Enabling Small Language Models for Text-to-Process Extraction: Balancing Accuracy and Efficiency through Distant Supervision

Julian Neuberger[1], Han van der Aa[2], Ivan Khrop[3], and Hugo A. López[1]

[1] Technical University of Denmark, Denmark
jtone@dtu.dk,hulo@dtu.dk
[2] University of Vienna, Vienna, Austria
han.van.der.aa@univie.ac.at
[3] University of Bayreuth, Bayreuth, Germany
ivan.khrop@uni-bayreuth.de

Abstract. In recent years, Large Language Models (LLMs) have received increasing attention for tackling open issues in information systems engineering. A prominent example of this is the extraction of processes from unstructured texts. However, the adoption of LLMs for this task is impeded in many real-world scenarios, due to issues such as hardware limitations, confidential processes, personal information, real-time settings, or concerns about ecological sustainability. In this paper, we, therefore, explore how specialized Small Language Models (SLM) can be used to extract process model information from textual descriptions. We propose a framework for synthesizing *distantly supervised* training data using LLMs, effectively shifting their use away from online inference toward offline training. We find that SLMs can match the extraction quality of modern LLMs orders of magnitude larger, measured on one of the most popular process model extraction datasets. This makes SLMs promising candidates for implementation in on-premise information systems without the need for prohibitively expensive specialized hardware. We make our approach publicly available[4].

Keywords: Business Process Model Generation · Information Extraction · Natural Language Processing · Text-to-Process Extraction.

1 Introduction

Numerous tasks in Business Process Management (BPM) expect conceptual business process models as an input. Specifying such models usually requires the collaboration of a business process analyst and domain experts, a procedure that is time-intensive, reportedly consuming up to 60% of the time allotted to new BPM projects [18]. Business processes are often documented in natural language, embedded in sources such as user manuals, operational guides, or technical documentation [4]. Automatically extracting structured representations of

[4] See https://github.com/JulianNeuberger/slm-for-process-extraction

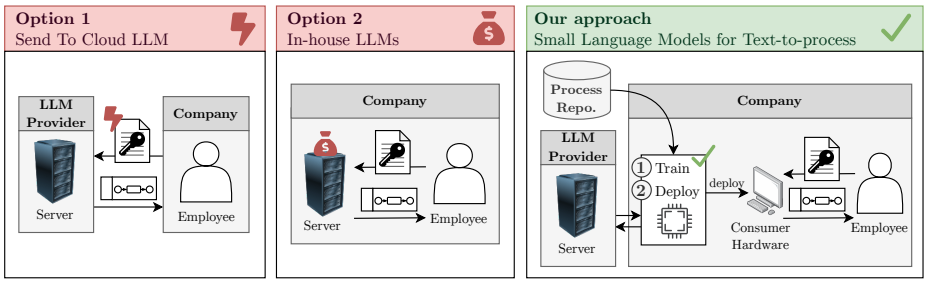


Fig. 1: Illustration of current issues of using LLMs in text-to-process extraction.

such processes from text is thus a key enabler for a range of downstream tasks, including process automation, compliance analysis, and integration with formal process modeling tools [1]. As a result, the task of extracting processes from unstructured documents has received large amounts of attention [8].

Although Large Language Models (LLMs) have been shown to achieve high accuracy for text-to-process extraction tasks [7, 16, 17, 24, 38], their usage also comes with downsides. Existing techniques can be resumed in two approaches (Options 1 & 2 in Fig. 1). The first one is simply to call an online LLMs, with privacy and compliance concerns when sending sensitive or confidential business process descriptions to external cloud providers [10]. An option is to use local LLMs, alleviating privacy concerns. This option incurs high hardware requirements [28]. LLMs that could be deployed on consumer hardware, such as personal computers or laptops, still lag far behind their large alternatives [38]. This means they require specialized hardware, which impedes their use in small and medium-sized companies. In these scenarios, the repeated use of LLMs during inference also raises concerns for environment-conscious companies [51].

While using smaller machine learning models to tackle the task would avoid the aforementioned issues, their adoption is currently hindered by the severely limited availability of labeled training datasets that are required for them. Creating such datasets requires substantial effort from BPM experts, owing to the ambiguous nature of textual business process descriptions [37]. For instance, the creation of one of the most popular datasets covers only 45 texts and required the collaboration of three BPM experts [9].

Recognizing this issue, we present a novel approach to train Small Language Models (SLM) using automatically synthesized training data following the idea of Distant Supervision (DS) [35]. We use the term SLM to describe a language model that is much *smaller* than modern LLMs [53]. In this paper, SLMs are BERT-like [11] (encoder-only) transformer architectures with less than one billion parameters. DS is a technique for automatically generating labeled training data by leveraging external knowledge [35]. Traditionally, this involves aligning text with structured resources such as knowledge bases or handcrafted rules. In BPM, conceptual process models can take this role. These are formal descriptions of business processes (extraction *target*), which can be described to

obtain a natural language description (extraction *input*). This description can be transformed into training data by automatically annotating process-relevant information in the text.

We thus enable the use of SLM for text-to-process extraction, avoiding the downsides of using LLMs for the extraction task[5]. We show that using our approach, we can leverage large, openly available process repositories as a source for training SLM text to process extractors. These are significantly smaller, faster, and energy-efficient than LLMs, while preserving extraction quality. In summary, this paper makes the following contributions:

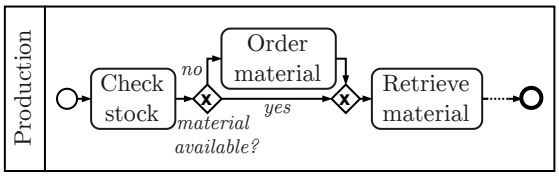
C1. We propose a novel approach for synthesizing large amounts of training data for text-to-process extraction. A key aspect of our approach is that it first describes the contents of a process model in a Controlled Natural Language (CNL) description, which can then be used as a basis for the transformation into and annotation of more realistic textual descriptions.

