UNIVERSITÄT
BAYREUTH

# Extracting Formal Process Model Information from Natural Language Text Descriptions using Machine Learning

Von der Universität Bayreuth
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

von
**Julian Tobias Neuberger**
aus Kronach

1. Gutachter: Prof. Dr.-Ing. Stefan Jablonski
2. Gutachter: Prof. Dr. Stefan Schönig

Tag der Einreichung: 27.02.2025
Tag des Kolloquiums: 16.04.2025

# Zusammenfassung

Geschäftsprozessmanagement hilft Unternehmen und Organisationen dabei, wiederkehrende Arbeitsabläufe — sogenannte Geschäftsprozesse — zu verstehen, zu verbessern und zu automatisieren. Ein zentraler Bestandteil davon ist ein formales Geschäftsprozessmodell, dessen Erstellung kostspielig ist, da es die Zusammenarbeit zwischen Prozessbeteiligten und Prozessanalysten erfordert. Die automatische Generierung formaler Geschäftsprozessmodelle ist ein Forschungsbereich, der die Erstellung solcher Modelle erleichtern soll, indem Informationen aus bestehenden natürlichsprachlichen Prozessbeschreibungen extrahiert werden, woraus automatisch ein entsprechendes formales Prozessmodell synthetisiert wird. Obwohl dieses Forschungsfeld in den letzten Jahren zunehmend an Bedeutung gewonnen hat, basieren viele Ansätze zur Extraktion von Prozessinformationen weiterhin auf Systemen mit handgeschriebenen Regeln. Dies erschwert deren Anwendung in neuen Domänen, Beschreibungssprachen und Prozessmodellierungssprachen. Ansätze, die auf maschinellem Lernen und Deep Learning basieren, bieten hier erhebliche Vorteile, da sie Extraktionsregeln automatisch aus Daten ableiten und dadurch leichter übertragbar sind. Dennoch sind solche Ansätze bisher selten, was sich durch den hohen Bedarf an Trainingsdaten erklären lässt. Diese Arbeit stellt die folgenden drei Ansätze vor, um den Mangel an ausreichend großen Trainingsdatensätzen in der Prozessinformationsextraktion zu beheben. Beschleunigte Sammlung von neuen Trainingsdaten durch angemessene digitale Unterstützung, effizientere Nutzung von vorhandenen Trainingsdaten und die Anwendung von vortrainierten Sprachmodellen. So ebnet diese Arbeit den Weg für zukünftige erfolgreiche Anwendungen von Text-zu-Modell-Methoden.

# Abstract

Business process management helps businesses and organizations with understanding, improving, and automating reoccurring workflows, called business processes. One integral part to this is a formal business process model, which is expensive to create, as it needs collaboration between process participants and process analysts. Automatic generation of formal business process models is a research field working on facilitating the creation of formal process models, by extracting the information contained in existing natural language process descriptions and synthesizing a formal process model. While this field of research has become increasingly popular in recent years, many process information extraction approaches still rely on hand-written rules, which makes them hard to transfer to new application domains, description languages, or process modeling languages. Approaches based on machine learning and deep learning would provide significant benefits in this regard, as extraction rules are derived from data automatically and are therefore easier to adjust. Yet, these approaches are still rare, which can be explained by their need for large amounts of training data. This thesis presents three options to mitigate the lack of large training datasets in process information extraction, which are as follows. Facilitating the collection of new training data via proper tool support, more efficient use of existing training data, and the application of pretrained language models. Thus, this thesis paves the way for successful applications of text-to-model methods in the future.

# Acknowledgments

This thesis has been one of the longest running projects that I worked on. Simultaneously, it was also the most fulfilling one thanks to a lot of people I have met along the way. First of all, I would like to thank my supervisor Prof. Dr. Stefan Jablonski, for sharing his deep knowledge about process management, for reading and revising my publications, including this thesis, and for his guidance along the way. I also want to express my gratitude towards Prof. Dr. Lars Ackermann, who mentored me since my first day at the chair, helped me improve my writing manyfold, and was always available as a sparring partner. Louise Schaub deserves special thanks for putting up with me after long nights of tight deadlines, revising my writing with attention to detail, and always supporting me when I needed it the most. I want to thank Prof. Dr. Han van der Aa for the fruitful collaboration and the insights he provided during our work on the application of natural language processing on process descriptions. Further, I would like to thank Prof. Dr. Daniel Buschek for sharing his expertise with user study design, resulting in one of the most exciting publications I had the pleasure of working on. I also would like to thank my colleagues Martin Käppel and Sebastian Petter, with whom I had many invigorating discussions, and who always made sure I would never forget about a well timed lunch break. Finally, I want to thank my family and friends for their unwavering support. This thesis would not have been possible without you all, and even if it was, it would not have been this fun.

# Contents

# Chapter 1

# Introduction

The central goal of Business Process Management is reducing the complexity in coordination of labor in modern organizations [75]. Benefits of proper management include, among others, increased productivity, quality, or compliance with regulations [23]. Typically, to achieve these benefits a formal model of the business process is needed, which is very expensive to create as it needs intensive collaboration of process analysts and process owners [29]. Research estimates the cost of creating an as-is process model without any improvements at 60% of the time allocated for the business process management project as a whole [33]. To facilitate the creation of process models, researchers turn to analyzing process data already available at organizations. Although process mining methods have shown practical success using such data by analyzing logs of workflow events, another widespread source is often overlooked by practitioners: Textual process documentation, such as employee notes, legislation, or standard operating procedures. In such documents the process is described in natural language and contains information needed to create a (rough) formal process model, which can subsequently be refined with process stakeholders.

Since process descriptions are not created with machine-readability in mind, text-to-model can be understood as a two-step task [8]. First, all process relevant information has to be extracted from the text, which includes finding process elements such as actors and activities. Second, the extracted process information is transformed into a formal process model in a given modeling language, such as Business Process Model And Notation (BPMN)[1]. Figure 1.1 visualizes the text-to-model task.

Extracting relevant process information is a challenging step, as various issues with the nature of language complicate information extraction [75]. These issues include, but are not limited to, linguistic ambiguity and implicit information [76], that require careful integration of expert knowledge into methods dealing with them. Thus, even today the extraction of process information is still done with rule-based systems [53]. Defining such rules requires extensive understanding of the targeted process modeling language, as well as natural language processing. The resulting rules are hard to transfer across organizations, data sources, and process modeling languages, which impedes reusing existing systems.

Research in the closely related field of natural language processing has shifted to machine learning approaches with great success, e.g., in identifying the role of words in a sentence [14],

[1] See `https://www.omg.org/bpmn/`, last accessed Jan 14, 2025.

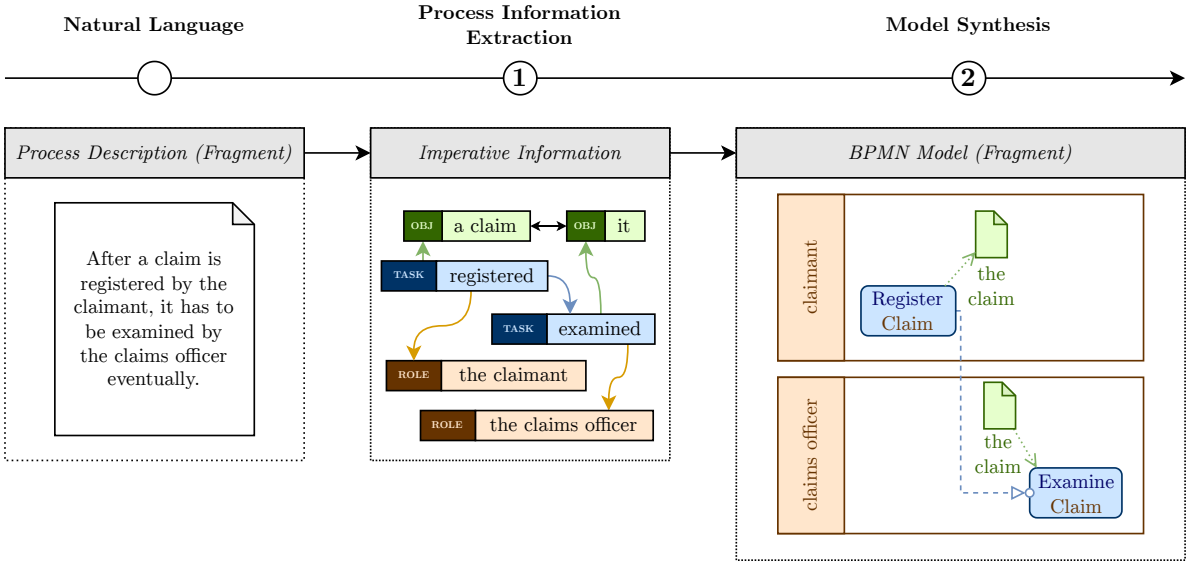



Figure 1.1: Motivating example of a short process description fragment, the corresponding process information, and finally the generated process model fragments.

