# A natural language querying interface for process mining

**Luciana Barbieri[1] · Edmundo Madeira[1] · Kleber Stroeh[2] · Wil van der Aalst[3,4]**

**Abstract**

In spite of recent advances in process mining, making this new technology accessible to non-technical users remains a challenge. Process maps and dashboards still seem to frighten many line of business professionals. In order to democratize this technology, we propose a natural language querying interface that allows non-technical users to retrieve relevant information and insights about their processes by simply asking questions in plain English. In this work we propose a reference architecture to support questions in natural language and provide the right answers by integrating to existing process mining tools. We combine classic natural language processing techniques (such as entity recognition and semantic parsing) with an abstract logical representation for process mining queries. We also provide a compilation of real natural language questions and an implementation of the architecture that interfaces to an existing commercial tool: Everflow. We also introduce a taxonomy for process mining related questions, and use that as a background grid to ana-lyze the performance of this experiment. Finally, we point to potential future work oppor-tunities in this field.

**Keywords** Process mining · Process querying · Natural language interface · Taxonomy

✉ Luciana Barbieri
luciana.barbieri@ic.unicamp.br

Edmundo Madeira
edmundo@ic.unicamp.br

Kleber Stroeh
kleber.stroeh@pega.com

Wil van der Aalst
wvdaalst@pads.rwth-aachen.de

[1] Institute of Computing, University of Campinas, Campinas, Brazil

[2] Pega Systems, São Paulo, Brazil

[3] Chair of Process and Data Science, RWTH Aachen University, Aachen, Germany

[4] Fraunhofer Institute for Applied Information Technology FIT, Sankt Augustin, Germany

## 1 Introduction

Process mining aims to discover, monitor and enhance processes using information extracted from event logs (van der Aalst 2016). There exist mature academic and commercial process mining techniques and tools that provide analyses over event log data (Viner et al. 2020). The use of these tools, however, requires knowledge of the technology itself and is mostly done by technical teams (process analysts, data scientists and alike).

To make process mining more ubiquitous, i.e., accessible on a daily basis by non-technical teams, we propose a natural language conversational querying interface. Business level and operations teams, for example, can take great benefit from the insights produced by process mining tools when accessed through such an intuitive interface.

In spite of recent advances in Natural Language Processing (NLP), understanding the semantics of a natural language question and translating it to a correct corresponding logical query is still a challenging task. Problems such as ambiguity (same natural language expression having multiple interpretations) and variability (many different expressions having the same meaning) are still difficult to handle.

On the other hand, other works that tackle the NLP side of the problem generally map questions to queries over the event log, itself. This approach is suboptimal as it misses the opportunity to leverage all the advanced algorithms and insights provided by more mature process mining techniques and tools.

Our previous work (Barbieri et al. 2021) proposes an *architecture for a process mining natural language query interface that takes questions and translates them to logical queries that can be run against existing process mining tools*. It is a general architecture, in the sense that, by interfacing with these tools, it is able to handle not only queries over event log data, but also over all types of analyses and algorithms they support. On the other hand, other aspects of our previous work, such as the logical representation of queries, the set of semantic rules, the implementation and the evaluation of the proposed method, focused only on questions over process execution data.

The main objective of this work is to expand our previous research by *tackling questions leveraging the analysis capabilities supported by process mining tools*, including the behavioral aspects of process execution. The contributions presented in this paper are:

- Extend the abstract logical representation for process mining queries proposed in Barbieri et al. (2021) and their corresponding set of semantic rules to *support queries over process behavior and process mining analyses*
- Introduce a *taxonomy* for natural language questions for process mining, to help guide the architecture implementation, better understand limitations and challenges, help frame the tests of the experiment (measure performance) and lay the foundation for future research (such as combining Artificial Intelligence (AI) and rule-based techniques for question understanding, for example)
- A *classification* of the initial set of questions presented in Barbieri et al. (2021) according to the proposed taxonomy, aiming to create a public dataset of process mining questions
- An *evaluation of the proposed architecture*, including integration to a commercial tool (Everflow Process Mining[1]) through its Representation State Transfer (REST or RESTful) Application Programming Interface (API), and an analysis on its performance vis-à-vis the proposed taxonomy for the questions

[1] https://everflow.ai/

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces our taxonomy for process mining natural language questions. Section 4 presents the proposed method for our natural language interface and logical representation of process mining queries. Section 5 describes the question dataset under construction. Section 6 presents the conducted experiment and the obtained results. Section 7 concludes this paper.

## 2 Related work

This section reviews work related to this research from the NLP application perspective, as well as from process management and process mining.

### 2.1 Natural language interfaces to databases

From the many existing NLP applications, the ones that are mostly related to this research are the so called Natural Language Interfaces to Databases (NLIDB). The main objective of NLIDB is to enable users who are not familiar with complex query languages such as Structured Query Language (SQL) to easily formulate queries over information stored in databases using natural language.

Even though NLIDB is not a new research topic (Hendrix et al. 1978; Androutsopoulos et al. 1995), recent advances in NLP have raised its importance and popularity during the last decade (Affolter et al. 2019; Mishra & Jain 2016). Current methods differ in the use of natural language itself (from queries restrictedly written according to specific grammatical constraints to full natural language sentences), as well as in the technical approaches used to parse and convert them to a machine-readable format such as SQL or SPARQL (recursive acronym for SPARQL Protocol and RDF - Resource Description Framework - Query Language). Most common parsing techniques are based on rule matching or machine learning. In either case, the types of queries that can be handled by the system are limited either by the set of rules, in the first case, or by the training data in the second.

SODA (Search over Data Warehouse) (Blunschi et al. 2012) is an example of an NLIDB system that takes keywords as input and automatically generates executable SQL queries. The method applies a graph pattern matching algorithm using a metadata model of the data warehouse that includes database schemas and domain ontologies. Athena (Saha et al. 2016) is also an ontology-driven rule-based NLIDB system. It uses domain specific ontologies mapped to database schemas to handle full natural language sentences. Queries are first translated into an intermediate query language over the ontology and subsequently converted to SQL.

Another category of methods apply machine learning to handle the translation problem. The main advantage of these approaches is that they support a richer language variability at the cost of requiring large training datasets. Seq2SQL (Zhong et al. 2017) is an example of such systems. The method applies a deep neural network with reinforcement learning to translate natural language questions to SQL and is able to address single table queries. Another neural approach for NLIDB is presented in Iyer et al. (2017), which proposes the adaptation of neural sequence models to map utterances directly to SQL. The system solicits user feedback on query results to reduce SQL annotation efforts and improve the model.

While these methods (like most of the existing NLIDB methods) are designed to handle queries over any domain (metadata and/or domain ontologies are usually taken as input to map domain terminology to database entities), their application to the process mining domain would be limited to querying raw event data stored in a database. Using specific process mining domain knowledge yields context to the design of a potentially more robust natural language interface.

## 2.2 Process querying

Processy Querying has become an important subfield of Business Process Management (BPM). The ability to retrieve knowledge from process models has propelled this area of study. Notable is the work presented in Polyvyanyy et al. (2017), where the authors propose a framework for devising process querying methods. The Process Querying Framework (PQF) connects and organizes elements in four different groups: (a) prepare; (b) execute; (c) model, simulate, record and correlate; and (d) interpret.

In Polyvyanyy, (2022), the editor describes the area of process querying, as well as compiles prominent works in the fields of Event Log Querying, Process Model Querying, combinations of both and other approaches (model similarity, logic-based, among others). We highlight the work presented in Polyvyanyy (2022), where the author proposes the Process Query Language (PQL) to query and manipulate process models. The language implements two classes of predicates: *behavioral predicates* to query behavior relations in processes and *process scenarios* to check for sequences of actions. On a more specific take, in Álvarez et al. (2022), the authors propose a new language named Process Instance Query Language (PIQL) to help measure KPIs or PPIs from logged events. The chapter defines the grammar for the language as well as examples of its use. It is worth to mention that none of those works addresses or includes natural language querying capabilities.

## 2.3 NLP applications in business process management and process mining

The possible applications of NLP techniques to the BPM and process mining domain are countless. In Section 2.3.1 we review work on general applications, while Section 2.3.2 focuses on natural language interfaces inside the domain.

### 2.3.1 General applications

As business processes are commonly documented by organizations in unstructured textual format rather than as structured models, one of the most important applications of NLP techniques to the BPM domain is the extraction of process models from natural language text (Riefer et al. 2016; van der Aa et al. 2018).

In Friedrich et al. (2011), the authors present an automatic approach to generate Business Process Model and Notation (BPMN) models from natural language text describing processes. The method combines a set of NLP techniques (tokenization, dependency parsing, etc.) and augment it with an anaphora resolution mechanism to identify actions and their inter-relationship and build a process model. Instead of textual descriptions, the method proposed in Epure et al. (2015) takes manual, text-based, process execution records as input to mine process models. The approach uses part-of-speech tagging, dependency parsing and rule-based semantic analysis to identify activities and the relationships between them and create a corresponding textual representation (a formal process model is not generated).

Another application of NLP to BPM is the automatic generation of textual descriptions from process models. The method presented in Leopold et al. (2012) addresses this task by taking a BPMN model as input, extracting linguistic information from its activity labels, events and gateways, among others, and building a syntactic tree, which is used to generate sentences in natural language.

NLP techniques were also used to compare process models to textual descriptions. The method proposed in van der Aa et al. (2017) aims to automatically detect inconsistencies between models and their corresponding descriptions, such as the representation of activities in different orders. Model and text are both subjected to a linguistic analysis to process sentences and activities so that their similarity can be assessed.