C2. We show that by using training data generated by our approach, we can train much smaller SLMs that rival the extraction capabilities of LLMs for the text-to-process extraction task, while being faster, requiring less powerful hardware, and consuming less energy.

The remainder of this paper is structured as follows. In Sect. 2 we describe the task of automated text-to-process extraction. In Sect. 3 we detail our approach to DS training of SLMs. In Sect. 4, we evaluate the external and internal validity of our approach. Sect. 5 discusses related work. We conclude this paper in Sect. 6

2 Text-to-Process extraction

This section defines the text-to-process extraction task. Text-to-process extraction deals with automatically generating a conceptual business process model from an unstructured process description in natural language. Such a description could look like the following. *After the process starts, the production department has to check the stock of materials. If no material is available, they order the material. Otherwise, production retrieves it.* This description could be expressed as the following conceptual process model.



The process model is given in Business Process Model and Notation (BPMN), a popular imperative process modeling language. Transforming the textual process description involves solving several sub-tasks and their respective challenges. We

[5] It is important to note that, in our proposal, LLMs are used only once to generate training data. These can then be reused indefinitely as a basis for downstream tasks, in contrast to repeatedly invoking LLMs at inference time for each new instance.

will now recall techniques and terms from Natural Language Processing (NLP) that are used to tackle these sub-tasks.

Segmentation, tokenization, and spans. A given text can be split into its constituent sentences via *segmentation*. Our example would be segmented into two sentences. Each sentence consists of tokens. Tokens can be thought of as words and punctuation for the sake of this example, but in recent years, many approaches consider sub-words during *tokenization* [11]. In our example, *'After', 'the', 'process', 'starts', ','* are tokens. Consecutive tokens are called *spans*.

Named Entity Recognition (NER). Named entities are spans that have a specific type. How we understand named entities in the context of BPM differs from the traditional definition, which primarily considers proper nouns [27, 47]. Instead, we are concerned with spans of text that describe process-relevant information. In our example, these would be, among others, *the production department* (a process role), but also *check* (an activity), and *Otherwise* (a decision point, i.e., XOR gateway).

Relation Extraction (RE). Extracting the interaction between named entities, i.e., process elements, is called relation extraction. RE is defined as extracting triples consisting of the relation type and its two arguments [58]. The type of a relation describes how its two arguments interact, e.g., extracting a relation of type *performer* between *checks* and *the production department* would let us assign the activity *checks* to the pool *the production department*.

Process model generation. Combining the information from NER and RE enables the automatic generation of a business process model using existing algorithms [9,31,37]. This step involves several challenges in itself, such as relabeling *the production department* to *Production*, as shown in our example, as well as laying out the process model. These challenges are out of scope for this paper.

3 Approach

In this section, we present our approach for generating training data for the distantly supervised training of SLMs for the text-to-process task. As visualized in Fig. 2, our approach assumes BPMN models as *input* and produces data for training SLMs for text-to-process extraction as *intermediary output*. This then allows for training SLMs as an *output*. Achieving this involves four distinct stages: *(1)* we generate Controlled Natural Language (CNL) descriptions of BPMN process models by filling templates. *(2)* We rephrase CNL into the style of existing natural language process descriptions using an LLM. In *(3)*, the natural-language description is automatically annotated by an LLM to obtain training data. This training data is used in *(4)* to train an SLM, with an optional fine-tuning step to improve extraction quality. Step *(4)* can be repeated for different SLMs using the same dataset, allowing rapid adoption of new SLMs approaches, without incurring additional costs from dataset generation.

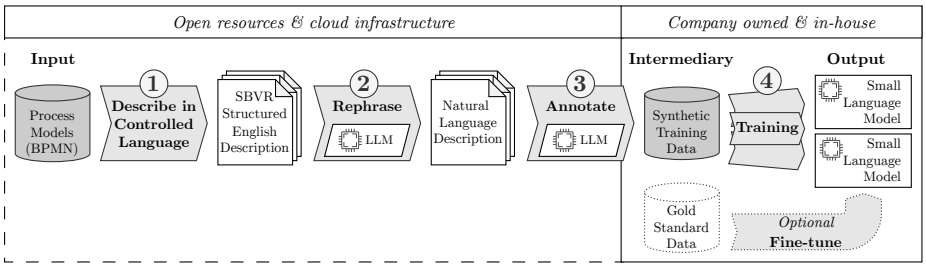


Fig. 2: High-level overview of our proposed DS approach for converting business process models in BPMN into data to train SLMs for text to process extraction.

3.1 Generation of CNL Descriptions

Given a BPMN process model, we first automatically generate a CNL process description algorithmically. In contrast to directly using conceptual models in formats like XML, or JSON, or graphical formats like PNG as input to LLMs, we propose to use CNL descriptions, which are more token-efficient, can be used across all types of LLMs, not just multi-modal ones, and allow ensuring process descriptions are complete, i.e., all elements of the process model are described.

We use Semantics of Business Vocabulary and Business Rules (SBVR) [6] as a CNL specification. SBVR is a formalism approved by the Object Management Group and was developed to express complex business rules.[6] We define SBVR business rule templates for common control-flow patterns [45], if they are fully supported by BPMN[7]. In total, we utilize eleven control flow patterns with corresponding SBVR business rule templates. These are listed in Tab. 1. Each template is associated with a BPMN pattern that we search in a given business process model, which we call a *search pattern*.

Pattern matching. Matching a search pattern in a given BPMN process model entails graph matching. We follow the approach of Dijkman et al. for business process graph matching [12], but modify it for matching sub-graphs of the given process model with our search patterns. Search patterns have different levels of specificity, which becomes clear when comparing the pattern for *Sequence* with that of *Exclusive Choice* in Tab. 1. Here, the first is contained in the latter. To avoid describing a workflow pattern, which was already fully described, we apply precedence to search patterns and search for them in descending order. If we find a search pattern, we mark all matched BPMN elements as `visited`. A search pattern is only counted as found if at least one element in the potential match is `unvisited`. We repeat looking for a search pattern until we find no more matches, and then move on to the next search pattern.