finding named entities [43], or automatic translation [82]. While machine learning approaches alleviate the most urgent issues of rule-based approaches, they require large amounts of training data, which needs expensive manual processing before use. We identify this fact as the main reason for slow adoption of powerful, yet flexible data-driven approaches towards process information extraction [3, 53, 52, 51]. Lack of data is still a significant hurdle in the development of new approaches, even when pretrained generative large language models (colloquially known as generative artificial intelligence) are used. Here, rigorous evaluation on many different data sources is essential to assess their practicality, especially in light of the large variation in terms of the structure, style, and contents of textual documents that contain process information [2].

This thesis and the reprinted publications, starting with Chapter 5, propose and evaluate three options of addressing the lack of high-quality training data for process information extraction systems. Each option can be assigned to a stage in the development of such a system, as shown in Figure 1.2.

**1** *Efficient Annotation* (**EA**): Simplifying the collection of training data for process information extraction methods, i.e., by identifying how to reduce the workload of the data annotation task.

**2** *Data Efficiency* (**DE**): Use existing training data more efficiently, and even create new, synthetic training data by leveraging data augmentation for process information extraction.

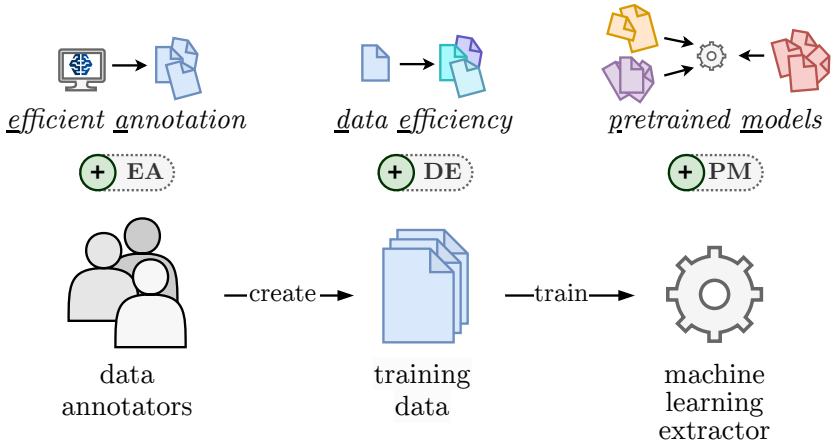



Figure 1.2: Contributions of this thesis for developing new machine learning approaches for process information extraction.

**3** *Pretrained Models* (**PM**): Using pretrained models from the field of natural language processing, either as core component of process information extraction methods, or as way of creating features for other, smaller models.

The remainder of this thesis is structured as follows. Chapter 2 defines basic terms and concepts needed throughout the rest of this thesis and the reprinted work. Chapter 3 discusses work related to this thesis. In Chapter 4 we give an extended overview over the work reprinted in this thesis. Chapter 5 presents our initial approach towards applying machine learning in process information extraction. Chapters 6 and 7 contain publications where we integrate pretrained machine learning models to reduce the amount of process information extraction data needed for training extractors, and can therefore be seen as representatives of option **3** (**PM**). In extension, Chapter 9 reprints our approach towards process information extraction using large language models, therefore removing the need for training data entirely. Chapters 8 and 12 reprint our investigations into creating synthetic data from a limited dataset using data augmentation techniques, i.e., option **2** (**DE**). Chapters 10 and 11 contain reprints of our papers discussing how to facilitate the creation of large, high-quality process information extraction datasets, i.e., option **1** (**EA**). Finally, in Chapter 13 we discuss our contributions, limitations, and present avenues of future work and further research.

This thesis would not have been possible in its current form without the help and support of my supervisors, colleagues, and students. To reflect this, the scientific "we" is used in the remainder of this thesis.

# Chapter 2

# Preliminaries

In the following chapter we define key concepts and terms we will use throughout the remainder of this thesis. In Section 2.1 we discuss business process models in general and explain fundamental differences in two approaches towards business process modeling. Section 2.2 discusses two main tasks that must be solved by all approaches towards automatic business process modeling, and presents datasets used for developing new solution approaches. In Section 2.3 we will define core concepts of supervised machine learning. Section 2.4 will give a brief overview of terminology and ideas of modeling natural language, and as an extension of that, how generative models are used to solve tasks that are not explicitly in their training objective. Finally, Section 2.5 introduce the terms data augmentation and oversampling.

## 2.1 Business Process Modeling

A business process model is a conceptual model that describes a business process, that is, a set of (partially) ordered events, activities, and decision points, involving actors and objects, collectively leading to a desired outcome [23]. Depending on the language used to model the business process, there are up to five main perspectives contained in the model.

1. The *functional* perspective defines how activities are to be done, which is usually information stored in activity labels, annotations of activities, etc.

2. The *behavior* perspective of a process is defined by its activities and their order of execution, the control flow.

3. The *data* perspective adds information about physical and digital data (and material) that is created, consumed, and transformed during a process.

4. The *organizational* perspective contains the process participant responsible for or affected by an activity, which additionally may include, for example, information about company structure or employee capabilities.

5. The *operational* perspective of a process defines the tools, machinery, and systems used during a process.

There are two major approaches to describing the behavioral perspective of a process, called imperative and declarative modeling.

**Imperative Modeling.** For processes where the sequence of tasks is well known, and deviates rarely, imperative modeling of a process can be useful. In imperative models, the order of execution is given as a sequence of steps, which may contain branching to express exclusive or parallel workflows. Popular imperative modeling languages include BPMN and UML.

**Declarative Modeling.** In contrast to imperative modeling, which describes the normal, expected workflow, declarative models define constraints that have to hold throughout the execution of a process. Constraints define, for example, if a given activity is required, if executing one activity prevents execution of a second, or if executing one activity requires execution of another activity first. As long as all constraints in a declarative process model are satisfied, any sequence of activity executions is allowed, which grants much greater freedom during execution of a process. A commonly used formalism for declarative process modeling in both academia and industry is *Dynamic Condition Response* (DCR) [70].

## 2.2 Automated Generation of Business Process Models

In this section we will discuss how a formal business process model can be extracted from a natural language process description automatically. We first give a high-level overview of the two steps in the text-to-model task, process information extraction and process model synthesis. We then define process information extraction in more detail (Section 2.2.1), challenges that need to be addressed during this step (Section 2.2.2), as well as two datasets used for developing new extraction approaches (Sections 2.2.3 and 2.2.4). Finally, we give a short algorithm for synthesizing a formal business process model from extracted process information (Section 2.2.5.

Generating a business process model from natural language texts can be described by a two step procedure [8], which is visualized in Figure 2.1. First, all the information that is relevant to a process model has to be *extracted*, i.e., fragments of a text that describe parts of the process. Next, the extracted information is transformed into an actual process model, i.e., the concrete process model described by the text is *synthesized*. Some work labels approaches as *direct text-model transformations*, if they use deep neural networks [8]. We argue that this is not the case, even if these approaches treat the text-to-model as a machine translation problem, where an input in a natural language has to be *translated* into a formal language describing the model (e.g., XML for BPMN). Such approaches still have to extract process relevant information implicitly, which is then present latently, e.g., in *embeddings* as visualized in Figure 2.1.

### 2.2.1 Business Process Information Extraction

As stated in the introduction to this section, generating a formal business process model from natural language text is a two step procedure. This section describes the first step, which is detecting and classifying the information relevant to the business process. Figure 2.2 conceptually visualizes how process information is usually extracted from natural language.