In addition to these applications, in van der Aa et al. (2018), the authors discuss future challenges for NLP applications in the BPM field, including the use of conversational systems to support the execution of business processes. The goal would be to allow process stakeholders to query the process through a natural language dialog system that would work on top of a formal model or textual description.

### 2.3.2 Natural language interfaces

In the work presented in Han et al. (2020), the authors propose a method to answer natural language queries over process automation event logs. The method extends the Athena NLIDB system (Saha et al. 2016) to translate natural language queries to queries over process execution data (event logs) stored in Elasticsearch. Although the experimental results show good accuracy (80%), the fact that the system answers natural language queries directly over event log data restricts the set of questions it is able to handle to those directly related to raw process execution data and statistics.

Similarly, in Kobeissi et al. (2021) the authors propose a natural language interface to query process execution event data. The method takes a textual natural language question and converts it into a query to be executed over a graph database where event data is stored. A hybrid approach is used for the conversion: machine learning-based techniques are used for intent detection and named entity recognition, while the construction of the database query is done using a rule-based component. Although the paper reports very high accuracy (95%) on question answering, it can only handle questions directly over event data (event attribute values and aggregations), not being able to address questions on conformance checking, bottleneck analysis and process behavior, among others.

Existing process mining techniques and tools can provide automatic analysis over event log data, which can be used to answer high-level user questions. To the best of our knowledge, this is the first research work aiming to automatically understand and answer high-level natural language questions by integrating a natural language interface to existing process mining tools that implement advanced analysis and algorithms. It is also the first to propose a taxonomy for natural language questions over process mining. An assessment of the types of questions handled by the works discussed in this section is presented in Section 3.3.

## 3 Question taxonomy

This work introduces a taxonomy for process mining questions asked in natural language. The idea is to create a classification framework around the questions that could help:

- Organize experiments and support comparison amongst different implementations of natural language querying interfaces to process mining.
- Identify shortcomings in implementations according to the dimensions of the taxonomy.
- Create the foundations for hybrid implementations (rule-based and AI-based) by segmenting the questions into groups that are more likely to be better processed using one technique or the other.

### 3.1 Foundations

The taxonomy introduced in this work is derived from a set of different inputs and influences:

- Examples of questions compiled in our previous work (Barbieri et al. 2021), as well as other works in healthcare (Mans et al. 2015), linguistic patterns applied to process mining (del-Río-Ortega et al. 2016) and conformance checking (Carmona et al. 2018).
- Experience in real life projects using Everflow in different scenarios and industries, as well as commonly asked questions, including aspects such as filtering and aggregation of information.
- Support to the basic concepts associated with process mining, including entities (cases, events, activities, etc.) and their relationships (van der Aalst 2016; Hompes et al. 2016).
- Answering commonly asked questions around Process Performance Indicators (PIIs) (del-Río-Ortega et al. 2013).
- Support the four types of questions presented in Mans et al. (2015), namely: (i) what happened, (ii) why did it happen, (iii) what will happen, and (iv) what is the best that can happen.
- Instantiating the linguistic patterns introduced in del-Río-Ortega et al. (2016), with emphasis on measures (time, count, conditions, data, derived, aggregates), targets (simple and composed) and scopes (number of instances, state condition, temporal condition), which we call *filters*.

Our proposition intends to shed some light onto classifying questions, so that patterns can be identified/created to process them. We don't claim this taxonomy to be complete. On the contrary, we encourage others to collaborate by extending it to deliver value to the research community and users in general.

### 3.2 Proposition

We propose a multi-dimensional, extensible, taxonomy, where different, independent, "points of view" are applied to classify the questions. These dimensions closely relate to the concepts mentioned in Section 3.1. Namely they are:

- *Perspective*: how a question relates to fundamental types of process mining, such as process execution (case/event) data, conformance checking, discovery, performance analyses, etc.
- *Relativity*: indicates if the question is absolute or relative to some set of entities (cases, analyses, etc.), i.e., it requires intermediate computations to be answered
- *Normativity*: does the question require normative information, such as a normative model or Service Level Agreement (SLA) information, to be answered or not (non-normative)?
- *Composition*: does the sentence indicate a single question (simple) or does it actually encompass multiple questions (composite), each requiring a separate answer?
- *Filtering*: is the question bounded by some filtering criteria (case identification, time, etc.) or is it applicable to the whole dataset/model?
- *Ambiguity*: indicates if the question has a single interpretation (unambiguous) or multiple ones (ambiguous) in the context of process mining.
- *Context*: indicates whether a question requires additional information/knowledge outside the question (contextual) itself, usually contained in previous questions/answers made through the natural language interface, or whether all that is needed is stated in the question itself (self-contained).

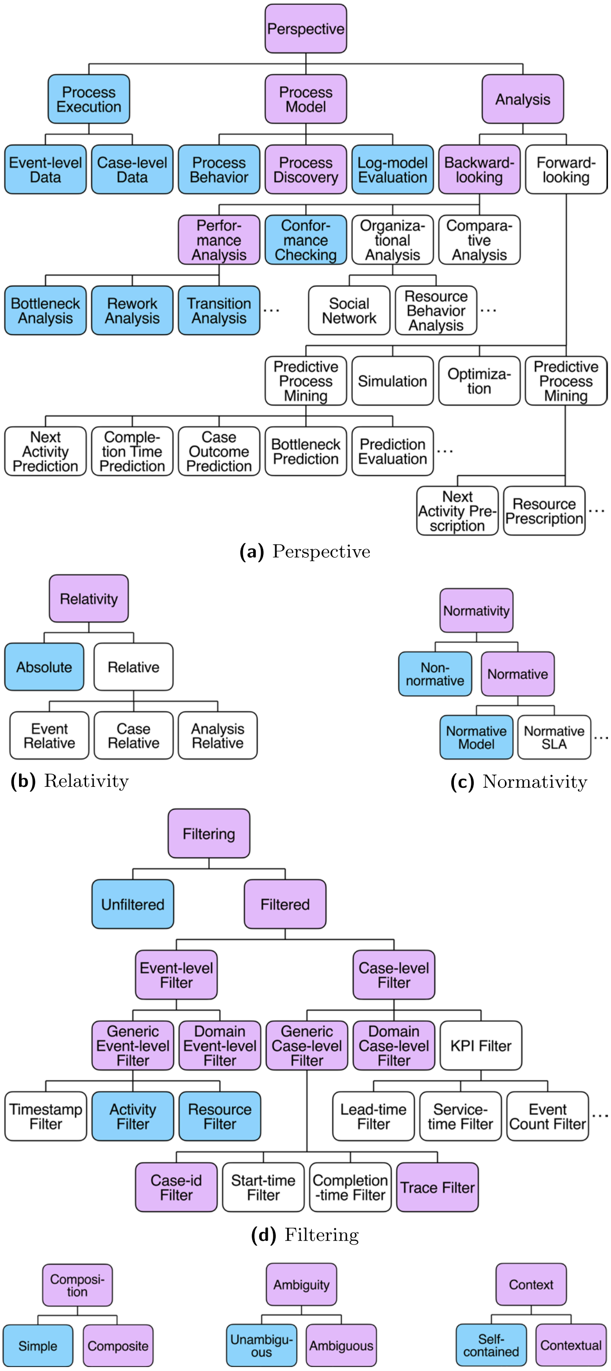
Figure 1 depicts the taxonomy breakdown for each dimension. Every dimension is organized in a hierarchical manner, so that a leaf of each tree (dimension) is assigned as part of the classification for a question. Colored nodes indicate the types of questions addressed in this work, as detailed in Section 3.3. Table 1 presents examples of questions and their corresponding classifications according to the proposed taxonomy.

The dimensions *relativity*, *normativity*, *composition* and *context* are very straightforward. Actually, the reduced size of their underlying tree structures gives little room for wondering what leaf to use to classify a question. The same holds true for the *filtering* dimension, which, although having a more complex tree structure, has a clear interpretation.

*Ambiguity*, on the other hand, does have a streamlined tree structure, but we observed that classifying a question as ambiguous (or not) leads to more debate and might pose a personal preference bias to the stage. This affects question classification not only in the *ambiguity* dimension itself, but also into the other dimensions, as different interpretations may imply fitting into a different *perspective*, *normativity* or *filtering* category, for example.

Finally, *perspective* presents the most complex tree structure and, arguably, the most interesting one too. It is a multi-layered hierarchy, where the root breaks down into the following subgroups:

- *Perspective: process execution*: questions that can be answered solely from the data extracted from the event log organized into two database-like tables (one for events and one for cases).