Each pattern in Tab. 1 makes use of *references*. References can be resolved directly if the referenced element is an activity, a process start, or a process end

[6] See https://www.omg.org/spec/SBVR/, last access Nov. 28th, 2025

[7] According to workflowpatterns.com/patterns/control/, last access Nov. 17th, 2025.

event. References to a gateway are resolved by replacing them with the ID of the SBVR rule describing it, e.g, after R0.

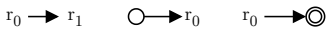
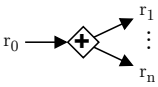
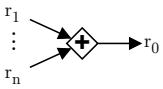
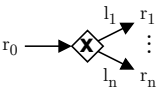
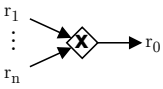
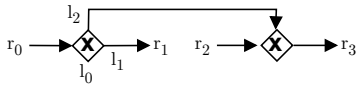
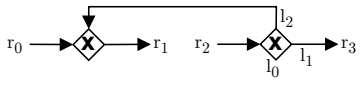
Workflow P.	Search Pattern	Templates
Sequence		It is obligatory that r_1 after r_0
Parallel Split		It is obligatory that **all of** r_1 ... **and** r_n after r_0
Synchron-ization		It is obligatory that r_0 after **all of** r_1 ... **and** r_n
Exclusive Choice		It is obligatory that **one of** r_1 if l_1 ... **or** r_n if l_n after r_0
Simple Merge		It is obligatory that r_0 after **one of** r_1 ... **or** r_n
Optional		It is obligatory that **either** r_1 if l_1 **or not** if l_0 after r_0 It is obligatory that r_3 after **either** r_2 **or** r_0
Structured Loop		It is obligatory that r_1 after **either** r_0 **or** r_2 It is obligatory that **either** r_1 **is repeated** if l_0 **or** r_3 if l_1 after r_2

Table 1: Workflow patterns, their corresponding search patterns, and templates for SBVR Structured English, ordered in descending order of precedence.

SBVR Structured English. Given our running example, applying the template *Sequence* (Tab. 1) would result, among others, in the following SBVR rule.

SBVR rule *R0*

It is obligatory that Production *Check* stock after the process starts.

SBVR rules begin with a key phrase describing the rule's modality. In our approach, we always use It is obligatory that, expressing the obligation to follow the (imperative) control flow. The keyword after expresses the semantics of the flow between the two elements it joins: *check* (an SBVR *verb*) stock (an SBVR *term*) and the process start. The *check* activity is performed by a role, here Production (an SBVR *name*). While the original SBVR annex does not prescribe a single,

definitive formalism to represent business rules, Structured English is commonly used research [6,33]. A business rule expressed in Structured English is split into several syntactic components, which are easy to parse [6]. This makes them a good representation of a process model for the following stages.

3.2 Rephrasing

CNL descriptions from stage *(1)* are repetitive, lack naturalness, and often do not use correct grammar, because we use labels of process elements without modification. Most openly available collections of business process descriptions exhibit characteristics of natural language, such as ambiguity [3, 14, 30], or co-references [15,39]. To get more natural-sounding process descriptions, we prompt an LLM to rephrase the CNL description. Applied to our running example, SBVR rule *R0* is rephrased as the following sentence.

Rephrased rule *R0*

After the process starts, the Production department checks the stock of materials.

Events in the process are now in chronological order, the SBVR name Production was expanded to *the Production department*, and the grammar surrounding *Check* stock is now proper. Phrasing like this can be found in public collections of process descriptions [9].

Prompting strategy. We frame rephrasing as a text style-transfer task by providing an example process description in our prompts. This means the LLM is tasked with transferring the CNL description to the style of a given target domain, such as a company handbook, law text, or medical guideline. LLMs are well-suited tools for this task [43]. The basic structure of this prompt is:

Rephrasing prompt structure

Task. Describe the following set of SBVR rules, in a coherent, continuous text. Make sure it sounds natural and fluid */.../*
Guidelines. Please follow the following guidelines:

- Describe the branches of decisions completely, one after the other
- SBVR rules have XML-style tags around activities, actors, and conditions. These should not appear in the final text
- */.../*

Style Reference. Here is a text in the expected writing style:
The process of Vacations Request starts when any employee of the organization submits a vacation request */.../*

3.3 Automated Annotation

Text from Step *(2)* cannot be used for training SLMs, as it lacks annotations of process-relevant information. This step, therefore, automatically annotates the

generated texts. Usually, this stage involves human annotators, but recent work showed LLMs can perform it with high accuracy and few hallucinations [38].

Prompting strategy. In most applications of text-to-process extraction, we are interested in both entities (process model elements) and relations (connections between model elements). Additionally, a separate co-reference resolution step is often needed to resolve aliases and anaphoras of model elements in natural language process descriptions [39]. We use previous work as the basis for our prompts [38], but combine their iterative prompting strategy into a single, unified prompt and use zero-shot prompting to minimize the number of input tokens.

Annotation hints. Following the idea of DS, we supplement the prompt for annotating entities and relations with the SBVR Structured English description. This description contains some version[8] of activities, actors, and relations, and is therefore helpful for detecting them. Text in SBVR Structured English conveys concepts with different font styles, as shown in previous examples. While this is useful for human readers, it is less so for machines. We instead introduce XML-like tags into the text, where SBVR would use font styles for certain concepts. These tags are only presented to the LLM in addition to the process description it has to annotate, and are generated independently of the CNL in step *(1)*. They therefore do not pose a source of errors to the rephrasing LLM (step *(2)*) or annotating LLM (step *(3)*). Annotation hints for our running example are:

Annotation hints

It is obligatory that `<actor>` Production `</actor>` `<activity>` Check stock `</activity>` after the process starts.