The input to this step is a text in natural language that describes a process, which we will call a (process description) *document* from now on. In essence, extracting business process information is closely related to general information extraction, that is, we use an extraction

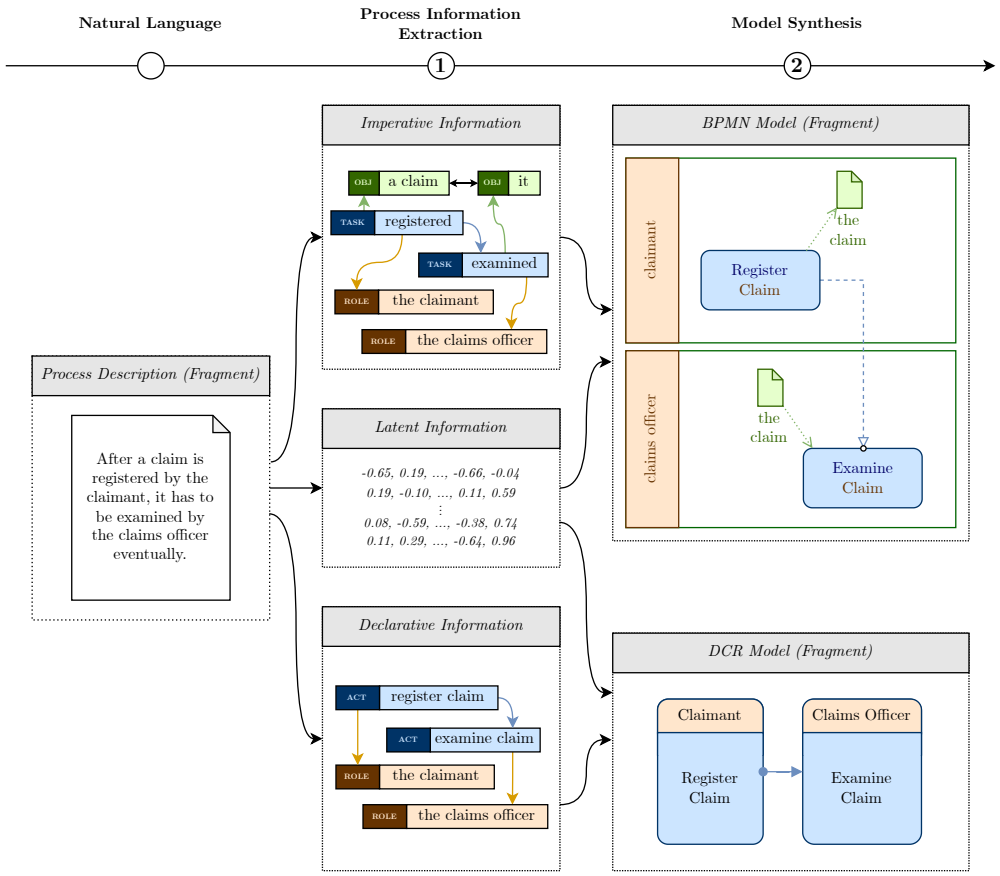



Figure 2.1: Visualization of the two steps in the text-to-model task. Information may be extracted implicitly and only be represented latently in the extraction model.

approach to find fragments of text in a document, which describe a concept we want to extract, or describe how these concepts relate to each other. To extract process relevant information from natural language text, the document is usually first *segmented* into sentences. Each sentence is then split into *tokens*, a procedure called *tokenization*. A token can usually be thought of as a single word, but some tokenization methods subdivide words further into *subword* tokens (e.g., *preprocessing* → *pre*, *process*, *ing*), or group words together if they belong to a known phrase (e.g. *New York*). A sequence of tokens is called a *span*. The surface form, i.e., the specific linguistic construct of a span may refer to a relevant concept, e.g., a process participant, which we then mark with a corresponding *type*. Such typed spans are called *entity mentions*, i.e., a single *mention* of a conceptual *entity*, which exists throughout the entire document, potentially even across documents. Detecting and extracting these entities is closely related to *named entity recognition*, a subfield of natural language processing [43]. It is the consensus that named entities refer to so called *rigid designators*, which are generic (e.g., proper names of persons or companies),

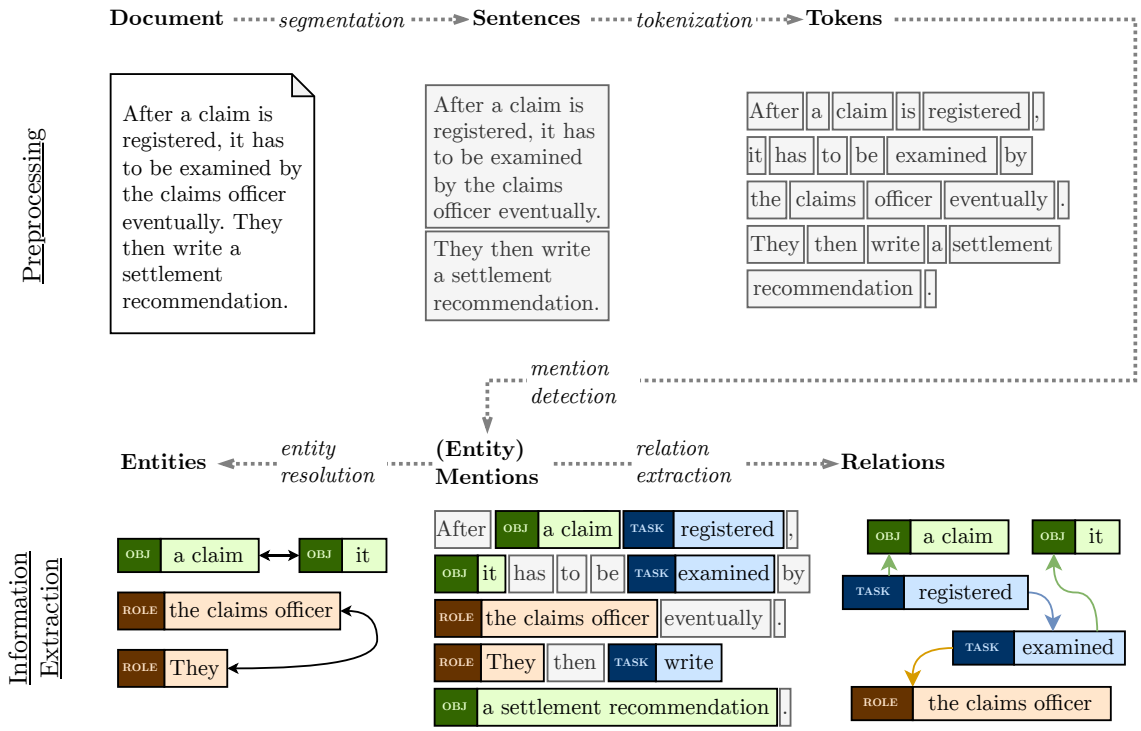



Figure 2.2: Overview of the process information extraction pipeline.

or natural (e.g., species or protein names) [43]. In process information extraction entities often refer to process relevant concepts, which can be rigid designators (think process participants), but also entities that are more akin to *events* [81], such as activities in the workflow. We therefore refrain from using the term *named* entity recognition, and instead use *entity mention detection*.

Resolving whether two given entity mentions refer to the same entity (concept), is called *entity resolution*. In process information extraction this step becomes especially relevant for the organizational perspective (process participants) and data perspective (data objects), where process elements are mentioned multiple times throughout the document, and are often referred to using pronouns or slight variations of the original surface form. Entity resolution is also needed for the other perspectives, for example to resolve multiple mentions of the same task. Yet, this is much less prevalent in the current state of the art and requires further research. If two entities interact with each other, they form a *relation*, e.g., to express that a task is performed by a specific process participant. Relations are usually directed, i.e., they have a source (*head*) entity argument and a target (*tail*) entity. Additionally, relations have a type describing the kind of interaction between their arguments. Relations contain important information about many perspectives of the process, such as the order of execution between activities, which is integral to the behavioral perspective.