**(a)** Perspective

**(b)** Relativity

**(c)** Normativity

**(d)** Filtering

**(e)** Composition

**(f)** Ambiguity

**(g)** Context

**Fig. 1** Taxonomic dimensions

**Table 1** Examples of questions classified according to the proposed taxonomy

| Question | Perspective | Relativity | Normativity | Composition | Filtering | Ambiguity | Context |
|---|---|---|---|---|---|---|---|
| What is the most common flow of activities? | Case-level data | Absolute | Non-normative | Simple | Unfiltered | Unambiguous | Self-contained |
| What is the mean lead time of the process instances completed today? | Case-level data | Absolute | Non-normative | Simple | Completion-time filter | Unambiguous | Self-contained |
| What is the biggest bottleneck in the log? | Bottleneck analysis | Absolute | Non-normative | Simple | Unfiltered | Ambiguous | Self-contained |
| What is the rate of non conformances today? | Conformance checking | Analysis relative | Normative model | Simple | Completion-time filter | Unambiguous | Self-contained |
| What are the most and least frequent tasks? | Event-level data | Absolute | Non-normative | Composite | Unfiltered | Unambiguous | Self-contained |
| How does the discovered model look like? | Process discovery | Absolute | Non-normative | Simple | Unfiltered | Unambiguous | Self-contained |
| Where may this instance hit a bottleneck before it completes? | Next-activity prediction | Analysis relative | Non-normative | Simple | Unfiltered | Unambiguous | Contextual |

- *Perspective: process model*: questions on discovered and normative process models, process behavior and model evaluation against the event log (comprising quality metrics such as model fitness, precision, etc.).
- *Perspective: analysis*: questions that require process mining analysis algorithms to be run against the event log data in order to be answered (conformance checking, performance analysis, case outcome prediction, etc.).

One could argue that the line that separates these subgroups is thin, and that a given question could belong both to *perspective: process execution* and *perspective: analysis*. An example of such a question could be: "Which activity has the highest mean resolution time?". In this case, some may say that this is clearly a *perspective: process execution* question about lead-time, while others would argue that this is a *perspective: performance analysis*-related question.

In order to tie-break situations like this, we propose to apply the Occam's razor principle. If a question can be answered based solely on process execution information, it will then belong to that subgroup. To be more specific, if a question could be answered with an SQL-like query over process execution data, without requiring the launch of a specific process mining algorithm or analysis, then it is classified as a *perspective:process execution* rather than a *perspective:analysis* question. To further help assist arbitrating situations like this, we have presented the *perspective* tree in order of precedence from left to right. In other words, as soon as a question matches the left-most classification node in the tree, that is its final classification. Using this strategy brings another practical advantage: it leads to an implementation that typically requires fewer functionality to be provided by the counterpart process mining tool, making it easier to integrate to multiple tools.

### 3.3 Benchmarking

Colored nodes in Fig. 1 indicate the types of questions addressed by this work. Blue nodes denote groups that are fully handled, while purple nodes are partially supported. White nodes are not addressed at this time and are left for future work. In comparison, when looking at the *perspective* dimension, the methods presented in Han et al. (2020); Kobeissi et al. (2021) address questions classified under the *perspective:process execution* subtree only. From the *normativity* point of view, only *normativity:non-normative* questions are handled, while from the *filtering* angle both methods are able to address questions classified under all but the *filtering:KPI filter* subtree. From Han et al. (2020); Kobeissi et al. (2021), it was not possible to assess how these methods deal with the other dimensions.

## 4 Natural language querying method

Our previous work (Barbieri et al. 2021) proposes the architecture depicted in Fig. 2 for a process mining natural language querying interface. The input is a question in regular natural language such as "What is the average execution time of the process?" or "How many conformance problems have been identified?".

Questions go initially through a *pre-processing and tagging* step, where the text is split into tokens and tagged with Part-of-Speech (POS), dependency relations and base word form information. Mentions to real-world entities such as people and geographic location

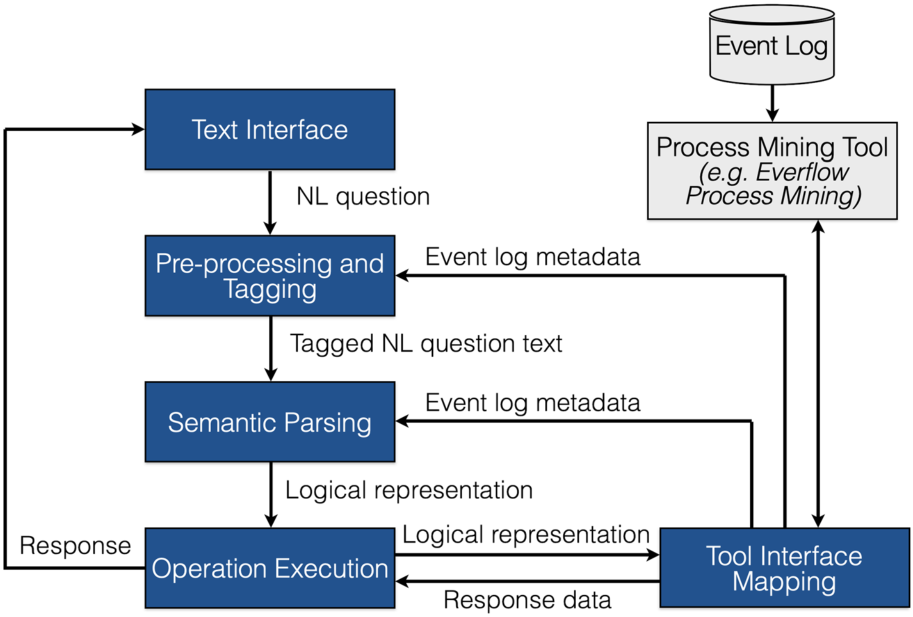


**Fig. 2** Process mining natural language querying interface architecture (Barbieri et al. 2021)

names are also identified and tagged during this step. The tagged text is then semantically parsed in order to extract its meaning and convert it to a logical representation that is machine readable and actionable. The logical representation is then executed by mapping it to calls of a process mining tool interface. These architecture components are described in more details in the following subsections.

## 4.1 Pre-processing and tagging

The raw question text is the input for this component, where the following processing occurs:

- Tokenization, which is the splitting of text into tokens.
- Part-of-Speech (POS) tagging, which performs morphological analysis over the text, marking tokens with tags such as “PRON” (pronoun), “ADJ” (adjective) and “VERB”.
- Dependency parsing, which marks tokens with grammatical structure information and dependency relations.
- Lemmatization, which finds the base (non-inflected) form of words.
- Entity recognition, which identifies and tags real-world entities in the text.

Entity recognition identifies general entities from pre-defined categories, such as names of people and organizations, geographic locations, time and quantities, among others. In addition to that, a natural language interface for process mining must be able to recognize the process mining entities present in sentences. Terms such as event, case, activity, resource and trace (along with its synonyms) must be recognized and tagged appropriately. The resulting tags are a crucial input for the next task in the processing pipeline (semantic

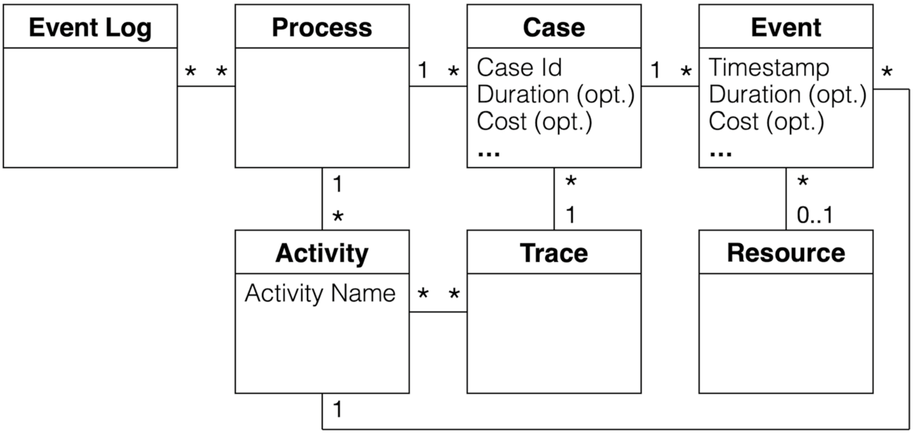


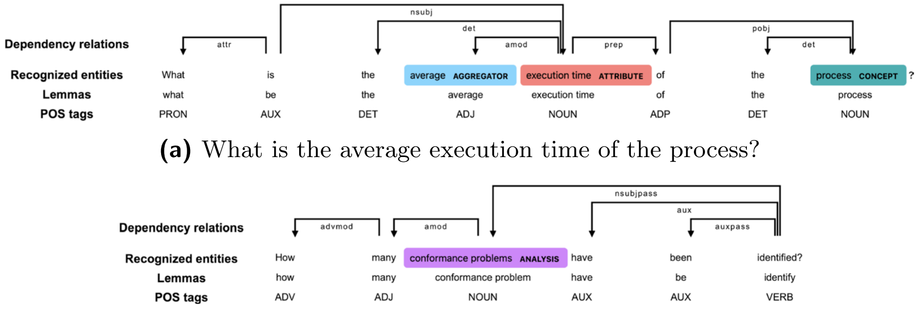
**Fig. 3** Process mining data model underlying entity recognition (Barbieri et al. 2021)

parsing). Figure 3 depicts the process mining data model that underlies the recognition of such terms. The model and the concepts it contains are standard and based on van der Aalst (2016).

Besides dealing with general process mining terms, the system must be able to recognize domain-specific terms. This includes the names of non-standard attributes present in the event log along with possible categorical values, among others. To be able to recognize such terms, this module uses event log metadata (names, types and possible values) of these attributes. The *tool interface mapping* layer takes the responsibility of interfacing with the process mining tool to gather these metadata. Figure 4 shows the output of *pre-processing and tagging*.

