end executions (i.e., an *event log* [15]) that correspdond to the process described in the text, which would allow the application of *process mining* algorithms.

`ATDP` models are defined over an input text, which is marked with *typed text fragments*, which may correspond to *entities*, or *activities*. Marked fragments can be related among them via a set of *fragment relations*.

Entity fragments. The types of entity fragments defined in `ATDP` are:

- *Role.* The role fragment type is used to represent types of autonomous actors involved in the process, and consequently responsible for the execution of activities contained therein.
- *Business Object.* This type is used to mark all the relevant elements of the process that do not take an active part in it, but that are used/manipulated by process activities.

Activity fragments. `ATDP` distinguishes two types of activity fragments:

- *Condition.* It is considered discourse markers that mark conditional statements, like: *if*, *whether* and *either*. Each discourse marker needs to be tailored to a specific grammatical structure.

[2] `http://nlp.cs.upc.edu/freeling`
[3] In this work we consider a flattened version of the `ATDP` language, i.e., without the notion of *scopes*.

– *Task and Event.* Those fragment types are used to represent the atomic units of work within the business process described by the text. Usually, these fragments are associated with verbs. Event fragments are used to annotate other occurrences in the process that are relevant from the point of view of the control flow, but are exogenous to the organization responsible for the execution of the process.

Fragment Relations. Text fragments can be related to each other by means of different relations, used to express properties of the process emerging from the text:

– *Agent.* Indicates the role responsible for the execution of an activity.
– *Patient.* Indicates the role or business object on which an activity is performed.
– *Coreference.* Induces a coreference graph where each connected component denotes a distinct process entity.
– *Sequential.* Indicates the sequential execution of two activity fragments `A1` and `A2` in a sentence. We consider two important relations from [10]: Precedence and Response. Moreover, to cover situations where ambiguities in the text prevent selecting any of the two aforementioned relations, we also incorporate a less restrictive constraint WeakOrder, that only applies in case both activities occur in a trace.
– *Conflicting.* A conflict relation between two condition activity fragments $\langle C1, C2 \rangle$ in a sentence indicates that one and only one of them can be executed, thus capturing a choice. This corresponds to the relation NonCoOccurrence from [10].

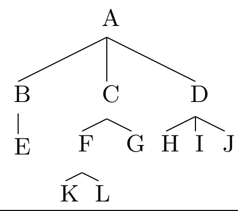
3.3 TRegex

In this paper, we use Tregex[4] [6], a query language that allows the definition of regular-expression-like patterns over tree structures. Tregex is designed to match patterns involving the content of tree nodes and the hierarchical relations among them. In our case we will be using Tregex to find substructures within syntactic dependency trees. Applying Tregex patterns on a dependency tree allows us to search for complex labeled tree dominance relations involving different types of information in the nodes. The nodes can contain symbols or a string of characters (e.g. lemmas, word forms, PoS tags) and Tregex patterns may combine those tags with the available dominance operators to specify conditions on the tree. Additionally, as in any regular expression library, subpatterns of interest may be specified and the matching subtree can be retrieved for later use. This is achieved in Tregex using unification variables as shown in pattern (2) in Figure 1.

Figure 1 describes the main Tregex operators used in this research to specify pattern queries.

[4] `https://nlp.stanford.edu/software/tregex.html`

Operator	Meaning
`X << Y`	X dominates Y
`X >> Y`	X is dominated by Y
`X !>> Y`	X is not dominated by Y
`X < Y`	X immediately dominates Y
`X > Y`	X is immediately dominated by Y
`X >, Y`	X is the first child of Y
`X >- Y`	X is the last child of Y
`X >: Y`	X is the only child of Y
`X $-- Y`	X is a right sibling of Y
`X $. Y`	X is the immediate left sibling of Y



(1) `E>>(A<<G)`	(4) `F!>>A`
(2) `E>>(A=x)>:(B=y)`	(5) `H>:D`
(3) `K!>>B>>(A<D)`	(6) `A<J`

Fig. 1. Some operators provided by Tregex (left). The tree on the right would match patterns (1), (2), (3), and would not match patterns (4), (5), (6). Note that unless parenthesized, all operators refer to the first element in the pattern. Pattern (2) uses operator `=` to capture nodes `A` and `B` into variables `x` and `y` respectively.