### 2.2.2 Challenges in Business Process Information Extraction

While our current description of the process information extraction task characterizes it as a direct application of general information extraction, there are several characteristics of process descriptions, which set the two fields apart. The following list does not claim exhaustiveness, but instead shall serve as an illustration of the differences between the two fields, and the challenges this thesis addresses. Consider the following process description fragment for the examples of business process specific challenges below. "*The process begins when a new order comes in. The contract is finalized by the sales department. It is then sent to the development team, which compiles a list of requirements.*"

1. **Ambiguous Information**. Process descriptions often contain ambiguities that need knowledge about the world to resolve. *It* might refer the contract or the sales department, both interpretations are at least feasible — sending a team is common in specific contexts, e.g., in consulting or construction. Knowledge about the world can disambiguate the sentence, since *sales teams* are rarely sent somewhere, and therefore sending the contract is more likely to be the correct interpretation.

2. **Implicit Information**. Sometimes information is not stated explicitly, since mentioning one action implies a corresponding reaction, or knowledge about the word dictates certain behavior (see ambiguous information). For the human reader it is obvious that after the sales team sends the contract, the development team experiences the event of *receiving* it, even though the text only implies it. Process information extraction approaches need to either identify that some information is implied (and therefore missing), or have models of the world that allow automatic completion, such as modern large language models do.

3. **Process Meta Information**. Descriptions of business processes often involve statements about the execution of process *instances*. This includes, for example, information when a process instance starts or ends. Simple rule-based methods for extracting process information based on dependency parsing would incorrectly extract *the process* (object) and the task *begin* (predicate) from the first sentence. These systems usually have to compensate for this, e.g., by means of dictionaries with black listed actor names and tasks.

### 2.2.3 The PET Annotation Schema and Dataset

The PET dataset is the currently largest one, and serves as a benchmark for comparing different extraction approaches in the realm of process information extraction from natural language text process descriptions [8]. PET contains a total of 45 process descriptions in natural language, where process relevant information was *annotated* (marked) by three process modeling experts [9]. The corresponding annotation schema is focused mainly on imperative process modeling, specifically BPMN [9] and the defines seven entity types and six relation types. Figure 2.3 shows how a fragment of a process description document might be annotated with PET and the corresponding formal process model in BPMN.

Note, that Bellan et al. do not use the concept of entity mentions in their original publication of PET [9], and therefore call typed spans *entities*, instead of *entity mentions*. We will continue to use the term *(entity) mention* to refer to typed spans of text in the PET dataset.

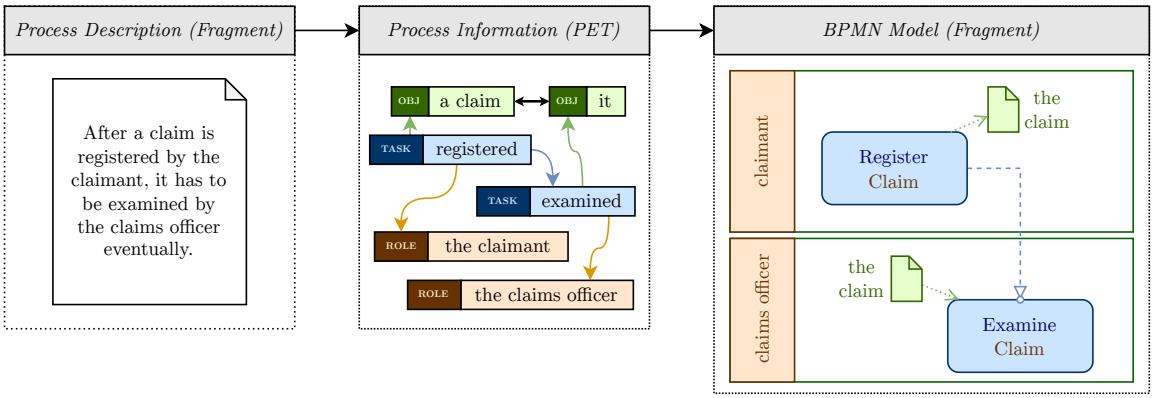



Figure 2.3: Document annotated with the PET annotation schema and corresponding formal process model in BPMN.

While an enumeration and explanation of all types of mentions and relations is out of scope for this section, it is important to note that mentions in the PET dataset directly map to modeling elements of BPMN. For example, *Actors* in PET can be mapped to swimlanes, or *Activity Data* mentions to data objects. Relations in PET define associations between elements, e.g., *Actor Performer* relations map *Actors* to the *Activities* they execute (perform), or *Uses* relations associate *Activity Data* with the *Activity* that uses said data object during execution.

The PET dataset is one of the most important datasets in current research on business process information extraction, as it combines and standardizes several datasets. Through PET extraction approaches become directly comparable. Yet, it also contains issues that need addressing in future research. First, while mentions of entities are annotated in the process description, coreferences between them are not given, i.e., the data used to resolve entity mentions to their corresponding entities is missing. We addressed this fact in one of our publications and provided a solution approach [53]. Next, the processes described in PET are only moderately complicated. Current research proposed datasets with far more complex underlying processes [31], albeit targeting declarative process modeling languages. Finally, the annotation schema of PET does not allow annotation of implicit process information, and is fairly rigid when it comes to overlapping (and non-continuous) token spans containing process information.

### 2.2.4 The ATDP Annotation Schema and Related Datasets

The *Annotated Textual Description of Processes* (ATDP) language has first been proposed by Sànchez et al. [65] to address interpretational ambiguity in natural language process descriptions. Figure 2.4 shows a fragment of a process description document annotated with the ATDP annotation schema (excluding scopes) and the corresponding process model in DCR.

They use the notion of *scopes*, into which certain parts of the textual description fall. Scopes are recursive, i.e, a single scope may contain any number of (smaller) child scopes. Non-leaf scopes are typed, which defines the control flow between child scopes, e.g., sequential ordering, exclusivity, or iteration. Leaf scopes (i.e., scopes with no child scopes) contain at least one *fragment* (entity), which are typed with one of the three fragment types *Activity* (workflow step),

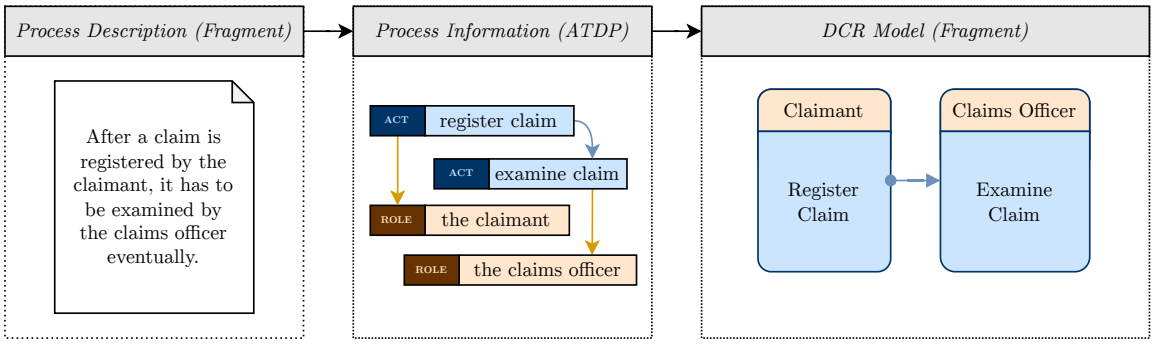



Figure 2.4: Document annotated with ATDP as used by Quishpi et al. [62] and corresponding formal process model in DCR.

*Role* (process participant), or *Business Object* (manipulated data). Fragments are connected with relations, which create associations between them, for example *Agent*, defining the role performing an activity, as well as control flow relations, such as *Response*, defining that executing one activity triggers the eventual execution of the other activity.

Annotating a textual process description with ATDP thus structures information in a tree, similar to a process tree, and fixes it to a single interpretation, that makes ambiguities explicit. Multiple interpretations allow for the generation of more than one realizations of the process model, of which a domain expert may then choose the most appropriate one.

To the best of our knowledge, Quishpi et al. [62] published the only use of ATDP in a concrete dataset for process information extraction, albeit in an incomplete version not considering the extraction of scopes. The dataset uses all three entity types, and an additional constraint, for a total of eight constraint (relation) types. Since this approach does not consider scopes, the interpretation mechanism explained earlier is not utilized, which makes the implementation limited in usefulness. Still, the approach of Quishpi et al. towards mention and relation extraction using pattern matching on dependency trees is powerful and fairly robust, even on new datasets [3]. Additionally, the ATDP dataset is useful for developing new process information extraction approaches, serving as fundamentally different extraction target compared to the PET dataset. We used the ATDP dataset for exactly this purpose in our work [3, 52].