We mark the whole of *Check stock* as an *activity*, while SBVR would consider that *verb* and term. While it might help distinguish between the two, it would entail having additional assumptions about the structure of task labels, i.e., them being in *verb-noun* form. Owing to the short labels in BPMN models, this bears the risk for many false negatives [26]. We therefore accept additional noise in the resulting data, to not further reduce the number of source process models.

3.4 Training SLMs

Using the annotated DS training data from Sect. 3.3 we can now train SLMs for the text-to-process task. The exact procedure depends largely on the chosen SLM, e.g., data format, choice of hyper-parameters, or training regime. Still, our approach considers the training of SLMs with DS training data from *(3)* as *pre-training*. This implies an optional second training run, called *fine-tuning*, which uses human-annotated gold-standard training data, if it is available. This allows an SLM to learn basic concepts about process model elements and their interaction from the noisy DS training data, and subsequently corrects some of the errors introduced by data noise. The trained SLMs represents the output of our approach and can now be used in downstream applications.

[8] The original labels might have been changed during the rephrasing step.

4 Evaluation

In this section, we run several experiments to evaluate the validity of our approach for training SLMs for text to process extraction through DS. In Sect.4.1 we describe how we select business process models from a large process model repository to use as inputs for our approach. We then describe the experimental setup in Sect. 4.2. To evaluate *external* validity, we train four SLMs for the text to model extraction task using our approach and report the results in Sect. 4.3. Sect. 4.4 evaluates *internal* validity through an ablation study of our DS approach. Sect. 4.5 discusses generalization and qualities of generated data. Finally, Sect. 4.6 lists threats to both internal and external validity.

4.1 Process Model Selection

Our approach takes conceptual business process models in the BPMN modeling language as input. For our experiments, we select process models from the SAP Signavio Academic Models (SAM) [50] dataset. The SAP SAM dataset is a collection of 1,021,471 process models with a wide range of properties. Many of the process models contained in SAP SAM are of low technical quality [50], meaning not all process models are suited for synthesizing data from them with our approach. We define seven general and four structural requirements to systematically select high-quality process models for evaluating our approach.

General requirements. Our seven *General Requirements* (**GR**) concern model properties and are as follows.

- **GR1** *Modeling Language*: Models need to be in BPMN 2.0, as the set of proposed templates in Tab. 1 correspond to patterns in BPMN 2.0. This set of templates can be extended to other modeling languages in future work.
- **GR2** *Natural Language*: Model element labels need to be in English, since labels are used to create the CNL representation of a model. However, most modern Large Language Model (LLM) are multi-lingual, so this assumption could be relaxed in the future, but is out of scope for this paper.
- **GR3** *Modeling Elements*: Following zur Muehlen and Recker [36], we limit the elements used in the process model to the most common ones, i.e., *Sequence*, *Tasks*, *Start* and *End Events*, *XOR* and *Parallel Gateways*, *Pools* and *Lanes*. We also exclude *Inclusive Choice Gateways*, as these are not supported by our gold-standard evaluation data (see Sect. 4.2).
- **GR4** *Model Complexity*: To ensure an appropriate level of complexity, models need to have 1–40 tasks, at least one start and end event, 1–10 gateways, and 1–10 actors. Smaller models yield overly simplistic descriptions, while larger models lead to descriptions that are hard to annotate with LLMs.
- **GR5** *Task Labels*: Task labels must describe meaningful activities, not just single-character labels (A, B, C), commonly used in teaching. Our approach relies on labels, such as "*check stock*", to create useful process descriptions.
- **GR6** *Explicit Actor*: Each task needs to be inside a labeled lane or pool. This ensures templates from Tab. 1 can be filled with the executing resource.

Structural requirements. In addition to general requirements for process model properties, we also have four *Structural Requirements* (**SR**) for the control-flow properties of the selected models.

- **SR1** *Weakly Connected*: We remove all process models that are not at least weakly connected. This removes process models with disconnected control flow, for example, missing message flows between BPMN pools. Disconnected control flow leads to incomplete CNL and unnatural process descriptions.
- **SR2** *Reachable*: There must be a path from at least one process start to each task and gateway in the process model. Different from weak connectivity, this considers the direction of sequence flows. It excludes process models where a single sequence flow is mistakenly connected opposite to all other control flow. Like weakly connected processes, this requirement ensures natural sounding process descriptions.
- **SR3** *Gateway Degree*: Each gateway must be connected to at least three sequence flows, with at least one incoming and one outgoing edge. This prevents gateways that neither split nor merge control flow in the process. Some of the models in SAP SAM are incomplete students' assignments, which would prevent matching patterns, e.g., "Exclusive Choice" and "Parallel Split".
- **SR4** *Sequence Flow Degree*: We require sequences to connect exactly two process elements, preventing "dangling" flows, some modeling tools allow. This ensures the "Sequence" pattern can be matched.

The presented sets of general and structural requirements are quite strict and limit the expressiveness of selected models. This decision is by design to make the study at hand feasible. In theory, requirements could be relaxed by introducing more templates and post-process models. However, this would increase the number of selected models to prohibitively large numbers. Instead, follow-up work should study which characteristics of models are useful for training SLMs.

Selected source models. Of the 1,021,471 models in SAP SAM overall, 4,505 fulfill all of our general and structural requirements. The main exclusion reasons are 402,664 non-BPMN models (**GR1**), 353,181 models with excluded elements (**GR3**), and complexity (**GR4**), with 120,456 models. Structural requirements (**SR1** – **SR4**) exclude 1,704 models in total. Future work could relax some of the requirements, especially **GR3**, which currently excludes models containing intermediary events. Selected models are then used as input for training SLMs.

LLMs for Description and Annotation. We use OpenAI's GPT5.1-nano (*gpt-5-nano-2025-08-07*) for the rephrasing step *(2)*. In step *(3)* we use GPT5.1-mini (*gpt-5-mini-2025-08-07*) for annotating rephrased process descriptions. We use the default temperature of 1.0 for both stages. Furthermore, we set reasoning effort to *minimal*, to minimize costs incurred by (reasoning) output tokens.