4 Generalized Approach

4.1 Basic Approach: Intra-Sentence Analysis

In this paper we describe an extension to the approach presented in [9]. This subsection summarizes the basic original approach, and following subsections provide details on the added extensions, which mainly consist of the extraction of relations between actions or conditions in different sentences, as well as an extended evaluation covering not only entities and actions, but also relations.

In [9] we presented a proposal to extract Business Process elements (entities, actions, conditions, events, and relations) from a process textual description.

The approach consists of: (a) Use a full-fledged NLP analysis pipeline [8] to analyze the text and extract verbal predicates, involved actors and objects, syntactic trees of all sentences, and coreferences between different mentions of the same actor/object, and (b) apply a cascade of TRegEx patterns on the output of the NLP preprocess to extract and elaborate the relevant process information. These patterns perform the following tasks:

1. Select the appropriate description for an entity or object. For instance, in the sentence "*The process starts when the female patient is examined by an outpatient physician, who decides whether she is healthy or needs to undertake an additional examination*" the results of the NLP semantic role labeling step for *Agent* would return the whole subtree headed by *physician* (i.e. *an outpatient physician, who decides... examination*).
The used Tregex patterns will strip down such a long description removing the determiner and the relative clause, while keeping the core actor/object and its main modifiers, thus extracting respectively `outpatient physician` as a role, and `female patient` as a business object.
2. Next step is identifying relevant activities. The NLP preprocess detects all predicates in the text (mainly, all verbs are considered a predicate, plus

some deverbal nouns such as "reception", "meeting", etc). However, although many verbs in a process description may be predicates from a linguistic perspective, they do not correspond to actual process activities. Thus, we use a set of patterns that discard predicates unlikely to be describing a relevant process task, or relabel them as *condition* or *event* fragments:

(a) More specifically, we use a set of predicates that check for syntactic structures involving conditional clauses (*if*, *whether*, *either*, ...) and the appropriate nodes in the tree are marked as *condition* fragments. In this step, we determine, for instance, that `she is healthy` and `needs an additional examination` are conditions in the sentence "*... who decides whether she is healthy or needs an additional examination.*".

(b) Another set of patterns deal with syntactic structures involving keywords like *when*, *once*, *as soon*, *whenever*, etc, and mark the related predicates as *event* fragments. These patterns allow us to identify the fragment `confirm(payment)` as an event fragment in the sentence "*Once the payment is confirmed, the ZooClub department can print the card...*"

(c) A third batch of patterns takes care of discarding activities that are not relevant to the process. To this end, we use two different strategies: one is removing all activities related to auxiliary, control, or subjective verbs (*be*, *have*, *start*, *want*, *think*, *believe*, etc.) which are unlikely to describe an actual process task. The second strategy relies on removing actions described in a subordinate clause. For instance, in the sentence "*..., the examination is prepared based on the information provided by the outpatient section*", the verbs *base* and *provide* would be removed as activities, since the main action described by this sentence is just *prepare (examination)*, and the subordinate clause just gives additional details on the object or procedure, but not on the actual process activity.

3. The last set of patterns deal with relations between activities. In our original work we tackled only relations between two activities in the same sentence. We considered different types of relations:

(a) Precedence: We use patterns to detect sentences relating one *event* and one *activity* in a precedence relation. E.g. in the sentence "*An intaker keeps this registration with him at times when visiting the patient*", it would extract the sequential relation from `visit(patient)` to `keep(registration)`.

(b) Response: This relation is identified between *condition* and *activity* fragments, which typically occur in conditional sentences such as "*If the patient signs an informed consent, a delegate of the physician arranges an appointment with one of the wards and updates the HIS selecting the first available slot*". From this sentence, we would extract the relation that `arrange (appointment)` *responds to* `sign(consent)`.

(c) Weak Order: There are many pairs of activities appearing in the same sentence where some kind of sequential order can be deduced, but it is not possible for an automatic system to determine their exact kind of relation. In these cases, we take a conservative approach and extract the least restrictive constraint, WeakOrder. For instance, in the sentence "*The Payment Office of SSP generates a payment report and then pays the vendor*", we could extract that `generate` and `pay` are in WeakOrder.