## 4.2 Semantic parsing

Semantic parsing aims to understand the meaning of a natural language sentence and map it to a logical (machine-readable) representation. The most common methods used for semantic parsing are rule-based and neural approaches. While rule-based methods are usually more appropriate to build domain specific systems targeted to understand a finite set of sentences, neural systems are more suitable to handle complex scenarios at the cost of


Dependency relations
Recognized entities
Lemmas
POS tags
attr
nsubj
det
amod
prep
pobj
det
What is the average AGGREGATOR execution time ATTRIBUTE of the process CONCEPT ?
what be the average execution time of the process
PRON AUX DET ADJ NOUN ADP DET NOUN
Dependency relations
Recognized entities
Lemmas
POS tags
advmod
amod
nsubjpass
aux
auxpass
How many conformance problems ANALYSIS have been identified?
how many conformance problem have be identify
ADV ADJ NOUN AUX AUX VERB

**(a)** What is the average execution time of the process?

**(b)** How many conformance problems have been identified?

**Fig. 4** Questions after pre-processing and tagging

**Table 2** Some QDMR operators used for PM question logical representation

| Operator | Description | Example | Logical Form |
|---|---|---|---|
| `select` | Return all instances of the given concept. | Show me all cases. | `select case` |
| `filter` | Return the referenced instances for which the given condition holds. | Show me all cases from Chicago. | `select case`<br>`filter city Chicago #1` |
| `project` | Return the given attribute/relation for the referenced instances. | How long does each process instance take to execute? | `select case`<br>`project duration #1` |
| `aggregate` | Apply the given aggregation to the referenced values. The supported aggregations are `max`, `min`, `average`, `sum` and `count`. | What is the average case duration? | `select case`<br>`project duration #1`<br>`aggregate average #2` |
| `group` | Apply the given aggregation to each subset of values corresponding to each key. | What is the average cost of each activity? | `select event`<br>`project cost #1`<br>`project activity #1`<br>`group average #2 #3` |
| `superlative` | Return the referenced instances for which the given value the highest/lowest. | What was the slowest case? | `select case`<br>`project duration #1`<br>`superlative max #1 #2` |

requiring large training corpora. Logical representations usually take the form of lambda calculus, query languages such as SQL and SPARQL or executable programs, among others.

### 4.2.1 Logical representation

The Question Decomposition Meaning Representation (QDMR) proposed in Wolfson et al. (2020) and inspired by SQL has been used, with some extensions, for the logical representation of process mining questions.

In QDMR questions are represented by a sequence of steps where each step corresponds to an operator. Each operator (except for `select`) is applied to the results of a previous step in the sequence. Additional parameters may be given to logical operators depending on the entities (concepts, attributes, aggregations, etc.) recognized in the natural language question. Table 2 presents the most relevant QDMR operators used in this research work to compose the logical representation of queries. For the complete set, please refer to Wolfson et al. (2020).

Notice that hash tags are used to refer to the results of a previous logical operation in the sequence, which may be a set of event or case instances or their attribute values. For example, in the following sequence (which logically represents a question such as "What is the average execution time of the process?"), `#1` refers to the results of `select case`, which are all case instances and `#2` refers to the values of the `duration` attribute for `#1`.

```
select case
project duration #1
aggregate average #2
```

The original set of QDMR operators was extended by this work to allow querying the behavioral aspects of process execution. Inspired by and initially based on the set of predicates defined by the PQL Polyvyanyy (2022), the `predicate` operator was introduced to logically represent questions over behavioral relations between executed activities. Supported behavioral predicates can be applied over cases or traces and are presented in Table 3. Optional parameters are enclosed in square brackets.

The following sequence, for example, represents a query of all activities that immediately precede activity "payment" over all available cases.

```
select case
predicate precede payment direct #1
```

Moreover, this work proposes another set of predicates to logically represent questions on process mining analyses, such as conformance checking, bottleneck and rework analysis, among others. Supported analysis predicates can be applied over cases and are presented in Table 4.

The following sequence, for example, logically represents a question such as "How many conformance issues have been identified?".

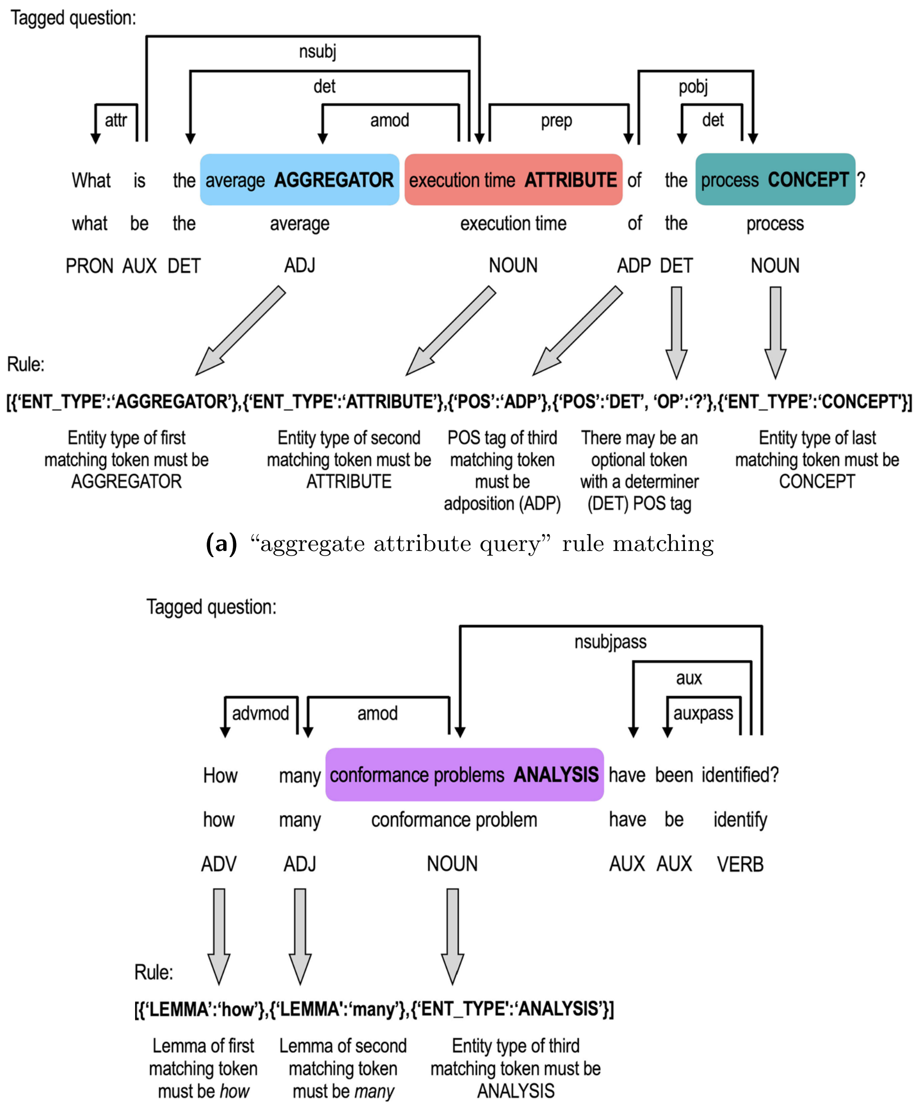
```
select case
predicate nonconformance #1
aggregate count #2
```

**Table 3** Behavioral predicates used for process mining question logical representation

| Predicate | Parameters | Description |
|---|---|---|
| occurs | activity | Return the referenced cases or traces that execute the given activity. |
| cooccur | activity1, [activity2] | Return the referenced cases or traces that execute both activity1 and activity2 or none. If only activity1 is given, return the activities that cooccur with it. |
| conflict | activity1, [activity2] | Return the referenced cases or traces that execute either activity1, activity2 or none. If only activity1 is given, return the activities that conflict with it. |
| causal | activity1, activity2 | Return the referenced cases or traces where any occurrence of activity1 precedes any occurrence of activity2. |
| concurrent | [activity1], [activity2] | Return the referenced cases or traces where some occurrence of activity1 occurs at the same time as some occurrence of activity2. If only activity1 is given, return the activities that occur concurrently to it. If neither activity is given, return the pairs of activities that occur concurrently. |
| precede | activity, [direct] | Return the activities that precede the given activity. The “direct” parameter is a flag that, when present, indicates that only the activities which directly precede the given activity must be returned. |
| succeed | activity, [direct] | Return the activities that succeed the given activity. The “direct” parameter is a flag that, when present, indicates that only the activities which directly succeed the given activity must be returned. |
| sequence | activity1, activity2, ... | Return the referenced cases or traces that execute the given sequence of activities. |
| start | – | Return the start activities for the referenced cases or traces. |
| end | – | Return the end activities for the referenced cases or traces. |
| activity-count | – | Return the number of activities executed by each referenced case or trace, including repetitions. |
| distinct-activity-count | – | Return the number of distinct activities executed by each referenced case or trace. |
| occurence-count | activity | Return the number of times the given activity is executed for each referenced case or trace. |

**Table 4** Analysis predicates used for process mining question logical representation

| Predicate | Parameters | Description |
|---|---|---|
| activity-bottleneck | – | Run activity bottleneck analysis for the referenced cases and return the identified bottleneck activities. |
| resource-bottleneck | – | Run resource bottleneck analysis for the referenced cases and return the identified bottleneck resources. |
| rework | – | Run rework analysis for the referenced cases and return the traces identified as containing rework. |
| transition | [activity1, activity2] | Run a temporal analysis over transitions and return statistics on their duration (minimum, maximum, average). If a pair of activities is given, run the analysis over the transitions between them. |
| slow-transition | – | Run slow transition analysis for the referenced cases and return the identified slow transitions. |
| slow-activity | – | Run slow activity analysis for the referenced cases and return the identified slow activities. |
| nonconformance | – | Run conformance checking for the referenced cases and return the identified nonconformances. |
| dfg | – | Return either an image or a Uniform Resource Locator (URL) to the Direct-follows Graph (DFG) that represents the referenced cases. |
| social-network | – | Return either an image or a URL to the social network discovered from the referenced cases. |