4.2 Experimental Setup

We run our experiments on a machine with a 2 × 16-core AMD EPYC 7313 Processor, 264GB Memory, and an NVIDIA A40 GPU with 48GB video memory.

Selected SLMs. We select two state-of-the-art SLMs for NER, based on a recent study [20], namely *PIQN* [48] and *ACE* [54]. Similarly we select two SLMs for RE, following a study for RE [58], namely *UniRel* [52] and *PL-Marker* [57]. All approaches use pretrained language models, e.g., BERT [11], to reach best-in-class performance on well-known information extraction datasets. As such, their predictive performance is representative of the current state-of-the-art. Selecting more than one SLM for each task also lets us test our assumption that the same generated data can be used to train multiple SLMs.

Evaluation data. We use the popular PET dataset [9] to evaluate SLM performance after training with our approach. PET is a collection of 45 process descriptions, annotated with process information by three BPM experts. A trained SLM is never evaluated (tested) on synthetic data, only on human-annotated gold standard data from PET. This avoids overfitting to the noise in DS data, which is not present in data curated by human experts. During training, an SLM only has access to synthetic data, not to gold standard data from PET.

Metrics. We report the F_1 score for both the NER and RE tasks, defined as the harmonic mean $F_1 = \frac{2PR}{P+R}$ for the precision $P = \frac{\#ok}{\#pred}$ and recall $R = \frac{\#ok}{\#gold}$, with $\#ok$ as the number of correct predictions, $\#pred$ the number of total predictions, and $\#gold$ the number of gold standard targets. We use the *micro-averaged* F_1 score following previous work [2, 5, 8]. A predicted entity is considered correct if its *type* and at least one *token* match those of a gold standard entity. Similarly, a relation prediction is considered correct if its *type* and both *source* and *target* entities match a gold standard relation.

Methodology. Each SLM is trained with the synthetic DS dataset generated by our approach. They are then fine-tuned with 80% of the PET dataset and tested on the remaining 20%. We report the average F_1 score of five seeds.

Baseline LLMs. We select four different versions of Llama3, with sizes ranging from 1 billion to 70 billion parameters as local LLM baselines for both NER and RE. Additionally, we use OpenAI's GPT-4o, which was previously used in [38] and has estimates regarding its energy consumption [21]. We use these estimates to compare all LLMs to SLMs not only in terms of F_1 score, time, and GPU memory use, but also in terms of energy consumption.

Measuring energy consumption. We use `nvidia-smi` to repeatedly measure the power draw of the GPU. For two points in time a and b, with power draw measurements $f(a), f(b)$, the energy E used between a and b is $E = (b-a) * f(a) + 0.5(b-a)[f(b) - f(a)]$. We inject power measurement into the code of all SLMs, to measure only during prediction, not while loading the model. To avoid the same bias for LLMs, we load them before starting measurements.

4.3 External Validity

The overall results we observed when running the experiments are presented in Tab. 2. In addition to the extraction performance measured using the F_1 score, it also shows average time and energy requirements, and the size on GPU.

	Extraction Model	Avg. F_1 Score and std. dev.	Time per document	Size on GPU	Energy per document
NER	ACE	74.2% ± 4.7%	**4.9s**	10.7GB	**0.12Wh**
	PIQN	**80.7%** ± 3.0%	5.8s	**2.2GB**	0.13Wh
	GPT-4o	80.1% ± 9.2%	10.0s	*N/A*	1.22Wh†
	Llama3.3 70B	70.8% ± 11.4%	96.1s	43.5GB	7.70Wh
	Llama3.1 70B	72.4% ± 13.5%	50.1s	41.1GB	3.88Wh
	Llama3.2 3B	47.3% ± 12.4%	6.4s	4.1GB	0.46Wh
	Llama3.2 1B	25.2% ± 17.1%	12.1s	3.0GB	0.79Wh
RE	PL-Marker	83.3% ± 4.7%	1.4s	5.1GB	0.06Wh
	UniRel*	44.6% ± 6.8%	**0.6s**	8.5GB	**0.01Wh**
	GPT-4o	**91.0%** ± 7.7%	3.85s	*N/A*	1.22Wh†
	GPT-4o*	69.8% ± 16.6%	15.6s	*N/A*	2.43Wh†
	Llama3.3 70B	84.1% ± 10.0%	34.8s	43.5GB	2.18Wh
	Llama3.1 70B	82.3% ± 12.9%	26.2s	41.1GB	2.13Wh
	Llama3.2 3B	28.9% ± 11.6%	2.78s	4.1GB	0.18Wh
	Llama3.2 1B	13.3% ± 9.5%	2.31s	**3.0GB**	0.12Wh

Table 2: Direct comparison between four SLMs and different LLMs.
*Results obtained from running the approach end-to-end, i.e., first extracting entities and then relations. Compounding errors lead to noticeable lower scores.
†Estimates based on Jegham et al. [21]. Note, estimates include optimizations we can not replicate in our setup, such as batching and context caching.

Overall, we find that training SLMs for text to process extraction is feasible when using our DS approach. All SLMs reach extraction quality comparable to that of their respective LLM baselines. Considering that these baselines are either much worse, in the cloud, or require much more energy and time for the same results, this is a very encouraging result. Comparing the time needed to process one process description between SLMs and LLMs, we can observe significant speed-ups. This is not surprising, when recalling that GPT-based LLMs are autoregressive models, i.e., to generate output that is ten tokens long, the entire model has to be evaluated ten times. On the other hand, BERT-based models are not, and one encoding step is enough. Since time and total energy consumption are correlated, the same is true for energy consumption. Smaller models, especially *Llama3.2 1B* tend to fail at instruction following, generating long "chat" responses, which result in higher run-times than its 3B variant.

4.4 Internal Validity

To assess the internal validity of our approach, we run several ablation studies. **No fine-tuning.** If we do not fine-tune the SLMs in step *(4)*, all SLMs have significantly worse extraction quality. *PIQN* drops by 10.4% , similar to *ACE* (−10.5%). For the RE SLMs, *UniRel*'s performance worsens by 3.8% to $F_1 = 34.9\%$, and *PL-Marker* worsens by 13.2%. These results show the importance of fine-tuning, and we recommend it, if gold-standard data is available.