### 2.2.5 Model Synthesis from Process Information

Generating a model from explicitly extracted process information is usually done with a deterministic layouting algorithm. We will sketch an example for an algorithm used in some of our papers here, but note that researching model synthesis is out of scope for this thesis, and requires further research (see Chapter 13).

Synthesizing a BPMN model from process information in the PET data annotation schema works in three major stages. First, in the ***Consolidation*** phase, we assign conditions to the mentions of their respective decision points (XOR gateways). Next, all mentions of gateways are merged, if they refer to the same decision point in the process. Finally, for all activities that are not assigned an executing actor, we find the closest actor mention in the text left of it. In

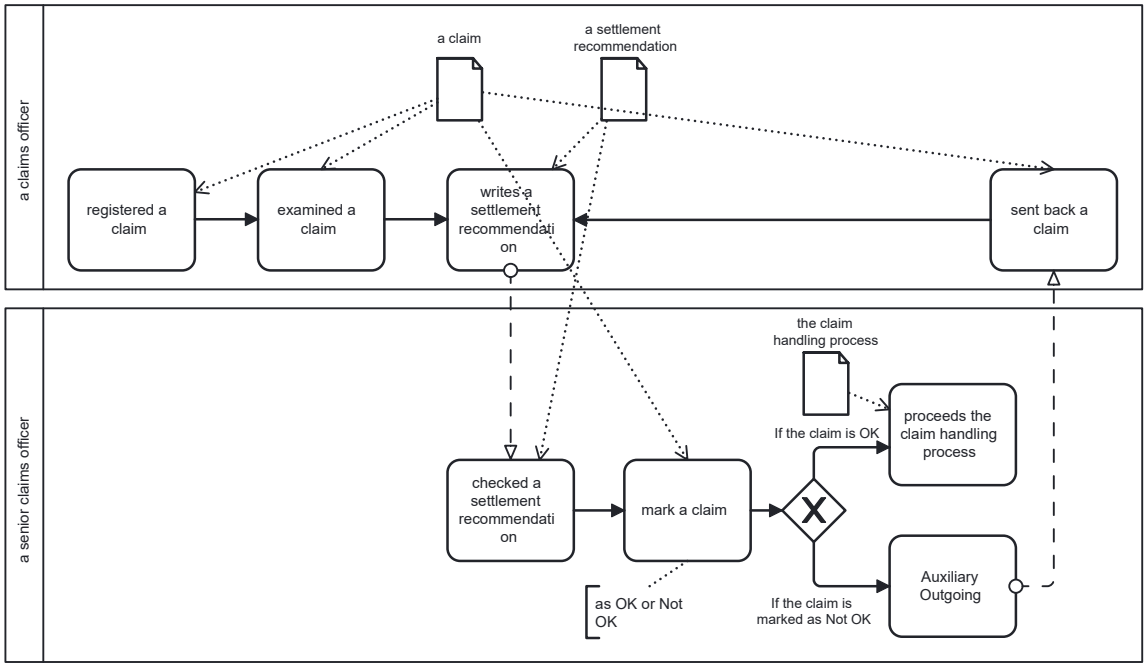



Figure 2.5: Result of a simple deterministic model synthesis algorithm.

the second stage, the ***Vertex*** stage, we create process elements for all mentions, e.g., *Tasks*, *Data Objects*, *Swimlanes* (actors), etc. The final ***Linking*** stage connects related elements, e.g., successive tasks with *Sequence Flows*, if they are located in the same Swimlane, or *Message Flows* otherwise. We also create *Data Associations* between data objects and tasks, adding the label of the data object to the label of the task, to generate labels like "*register the claim*" instead of just "*register*". Sometimes, we extracted control flow relations between a gateway in one lane and a task in another lane. In such a case we also have to insert *auxiliary* tasks, since message flows are only allowed between Tasks in BPMN. The resulting model is then interpreted as a directed graph, which we traverse and draw into a graphical diagram, see Figure 2.5.

## 2.3 Artificial Intelligence

For a human it might appear easy to read a description of a business process and draw a business process model that reflects the information contained in it. Yet, to formalize the skills needed to do so, encode them so a machine may use them is non-trivial. Doing so would make this machine a *weak artificial intelligence* [67]. Despite how the term is used colloquially in recent years, artificial intelligence fundamentally describes systems that mimic the human ability to make informed decisions [80]. While strong artificial intelligence describes a computer system that is able to actually *understand*, weak artificial intelligence merely simulates this understanding [67].

In this section we will discuss the fundamental approaches to simulate human decision making and understanding of problems. First, Section 2.3.1 describes systems that directly use human knowledge encoded in rules. Section 2.3.2 explains, how machines can derive these (or similar)

rules directly from data, a paradigm called machine learning. Finally, we will briefly discuss artificial neural networks and their use as statistical models in machine learning in Section 2.3.3.

### 2.3.1 Rule-based Systems

Traditionally, simulating human decision making was (and is) achieved through *expert systems*, which come to solutions to complex problems by applying expert knowledge. This knowledge has to be represented in such a way that computers can reason over it, commonly in *If-Then* constructs [80]. Rule-based systems require interpretable features as inputs, i.e., information needs to be processed first, before it can be used in a rule-based system. In the context of process information extraction, this requires, among others, transforming a string input (text) into tokens, tagging those tokens with part-of-speech tags, parsing dependencies, or calculating the distance between tokens. Only then rules can be applied, for example, "If two actions are consecutive, then there is a control flow connection between them." (paraphrased from the annotation guidelines of PET [8]), or "Any verb that is the first dependency of the word 'whether', is an action." (paraphrased dependency tree pattern from [62]).

### 2.3.2 Machine Learning

In contrast to rule based systems, where a human engineer defines rules that derives one or more outputs from a number of inputs, machine learning aims to derive these rules automatically. To this end a model has to be configured in such a way that a given metric is optimized on a given dataset. What metric is used, is dictated by the method, for example, in *linear regression* a line is fitted to a set of points, in such a way that the mean squared error between line and data is minimal. If the calculation of said metric relies on a known true output, i.e., it is *labeled*, we call the corresponding machine learning method *supervised*. *Unsupervised* methods rely on metrics that solely use observable inputs, e.g., for k-means clustering, where a set of $n$ clusters is formed, such that the average inner-cluster distance is minimal [48]. Using this metric, we only need to know about the position of a data point in the feature space to automatically assign it to a cluster of similar points, which then can be interpreted by a human.

Methods using datasets where only a portion is labeled and the remainder unlabeled, are called *semi-supervised*. The fundamental idea behind semi-supervised learning is to learn a representation of inputs $x$ from unlabeled and labeled data, and use labeled data to learn a mapping from that representation to outputs, which follows from the assumption, that if two inputs $x$ are close, so should the corresponding outputs $y$ [12]. Finally, *reinforcement* learning methods optimize a *policy* of when to take specific actions given environmental inputs, so that the total *reward* for a sequence of actions is maximal, given a reward function. Unlike the previous classes of machine learning, reinforcement learning uses simulation, i.e., dynamic data, instead of a fixed dataset for learning.

### 2.3.3 Neural Networks

Artificial neural networks are inspired by biological neural networks in their fundamental component, the neuron [30]. For the remainder of this thesis, we will use the terms neural network and neuron to describe the artificial variant, and not biological one.

Artificial neurons, like their biological counterparts, form *weighted connections* with each other. These connections are directed, meaning the output of one artificial neuron serves as the input of others. According to the definition by Goodfellow et al. [30], modern artificial neural networks organize artificial neurons in *layers* with a shared *activation function*. Such an activation function is a vector-valued function that maps the weighted layer inputs to an *activation* (output). Stacking multiple layers can then be understood as connecting them in a function chain [30]. A neural network composed of stacked layers is called *feed forwards*, if it does not contain cycles. If there are connections between layers, such that the output of a layer eventually becomes its input again, we call such a neural network *recurrent* [30].

