Extracting Semantic Process Information from the Natural Language in Event Logs

Adrian Rebmann[(✉)] and Han van der Aa

Data and Web Science Group, University of Mannheim, Mannheim, Germany
{rebmann,han}@informatik.uni-mannheim.de

Abstract. Process mining focuses on the analysis of recorded event data in order to gain insights about the true execution of business processes. While foundational process mining techniques treat such data as sequences of abstract events, more advanced techniques depend on the availability of specific kinds of information, such as resources in organizational mining and business objects in artifact-centric analysis. However, this information is generally not readily available, but rather associated with events in an ad hoc manner, often even as part of unstructured textual attributes. Given the size and complexity of event logs, this calls for automated support to extract such process information and, thereby, enable advanced process mining techniques. In this paper, we present an approach that achieves this through so-called semantic role labeling of event data. We combine the analysis of textual attribute values, based on a state-of-the-art language model, with a novel attribute classification technique. In this manner, our approach extracts information about up to eight semantic roles per event. We demonstrate the approach's efficacy through a quantitative evaluation using a broad range of event logs and demonstrate the usefulness of the extracted information in a case study.

Keywords: Process mining · Natural language processing · Semantic labeling

1 Introduction

Process mining [1] enables the analysis of business processes based on event logs that are recorded by information systems in order to gain insights into how processes are truly executed. Process mining techniques obtain these insights by analyzing sequences of recorded events, also referred to as *traces*, that jointly comprise an *event log*. Most foundational process mining techniques treat traces as sequences of abstract symbols, e.g., $\langle a, b, c, d \rangle$. However, more advanced techniques, such as social network analysis [3] and object-centric process discovery [2] go beyond this abstract view and consider specific kinds of information contained in the events' labels or attributes, such as *actors*, *business objects*, and *actions*.

A key inhibitor of such advanced process mining techniques is that the required pieces of information, which we shall refer to as *semantic components*,

are not readily available in most event logs. A prime cause for this is the lack of standardization of attributes in event logs. While the XES standard [4] defines certain standard extensions for attributes (e.g., `org:resource`), the use of these conventions is not enforced and, thus, not necessarily followed by real-life logs (cf., [9]). Furthermore, the standard only covers a limited set of attributes, which means that information on components such as *actions* and *business objects*, are not covered by the standard at all and, therefore, often not explicitly represented in event logs.

Rather, relevant information is often captured as part of unstructured, textual data attributes associated with events, most commonly in the form of an event's label. For example, the "*Declaration submitted by supervisor*" label from the most recent BPI Challenge [10] captures information on the business object (*declaration*), the action (*submitted*), and the actor (*supervisor*). Since these components are all encompassed within a single, unstructured text, the information from the label cannot be exploited by process mining techniques. Enabling this use, thus, requires the processing of each individual attribute value in order to extract the included semantic information. Clearly, this is an extremely tedious and time-consuming task when considered in light of the complexity of real-life logs, with hundreds of event classes, dozens of attributes, and thousands of instances. Therefore, this calls for automated support to extract semantic components from event data and make them available to process mining techniques.

To achieve this, we propose an approach that automatically extracts semantic information from events while imposing no assumptions on a log's attributes. In particular, it aims to extract information on eight *semantic roles*, covering various kinds of information related to business objects, actions, actors, and other resources. The choice for these specific roles is based on their relevance to existing process mining techniques and presence in available real-life event logs. To achieve its goal, our approach combines state-of-the-art natural language processing (NLP) techniques, tailored to the task of semantic role labeling, with a novel technique for semantic attribute classification.

Following an illustration of the addressed problem (Sect. 2) and presentation of our approach itself (Sect. 3), the quantitative evaluation presented in Sect. 4 demonstrates that our approach achieves accurate results on real-life event logs, spanning various domains and varying considerably in terms of their informational structure. Afterwards, Sect. 5 highlights the usefulness of our approach by using it to analyze an event log from the 2020 BPI Challenge (BPI20). Finally, Sect. 6 discusses streams of related work, before concluding in Sect. 7.

2   Motivation

This section motivates the goal of semantic role labeling of event data (Sect. 2.1) and discusses the primary challenges associated with this task (Sect. 2.2).

2.1   Semantic Roles in Event Data

Given an event log, our work sets out to label pieces of information associated with events that correspond to particular *semantic roles*. In this work, we focus on various roles that support a detailed analysis of business process execution from a behavioral perspective, i.e., we target semantic roles that are commonly observed in event logs and that are relevant for an order-based analysis of event data. Therefore, we consider information related to four main categories: *business objects*, *actions*, as well as *active* and *passive* resources involved in a process' execution. For each category, we define multiple semantic roles, which we jointly capture in a set $\mathcal{R}$:

Business Objects. In line with convention [19], we use the term *business object* to broadly refer to the main object(s) relevant to an event. Particularly, we define (1) $\texttt{obj}$ as the type of business object to which an event relates, e.g., a *purchase order*, an *applicant*, or a *request* and (2) $\texttt{obj}_{status}$ as an object's status, e.g., *open* or *completed*.

Actions. We define two roles to capture information on the actions that are applied to business objects : (1) $\texttt{action}$, as the kind of action, e.g., *create*, *analyze*, or *send*, and (2) $\texttt{action}_{status}$, as further information on its status, e.g., *started* or *paused*.

Actors. Information regarding the active resource in the event is captured in the following two roles: (1) $\texttt{actor}$ as the type of active resource in the event, e.g., a "*supervisor*" or a "*system*", and (2) $\texttt{actor}_{instance}$ for information indicating the specific actor instance, e.g., an employee identifier.