Description generation. Our DS approach generates process descriptions not directly from a conceptual process model, but from SBVR Structured English. We do so to reduce the number of input tokens sent to the LLM, generating and annotating the process description. To quantify the difference, we generated three smaller versions of our DS dataset (roughly 10%, or 392 process descriptions), using *(1)* our proposed SBVR Structured English description as input, *(2)* a PNG image of the process model, and *(3)* both image and SBVR description. We find that using just SBVR in the description prompt resulted in an average of 1,014 input tokens, while using an image resulted in double that (2,063 tokens). Combining both resulted in 2,838 input tokens[9]. Using all three datasets for training the SLMs, we find that there is no noteworthy difference between the SLMs for NER. SLMs for RE on the other hand seem to be affected. *UniRel* improves by +6.4%, *PL-Marker* by +5.0%, when using images. This improvement could be explained by the images, resulting in a better description of relations. In future research, we would like to close this gap while preserving the token efficiency of SBVR. We still recommend rephrasing CNL to obtain process descriptions, as it effectively halves the cost incurred by input tokens.

No annotation hints. To measure how providing the expected targets affects annotation quality, we generate a fourth data set, of the same size as the previous three (392 descriptions). This dataset results from giving no annotation hints during annotation, but otherwise follows our approach. We train all SLMs again using this dataset and compare it to the SLMs trained on a dataset using hints during annotation. *UniRel* (RE) is severely affected by this change, dropping by 15.05% in F_1. We also observed SLMs that were able to compensate for lower data quality, i.e., *PL-Marker* (−0.07%). Still, providing the conceptual process model has no adverse effects at worst, and is helpful at best. We therefore recommend providing it during annotation in the form of CNL.

4.5 Discussion

The set of patterns and templates from Sect. 3.1 currently focus on input process models in BPMN, which is an imperative modeling language. To apply our approach to other modeling languages, particularly declarative ones, such as Declare [40] or DCR [19], the set of templates needs to be extended. While this does not limit the applicability of the concept of our approach, it would be an interesting avenue of future research. We currently use the PET annotation schema for automated annotations, although adapting it to a different schema would only require changing the annotation prompts accordingly.

Automatically generated training data contains errors that might affect SLM performance. Fig. 3 shows a fragment of automatically generated data for the running example process (Fig. 1). It contains annotation errors, e.g., the first "manufacturing", or multiple wrong further specifications, e.g., "first", "only if needed". Some sequence flows are missing, e.g., between "checking" and the XOR

[9] Which is less than the combination of the previous two values, as the common parts of the prompt (e.g., *"You are a process analyst, ..."*) are not duplicated.

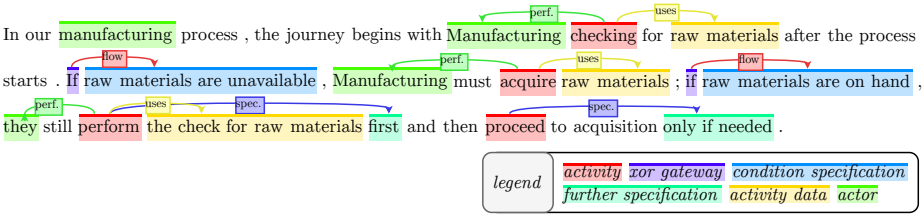


Fig. 3: Fragment of generated data for the running example process.

gateway ("If"). Parts of the description are overspecified, e.g., restating the check for raw materials, which are artifacts of CNL specifications. Ultimately, lower quality is expected from an automated approach and are balanced out by the overall quantity of data. How many errors in process models, text generation, and data annotation affect SLM performance will be explored in future work.

4.6 Threats to Validity

Our work has some limitations that should be considered when applying our findings. First, we used four SLMs for the main experiments in Sect. 4.3. While these models were picked based on scores in recent studies, these studies are focused on information extraction from texts outside the BPM domain. This could introduce bias in model selection, and requires further experiments with more varied SLMs. Similarly, we chose the default hyper-parameter configuration for these SLMs, and did not run a hyper-parameter optimization ourselves. This can introduce bias, as hyperparameters are optimized for datasets outside the BPM domain. Our results should therefore be understood as a lower bound, which could be improved through hyper-parameter optimization. Regarding power measurements, we used `nvidida-smi`, which deviates reportedly as much as 5% from physical power readings, and can only measure with a frequency of 10Hz [55]. To obtain more accurate results, especially for *UniRel*, would require a physical experiment setup, not just software, as we used. Since all LLMs and SLMs are subject to this limitation, our results are still a good indication of relative energy consumption. Next, real-world process descriptions often contain ambiguous statements [3,30]. While the rephrasing step from CNL to natural language might introduce ambiguities, we currently have no mechanism to ensure they do. This might limit the applicability to datasets other than the one we used in our study. Finally, our requirements for selecting business process models from SAP-SAM are strict, resulting in just over 4,500 process models. We do not explore the effects of relaxing requirements, e.g., allowing intermediary events and transforming them to activities during CNL construction.

5 Related Work

Distant Supervision. Synthesizing DS training data is a popular technique in the field of information extraction. Mintz et al. [35] initially proposed distant

supervision for RE using the now-discontinued FreeBase knowledge database. The NYT [44] and DocRed [56] datasets are two examples of datasets that use a knowledge base to automatically generate large amounts of noisy training data.

Automatic Model Generation. Automatic text-to-process extraction is a field of research that became increasingly popular in recent years. Starting with the seminal work by Friedrich et al. [15], many other rule-based approaches have been proposed [9,13,42,46]. While these approaches are very data-efficient, they are hard to extend to new types of process-relevant information, giving rise to SLM approaches [5, 9, 39, 41]. These SLM approaches usually possess only a very limited number of trainable parameters, which makes them data-efficient, but less powerful than larger SLMs. With the advent of powerful LLMs, many approaches using LLMs have been proposed, either for text-to-process extraction [7,38], or for conversational business process modeling [22–24].