(d) Conflict: Conflict relations can be determined between condition fragments, provided they are in the right syntactic structure. In this way, we can extract the constraint that the sample can not be safely used and contaminated at the same time from the sentence "*... decides whether the sample can be used for analysis or whether it is contaminated*", or that conditional fragments `approve` and `deny` from the sentence "*The next step is for the IT department to analyse the request and either approve or deny it.*" are considered in conflict.

4.2 Inter-Sentence Analysis

Patterns used in [9] for relation extraction summarized in Section 4.1 aimed to capture relations between two activities/events mentioned in the same sentence. The main contribution of this paper is the extension of these patterns to capture also relations between activities or events located in different sentences.

To achieve this goal, since TRegEx is able to handle a single tree at a time, we first need to join together the syntactic trees for all sentences in the text in a single tree. For this, we add two kinds of artificial parent nodes: A `<PARAGRAPH>` node that has as children the root nodes for each of the sentences in the same paragraph, and a `<DOCUMENT>` node that has as children all the `<PARAGRAPH>` nodes. With that, we obtain a unique tree for all the document, and we can apply TRegEx patterns that span over more than one sentence. Figure 2 shows an example of a tree representing a short document. We apply patterns on the document tree to extract conflict and sequence relations between activities, events, or conditions detected in previous steps (see sec. 4.1.)

Conflicts. Conflicts between activities in the same sentence are detected using patterns described in [9]. The following patterns deal with conflicts between activities in different sentences. Their goal is to instantiate in variables `originRef` and `destinationRef` verbs that head sentences which may contain nodes marked as `<ACTIVITY>` or `<CONDITION>` on which the relation will be extracted.

```
PC1  /verb/=originRef > /<PARAGRAPH>/
                      << /<CONDITION>/
                      $. (/verb/=destinationRef << /<CONDITION>/)
PC2  /verb/=originRef > /<PARAGRAPH>/
                      << /<CONDITION>/
                      $. (/verb/ !<< /<CONDITION>/
                             $. (/verb/=destinationRef << /<CONDITION>/))
```

Pattern PC1 checks for a verb directly under a `<PARAGRAPH>` (i.e. main sentence verb) that has a condition as a child, and that its right sibling (i.e, main verb in the following sentence) also has a condition. This would extract a conflict