Passive Resources. Aside from the actor, events may also store information on *passive* resources involved in an event, primarily in the form of *recipients*. For this, we again define two roles: (1) $\texttt{passive}$ as the type of passive resource related to the event, e.g., the role of an employee receiving a document or a system on which a file is stored or transferred through, and (2) $\texttt{passive}_{instance}$ for information indicating the specific resource, e.g., an employee or system identifier.

The considered semantic roles enable a broad range of fine-granular insights into the execution of a process. For example, the *business object* and *action* categories allow one to obtain detailed insights into the business objects moving through a process, their inter-relations, and their life-cycles. Furthermore, by also considering the resource-related roles, one can, for instance, gain detailed insights into the resource behavior associated with a particular business object, e.g., how resources jointly collaborate on the processing of a specific document. While the covered roles, thus, support a wide range of analyses and are purposefully selected based on their relevance in real-life event logs, our approach is by no means limited to these specific roles. Given that we employ state-of-the-art NLP technology that generalizes well, the availability of appropriate event data allows our approach to be easily extended to cover additional semantic roles, both within and outside the informational categories considered here.

2.2   The Semantic Role Labeling Task

To ensure that all relevant information is extracted from an event log, our work considers two aspects of the *semantic role labeling* task, concerned with two kinds of event attributes: *attribute-level classification* for attributes dedicated to a single semantic role and *instance-level labeling* for textual attributes covering various roles:

Attribute-Level Classification. Attribute-level classification sets out to determine the role of attributes that correspond to the same, dedicated semantic role throughout an event log, e.g., a `doctype` attribute indicating a business object. Although the XES standard [4] specifies several standard event attributes, such as `org:resource` and `org:role`, these only cover a subset of the semantic roles we aim to identify. They omit roles related to business objects, actions, and passive resources. These other semantic roles may, thus, be captured in attributes with diverse names, e.g., the obj_{status} role corresponds to event attributes such as `isClosed` or `isCancelled` in the Hospital log[1]. Furthermore, even for roles covered by standard attributes, there is no guarantee that event logs adhere to the conventions, e.g., rather than using `org:group`, the BPI14 log captures information on actors in an `Assignment_Group` attribute.

Instance-Level Labeling. Instance-level labeling, instead, sets out to derive semantic information from attributes with unstructured, textual values that encompass various semantic roles, differing per event instance. This task is most relevant for so-called event labels, often stored in a `concept:name` attribute. These labels contain highly valuable semantic information, yet also present considerable challenges to their proper handling, as illustrated through the real-life event labels in Table 1. The examples highlight the diversity of textual labels, in terms of their structure and the semantic roles that they cover. It is worth mentioning that such differences may even exist for labels within the same event log, e.g., labels l_5 and l_6 differ considerably in their textual structure and the information they cover, yet they both stem from the BPI19 log. Another characteristic to point out is the possibility of recurring roles within a label, such as seen for label l_1, which contains two `action` components: *draft* and *send*. Hence, an approach for instance-level labeling needs to be able to deal with textual attribute values that are highly variable in terms of the information they convey, as well as their structure.

---

[1] We kindly refer to Sect. 4.1 for further information on the event logs referenced here.

Table 1. Exemplary event labels from real-life event logs.

Log	ID	Event label	Contained semantic roles
WABO	l_1	Draft and send request for advice	`action` (×2), `obj`
BPI15	l_2	Send design decision to stakeholders	`action`, `obj`, `passive`
BPI15	l_3	Send letter in progress	`action`, `obj`, `action`$_{status}$
RTFM	l_4	Insert date appeal to prefecture	`action`, `obj`, `passive`
BPI19	l_5	Vendor creates invoice	`actor`, `action`, `obj`
BPI19	l_6	SRM: In Transfer to Execution Syst.	`action`, `passive`
BPI20	l_7	Declaration final_approved by supervisor	`obj`, `action`$_{status}$, `action`, `actor`

3   Semantic Event Log Parsing

This section presents our approach for the semantic labeling of event data. Its input and main steps are as follows:

Approach Input. Our approach takes as input an event log L that consists of events recorded by an information system. We denote the universe of all events as $\mathcal{E}$, where each event $e \in \mathcal{E}$ carries information in its payload. This payload is defined by a set of (data) *attributes* $\mathcal{D} = \{D_1, \ldots, D_p\}$ with $\mathrm{dom}(D_i)$ as the domain of attribute D_i, $1 \leq i \leq p$ and $\mathrm{name}(D_i)$, its name. We write $e.D$ for the value of D for an event e.

Note that we do not impose any assumptions on the attributes contained in an event log L, meaning that we do not assume that attributes such as `concept:name` and `org:role` are included in $\mathcal{D}$.

Fig. 1. Overview of the approach.

Approach Steps. The goal of our approach is to label the values of event attributes with their semantic roles. To achieve this, our approach consists of three main steps, as visualized in Fig. 1. Given a log L and its set of event attributes $\mathcal{D}$, Step 1 first identifies sets of *textual attributes* $\mathcal{D}^T \subseteq \mathcal{D}$ and of *miscellaneous attributes* $\mathcal{D}^M \subseteq \mathcal{D}$. Afterwards, Step 2 labels the values of textual attributes in $\mathcal{D}^T$ to extract the parts that correspond to semantic roles, e.g., recognizing that a "*document received*" event label contains the business object "*document*" and the action "*received*". Step 3 focuses on the attribute-level classification of miscellaneous attributes in $\mathcal{D}^M$, as well as some textual attributes $\mathcal{D}_n^T \subseteq \mathcal{D}^T$ that were deemed unsuitable for instance-level labeling during the previous step. This classification step aims to determine the semantic role that

corresponds to all values of a certain attribute in $\mathcal{D}^M \cup \mathcal{D}_n^T$, e.g., recognizing that all values of a `doctype` attribute correspond to the `obj` role.