Tagged question:
nsubj
det
amod
prep
pobj
det
attr
What is the average AGGREGATOR execution time ATTRIBUTE of the process CONCEPT ?
what be the average execution time of the process
PRON AUX DET ADJ NOUN ADP DET NOUN
Rule:
[{'ENT_TYPE':'AGGREGATOR'},{'ENT_TYPE':'ATTRIBUTE'},{'POS':'ADP'},{'POS':'DET', 'OP':'?'},{'ENT_TYPE':'CONCEPT'}]
Entity type of first matching token must be AGGREGATOR
Entity type of second matching token must be ATTRIBUTE
POS tag of third matching token must be adposition (ADP)
There may be an optional token with a determiner (DET) POS tag
Entity type of last matching token must be CONCEPT
Tagged question:
nsubjpass
aux
auxpass
advmod
amod
How many conformance problems ANALYSIS have been identified?
how many conformance problem have be identify
ADV ADJ NOUN AUX AUX VERB
Rule:
[{'LEMMA':'how'},{'LEMMA':'many'},{'ENT_TYPE':'ANALYSIS'}]
Lemma of first matching token must be *how*
Lemma of second matching token must be *many*
Entity type of third matching token must be ANALYSIS

**(a)** "aggregate attribute query" rule matching

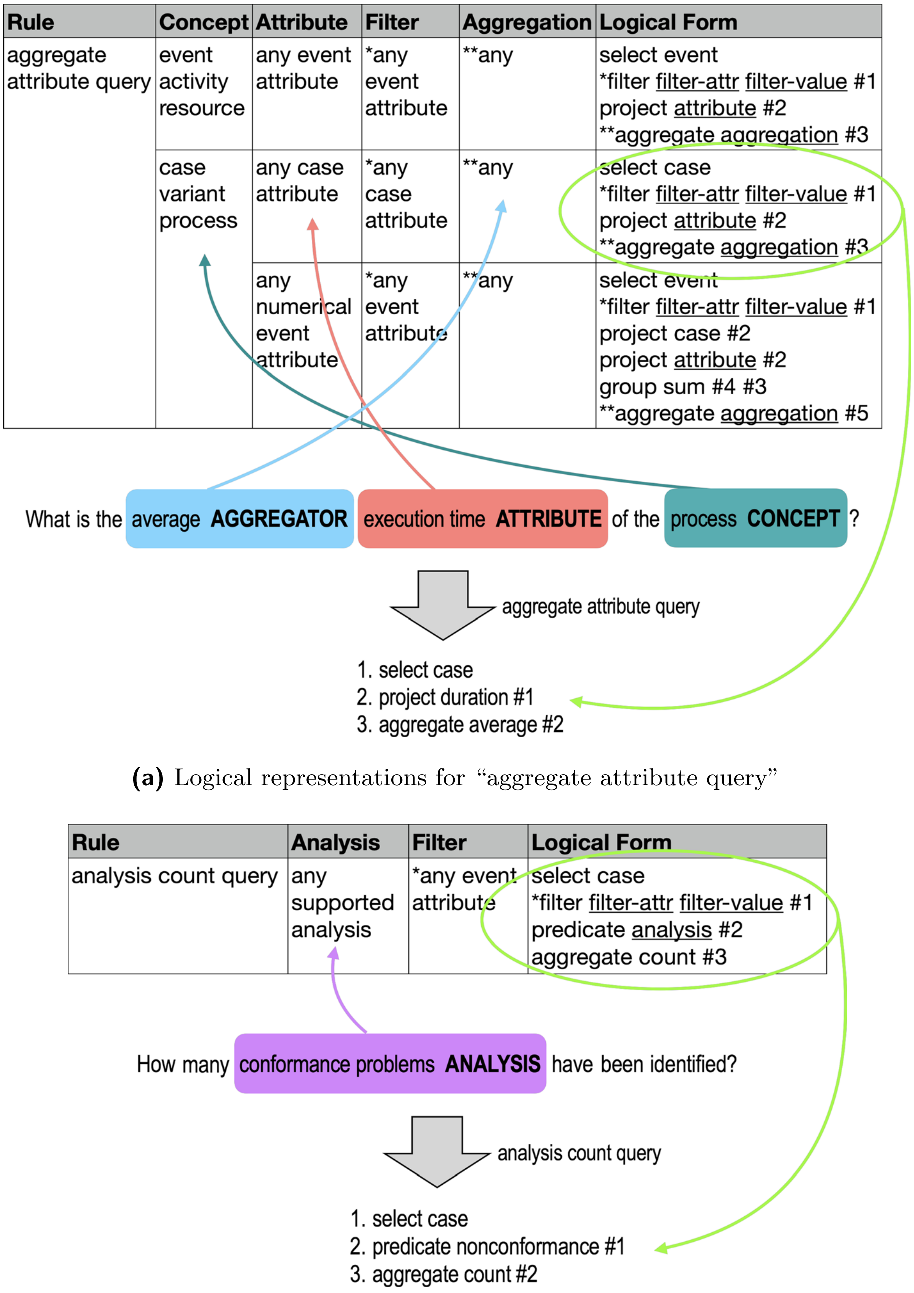
**(b)** "analysis count query" rule matching

**Fig. 5** Rule matching examples

### 4.2.2 Rule-based semantic parsing

We have initially adopted a rule matching approach for semantic parsing. Besides requiring no training data, the method has the advantage of achieving high accuracy in answering predictable questions.

In our experimental implementation, we used the spaCy open-source natural language processing library (Honnibal et al. 2020,). Its Rule Matcher component allows the definition of rules that match sequences of tokens. Rules are based on tags filled in the previous steps in the pipeline (part-of-speech tags, dependency parsing results, entity recognition



| Rule | Concept | Attribute | Filter | Aggregation | Logical Form |
|---|---|---|---|---|---|
| aggregate attribute query | event<br>activity<br>resource | any event attribute | *any event attribute | **any | select event<br>*filter filter-attr filter-value #1<br>project attribute #2<br>**aggregate aggregation #3 |
| | case<br>variant<br>process | any case attribute | *any case attribute | **any | select case<br>*filter filter-attr filter-value #1<br>project attribute #2<br>**aggregate aggregation #3 |
| | | any numerical event attribute | *any event attribute | **any | select event<br>*filter filter-attr filter-value #1<br>project case #2<br>project attribute #2<br>group sum #4 #3<br>**aggregate aggregation #5 |

What is the average AGGREGATOR execution time ATTRIBUTE of the process CONCEPT ?

aggregate attribute query

1. select case
2. project duration #1
3. aggregate average #2

**(a)** Logical representations for “aggregate attribute query”

| Rule | Analysis | Filter | Logical Form |
|---|---|---|---|
| analysis count query | any supported analysis | *any event attribute | select case<br>*filter filter-attr filter-value #1<br>predicate analysis #2<br>aggregate count #3 |

How many conformance problems ANALYSIS have been identified?

analysis count query

1. select case
2. predicate nonconformance #1
3. aggregate count #2

**(b)** Logical representations for “analysis count query”

**Fig. 6** Rule to logical representation mapping

**Table 5** Everflow's RESTful API endpoints

| Endpoint | Description |
|---|---|
| /cases | Return cases and corresponding attributes. |
| /events | Return events and corresponding attributes. |
| /analyses/modelViolations | Run conformance checking and return the nonconforming traces. |
| /analyses/bottlenecks | Run activity bottleneck analysis and return the identified bottleneck activities. |
| /analyses/reworks | Run rework analysis and return the traces identified as containing rework. |
| /analyses/transitions | Return statistics on transitions, such as average duration. |
| /analyses/slowTransitions | Run slow transition analysis and return the identified slow transitions. |

labels), together with actual expressions (in our case, words used to express the sort of process mining relationship/analysis being queried). They are formed by sequences of conditions, where each condition is composed by *tag:value* pairs that must match a single token from the tagged question obtained after *pre-processing and tagging*, which is the input for the *semantic parsing* step).

The set of questions introduced in Barbieri et al. (2021) was the base for the elaboration of the rules used for semantic parsing, resulting in 62 rules covering event log attribute querying, instance querying and counting, behavioral relations, process mining analysis, filters, aggregations and superlatives ("most", "least"). Figure 5 presents examples of semantic rules and their matchings (the matched rules are highlighted, with the arrows indicating which token in the question matched each condition in the rule). The "aggregate attribute query" rule (Fig. 5a) aims to match questions asking for aggregations (minimum, maximum and average, among others) of attribute values, while "analysis count query" (Fig. 5b) focuses on requests for counting analysis results (for example, the number of non-conformances identified by a conformance checking analysis). For the complete set of rules, please see https://ic.unicamp.br/~luciana.barbieri/pmnli-src.