Automatic Model Description. Leopold et al. [25,26] propose an approach for generating natural language process descriptions for supporting process analysts in model validation. There is also a stream of work implementing the transformation of conceptual process models into SBVR as plugins for *MagicDraw* [32–34,49]. Since these approaches are only available through this tool, we cannot use them directly, although we adopt their ideas partially.

6 Conclusion

We presented a novel approach using Distant Supervision (DS) to train Small Language Models (SLMs) for text-to-process extraction. Our approach entails four distinct steps: *(1)* algorithmically generating Controlled Natural Language (CNL) from business process models, *(2)* rephrasing CNL into natural language using an LLM, *(3)* annotating this natural language process description using an LLM, and *(4)* training SLMs for text to process extraction. We find that SLMs trained with our DS approach achieves extraction quality similar to LLMs, such as GPT4o or Llama3.3 70B, while requiring less hardware, time, and energy.

In future work, we plan to extend the comparison of local LLMs and SLMs, especially regarding energy usage and efficiency. This aspect of artificial intelligence becomes increasingly relevant and should also be investigated in BPM research. In this work, we focused on BPMN, but our approach supports the generation of declarative process descriptions, a key resource for improving Text-to-Process extraction techniques for declarative process models [29].

Acknowledgments. We would like to thank Lars Ackermann for his helpful insights and comments on the use of SBVR for business process descriptions. This work has been supported by the grant "Center for Digital CompliancE (DICE)" (VIL57420) from VILLUM FONDEN, and by the Innovation Foundation project "Explainable Hybrid-AI for Computational Law and Accurate Legal Chatbots" 4355-00018B XHAILe.

References

1. van der Aa, H., Carmona Vargas, J., Leopold, H., Mendling, J., Padró, L.: Challenges and opportunities of applying natural language processing in business process management. In: COLING (2018)
2. van der Aa, H., Di Ciccio, C., Leopold, H., Reijers, H.A.: Extracting declarative process models from natural language. In: CAiSE (2019)
3. van der Aa, H., Leopold, H., Reijers, H.A.: Dealing with behavioral ambiguity in textual process descriptions. In: BPM. pp. 271–288. Springer (2016)
4. van der Aa, H., Leopold, H., van de Weerd, I., Reijers, H.A.: Causes and consequences of fragmented process information: Insights from a case study. In: AMCIS (2017)
5. Ackermann, L., Neuberger, J., Jablonski, S.: Data-driven annotation of textual process descriptions based on formal meaning representations. In: CAiSE (2021)
6. Bajwa, I.S., Lee, M.G., Bordbar, B.: Sbvr business rules generation from natural language specification. In: AAAI spring symposium: AI for business agility. pp. 2–8 (2011)
7. Bellan, P., Dragoni, M., Ghidini, C.: Extracting business process entities and relations from text using pre-trained language models and in-context learning. In: EDOC (2022)
8. Bellan, P., Dragoni, M., Ghidini, C., van der Aa, H., Ponzetto, S.P.: Process extraction from text: benchmarking the state of the art and paving the way for future challenges. arXiv:2110.03754 (2021)
9. Bellan, P., Ghidini, C., Dragoni, M., Ponzetto, S.P., van der Aa, H.: Process extraction from natural language text: the PET dataset and annotation guidelines. In: NL4AI (2022)
10. Das, B.C., Amini, M.H., Wu, Y.: Security and privacy challenges of large language models: A survey. ACM Computing Surveys (2025)
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018)
12. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: BPM. pp. 48–63. Springer (2009)
13. Ferreira., R.C.B., Thom., L.H., Fantinato., M.: A semi-automatic approach to identify business process elements in natural language texts. In: ICEIS (2017)
14. Franceschetti, M., Seiger, R., López, H.A., Burattin, A., García-Bañuelos, L., Weber, B.: A characterisation of ambiguity in bpm. In: ER (2023)
15. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: CAiSE (2011)
16. Grathwol, D., van der Aa, H., López, H.A.: Automating pathway extraction from clinical guidelines: A conceptual model, datasets and initial experiments. In: CoopIS (2024)
17. Grohs, M., Abb, L., Elsayed, N., Rehse, J.R.: Large language models can accomplish business process management tasks. In: BPM (2023)
18. Herbst, J., Karagiannis, D.: An inductive approach to the acquisition and adaptation of workflow models. In: Proceedings of the IJCAI. vol. 99, pp. 52–57. Citeseer (1999)
19. Hildebrandt, T.T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. arXiv preprint arXiv:1110.4161 (2011)