In the remainder, Sects. 3.1 through 3.3 describe the steps of our approach in detail, whereas Sect. 3.4 discusses how their outcomes are combined in order to obtain an event log L' augmented with the extracted semantic information.

3.1   Step 1: Data Type Categorization

In this step, our approach sets out to identify the sets of textual attributes $\mathcal{D}^T$ and miscellaneous attributes $\mathcal{D}^M$. As a preprocessing step, we first identify `string`, `timestamp`, and `numeric` attributes using standard libraries, e.g., *Pandas* in Python[2].

Identifying Textual Attributes. To identify the set of textual attributes $\mathcal{D}^T$, we need to differentiate between `string` attributes with true natural language values, e.g., "*document received*" or "*Create_PurchaseOrder*", and other kinds of alphanumeric attributes, with values such as "*A*", "*USER_123*", and "*R_45_2A*". Only the former kind of attributes will be assigned to $\mathcal{D}^T$ and, thus, analyzed on an instance-level in the remainder of the approach. We identify such true textual attributes as follows:

1. Given a `string` attribute, we first apply a tokenization function *tok*, which splits an attribute value into lowercase tokens (based on whitespace, camelcase, underscores, etc.) and omits any numeric ones. E.g., given $s_1 =$ "*Create_PurchaseOrder*", $s_2 =$ "*USER_123*", and $s_3 =$ "*08_AWB45_005*", we obtain: $tok(s_1) = [create, purchase, order]$, $tok(s_2) = [user]$ and $tok(s_3) = [awb]$.
2. We apply a *part-of-speech tagger*, provided by standard NLP tools (e.g., Spacy [14]), to assign a token from the Universal Part of Speech tag set[3] to each token. In this manner, we obtain [(create, *VERB*) (purchase, *NOUN*), (order, *NOUN*)] for s_1, [(user, *NOUN*)] for s_2, and [(awb, *PROPN*)] for s_3.
3. Finally, we exclude any attribute from $\mathcal{D}^T$ that only has values with the same token in $tok(s)$ or do not contain any `NOUN`, `VERB`, `ADV`, or `ADJ` tokens. In this way, we omit attributes with values such as $s_2 =$ "*USER_123*" and $s_3 =$ "*08_AWB45_005*", which are identifiers, rather than textual attributes. The other attributes, which have diverse, textual values, e.g., $s_1 =$"Create_PurchaseOrder", are assigned to $\mathcal{D}^T$.

Selecting Miscellaneous Attributes. We also identify a set of non-textual attributes that are candidates for semantic labeling, referred to as the set of miscellaneous attributes, $\mathcal{D}^M \subseteq \mathcal{D} \setminus \mathcal{D}^T$. This set contains attributes that are not included in $\mathcal{D}^T$, yet have a data type that may still correspond to a semantic role in $\mathcal{R}$.

To achieve this, we discard those attributes in $\mathcal{D}\backslash\mathcal{D}^T$ categorized as `timestamp` attributes, as well as `numeric` attributes that include *real* or *negative* values. We exclude these because they are not used to capture semantic information. By contrast, the remaining attributes have data types that may correspond to roles in $\mathcal{R}$, such as `boolean` attributes that can be used to indicate specific states, e.g., `isClosed`, whereas non-negative integers are commonly used as identifiers. Together with the `string` attributes not selected for $\mathcal{D}^T$, the retained attributes are assigned to $\mathcal{D}^M$.

3.2   Step 2: Instance-Level Labeling of Textual Attributes

In this step, our approach sets out to label the values of textual attributes in order to extract the parts that correspond to certain semantic roles, e.g., recognizing that a "*create purchase order*" event label contains "*purchase order*" as the `obj` and "*create*" as the `action`. As discussed in Sect. 2.2, this comes with considerable challenges, given the high diversity of textual attribute values in terms of their linguistic structure and informational content. To be able to deal with these challenges, we therefore build on state-of-the-art developments in the area of natural language processing.

Tagging Task. We approach the labeling of textual attribute values with semantic roles as a text tagging task. Therefore, we instantiate a function that assigns a semantic role to chunks (i.e., groups) of consecutive tokens from a tokenized textual attribute value. Formally, given the tokenization of an attribute value, $tok(e.D) = \langle t_1, \ldots, t_n \rangle$, for an attribute $D \in \mathcal{D}^T$, we define a function $tag(\langle t_1, \ldots, t_n \rangle) \rightarrow \langle c_1 \backslash r_1, \ldots, c_m \backslash r_m \rangle$, where c_i for $1 \leq i \leq m$ is a chunk consisting of one or more consecutive tokens from $\langle t_1, \ldots, t_n \rangle$, with $r_i \in \mathcal{R} \cup \{\texttt{other}\}$ its associated semantic role. For instance, $tag(\langle create, purchase, order \rangle)$ yields: $\langle create \backslash \texttt{action}, purchase\ order \backslash \texttt{obj} \rangle$.

BERT. To instantiate the *tag* function, we employ BERT [8], a language model that is capable of dealing with highly diverse textual input and achieves state-of-the-art results on a wide range of NLP tasks. BERT has been pre-trained on huge text corpora in order to develop a general understanding of a language. This model can then be *fine-tuned* by training it on an additional, smaller training data collection to target a particular task. In this manner, the trained model combines its general language understanding with aspects that are specific to the task at hand. In our case, we thus fine-tune BERT in order to tag chunks of textual attribute values that correspond to semantic roles.