Once a rule fires, a corresponding logical representation must be put together (this is the output of the *semantic parsing* step). This depends not only on what rule has been matched, but also on the entities (concepts, attributes, etc.) recognized in the sentence. One should notice at this point that one of the architectural goals of the proposed method is to allow seamless integration with any process mining tool. Therefore, we make as few assumptions as possible on how the integrated tool models the event log data. As a result, a minimal process mining data model based in the eXtensible Event Stream (XES) standard event log format (Verbeek et al. 2010) drives the mapping of matched rules to logical representation. Some of the entities tagged and handled as concepts during entity recognition and rule matching (activity, resource, trace) are, at this point, mapped to attributes of event and case, which are the only selectable concepts. Non-standard attributes contained in the event log are mapped based on the metadata obtained from the process mining tool.

As an example, Fig. 6 depicts the possible logical representations to be created when the "aggregate attribute query" and "analysis count query" rules are matched.

The matched rule indexes the first column in the tables, while the entities tagged in the sentence index the next (concept, attribute, filter and aggregation in Fig. 6a, analysis and filter in Fig. 6b). The last column corresponds to the logical representation that will be used to drive the calls to the process mining tool API. Asterisks indicate optional entities and the corresponding logical operations that are added to the sequence when they are present.

**Table 6** Everflow's RESTful API parameters

| Parameter | Description | Example |
|---|---|---|
| fields | Only the given attributes are returned for instances matching the query. | `/events?fields=activity,timestamp` |
| filters | Only instances matching the given conditions are returned. | `/events?activity=payment` |
| aggregate | The query returns the specified aggregation for the given attribute. Supported aggregations are `count()`, `avg()`, `sum()`, `max()` and `min()`. | `/cases?aggregate=avg(duration)` |
| group_by | Results are grouped by the given attribute before a corresponding aggregation is applied. | `/events?group_by=activity &aggregate=count(activity)` |
| sort_by | Results are sorted by the given attribute. If a "-" is used in front of the attribute name reverse (descending) sorting will be applied. | `/cases?sort_by=-duration` |
| limit | Limits the number of instance to be returned. | `/cases?sort_by=-duration &limit=10` |
| trace | This parameter applies exclusively to the `/cases` endpoint. The query returns the cases whose trace matches the specified behavioral activity relations. Supported relations are `alwaysOccurs()`, `cooccur()`, `conflict()`, `existCausal()`, `executes()`. | `/cases?trace=cooccur(approval,payment)` |

**Table 7** Logical representation to API mapping examples

| Question | Logical Representation | API Call |
|---|---|---|
| What is the average execution time of the process? | `select case`<br>`project duration #1`<br>`aggregate average #2` | `/cases?aggregate=avg(duration)` |
| What is the most common flow of activities? | `select case`<br>`project trace #1`<br>`group count #1 #2`<br>`superlative max #2 #3` | `/cases?fields=trace &aggregate=count(trace) &group_by=trace & sort_by=-ef_agg_0 &limit=1` |
| What is the average cost of the approval task? | `select event`<br>`filter activity approval #1`<br>`project cost #2`<br>`aggregate average #3` | `/events?filter=activity= approval &aggregate=avg(cost)` |
| What conformance problems have been identified? | `select case`<br>`predicate nonconformance #1` | `/analyses/modelViolations` |
| What are the bottlenecks? | `select case`<br>`predicate bottleneck #1` | `/analyses/bottlenecks` |
| What are the cases in which the approval and payment tasks cooccur? | `select case`<br>`predicate cooccur approval payment #1` | `/cases?trace=cooccur(approval,payment)` |

## 4.3 Operation execution

After a question has been understood and translated to a logical representation, the *operation execution* component orchestrates its execution. This is a simple step which takes the logical representation obtained after *semantic parsing* as input and executes each operation contained in it by invoking the *tool interface mapping* component. Intermediate operation results are passed as input to other operations that follow it in the sequence according the references they contain (# parameters described in Section 4.2.1). Final execution results are returned back as the response to the question.

## 4.4 Tool interface mapping

The final step in the question processing pipeline is to map the operations contained in the logical representation of the query into real API calls provided by a process mining tool. Operations are not necessarily mapped to API calls one by one, i.e., multiple operations contained in a logical representation may be grouped and mapped to a single API call, while a single operation may be mapped to multiple calls, depending on the underlying process mining tool. This mapping is done by the *tool interface mapping* component.

### 4.4.1 Everflow API

In this work, we integrated the architecture into Everflow's RESTful API. This API presents endpoints that mimic process mining main concepts and naturally maps into the process mining data model used to create logical representations. Tables 5 and 6 respectively depict the API's endpoints and the optional parameters that can be used for filtering, sorting and aggregating results, among others.

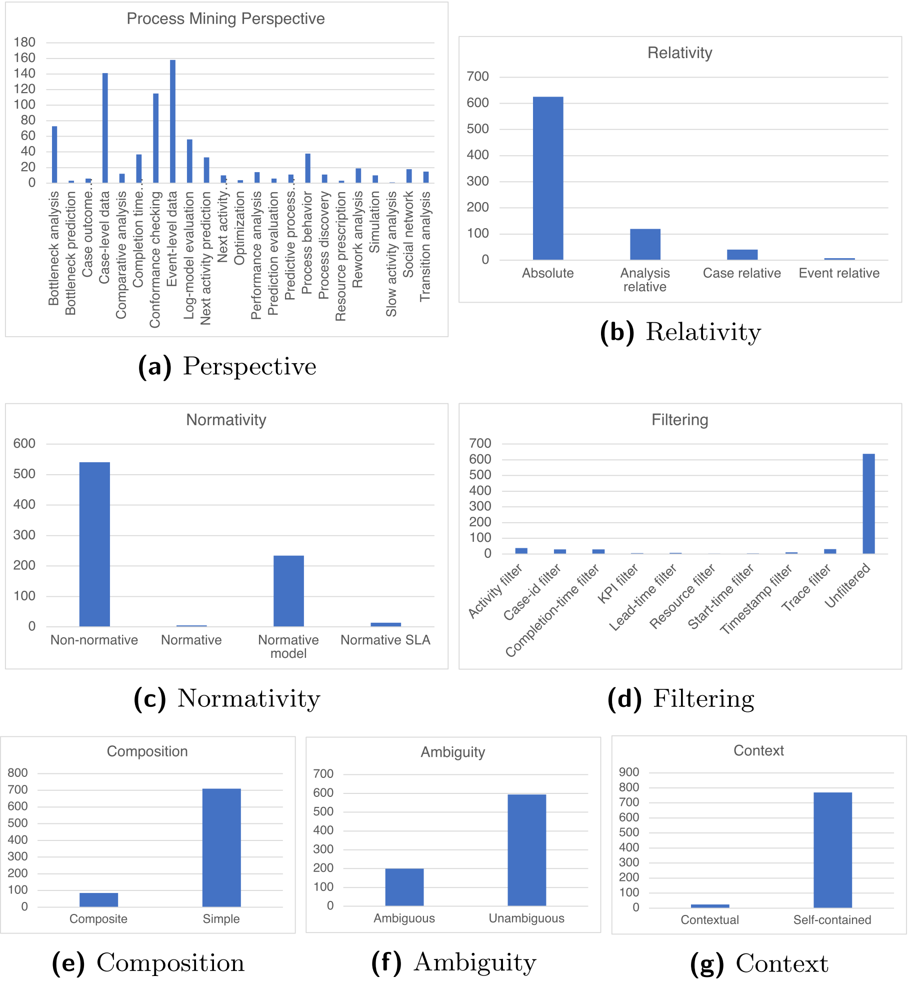
### 4.4.2 Logical representation to tool API mapping

Using the aforementioned endpoints and parameters, it is straightforward to map the logical representation into actual API calls. Table 7 presents examples of this mapping.

The integration of a new process mining tool currently requires a different instantiation of the *tool interface mapping* component to map the generic logical representation generated during semantic parsing to its particular query mechanism. Planned future work includes the definition of a standard API to replace this component and allow process mining tools to easily integrate our natural language conversational interface, as well as integration with another process mining tool, such as PM4Py (Berti et al. 2019).

# 5 Sample questions

An initial set of natural language questions was collected from graduate students with beginner to intermediate level of expertise in process mining, resulting in 250 general (not specific to any existing event log) questions originally written in Portuguese. Free translation was performed by 3 volunteers resulting in 794 questions in English (multiple translations were done by the volunteers for some of the questions). The objective of having



(a) Perspective

(b) Relativity

(c) Normativity

(d) Filtering

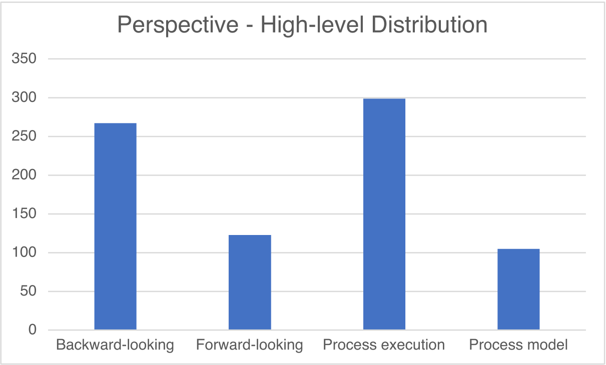
(e) Composition

(f) Ambiguity

(g) Context

**Fig. 7** Question classification distribution

**Fig. 8** Perspective distribution using a high-level classification



different translations for the same question is to add language variability to the test set and, as a result, enhance coverage from the natural language processing perspective, while the resulting logical query to be answered remains the same.

Variability in both language and contents of questions can be further improved in the future by collecting questions directly in English. Furthermore, the collection of question associated to selected, domain-specific, event logs may lead to more realistic questions (closer to what real business users would ask in a specific domain) and is part of planned future work. Nonetheless, as no public dataset of process mining questions is currently available, the samples collected so far played an important role in setting the ground for this research and being the initial input for building the rules used for semantic parsing.