20. Hu, Z., Hou, W., Liu, X.: Deep learning for named entity recognition: a survey. Neural Computing and Applications (2024)
21. Jegham, N., Abdelatti, M., Koh, C.Y., Elmoubarki, L., Hendawi, A.: How hungry is ai? benchmarking energy, water, and carbon footprint of llm inference. arXiv preprint arXiv:2505.09598 (2025)
22. Köpke, J., Safan, A.: Efficient llm-based conversational process modeling. In: BPM (2024)
23. Köpke, J., Safan, A.: Introducing the bpmn-chatbot for efficient llm-based process modeling. In: BPM (2024)
24. Kourani, H., Berti, A., Schuster, D., van der Aalst, W.M.: Process modeling with large language models. arXiv:2403.07541 (2024)
25. Leopold, H., Mendling, J., Polyvyanyy, A.: Generating natural language texts from business process models. In: CAiSE. pp. 64–79. Springer (2012)
26. Leopold, H., Mendling, J., Polyvyanyy, A.: Supporting process model validation through natural language generation. IEEE TSEN **40**(8), 818–840 (2014)
27. Li, J., Sun, A., Han, J., Li, C.: A survey on deep learning for named entity recognition. IEEE Transactions on Knowledge and Data Engineering (2020)
28. Li, J., Xu, J., Huang, S., Chen, Y., Li, W., Liu, J., Lian, Y., Pan, J., Ding, L., Zhou, H., et al.: Large language model inference acceleration: A comprehensive hardware perspective. arXiv:2410.04466 (2024)
29. Lindner, J., López, H.A.: Discovering declarative processes in textual descriptions using large language models. In: ICPM Workshops. Springer (2025)
30. López, H.A., Feng, B., Lindner, J., Franceschetti, M., Abbad-Andaloussi, A.: Ambiguity detection in business process descriptions: An evidence and an automated approach. In: BPM. pp. 379–396. Springer (2025)
31. López, H.A., Strømsted, R., Niyodusenga, J.M., Marquard, M.: Declarative process discovery: Linking process and textual views. In: International Conference on Advanced Information Systems Engineering (2021)
32. Malik, S., Bajwa, I.S.: Back to origin: Transformation of business process models to business rules. In: International Conference on Business Process Management. pp. 611–622. Springer (2012)
33. Mickeviciute, E., Nemuraite, L., Butleris, R.: Applying sbvr business vocabulary and business rules for creating bpmn process models. In: International Conference on Business Information Systems. pp. 105–116. Springer (2014)
34. Mickevičiūtė, E., Nemuraitė, L., Butleris, R.: Improving bpmn2 business process model to sbvr business vocabulary and business rules transformation with bpmn2 event naming patterns. Information technology and control. **45**(4), 443–451 (2016)
35. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: ACL-IJCNLP (2009)
36. Muehlen, M.z., Recker, J.: How much language is enough? theoretical and practical use of the business process modeling notation. In: Seminal Contributions to Information Systems Engineering: 25 Years of CAiSE, pp. 429–443. Springer (2013)
37. Neuberger, J., van der Aa, H., Ackermann, L., Buschek, D., Herrmann, J., Jablonski, S.: Assisted data annotation for business process information extraction from textual documents. In: CoopIS (2024)
38. Neuberger, J., Ackermann, L., van der Aa, H., Jablonski, S.: A universal prompting strategy for extracting process model information from natural language text using large language models. In: ER (2024)
39. Neuberger, J., Ackermann, L., Jablonski, S.: Beyond rule-based named entity recognition and relation extraction for process model generation from natural language text. In: CoopIS (2023)

40. Pesic, M., Schonenberg, H., Van der Aalst, W.M.: Declare: Full support for loosely-structured processes. In: 11th IEEE international enterprise distributed object computing conference (EDOC 2007). pp. 287–287. IEEE (2007)
41. Qian, C., Wen, L., Kumar, A., Lin, L., Lin, L., Zong, Z., Li, S., Wang, J.: An approach for process model extraction by multi-grained text classification. In: CAiSE (2020)
42. Quishpi, L., Carmona, J., Padró, L.: Extracting annotations from textual descriptions of processes. In: BPM 2020 (2020)
43. Reif, E., Ippolito, D., Yuan, A., Coenen, A., Callison-Burch, C., Wei, J.: A recipe for arbitrary text style transfer with large language models. In: ACL. pp. 837–848 (2022)
44. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Joint European conference on machine learning and knowledge discovery in databases. pp. 148–163. Springer (2010)
45. Russell, N.C., Ter Hofstede, A.H., Van Der Aalst, W.M., Mulyar, N.A.: Workflow control-flow patterns: A revised view (2006)
46. Sànchez-Ferreres, J., Burattin, A., Carmona, J., Montali, M., Padró, L., Quishpi, L.: Unleashing textual descriptions of business processes. SoSyM (2021)
47. Sharnagat, R.: Named entity recognition: A literature survey. Center For Indian Language Technology (2014)
48. Shen, Y., Wang, X., Tan, Z., Xu, G., Xie, P., Huang, F., Lu, W., Zhuang, Y.: Parallel instance query network for named entity recognition. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics (2022), https://arxiv.org/abs/2203.10545
49. Skersys, T., Danenas, P., Mickeviciute, E., Butleris, R.: Transforming bpmn processes to sbvr process rules with deontic modalities. Applied Sciences **12**(18), 8976 (2022)
50. Sola, D., Warmuth, C., Schäfer, B., Badakhshan, P., Rehse, J.R., Kampik, T.: Sap signavio academic models: A large process model dataset (2022)
51. Stojkovic, J., Choukse, E., Zhang, C., Goiri, I., Torrellas, J.: Towards greener llms: Bringing energy-efficiency to the forefront of llm inference. arXiv:2403.20306 (2024)
52. Tang, W., Xu, B., Zhao, Y., Mao, Z., Liu, Y., Liao, Y., Xie, H.: Unirel: Unified representation and interaction for joint relational triple extraction. arXiv:2211.09039 (2022)
53. Wang, F., Zhang, Z., Zhang, X., Wu, Z., Mo, T., Lu, Q., Wang, W., Li, R., Xu, J., Tang, X., et al.: A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. ACM Transactions on Intelligent Systems and Technology **16**(6), 1–87 (2025)
54. Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F., Tu, K.: Automated Concatenation of Embeddings for Structured Prediction. In: ACL-IJCNLP (2021)
55. Yang, Z., Adamek, K., Armour, W.: Part-time power measurements: nvidia-smi's lack of attention. arXiv preprint arXiv:2312.02741 (2023)
56. Yao, Y., Ye, D., Li, P., Han, X., Lin, Y., Liu, Z., Liu, Z., Huang, L., Zhou, J., Sun, M.: Docred: A large-scale document-level relation extraction dataset. arXiv:1906.06127 (2019)
57. Ye, D., Lin, Y., Li, P., Sun, M.: Packed levitated marker for entity and relation extraction. arXiv:2109.06067 (2021)
58. Zhao, X., Deng, Y., Yang, M., Wang, L., Zhang, R., Cheng, H., Lam, W., Shen, Y., Xu, R.: A comprehensive survey on relation extraction: Recent advances and new frontiers. ACM Computing Surveys **56**(11), 1–39 (2024)