Fine-Tuning. For the fine-tuning procedure, we manually labeled a collection of 13,231 unique textual values stemming from existing collections of process models [15], textual process descriptions [16], and event logs (see Sect. 4.1). As expected, the collected samples do not capture information on resource instances, and rather contain information on the type level (i.e., `actor` and `passive`). For those semantic roles that are included in the samples, we observe a considerable imbalance in their commonality, as depicted in Table 2. In particular, while roles

such as `obj` (14,629 times), `action` (12,573), and even `passive` (1,191) are relatively common, we only found few occurrences of `actor` (135), obj_{status} (92), and action_{status} (30) roles.

Table 2. Training data used to fine-tune the language model, with $s = status$

Source	Count	obj	obj_s	action	action_s	actor	passive	other
Process models	11,658	13,543	50	11,445	3	58	1,058	4,966
Textual desc.	498	503	11	498	0	8	114	206
Event logs	625	583	31	630	27	69	19	291
Augmentation	450	350	100	350	150	200	0	150
Total	13,231	14,979	192	12,923	180	335	1,191	5,613

To counter this imbalance, we created additional training samples with obj_{status}, action_{status}, and `actor` roles through established data augmentation strategies. In particular, we created samples by complementing randomly selected textual values with (1) known `actor` descriptions, e.g., "*purchase order created*" is extended to "*purchase order created by supervisor*", and (2) common life-cycle transitions from [1, p.131] to create samples containing obj_{status} and action_{status} roles, e.g., "*check invoice*" is extended to "*check invoice completed*". However, as shown in Table 2, we limited the number of extra samples to avoid overemphasizing the importance of these roles.

Given this training data, we operationalize the *tag* function using the *BERT base uncased pre-trained language model*[4] with 12 transformer layers, a hidden state size of 768 and 12 self-attention heads. As suggested by its developers [8], we trained 2 epochs using a batch size of 16 and a learning rate of 5e − 5.

Reassigning Noun-Only Attributes. After applying the *tag* function to the values of an attribute $D \in \mathcal{D}^T$, we check whether the tagging is likely to have been successful. In particular, we recognize that it is hard for an automated technique to distinguish among the `obj`, `actor`, and `passive` roles, when there is no contextual information, since their values all correspond to nouns. For instance, a "*user*" may be tagged as `obj` rather than `actor`, given that business objects are much more common in the training data and there is no context that indicates the correct role. Therefore, we establish a set $\mathcal{D}_n^T \subseteq \mathcal{D}^T$ that contains all such *noun-only* attributes, i.e. attributes of which all values correspond solely to the `obj` role. This set is then forwarded to Step 3, whereas the tagged values of the other attributes directly become part of our approach's output.

3.3   Step 3: Attribute-Level Classification

In this step, the approach determines the semantic role of miscellaneous attributes, $\mathcal{D}^M$ identified in Step 1, and the noun-only textual attributes, $\mathcal{D}_n^T$,

---

[4] https://github.com/google-research/bert.

identified in Step 2. We target this at the attribute level, i.e., we determine a single semantic role for each $D \in \mathcal{D}^M \cup \mathcal{D}_n^T$ and assign that role to each occurrence of D in the event log. For attributes in $\mathcal{D}^M$, the approach determines the appropriate role (if any) based on an attribute's name, whereas for attributes in $\mathcal{D}_n^T$, it considers the name as well as its values. Note that we initially assign each attribute a role $r \in \mathcal{R}'$, where $\mathcal{R}'$ excludes the *instance* resource roles, i.e. $\texttt{actor}_{instance}$ and $\texttt{passive}_{instance}$, and later distinguish between type-level and instance-level based on the attribute's domain.

Classifying Miscellaneous Attributes. To determine the role of miscellaneous attributes, we recognize that their values, typically alphanumeric identifiers, integers or Booleans, are mostly uninformative. Therefore, we determine the role of an attribute $D \in \mathcal{D}^M$ based on its name. In particular, we build a classifier that compares a $name(D)$ to a set of manually labeled attributes $\mathcal{D}^{\mathcal{L}}$, derived from real-life event logs $\mathcal{L}$ (with $L \notin \mathcal{L}$).

Using $\mathcal{D}^{\mathcal{L}}$, we built a multi-class text classifier function $classify(D)$ that, given an attribute D, returns $r_D \in \mathcal{R}' \cup \{\texttt{other}\}$ as the semantic role closest to $name(D)$, with $conf(r_D) \in [0,1]$ as the confidence. To this end, we encode the names from $\mathcal{D}^{\mathcal{L}}$ using the GloVe [20] vector representation for words. Subsequently, we train a logistic regression classifier on the obtained vectors, which can then be used to classify unseen attribute names. Since GloVe provides a state-of-the-art representation to detect *semantic similarity* between words, the classifier can recognize that, e.g., an $\texttt{item}$ attribute is more similar to $\texttt{obj}$ attributes like $\texttt{product}$ than to $\texttt{actor}$ attributes in $\mathcal{D}^{\mathcal{L}}$.

Classifying Noun-Only Attributes. Given an attribute in $D \in \mathcal{D}_n^T$, we first apply the same classifier as used for miscellaneous attributes. If $classify(D)$ provides a classification with a high confidence value, i.e., $conf(r_D) \geq \tau$ for a threshold τ, our approach uses r_D as the role for D. In this way, we directly recognize cases where $name(D)$ is equal or highly similar to some of the known attributes in $\mathcal{D}^{\mathcal{L}}$. However, if the classifier does not yield a confident result, we instead analyze the textual values in $\text{dom}(D)$.