By forward propagation of an input vector through the network, i.e. evaluating the corresponding chain of functions, we eventually obtain an output vector. This output can then be interpreted as the prediction of the neural network, which can be used to calculate a prediction error. The exact way of calculating this error depends on the dataset as described in the previous paragraph. After obtaining the error, the contributions of each weight are calculated in a process called *back propagation* [30]. Weights are then adjusted according to their individual contributions, usually with *gradient descent* methods, to minimize the error [30]. By repeating forward propagation, backward propagation, and corresponding weight adjustments, the neural network converges to an optimal configuration — it *learned* to model the given dataset.

Generally, neural networks with more parameters can describe more complex data. The theorem of *general universality* of neural networks states, that a neural network with at least one hidden layer, and enough neurons in this layer can describe any arbitrary function [15]. In practice, the depth of neural networks has become the determining factor of model expressiveness, giving rise to the era of deep learning. With increases in number of parameters and layers comes also an increase in required training data. This leads to correspondingly large training data sets, sometimes containing hundreds of thousands, millions, or even billions of training examples. Collecting such datasets has therefore become increasingly infeasible for *every* learning task, giving rise to the idea of training models on generally applicable knowledge, and then adjusting them for specific problems.

## 2.4 Large Pretrained Models in Natural Language Processing

The following Section 2.4.1 will expand on the idea of pretraining large models with huge datasets. Section 2.4.2 will briefly discuss the model architecture enabling pretraining on this large scale. Finally, Section 2.4.3 explains how modern generative language models are used, even without fine tuning them on specific tasks.

### 2.4.1 Pretraining and Fine Tuning Models

The concept of pretraining was made popular by the use of neural networks in language modeling. Here, large collections of natural language documents are used in training models for predicting *masked* tokens [20], or the continuation of sentences [63]. During this training, the model learns many rules about natural language, such as grammar, as this is needed for optimal prediction performance. Such a neural network is then called a *(large) language model* (LLM) and can be used for other tasks, by minor adjustments and additional training on a much smaller dataset.

This step is called *fine tuning*, and reduces the data needed to train useful models, since many of the concepts learned during pretraining can still be applied in this new task.

While pretrained models are very common in language modeling, other tasks make use of pretraining as well. This holds generally for tasks that require the extraction and subsequent composition of low-level abstract features into high-level rules. A prominent example of such a task is computer vision, where edges, shapes, and textures are features of increasing complexity, which might be extracted by a pretrained model, and then combined in rules during a fine tuning step on application-specific data [13].

Pretrained models are generally very large, which makes them computationally expensive to train and use. For this reason pretrained models became only popular in recent years through innovations in computation-efficient model architectures. Most notably, the *transformer* architecture allows more parallel computations during training, enabling the training of massive neural networks on very large datasets [77].

### 2.4.2 Transformers

The core concept of a transformer are stacks of *attention* layers [77]. These attention layers are trained to assign weights (attention) to elements of an input sequence based on their importance for answering a given query [77]. Through this mechanism transformer architectures are able to consider the entire input sequence at once, contrary to previous architectures, which had to process the input element by element. This enables better parallelization during training the model, compared to the sequential computation needed in previous architectures [77].

Transformers originally use two separate embeddings for encoding inputs and decoding the corresponding outputs, which is useful for many tasks, where input sequences significantly differ from expected output sequences, such as machine translation. Today, transformers are often either encoder-only, or decoder-only, depending on the task they are trained on. Encoder-only architectures like BERT [20] use training objectives, where the target can benefit from bi-directional encoding of the input, such as predicting a missing (masked) token in a given sentence. Since for these tasks the input and output sequences are part of the same distribution (e.g., language), encoding them both using the same embedding is permissible, and thus, more efficient.

On the other hand, decoder-only architectures are autoregressive (non bi-directional), meaning any element of their output sequence depends only on previous elements, never on future ones. They are usually trained on a next element prediction task, i.e., given an incomplete sequence the model has to predict the most likely next element in that sequence. Generative pretrained transformers, such as GPT2 [64], are pretrained using large collections of natural language texts, and show impressive performance when applied to previously unseen tasks. Modern large language models are generative pretrained transformers with many billion trainable parameters.

### 2.4.3 Prompting

The way such generative pretrained large language models are used today, differs fundamentally from traditional deep learning, where one would design a neural network architecture and train it with a large dataset, or even use a pretrained model as basis for a custom architecture that is

then fine tuned. Instead, models like GPT3.5 [10] are now used as so-called *few-shot learners* through prompting them in natural language. Few-shot learners are machine learning models able to learn a new task or class with a small number of examples, while *zero-shot learners* can do so without any training data. If large language models are given example turns of natural language input (prompt) and expected natural language output, they are able to answer a final prompt analogously to the examples given before, making them few-shot learners[2]. Since the training data is given in the *context* of the input, i.e., before stating the actual problem, this method of prompting is also called *in-context learning* [10].

## 2.5 Data Augmentation

One of the options of mitigating the lack of data in process information extraction is using the existing data more efficiently. This can either be achieved by smaller models, or by inflating the datasets procedurally. Data augmentation represents a means of doing just that, by either perturbing examples in the dataset, such that their label is preserved, or by oversampling examples [68]. Literature defines data augmentation as methods, that change existing examples, without inflating the dataset, while oversampling adds synthetic examples to the dataset. In practice, the two terms often intersect, e.g., when applying data augmentation to synthetic data. In this thesis we will use the term data augmentation to denote techniques that create new, synthetic training examples on the basis of existing data.

The intuition behind data augmentation can best be understood visually, that is, for image classification data, as Figure 2.6 shows. There, targeted changes (*perturbations*) are performed on the image of hand-written "*9*". This image is part of the *MNIST database of handwritten digits* [41], which contains 16x16 pixel pictures of digits written by humans. As such, it can be used for an image classification task — each picture needs to be mapped to the digit it contains. The style with which humans write digits greatly varies from person to person, some might angle them more, some might embellish them with serifs, some might keep them extremely simplistic. Furthermore, the conditions under which a picture is taken can vary, such as lighting, the pen used for writing, or the camera used for taking the picture. All these factors are not guaranteed to be contained in the training data, and thus influence the performance of a model during real-world application. Data augmentation aims to simulate these variations procedurally, changing original data examples via controlled perturbations.

Figure 2.6 shows the original picture of a handwritten "9", and three types of such perturbations — rotation, translation, and the addition of noise. The second row shows the same perturbations, but with much higher intensity. Here, the label of the picture is invalidated, i.e., the picture no longer shows a "9", but a "6", "0", or just noise. This highlights the importance of properly controlling data augmentation techniques. We will discuss this in more detail in Chapters 8 and 12.

[2]Note that many modern large language models are able to function in zero-shot settings as well.

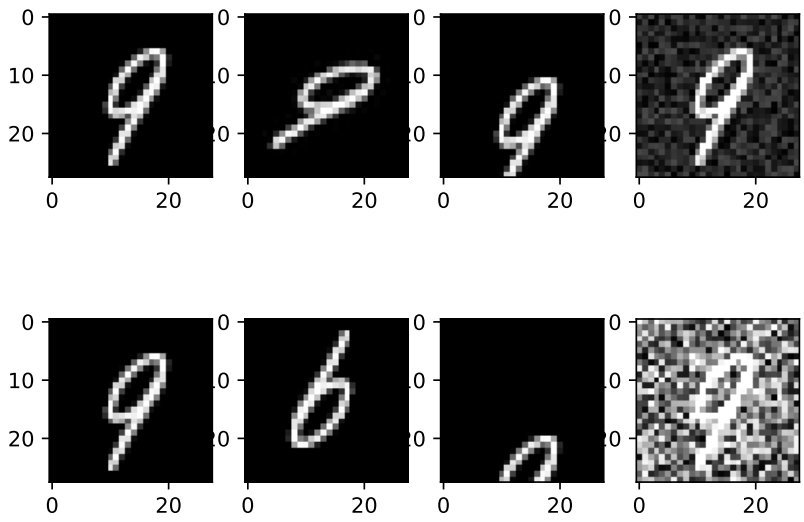



Figure 2.6: Examples of perturbations on an image from the MNIST digits dataset, containing the digit "9". The first row contains the original and three perturbations that preserve the label, while the second row contains the original and three perturbations that invalidate the label.