Questions were manually classified into the seven dimensions of the proposed taxonomy. Different translations for the same original question resulted in the same classification. Figure 7 depicts the breakdown of the questions into the different dimensions and their corresponding categories.

Dimension *relativity* is dominated by *relativity:absolute* and *relativity: analysis relative* questions. When it comes to *normativity*, most questions are either *normativity: non-normative* or *normativity: normative model*. The latter represents the questions that take a process model as a norm to be used for conformance checking, for example.

On *composition*, we see that the majority of the questions are classified as *composition:simple*, although we still have 85 questions that are *composition:composite* (two or more nuclei in the question).

When it comes to *filtering*, most questions are *filtering: unfiltered*. This may be a consequence of the way the questions were compiled, as they were detached from a specific event log or context. We would expect more questions to be filtered if there were given context. The same actually applies to dimension *context*, where the vast majority of the questions are *context: self-contained*.

Another striking characteristic of this question dataset is the high number of *ambiguity: ambiguous* questions. We noticed that ambiguity rises from two different factors: (1) language, and (2) unrealistic expectations. In the former, we witness what has been repeatedly reported in NLP, that is, constructions may naturally lead to more than one interpretation of the sentence. However, we would argue that unrealistic expectations played an even more important role in ambiguity. The idea is that the user, once faced with a natural language interface, tends to ask more open questions, as if the system was expected to be a "know-all" kind of oracle. What seems to happen is that this interface raises expectations about the capacity of the underlying system to comprehend very abstract questions, which is not necessarily true. One example of such a question in the dataset is: "Is there something special I should worry about this weekend?". No need to say how hard it is to properly answer such a question.

Finally, the *perspective* dimension depicts a much wider variety of values. Amongst all the possible values, special attention goes to *perspective: case-level data* and *perspective:event-level data*, as well as *perspective:bottleneck analysis* and *perspective: conformance checking*.

Given the myriad of categories under *perspective*, we grouped questions under a higher-level classification, as seen in Fig. 8. In this more summarized classification, analyses subtrees are collapsed under *perspective: backward-looking* and *perspective: forward-looking* only.

Last, but not least, we did a final classification outside the taxonomy realm. In this additional classification, we basically check whether the question is "corrrect" or if it

**Table 8** Experimental results

| Result | Count | Ratio |
|---|---|---|
| Understood | 304 | 63.9% |
| Answered | 266 | 55.9% |
| Partially Understood | 42 | 8.8% |
| Partially Answered | 42 | 8.8% |
| No match | 67 | 14.1% |
| Wrong match | 63 | 13.2% |
| Total | 476 | 100.0% |

has problems related to language (malformed sentences, misspellings, unknown abbreviations) or process mining comprehension (misconception, unknown type of analysis). Examples of incorrect questions are: “Size of my log” (malformed sentence), and “What should I do to view the most frequent cases?” (process mining misconception - cases are unique). Basically, we should not expect the implementation to be able to process most incorrect questions (some malformed sentences and misspellings may be fixable and answerable through the application of specific natural language processing techniques). Luckily, 678 of the 794 questions were deemed “correct”.

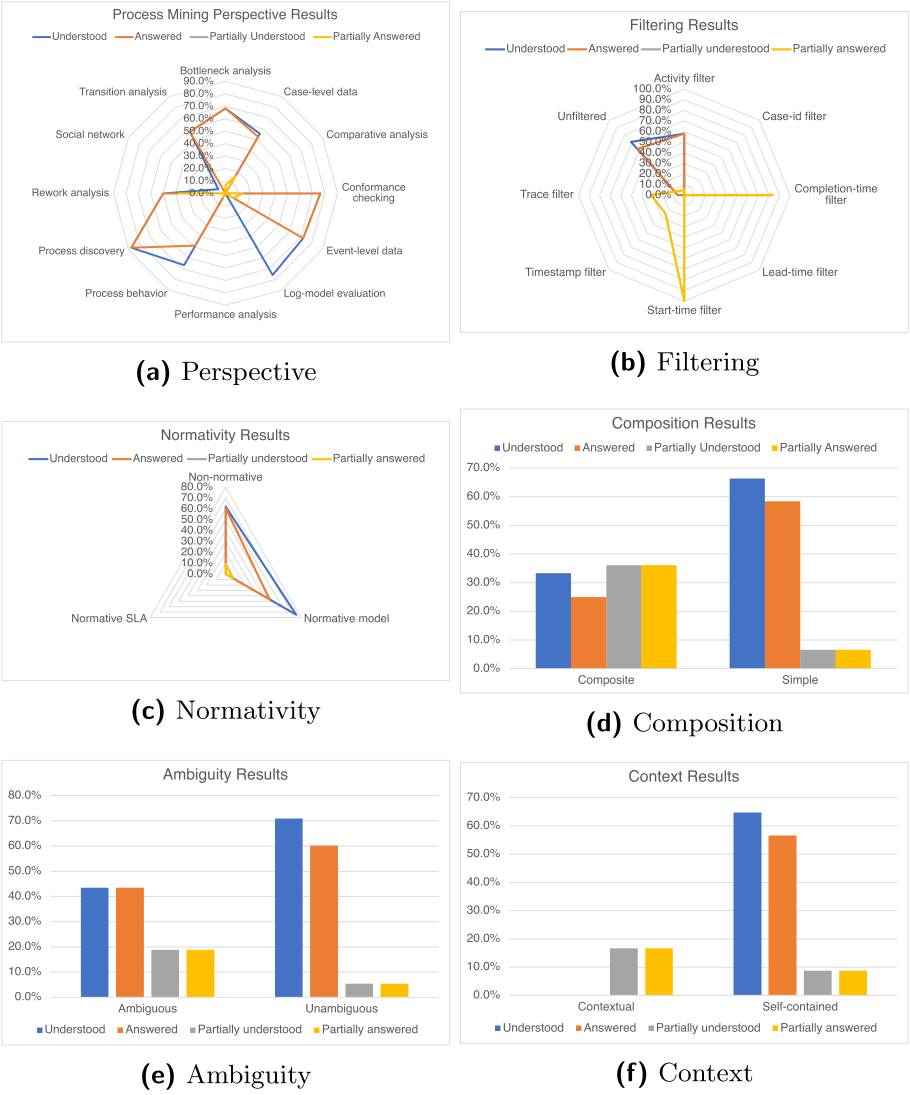
## 6 Experimental results

In order to verify the applicability of the proposed method, we implemented a subset of the architecture (blue colored components) presented in Fig. 2. For the experiment, we used the spaCy open-source natural language processing library, targeting the questions described in Section 5. The library’s Rule Matcher component was used and fed with the semantic rules discussed in Section 4. The source code is available at https://ic.unicamp.br/~luciana.barbieri/pmnli-src.

The testing set was executed against a work force management-based event log that was uploaded into the Everflow Process Mining tool. However, any process mining event log could be used, as the collected questions are not context-specific (not bounded to any particular event log).

Every executed test (question) was labeled with the following statuses:

- *No match*: no rule was fired
- *Wrong match*: the wrong rule was fired
- *Understood*: the question was completely understood, the right rule was fired
- *Answered*: the question was completely understood and the right rule was fired, and the integration with the Process Mining tool returned a complete and correct answer to the question
- *Partially understood*: the question was partially understood and the right rule was fired for the comprehended part (one of two parts of a composite question was understood, for example)
- *Partially answered*: the question was partially understood and the right rule was fired for the comprehended part, or the integration with the Process Mining tool returned a partial answer to the question



**(a)** Perspective

**(b)** Filtering

**(c)** Normativity

**(d)** Composition

**(e)** Ambiguity

**(f)** Context

**Fig. 9** Experimental results

Notice that some of the labels intentionally overlap, as *answered* questions are also *understood* and *partially answered* are (at least) *partially understood*.

In order to make the analysis meaningful, we discarded the questions that were marked "incorrect", for obvious reasons. We also excluded questions associated to the *perspective: forward-looking* subtree or to the *relativity: relative* subtree, as we did not implement the semantic rules to support those categories.

This led to a refined subset of 476 questions (59,6% of the original dataset), for which the summarized results of the processing can be seen in Table 8. Once more, please observe that the counting of *understood* questions encompasses *answered*, and *partially understood* encompasses *partially answered*.

Figure 9 depicts the breakdown of the results for each taxonomic dimensions (except for *relativity*, as *relativity: relative* questions were excluded from the experiment).

When we first analyze the results of the experiment, one can easily realize from Fig. 9a that the performance varies for the different *perspectives*. Amongst the poor performing ones are *perspective: comparative analysis*, *perspective: performance analysis* (generic, non-leaf, category assigned to open questions such as “Where am I wasting the most time in the process?”), and *perspective: social network*. The reason for that is that the implementation did not provide explicit support (including the required semantic rules) to these groups.