Since noun-only attributes were previously re-assigned due to their lack of context, we here analyze them by artificially placing each attribute value into contexts that correspond to different semantic roles. In particular, as shown in Fig. 2, we insert a candidate value (e.g., "*vendor*") into different positions of a set T of highly expressive textual attribute values (i.e., ones with at least 3 semantic roles). The resulting texts are then fed into the language model employed in Step 2, allowing our approach to recognize which context and, therefore, which semantic role, best suits the candidate value (i.e., $\texttt{passive}$ in Fig. 2). Finally, we assign $r_D \in \mathcal{R}' \cup \{\texttt{other}\}$ as the role that received the most votes across the different texts in T and values in $\text{dom}(D)$.

Recognizing Instance-Level Attributes. Since we only focused on the type-level roles $\mathcal{R}'$ in the above, we lastly check for every resource-related attribute $D \in \mathcal{D}^M$, with $r_D \in \{\texttt{actor}, \texttt{passive}\}$, if it actually corresponds to an instance-level role instead. Particularly, we change r_D to the corresponding instance-level

Original sentence	confirm to customer that paperwork is ok	
Replacing "confirm"	*vendor* to customer that paperwork is ok ⟶	prob(action) = 0.03
Replacing "customer"	confirm to *vendor* that paperwork is ok ⟶	prob(passive) = 0.90

Fig. 2. Exemplary insertion of a value from an attribute in $\mathcal{D}_n^T$ into an existing context.

role if dom(D) has values that contain a numeric part or only consist of named-entities (e.g., "*Pete*"). For instance, an attribute D_1 with values like *user_019* and *batch_06*, contains numeric parts and is, thus reassigned to $\texttt{actor}_{instance}$, while an attribute D_2 with dom(D_2)= {*staff member, system*} will retain its $\texttt{actor}$ role.

3.4   Output

Given an event e, our approach returns a collection of tuples (r, v) with $r \in \mathcal{R}$ a semantic role and v a value, where v either corresponds to an entire attribute value $e.D$ (for attribute-level classification applied to attributes in $\mathcal{D}^M \cup \mathcal{D}_n^T$) or to a part thereof (stemming from the instance-level labeling applied to $\mathcal{D}^T \setminus \mathcal{D}_n^T$).

To enable the subsequent application of process mining techniques, the approach returns an XES event log L' that contains these labels as additional event attributes, i.e., it does not override the names or values of existing ones. Note that we support different ways to handle cases where an event has multiple tuples with the same semantic role, e.g., the "*draft*" and "*send*" actions stemming from a "*draft and send request*" label: the values are either collected into one attribute, i.e., $\texttt{action} = [draft, send]$, or into multiple, uniquely-labeled attributes, i.e., $\texttt{action:0} = draft$, $\texttt{action:1} = send$. Furthermore, if multiple $\texttt{obj}_{status}$ (or $\texttt{action}_{status}$) attributes exist that each have Boolean values, e.g., $\texttt{isCancelled}$ and $\texttt{isClosed}$ for the Hospital log, these are consolidated into a single attribute, for which events are assigned a value based on their original Boolean attributes, e.g., $\{\bot, isCancelled, isClosed\}$.

4   Evaluation

We implemented our approach as a Python prototype[5], using the PM4Py library [5] for event log handling. Based on this prototype, we evaluated the accuracy of our approach and individual steps on a collection of 14 real-life event logs.

---

[5] https://gitlab.uni-mannheim.de/processanalytics/extracting-semantic-process-information.

4.1   Evaluation Data

To conduct our evaluation, we selected all real-life event logs publicly available in the common 4TU repository[6], except from those capturing data on software interactions or sensor readings, given their lack of natural language content. For collections that included multiple event logs with highly similar attributes, i.e., BPI13, BPI14, BPI15 and BPI20, we only selected one log per collection, to maintain objectivity of the obtained results. Table 3 depicts the details on the resulting collection of 14 event logs. They cover processes of different domains, for instance financial services, public administration and healthcare. Moreover, they vary significantly in their number of event classes, textual attributes, and miscellaneous attributes.

Table 3. Characteristics of the considered event logs, with $\mathcal{C}$ as the set of event classes

ID	Log name	$|\mathcal{C}|$	$|\mathcal{D}|$	$|\mathcal{D^T}|$	ID	Log name	$|\mathcal{C}|$	$|\mathcal{D}|$	$|\mathcal{D^T}|$
$L1$	BPI12	24	4	2	$L8$	BPI20	51	5	4
$L2$	BPI13	4	11	4	$L9$	CCC19	29	11	4
$L3$	BPI14	39	5	2	$L10$	Credit Req.	8	4	3
$L4$	BPI15	289	13	3	$L11$	Hospital	18	22	2
$L5$	BPI17	26	13	4	$L12$	RTFM	11	15	2
$L6$	BPI18	41	13	5	$L13$	Sepsis	16	31	1
$L7$	BPI19	42	4	2	$L14$	WABO	27	6	2

4.2   Setup

As a basis for our evaluation, we jointly established a *gold standard* in which we manually annotated all unique textual values (for instance-level labeling) and attributes (for attribute-level classification) with their proper semantic roles[7]. Since our approach requires training for the language model used in the instance-level labeling (Sect. 3.2) and for the attribute-name classifier (Sect. 3.3), we perform our evaluation experiments using *leave-one-out* cross-validation, in which we repeatedly train our approach on 13 event logs and evaluate it on the 14th. This procedure is repeated such that each log in the collection is considered as the test log once.