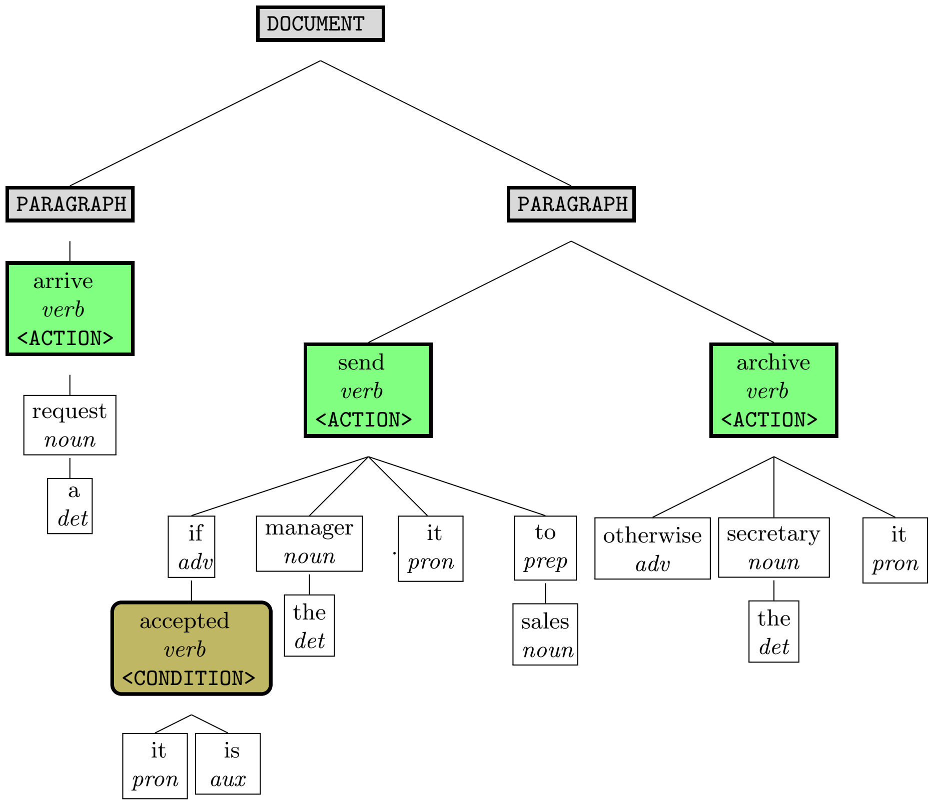


Fig. 2. Document tree for a text with two paragraphs: The first one with the sentence "*A request arrives*", the second with two sentences: "*If it is accepted, the manager sends it to sales. Otherwise, the secretary archives it*". Nodes in the syntactic dependency trees have been marked as `<ACTION>` or `<CONDITION>` in previous steps.

between *proceed* and *repeat* in the pair of sentences "*If sample is ok, proceed with examination. If contamination is detected, repeat sampling.*" Pattern PC2 captures the same kind of structure, when there is an additional sentence without a condition in between (e.g. "*If sample is ok, proceed with examination. Fill out treatment request form. If contamination is detected, repeat sampling.*")

Sequences. A second batch of patterns takes care of extracting sequence relations between activities in contiguous sentences. As in the case of conflicts, the patterns instantiate the variables `originRef` and `destinationRef` to candidate subtrees that are then searched for `<ACTIVITY>` or `<CONDITION>` nodes. Some patterns directly instantiate the variable `destination`, the actual target of the extracted relation.

```
PS1  /verb/=originRef > /<PARAGRAPH>/
                         $. (/verb/=destinationRef
                                  < /afterwards|then|immediately/)
PS2  /verb/=originRef > /<PARAGRAPH>/
                      $. (/verb/ << (/<CONDITION>/=destination << /or/))
PS3  /verb/=originRef > /<PARAGRAPH>/
                      $. (/verb/ << (/or/ << /<CONDITION>/=destination))
PS4  /verb/=originRef > /<PARAGRAPH>/
                      $. (/verb/ << /<CONDITION>/=destination
                                 < /otherwise|else/)
PS5  /verb/=originRef > /<PARAGRAPH>/
                      $. (/verb/ << /<CONDITION>/
                                 < (/otherwise|else/=destination)
```

Pattern PS1 extracts a sequence relation between the main verb of a sequence and the main verb of the next one provided the latter has a modifier such as *afterwards*, *then*, *immediately*, etc. Patterns PS2 and PS3 establish sequence relations between an activity and or-ed conditions in the following sentence (e.g extract sequences *send→fill* and *send→reject* in sentences "*Send form to customer. The customer can fill the form or reject to do it.*") Patterns PS4 and PS5 check a similar case, but where the second sentence has an "if-else" structure. They would extract the sequence relations *send→accept* and *send→cancel* in the sentence "*A budget is sent to the customer. If he accepts it, the bill is issued, otherwise the operation is cancelled.*"

5 Tool Support and Experiments

This section presents experiments evaluating the performance gain obtained when including patterns to capture relations between activities or events located in different sentences. We report two different results: First, we report relations extraction performance using a baseline based on [9] where we extract relations only for pairs in the same sentence. Second, we report results applying patterns to extract both intra- and inter-sentence relations.

The evaluation is performed comparing the relations extracted against gold standard manual annotations. For both table 1 and table 2, the test data set used in our experiments are the same as that used in the original proposal [9], which consists of 18 text-model pairs, each example includes a textual process description paired with the corresponding BPMN models created by a human.

The first 13 models stem from material in the appendix of [3], and the last 5 from our academic dataset[5] used in [11].

As a gold reference for evaluation, we manually created one ATDP for each example following the activities and relations in those BPMN models, i.e. marking as activity fragments only the text pieces that had a corresponding element in the BPMN model and connecting only the activities fragments that had a corresponding relation in the BPMN model.