# Chapter 3

# Related Work

The goal of this section is to give the reader context of previous and parallel work in the fields of research this thesis contributes to. We will first give an overview of approaches towards process model generation from natural language text in Section 3.1. Next, we will discuss general methods of data augmentation, as well as specifically for natural language processing and business process management in Section 3.2. Finally, Section 3.3 presents datasets that are useful for developing new process information extraction methods.

## 3.1 Automatic Model Generation

Related work on automatic process model generation can be organized into four distinct categories, based on the main approach towards extracting the information contained in a textual description, and thus, defining an automatically generated process model.

**Constrained Language Methods.** Ivanchikj et al. [35] present SketchMiner, a tool for describing the process in constrained language, from which BPMN diagrams are extracted automatically. Caporale [11] proposes a controlled language based on sentence templates and corresponding parser to describe business processes and generate process models in BPMN. Methods that require constrained language present a tradeoff between the complexity of a precise formal modeling notation and the convenience of imprecise natural language descriptions. No knowledge of the formal modeling language is needed to produce a formal model, instead users only have to fill in *templates* of descriptions in constrained language. While such an approach makes parsing of the information contained in descriptions trivial, it best serves as a tool for supporting the communication between process owners and process analysts [11]. For this reason methods using constrained language are not well suited for exploiting *existing* natural language process descriptions.

**Rule-Based Methods.** Related work in this category usually showcases striking performance during evaluation on the dataset they are designed for, extracting process information with high levels of precision and recall. Yet, all methods share a common inflexibility, i.e., the rules they define are specific to a particular subset of process perspectives and languages. Changing and extending them requires expert knowledge in both natural language processing, as well as the target business process modeling language. This is mainly founded in the sophisticated

use of natural language processing for feature engineering and subsequent, manually defined rules. Friedrich et al. [29] use part-of-speech tags, the dependency graph of sentences, as well as word information from a lexical database to extract process relevant information and generate a process model in BPMN. Sokolov et al. [71] suggest integrating semantic unification into process model extraction pipelines could improve extraction quality, when using partial descriptions or external knowledge sources, but do not report empiric results. Lopez et al. [46] focus on extracting declarative process models in the DCR formalism from natural language, using a tool called the *Process Highlighter*. Van der Aa et al. [1] present the currently leading method for extracting declarative process models from text, using the results of syntactic parsing and word-level features in a set of manually defined rules. Sanchez et al. [66] and Quishpi et al. [62] employ process information extraction techniques based on regular expressions for syntactic dependency trees and part-of-speech tags. Sintoris and Vergidis [69] propose a transformation pipeline for the automatic generation of BPMN process models, albeit without reporting its application or evaluation. Sonbol et. al [72] treat the generation of a formal process model from natural language text descriptions as a machine translation problem, and present a natural language processing pipeline to extract process relevant information and layout it in BPMN.

**Supervised Machine Learning Methods.** Unlike rule-based methods, supervised machine learning methods automatically *learn*, that is, derive what parts of a process description are relevant for generating process models. This is usually done by adjusting parameters of a machine learning model, until its predictions, i.e., the extracted process relevant information closely resembles the expectation found in a so-called training dataset. For this reason, machine learning methods can potentially be transferred to new process domains and languages, provided they are retrained with appropriate data. In [61], Qian et al. propose a neural method for entity and relation classification, which assumes relevant text fragments to already be extracted in a preprocessing step.

**Methods Based on Pre-Trained Models.** The limiting factor of applying supervised machine learning methods to process model extraction, is the lack of readily available training data. Pre-training machine learning models on out-of-domain data, i.e., data not immediately related to the target task has become more popular with the advent of so-called large language models, such as BERT [20]. This reduces the amount of training data needed to obtain a useful extraction model drastically, as the pretrained model only needs fine tuning. Recent generative approaches further reduce the amount of training data, often to less than three examples, sometimes to none. Thus, methods based on generative large language models reduce the problem to text completion, which is often referred to as *prompting*. In [44], Licardo et al. define an extraction pipeline, which revises process descriptions, extracts process relevant entities and activity execution order using a transformer fine-tuned on additional training data, and finally creates a BPMN diagram. Bellan et al. [7] use in-context learning with GPT3 to extract process relevant facts from natural language process descriptions. In [40], Kourani et al. present an approach for extracting activities and their execution order from a process description using a pretrained large language model and human review. In [16], Daclin et al. present an early-stage concept for using pretrained large language models and structured data to transform speech to text and subsequently extract a formal process model in BPMN, qualitatively evaluated on just a small number of easy examples. Klievtsova et al. [38] present six prompts for extracting graphical process models from natural language texts, concluding that modern large language models are

ready to be applied in real-world text-to-model scenarios. A empirical study of the application of different pretrained large language models on multiple datasets did not exist before this thesis, instead previous work often just studied feasibility.

## 3.2 Data Augmentation

Data augmentation is a well explored technique for making machine learning models more robust and use training data more efficiently. For this reason it is a possibility to mitigate the lack of training data present in process information extraction. For this reason we will first discuss prominent examples of data augmentation applications in research fields, then review how data augmentation is applied in natural language processing, and finally how it is applied in business process management tasks.

**Examples of Data Augmentation.** Data augmentation has long been a technique to complement incomplete or partial data in many classical algorithms, such as expectation maximization [18], or the calculation of posterior distributions [74]. Here data augmentation is understood as a ways of altering numerical data, so that algorithms converge to more suitable results or lead to more robust parametrization of stochastic models. The original idea for data augmentation on images is often attributed to Baird [5] and his work on modeling defects in images (scans) of physical documents, resulting in synthetic data useful in developing robust models for document image analysis. Wang et al. [79] published one of the earliest systematic reports of data augmentation as a technique to improve deep neural networks for image recognition, and propose it as an alternative to other regularization approaches, such as dropout [73].

**Natural Language Processing.** The *NL-Augmenter* framework [21] is a shared research effort that collects a total of 117 augmentation techniques to date [3]. These augmentations are targeted specifically at natural language processing tasks, including, but not limited to, named entity recognition, machine translation, question answering, or sentiment analysis. Li et al. [42] collect and categorize 16 data augmentation techniques, but focus on techniques applicable to text classification, text generation, and structured prediction, all of which are not directly applicable to process information extraction. Similarly, Pellicer et al. [57] survey data augmentation techniques for datasets used in sentence classification and emotion detection, which are not immediately useful in information extraction tasks. Wang et al. [78] propose a data augmentation method based on reinforcement learning, using four simple perturbations and reporting impressive improvements in entity relation prediction for medical datasets.

**Business Process Management.** In [36] Käppel et al. evaluate nine simple data augmentation techniques (e.g., random deletion) for predictive process monitoring, i.e., predicting how a business process evolves over time. Huo et al. [34] propose data augmentation based on large language models for intent recognition in robotic process automation, more specifically for automating repetitive tasks through chatbots.

[3] See the NLAugmenter repository in GitHub for an up to date list: `https://github.com/GEM-benchmark/NL-Augmenter/tree/main/nlaugmenter/transformations`, last accessed Feb. 7th, 2025.

## 3.3 Dataset Initiatives and Support

López et al. [45] collect, annotate, and publish 37 descriptions of declarative processes, focusing on the extraction of process roles, activities, and three types of constraints (relations) between them. In [9], Bellan et al. publish the PET dataset, which consists of 45 natural language process descriptions, collected from previous dataset initiatives, and annotated with extraction targets for process elements and their relations. Bellan et al. present a tool for visualizing the annotations of process descriptions in [6], supporting future dataset collection efforts, if they use the PET annotation schema. Caporale [11] proposes a tool for the collaborative collection of process models, with a mechanism for recommending similar process models. The *Model Judge* [17] is a tool developed by Delicado et al., aimed at novices in using BPMN for business process modeling, supporting them with recommendations of potential process elements in a textual description, and aligning existing process elements with their textual counterparts.

# Chapter 4

# Overview of Relevant Publications

In Chapter 1 we discussed how the application of machine learning for process information extraction requires amounts of training data currently not available. At the start of this thesis this fact was less obvious, and became only clear after working on the first machine learning based process information extraction approach [3]. After this publication it became apparent that investigating ways of mitigating this issue are needed, which motivated all subsequent publications.