   To assess the performance of our approach, we compare the annotations obtained using our approach against the manually created ones from the gold standard. Specifically, we report on the standard *precision*, *recall*, and the F_1-*score*. Note that for instance-level labeling, we evaluate correctness per chunk, e.g., if a chunk (*purchase order*, obj) is included in the gold standard, both "*purchase*" and "*order*" need to be associated with the obj role in the result, otherwise, neither is considered correct.

---

[6] https://data.4tu.nl/search?q=:keyword:%20%22real%20life%20event%20logs%22.
[7] For reproducibility, the gold standard is published alongside the implementation.

4.3   Results

Table 4 provides an overview of the main results of our evaluation experiments. In the following, we first consider the performance of the instance-level labeling and attribute-level classification steps separately, before discussing the overall performance.

Table 4. Results of the evaluation experiments

Semantic role	Instance-level				Attribute-level				Overall			
	Count	Prec.	Rec.	F_1	Count	Prec.	Rec.	F_1	Count	Prec.	Rec.	F_1
`obj`	583	0.89	0.88	0.88	2	0.50	0.50	0.50	585	0.89	0.88	0.88
`obj`$_{status}$	31	0.85	0.77	0.78	6	0.50	0.33	0.40	37	0.79	0.70	0.72
`action`	630	0.94	0.95	0.94	0	-	-	-	630	0.94	0.95	0.94
`action`$_{status}$	27	0.85	0.81	0.82	6	1.00	1.00	1.00	33	0.88	0.84	0.85
`actor`	69	0.93	0.84	0.88	0	-	-	-	69	0.93	0.84	0.88
`actor`$_{instance}$	0	-	-	-	16	1.00	0.94	0.97	16	1.00	0.94	0.97
`passive`	19	0.84	1.00	0.91	0	-	-	-	19	0.84	1.00	0.91
Overall	1,359	0.91	0.91	0.91	30	0.87	0.79	0.83	1,389	0.91	0.90	0.90

Instance-Level Labeling Results. The table reveals that our instance-level labeling approach is able to detect semantic roles in textual attributes with high accuracy, achieving an overall F_1-score of 0.91. The comparable precision and recall scores, e.g. 0.94 and 0.95 for `action` or 0.89 and 0.88 for `obj`, each suggest that the approach can accurately label roles while avoiding false positives. This is particularly relevant, given that nearly half of the textual attribute values also contain information beyond the scope of the semantic roles considered here (see also Table 2). An in-depth look reveals that the approach even performs well on complex values, such as "*t13 adjust document x request unlicensed*". It correctly recognized the business objects (*document* and *request*), the action (*adjust*) and status (*unlicensed*), omitting the superfluous content (*t13* and *x*).

Challenges. We observe that the primary challenge for our approach relates to the differentiation between relatively similar semantic roles, namely between the two kinds of statuses, `obj`$_{status}$ and `action`$_{status}$, as well as the two kinds of resources, `actor` and `passive`. Making this distinction is particularly difficult in cases that lack sufficient contextual information or proper grammar. For example, an attribute value like "*denied*" can refer to either type of status, whereas it is even hard for a human to determine whether the "*create suspension competent authority*" label describes *competent authority* as a primary actor or a passive resource.

Baseline Comparison. To put the performance of our approach into context, we also compared its instance-level labeling step to a baseline: a state-of-the-art technique for the parsing of process model activity labels by Leopold et al. [15]. For a fair comparison, we retrained our approach on the same training data as

used to train the baseline (corresponding to the collection of process models in Table 2) and only assess the performance with respect to the recognition of *business objects* and *actions*, since the baseline only targets these. Table 5 presents the results obtained in this manner for the event labels from all 14 considered event logs.

The table shows that our approach greatly outperforms the baseline, achieving an overall F_1-score of 0.75 versus the baseline's 0.47. Post-hoc analysis reveals that this improved performance primarily stems from event labels that are more complex (e.g., multiple actions, various semantic roles or compound nouns spanning multiple words) or lack a proper grammatical structure. This is in line with expectations, given that the baseline approach has been developed to recognize several established labeling styles, whereas we observe that event data often does not follow such expectations. Finally, it is worth observing that the performance of our approach in this scenario is considerably lower than when trained on the full data collection (e.g., an F_1 of 0.66 versus 0.88 for the `obj` role), which highlights the benefits of our data augmentation strategies.

Table 5. Comparison of our instance-level labeling approach against a state-of-the-art label parser; both trained on process model activity labels and evaluated on event labels.

Semantic role	Count	Our approach			Baseline [15]		
		Prec.	Rec.	F_1	Prec.	Rec.	F_1
`obj`	562	0.65	0.68	0.66	0.40	0.40	0.40
`action`	618	0.86	0.81	0.83	0.59	0.48	0.53
Overall	1,180	0.76	0.75	0.75	0.50	0.44	0.47

Attribute-Level Classification Results. As shown in Table 4, our also approach achieves good results on the attribute-level classification of attributes, with an overall precision of 0.87, recall of 0.79, and an F_1 of 0.83. We remark that the outstanding performance of our approach with respect to the $action_{status}$ and $actor_{instance}$ roles is partially due to the usage of standardized XES names for some of these attributes, enabling easy recognition. Yet this is not always the case. For instance, 7 out of 16 $actor_{instance}$ attributes handled by this step use alternatives to the XES standard, such as `User` or `Assingment_Group`. Our approach maintains a high accuracy for these cases, correctly recognizing 6 out of 7 of such attributes. Notably, the overall precision of our attribute-classification technique reveals that it is able to avoid false positives well, even though a substantial amount of event attributes are beyond the scope of our semantic roles, such as monetary amounts or timestamps. This achievement can largely be attributed to the domain analysis employed in our approach's first step.