[5] `https://github.com/setzer22/alignment_model_text/tree/master/datasets/NewDataset`

Source	Conflict						Sequence					
	#gold	#pred	#ok	P	R	F_1	#gold	#pred	#ok	P	R	F_1
1-1_bicycle_manufacturing	2	1	1	100	50	67	59	8	6	75	10	18
1-2_computer_repair	1	0	0	0	0	0	59	9	7	78	12	21
2-1_sla_violation	5	2	0	0	0	0	372	46	14	30	4	7
3-1_2009-1_mc_finalice_sct	0	0	0	0	0	0	52	4	3	75	6	11
3-2_2009-2_conduct	1	0	0	0	0	0	20	8	4	50	20	29
3-6_2010-1_claims_notification	2	0	0	0	0	0	63	9	7	78	11	19
4-1_intaker_workflow	0	0	0	0	0	0	596	17	5	29	1	2
5-1_active_vos_tutorial	3	0	0	0	0	0	15	4	3	75	20	32
6-1_acme-	1	0	0	0	0	0	340	22	11	50	3	6
7-1_calling_leads	1	0	0	0	0	0	13	2	1	50	8	13
8-1_hr_process_simple	0	0	0	0	0	0	15	7	6	86	40	55
9-2_exercise_2	3	0	0	0	0	0	11	6	6	100	55	71
10-2_process_b3	3	0	0	0	0	0	114	7	3	43	3	5
1081511532_rev3	1	0	0	0	0	0	47	9	5	56	11	18
1120589054_rev4-	0	0	0	0	0	0	66	9	6	67	9	16
1364308140_rev4	1	0	0	0	0	0	21	8	4	50	19	28
20818304_rev1	3	2	2	100	67	80	36	11	6	55	17	26
784358570_rev2	2	0	0	0	0	0	126	11	6	55	5	9
TOTAL	29	5	3	**60**	**10**	**18**	2025	197	103	**52**	**5**	**9**

Table 1. Evaluation of relation extraction using only intra-sentence patterns. Sequence relations are evaluated on the transitive clausure of both the sets of gold annotations and annotations produced by the system.

Intra-Sentence. Results for the first scenario (only intra-sentence patterns) are shown in Table 1, and correspond to results obtained using the patterns described in [9], which rely on extracting relations just within sentences. Precision is the percentage of right relations over predicted relations ($P = \#ok/\#pred$). Recall is the percentage of expected relations extracted ($R = \#ok/\#gold$). F_1 score is the harmonic mean of precision and recall ($F_1 = 2PR/(P+R)$). We only count extracted relations as right if they match the gold annotations in type (`<SEQUENCE>`, `<CONFLICT>`). In both experiments, sequence relations are evaluated over the transitive closure of the sequence annotations.

Inter-Sentence. In the second evaluation scenario, in addition to patterns created in [9], we use inter-sentence patterns described in Section 4.2.

Obtained results presented in Table 2 show that our new contribution extracts more relations, thus obtaining a large boost in recall (from 0.05 to 0.70 overall) with a very mild loss of precision (from 0.52 to 0.50 overall). Recall is boosted both for conflict and sequence relations, while precision is increased for conflicts, but slightly decreased for sequences.

Source	Conflict						Sequence					
	#gold	#pred	#ok	P	R	F_1	#gold	#pred	#ok	P	R	F_1
1-1_bicycle_manufacturing	2	2	2	100	100	100	59	90	54	60	92	72
1-2_computer_repair	1	0	0	0	0	0	59	65	33	51	56	53
2-1_sla_violation	5	4	2	50	40	44	372	572	152	27	41	32
3-1_2009-1_mc_finalice	0	0	0	0	0	0	52	57	42	74	81	77
3-2_2009-2_conduct	1	0	0	0	0	0	20	33	19	58	95	72
3-6_2010-1_claims	2	1	1	100	50	67	63	76	53	70	84	76
4-1_intaker_workflow	0	0	0	0	0	0	596	906	455	50	76	61
5-1_active_vos_tutorial	3	3	3	100	100	100	15	16	12	75	80	77
6-1_acme-	1	0	0	0	0	0	340	561	287	48	84	61
7-1_calling_leads	1	1	1	100	100	100	13	41	13	32	100	48
8-1_hr_process_simple	0	0	0	0	0	0	15	21	15	71	100	83
9-2_exercise_2	3	6	3	50	100	67	11	10	9	90	82	86
10-2_process_b3	3	1	1	100	33	50	114	138	83	60	73	66
1081511532_rev3	1	1	1	100	100	100	47	41	30	73	64	68
1120589054_rev4	0	0	0	0	0	0	66	78	66	85	100	92
1364308140_rev4	1	0	0	0	0	0	21	26	10	38	48	43
20818304_rev1	3	3	3	100	100	100	36	29	19	66	53	58
784358570_rev2	2	3	2	67	100	80	126	118	85	72	67	70
TOTAL	29	25	19	**76**	**66**	**70**	2025	2878	1437	**49**	**71**	**59**