The same analysis holds true for the *filtering*, *normativity* and *context* dimensions, where the temporal filter (*filtering: timestamp filter*, *filtering: start-time filter*, *filtering: completion-time filter*, etc.), *normativity:normative SLA* and *context:contextual* categories, respectively, did not perform well due to limited support from the implementation. Some other interesting conclusions can be drawn from the experiment:

- The suggested architecture does go beyond direct queries on the event log. We can see that the implementation performed quite well in groups such as *perspective: bottleneck analysis*, *perspective:conformance checking* and *perspective:process discovery*. Answering questions in these groups requires triggering specific process mining algorithms, that are more commonly found in commercial or academic tools, such as is the case in this experiment (with Everflow Process Mining).
- Understanding a question is a necessary step towards a complete successful answer. This can be seen in the radar chart depicted in Fig. 9a, where the polygon drawn by the lines associated with *understood* questions has a bigger area than the one associated with *answered*. If we want to understand the NLP side of the equation, we should focus on the *understood* group. Basically, the experiment shows that the implementation is able to completely understand 63.9% of the questions, while it can completely answer 55.9%. This decrease is associated with elements beyond NLP, such as the comprehensiveness of the process mining tool’s API, lack of specific implementation of the integration part of some dimensions (such as *perspective: log-model evaluation* and *perspective: comparative analysis*), among others.
- Some groups present a better “understanding rate” than others. This is likely associated to the presence of more defined keywords in the questions, specially around concepts like *perspective: conformance checking*, *perspective: bottleneck analysis*, *perspective:rework analysis* and *perspective: process discovery*. Since the implementation is rule-based, identifying well-defined keywords makes the rule firing more assertive.
- Very open questions, such as the ones in the *perspective: performance analysis* non-leaf category, do harm the ability to properly answer them. Some of these questions are actually *composition: simple* questions (single nucleus) that require the execution of multiple queries to the underlying process mining tool for a correct answer. These would require additional rules to be fully addressed.
- Some questions are really very open, and exemplify the unrealistic expectations mentioned before. Examples are: “Where am I wasting the most time in the process?”, “How can I optimize this process?”, “What is the performance of the model?”. We would argue that these questions would be hard to answer even by a human being.

In general, we consider that the rate of questions that were *understood* is surprisingly high given the limited set of rules that were used and the high number of different *perspectives* to be addressed.

We also believe that a better defined "business" context would probably help create more specific questions, and probably improve understanding and execution, but this remains to be seen in future work.

## 7 Conclusions

A natural language querying interface has the potential of taking process mining applications to the next level by making it easily accessible to non-technical users, such as business and operations teams. Although approaches exist that address this kind of interface in a generic way, none has presented a method that is tailored to deal with process mining data and analyses. In this context, this research presents a reference architecture for such an interface, a taxonomy for process mining related questions, and a public natural language question dataset.

In our experiments, implementing the proposed reference architecture and testing it against the aforementioned sample question dataset has led to several interesting conclusions. Rule-based semantic parsing was an appropriate choice for bootstrapping a natural language interface for process mining as no training data set of any kind is currently available to train any supervised or semi-supervised machine learning technique. Furthermore, as the process mining general ontology is small (few entities and relations), it was possible to answer questions for a selected, pre-defined, set with high accuracy using a relatively small number of rules.

However, evaluation shows that the approach has some limitations. Rule-based semantic parsing does not generalize well, with new rules being required for most new/unpredicted questions. If, on the one hand, this experiment uncovered these challenges, on the other, it verified the viability of the proposed architecture.

It is paramount to highlight that the architectural choice to integrate with other process mining tools gave the approach the ability to add more value to users by exposing advanced algorithms (discovery, conformance, etc) and going beyond queries on the event log file information alone.

The proposed taxonomy alongside the reference architecture lay the foundation for additional work around hybrid approaches (mixing AI and rule-based techniques) to natural language processing in the context of process mining, extending the question dataset (and still be able to compare results due to the taxonomy), automatic question classification (into the proposed taxonomy), additional functionality (forward-looking analyses, natural language response generation, speech-to-text and text-to-speech, etc.) and more process mining tool coverage.

**Acknowledgements** We would like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, for providing the financial support for this work.

**Data availability** The question dataset analysed during the current study is available in https://ic.unicamp.br/~luciana.barbieri/20220306-classifiedpmquestions.csv.

## Declarations

**Conflicts of interest** The authors have no competing interests to declare that are relevant to the content of this article.

## References

Affolter, K., Stockinger, K., & Bernstein, A. (2019). A comparative survey of recent natural language interfaces for databases. *The VLDB Journal, 28*(5), 793–819.

Álvarez, J.M.P., Díaz, A.C., Parody, L., Quintero, A.M.R., Gómez-López, M.T. (2022). Process instance query language and the process querying framework. In: Polyvyanyy, A. (ed.) *Process Querying Methods*, pp. 85–111. Springer.

Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases - an introduction. *Natural Language Engineering, 1*(1), 29–81.

Barbieri, L., Madeira, E.R.M., Stroeh, K., van der Aalst, W.M.P. (2021). Towards a natural language conversational interface for process mining. In: Process Mining Workshops, ICPM 2021. Springer.

Berti, A., van Zelst, S.J., van der Aalst, W. (2019). Process mining for python (pm4py): bridging the gap between process-and data science. In: Proceedings of the ICPM Demo Track 2019, Co-located with 1st International Conference on Process Mining, CEUR Workshop Proceedings 2374, pp. 13–16.

Blunschi, L., Jossen, C., Kossmann, D., Mori, M., & Stockinger, K. (2012). Soda: Generating sql for business users. *Proceedings of the VLDB Endowment, 5*(10), 932–943.

Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M. (2018). Conformance Checking: Relating Processes and Models. Springer.

del-Río-Ortega, A., Resinas, M., Cabanillas, C., Ruiz-Cortés, A. (2013). On the definition and design-time analysis of process performance indicators. Information Systems 38(4), 470–490.

del-Río-Ortega, A., Resinas, M., Durán, A., Ruiz-Cortés, A. (2016). Using templates and linguistic patterns to define process performance indicators. Enterprise Information Systems 10(2), 159–192.

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

Friedrich, F., Mendling, J., Puhlmann, F. (2011). Process model generation from natural language text. In: Advanced Information Systems Engineering, pp. 482–496. Springer.

Han, X., Hu, L., Sen, J., Dang, Y., Gao, B., Isahagian, V., Lei, C., Efthymiou, V., Özcan, F., Quamar, A., Huang, Z., Muthusamy, V. (2020). Bootstrapping natural language querying on process automation data. In: 2020 IEEE International Conference on Services Computing (SCC), pp. 170–177.

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D., & Slocum, J. (1978). Developing a natural language interface to complex data. *ACM Trans. Database Syst., 3*(2), 105–147.

Hompes, B.F., Buijs, J.C., van der Aalst, W.M. (2016). A generic framework for context-aware process performance analysis. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", pp. 300–317. Springer.

Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A.: spaCy: Industrial-strength Natural Language Processing in Python.

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., Zettlemoyer, L. (2017). Learning a neural semantic parser from user feedback. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 963–973. Association for Computational Linguistics.

Kobeissi, M., Assy, N., Gaaloul, W., Defude, B., Haidar, B. (2021). An intent-based natural language interface for querying process execution data. In: 3rd International Conference on Process Mining (ICPM), pp. 152–159. IEEE.

Leopold, H., Mendling, J., Polyvyanyy, A. (2012). Generating natural language texts from business process models. In: Advanced Information Systems Engineering, pp. 64–79. Springer.

Mans, R.S., van der Aalst, W.M.P., Vanwersch, R.J.B. (2015). Process Mining in Healthcare: Evaluating and Exploiting Operational Healthcare Processes. Springer.

Mishra, A., & Jain, S. K. (2016). A survey on question answering systems with classification. *Journal of King Saud University - Computer and Information Sciences, 28*(3), 345–361.

Polyvyanyy, A. (2022). Process query language. In: Polyvyanyy, A. (ed.) *Process Querying Methods*, pp. 313–341. Springer.

Polyvyanyy, A. (2022). Process Querying Methods. Springer.

Polyvyanyy, A., Ouyang, C., Barros, A., & van der Aalst, W. M. P. (2017). Process querying: Enabling business intelligence through query-based process analytics. *Decision Support Systems, 100,* 41–56.

Riefer, M., Ternis, S.F., Thaler, T. (2016) Mining process models from natural language text: A state-of-the-art analysis. Multikonferenz Wirtschaftsinformatik (MKWI-16), March, 9–11.

Saha, D., Floratou, A., Sankaranarayanan, K., Minhas, U. F., Mittal, A. R., & Özcan, F. (2016). Athena: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment, 9*(12), 1209–1220.

van der Aa, H., Leopold, H., & Reijers, H. A. (2017). Comparing textual descriptions to process models - the automatic detection of inconsistencies. *Information Systems, 64,* 447–460.

van der Aalst, W.M.P. (2016). Process mining: data science in action. Springer.

van der Aa, H., Carmona Vargas, J., Leopold, H., Mendling, J., Padró, L. (2018). Challenges and opportunities of applying natural language processing in business process management. In: International Conference on Computational Linguistics, pp. 2791–2801. Association for Computational Linguistics.

Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P. (2010). Xes, xesame, and prom 6. In: Information Systems Evolution, vol. 72, pp. 60–75. Springer.

Viner, D., Stierle, M., Matzner, M. (2020). A process mining software comparison. In: Proceedings of the ICPM Doctoral Consortium and Tool Demonstration Track 2020 Co-located with the 2nd International Conference on Process Mining (ICPM2020), Volume 2703 of CEUR Workshop Proceedings, pp. 19–22.

Wolfson, T., Geva, M., Gupta, A., Gardner, M., Goldberg, Y., Deutch, D., & Berant, J. (2020). Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics, 8,* 183–198.

Zhong, V., Xiong, C., Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv:1709.00103