Nevertheless, it is important to consider that these results were obtained for a relatively small set of 30 non-textual attributes. Therefore, the lower results for certain uncommon semantic roles (e.g., `obj`), as well as the overall high accuracy for this step should be considered with care. This caveat also highlights the need additional training data, in order to expand the generalization of this part of our approach.

Overall Results. The overall performance of the approach can be considered as the average over the instance-level and attribute-level results, weighted against the number of entities that were annotated (cf., *count* in Table 4), i.e., a unique textual attribute value (instance-level) or an entire attribute (attribute-level).

We observe that the approach achieves highly accurate overall results, with a micro-average precision of 0.91, and a recall and F_1-score of 0.90. Still, when considering the results per semantic role, we observe that there exist considerable differences. These differences are largely due to the lower scores obtained for the underrepresented roles in the data set, since it is clear that our approach is highly accurate on more common roles, such as the F_1 score of 0.94 for the recognition of actions.

5   Case Study

This section demonstrates some of the benefits to be obtained by using the semantic information extracted by our proposed approach. To this end, we applied our approach to the *Permit Log* published as part of the BPI20 collection [10], which contains 7,065 cases and 86,581 events, divided over 51 event classes (according to the event label, i.e., the `concept:name` attribute). By applying our approach on the log, we identify information on five semantic roles. Most prominently, our approach is able to extract information about the `action`, action_{status}, `obj`, and `actor` roles from the log's unstructured, textual event labels. The availability of these semantic roles as attributes in the augmented event log, created by our approach, enables novel analyses, such as:

Event Class Refinement. The event log contains event labels that are polluted with superfluous information, e.g., by including resource information such as '*by budget owner*', resulting in a total of 51 event classes. Any process model derived on the basis of these classes, therefore, automatically exceeds the recommended maximum of 50 nodes in a process model [18], which impedes its understandability. To alleviate this, we can use the output of our approach to refine the event classes by grouping together events that involve the same `action` and `obj`. For instance, we group events with labels like "*declaration approved by budget owner*" and "*declaration approved by administration*", while deferring the actor information to a dedicated `actor` attribute. In this manner, we reduce the number of event classes from 51 to 21, which yields smaller and hence more understandable process models through process discovery techniques.

Object-Centric Analysis. The extracted semantic information also enables us to investigate the behavior associated with specific business objects. Through the analysis of event labels, our approach recognizes that the log contains six of these: *permit*, *trip*, *request for payment*, *payment*, *reminder*, and *declaration*. In Fig. 3 we show the directly-follows graph computed for the latter, obtained by selecting all events with e.obj = '*declaration*', and using the identified actions to establish the event class. The figure clearly reveals how declarations are handled the process. Mostly, declarations are *submitted*, *approved*, and then *final approved*. Interestingly, though, we also see 112 cases in which a declaration was definitely approved, yet rejected afterwards.
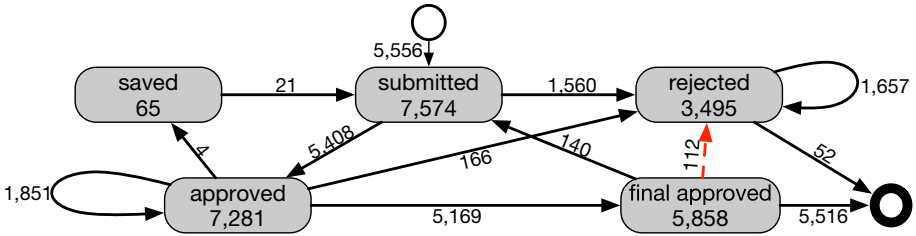

Fig. 3. Example for object-centric analysis. The directly-follows graph shows the actions applied to the object *declaration* in the log (includes 100% activities, 50% paths).

It is important to stress that both the *event class refinement* and *object-centric analysis* are based on information extracted from the unstructured, textual labels of the `concept:name` attribute in the original log. Therefore, the presented insights cannot be obtained by manually categorizing the attributes of the event log, but rather require the thorough, instance-level event analysis provided by our approach.

6   Related Work

Our work primarily relates to streams of research focused on the analysis of event and process model activity labels, as well as to the semantic role labeling task in NLP.

Various approaches strive to either disambiguate or consolidate labels in event logs. Lu et al. [17] propose an approach to detect duplicate event labels, i.e., labels that are associated with events that occur in different contexts. By refining such duplicates, the quality of subsequently applied process discovery algorithms can be improved. Work by Sadeghianasl et al. [22] aim to detect the opposite case, i.e., situations in which different labels are used to refer to behaviorally equivalent events. Other approaches strive for the semantic analysis of labels, such as work by Deokar and Tao [7], which group together event classes with semantically

similar labels, as well as the label parsing approach by Leopold et al. [15] against which we compared our work in the evaluation. Finally, complementary to our approach, work by Tsoury et al. [23] strives to augment logs with additional information derived from database records and transaction logs.

Beyond the scope of process mining, our work also relates to semantic annotation applied in various other contexts. Most prominently, semantic role labeling is a widely recognized task in NLP [6,12], which labels spans of words in sentences that correspond to semantic roles. The tasks' goal is to answer questions like *Who is doing what, where and to whom?* While early work in this area mostly applied feature engineering methods [21], recently deep learning-based techniques have been successfully applied, e.g., [13,24]. In the context of web mining, semantic annotation focuses on assigning semantic concepts to columns of web tables [25], while in the medical domain it is e.g. used to extract the symptoms and their status from clinical conversations [11].