Table 2. Evaluation of relation extraction using both intra- and inter- sentence patterns. Sequence relations are evaluated on the transitive clausure of both the sets of gold annotations and annotations produced by the system.

6 Conclusions and Future Work

We have presented an extension of our work in [9], consisting in adding syntax-tree based patterns to capture relations between activities or events located in different sentences. Results show that crossing the sentence boundaries is a highly productive strategy, since many more relations can be extracted. Also, the fact of using syntax-aware patterns, and not just flat regular expressions allows this extension to be done with almost no loss of precision.

Acknowledgments This work has been supported by MINECO and FEDER funds under grant TIN2017-86727-C2-1-R, and by the Ecuadorian National Secretary of Higher Education, Science and Technology (SENESCYT).

References

1. João Carlos de A. R. Gonçalves, Flávia Maria Santoro, and Fernanda Araujo Baião. Business process mining from group stories. In *Proceedings of the 13th International Conference on Computers Supported Cooperative Work in Design, CSCWD 2009, April 22-24, 2009, Santiago, Chile*, pages 161–166. IEEE, 2009.
2. Giuseppe De Giacomo, Riccardo De Masellis, and Marco Montali. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1027–1033. AAAI Press, 2014.

3. Fabian Friedrich. Automated generation of business process models from natural language input. *School of Business and Economics. Humboldt-Universität*, 2010.
4. Fabian Friedrich, Jan Mendling, and Frank Puhlmann. Process model generation from natural language text. In Haralambos Mouratidis and Colette Rolland, editors, *Advanced Information Systems Engineering*, pages 482–496, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
5. Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, 2013.
6. Roger Levy and Galen Andrew. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *LREC*, pages 2231–2234. Citeseer, 2006.
7. Hugo A. López, Søren Debois, Thomas T. Hildebrandt, and Morten Marquard. The process highlighter: From texts to declarative processes and back. In *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018), Sydney, Australia, September 9-14, 2018*, pages 66–70, 2018.
8. Lluís Padró and Evgeny Stanilovsky. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*, pages 2473–2479, 2012.
9. Luis Quishpi, Josep Carmona, and Lluís Padró. Extracting annotations from textual descriptions of processes. In *Proceedings of 18th International Conference on Business Process Management (BPM 2020), Sevilla, Spain, September 13-18, 2020*, pages 66–70, 2020.
10. Josep Sànchez-Ferreres, Andrea Burattin, Josep Carmona, Marco Montali, and Lluís Padró. Formal reasoning on natural language descriptions of processes. In *Int. Conference on Business Process Management*, pages 86–101. Springer, 2019.
11. Josep Sànchez-Ferreres, Han van der Aa, Josep Carmona, and Lluís Padró. Aligning textual and model-based process descriptions. *Data & Knowledge Engineering*, 118:25–40, 2018.
12. Han van der Aa, Josep Carmona, Henrik Leopold, Jan Mendling, and Lluís Padró. Challenges and opportunities of applying natural language processing in business process management. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 2791–2801, 2018.
13. Han van der Aa, Claudio Di Ciccio, Henrik Leopold, and Hajo A Reijers. Extracting declarative process models from natural language. In *International Conference on Advanced Information Systems Engineering*, pages 365–382. Springer, 2019.
14. Han van der Aa, Henrik Leopold, and Hajo A. Reijers. Dealing with behavioral ambiguity in textual process descriptions. In Marcello La Rosa, Peter Loos, and Oscar Pastor, editors, *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, volume 9850 of *Lecture Notes in Computer Science*, pages 271–288. Springer, 2016.
15. Wil M.P. van der Aalst. *Process Mining.* Springer, second edition, 2016.