7   Conclusion

In this paper, we proposed an approach to extract semantic information from events recorded in event logs. Namely, it extracts up to eight semantic roles per event, covering business objects, actions, actors, and other resources, without imposing any assumptions on the structure of an event log's attributes. We demonstrated our approach's efficacy through evaluation experiments using a wide range of real-life event logs. The results show that our approach accurately extracts the targeted semantic roles from textual attributes, while considerably outperforming a state-of-the-art activity label parser in terms of both scope and accuracy, whereas our attribute classification techniques were also shown to yield satisfactory results when dealing with the information contained in non-textual attributes. Finally, we highlighted the potential of our work by illustrating some of its benefits in an application scenario based on real-life data. Particularly, we showed how our approach can be used to refine and consolidate event classes in the presence of polluted labels, as well as to obtain object-centric insights about a process.

In the future, we aim to expand our work in various directions. To improve its accuracy, we aim to include data from external resources such as common sense knowledge graphs or dictionaries of domain-specific vocabulary into the approach. Furthermore, we intend to broaden its scope by introducing additional kinds of semantic roles, such as roles that disambiguate between human actors and systems. However, most importantly, through its identification of semantic information, our work provides a foundation for the development of wholly novel, semantics-aware process mining techniques.

Reproducibility: The implementation, dataset, and gold standard employed in our work are all available through the repository linked in Sect. 4.

References

1. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
2. Aalst, W.M.P.: Object-centric process mining: dealing with divergence and convergence in event data. In: Ölveczky, P.C., Salaün, G. (eds.) SEFM 2019. LNCS, vol. 11724, pp. 3–25. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30446-1_1
3. van der Aalst, W.M.P., Reijers, H.A., Song, M.: Discovering social networks from event logs. Comput. Support. Coop. Work (CSCW) **14**(6), 549–593 (2005). https://doi.org/10.1007/s10606-005-9005-9
4. Acampora, G., Vitiello, A., Di Stefano, B., van der Aalst, W., Günther, C., Verbeek, E.: IEEE 1849tm: The XES standard. IEEE Comput. Intell. Mag. **12**(2), 4–8 (2017). https://ieeexplore.ieee.org/document/7895272
5. Berti, A., van Zelst, S.J., van der Aalst, W.: Process mining for python (PM4Py): bridging the gap between process-and data science. ICPM Demo Track **2019**, 13–16 (2019)
6. Carreras, X., Màrquez, L.: Introduction to the CoNLL-2005 shared task: semantic role labeling. CoNLL **2005**, 152–164 (2005)
7. Deokar, A.V., Tao, J.: Semantics-based event log aggregation for process mining and analytics. Inf. Syst. Front. **17**(6), 1209–1226 (2015). https://doi.org/10.1007/s10796-015-9563-4
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL, pp. 4171–4186. ACL (2019)
9. van Dongen, B.F.: BPI challenge (2014). https://doi.org/10.4121/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35
10. van Dongen, B.F.: BPI challenge (2020). https://doi.org/10.4121/uuid:52fb97d4-4588-43c9-9d04-3604d4613b51
11. Du, N., Chen, K., Kannan, A., Tran, L., Chen, Y., Shafran, I.: Extracting symptoms and their status from clinical conversations. In: ACL. pp. 915–925 (2019)
12. Gildea, D., Jurafsky, D.: Automatic labeling of semantic roles. Comput. Linguist. **28**(3), 245–288 (2002)
13. He, L., Lee, K., Lewis, M., Zettlemoyer, L.: Deep semantic role labeling: what works and what's next. In: ACL, pp. 473–483 (2017)
14. Honnibal, M., Montani, I.: spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing (2017, To appear)
15. Leopold, H., van der Aa, H., Offenberg, J., Reijers, H.A.: Using hidden Markov models for the accurate linguistic analysis of process model activity labels. Inf. Syst. **83**, 30–39 (2019)
16. Leopold, H., van der Aa, H., Reijers, H.A.: Identifying candidate tasks for robotic process automation in textual process descriptions. In: Gulden, J., Reinhartz-Berger, I., Schmidt, R., Guerreiro, S., Guédria, W., Bera, P. (eds.) BPMDS/EMMSAD -2018. LNBIP, vol. 318, pp. 67–81. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91704-7_5
17. Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P.: Handling duplicated tasks in process discovery by refining event labels. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 90–107. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_6

18. Mendling, J., Reijers, H.A., van der Aalst, W.M.: Seven process modeling guidelines (7PMG). Inf. Softw. Technol. **52**(2), 127–136 (2010)
19. Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: empirical insights and recommendations. Inf. Syst. **35**(4), 467–482 (2010)
20. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: EMNLP, pp. 1532–1543 (2014)
21. Pradhan, S., Ward, W., Hacioglu, K., Martin, J.H., Jurafsky, D.: Semantic role labeling using different syntactic views. In: ACL, pp. 581–588 (2005)
22. Sadeghianasl, S., ter Hofstede, A., Suriadi, S., Turkay, S.: Collaborative and Interactive Detection and Repair of Activity Labels in Process Event Logs. In: ICPM, pp. 41–48 (2020)
23. Tsoury, A., Soffer, P., Reinhartz-Berger, I.: A conceptual framework for supporting deep exploration of business process behavior. In: Trujillo, J.C., et al. (eds.) ER 2018. LNCS, vol. 11157, pp. 58–71. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00847-5_6
24. Zhang, Z., et al..: Semantics-aware BERT for language understanding. In: AAAI, vol. 34, issue number 05, pp. 9628–9635 (2020)
25. Zhang, Z.: Effective and efficient semantic table interpretation using tableminer+. Seman. Web **8**(6), 921–957 (2017)