In this chapter we will provide an overview over the publications relevant for this thesis. We will highlight key ideas, contributions, and limitations of these publications in chronological order, to illustrate the systematic exploration of our initial, fundamental research question RQ-0.

> **RQ-0** *How can deep learning be applied to process information extraction from natural language process descriptions?*

Motivated by the academic success of rule-based process information extractors, we looked for reasons why these systems find little application in practice, with only one viable commercial system available at the time[4]. One key contributing factor is the inflexibility of rule-based systems during application in new domains, which also explains the success of machine learning in other fields of research. Proper machine learning approaches could be transferred to new domains by retraining them on suitable datasets.
RQ-0 represents our initial research motivation at the beginning of this thesis. Additional questions arose while investigating it. Figure 4.1 visualizes the timeline of publications made for this thesis, along with the research questions these publications raise and investigate.

We published an initial, direct approach towards process information extraction in "*Data-driven annotation of textual process descriptions based on formal meaning representations*" [3]. At the time, this work represented the first machine learning approach towards identifying mentions of process information in *complete* sentences of textual process descriptions. Other work is not able to distinguish between sentences that contain or do not contain process information [61]. During work on this approach, we found that training a traditional sequence-to-sequence model, i.e., a sequence of tokens as input and a sequence of entity mentions as output, is not feasible due

[4]DCR solutions offer a commercial system for semi-automated extraction of declarative process models from natural language text, see `https://dcrsolutions.net/`, last accessed Feb. 7th, 2025.

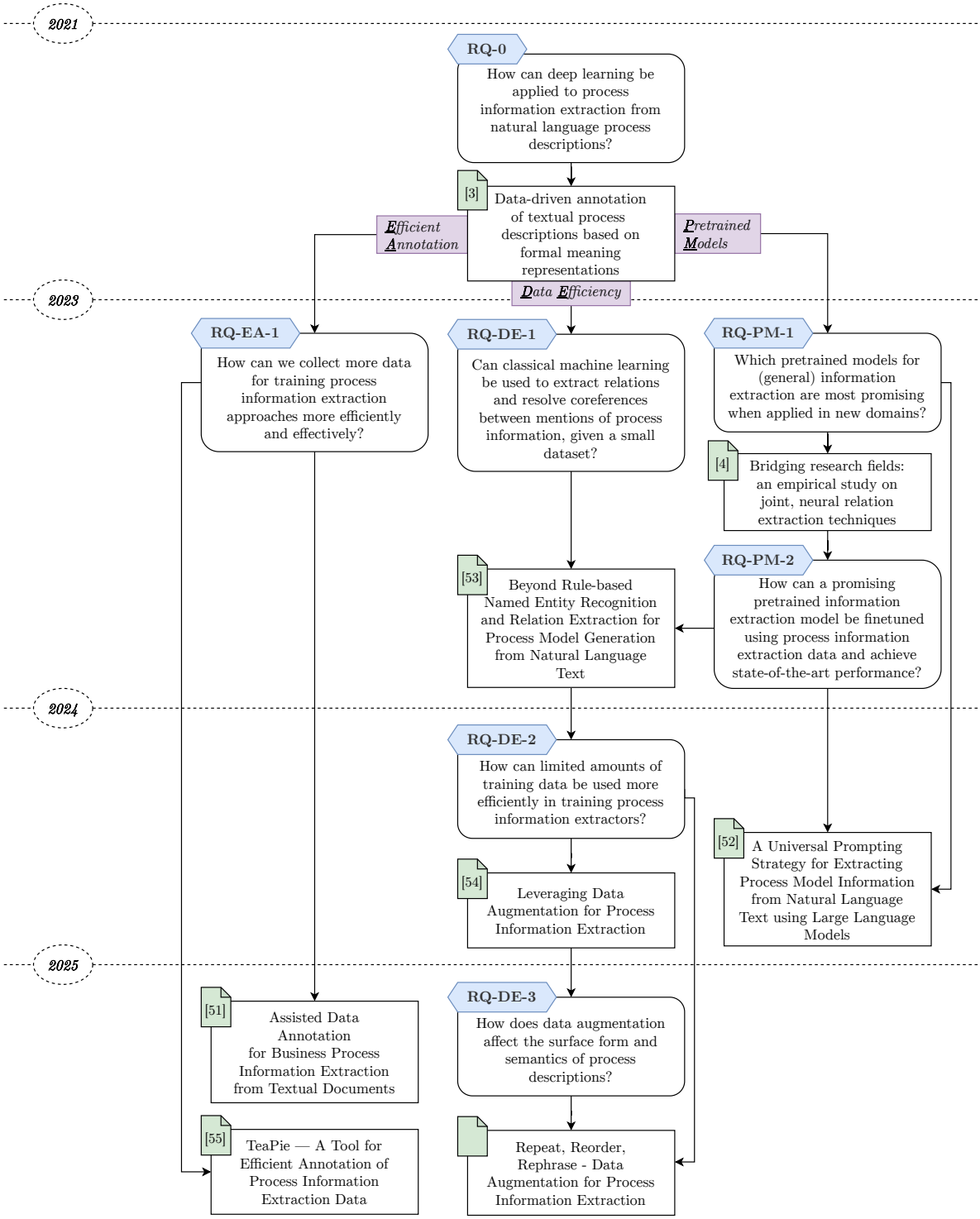



Figure 4.1: Timeline of publication for this thesis, along with the leading research questions brought up and answered by publications.

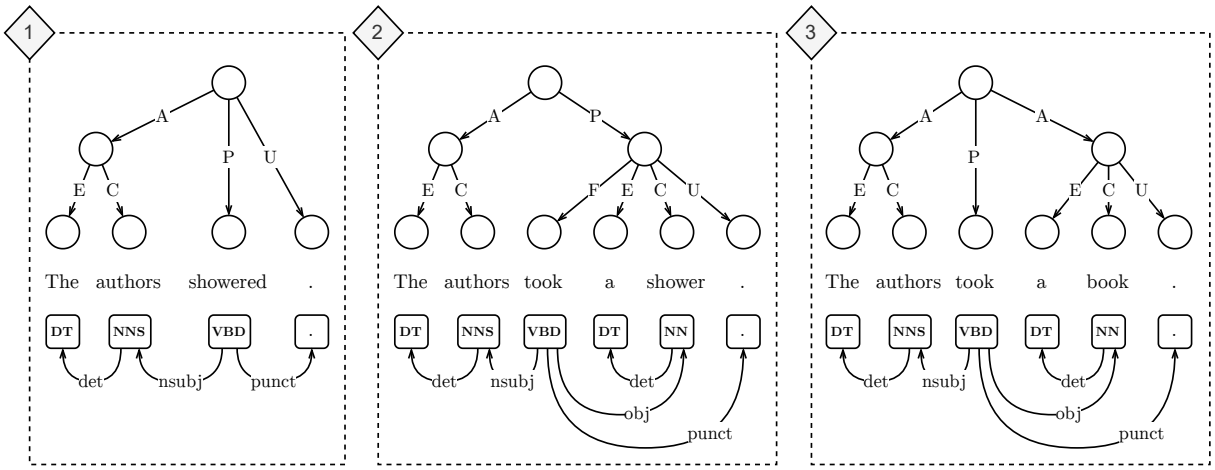



Figure 4.2: Comparing formal meaning representations to syntactic dependencies. We use Penn Treebank part-of-speech tags [49], see `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`.

to the lack of data, and complexity of the problem [3]. Instead, we made use of out-of-domain knowledge in the form of *formal meaning representations*, which are graphs representing the semantics of a sentence. Figure 4.2 shows the syntactic dependency graph below, and the formal meaning representation above the sentences we used in this paper: (1) "*The authors showered.*", (2) "*The authors took a shower.*", and (3) "*The authors took a book.*" [3]. Unlike syntactic dependency graphs, formal meaning graphs are largely invariant to changes in the *surface form*, i.e., what words or grammatical constructs we use to represent what we mean. Notice, how (1) and (2) have the same *meaning*, but very different surface form, while (2) and (3) have very different meaning, but very similar surface form. The formal meaning representation is able to capture this fact, as in both cases there are nodes describing an actor (**A**) and a process (**P**), i.e., something that happens over time — *taking a shower*, i.e., *showering*. In contrast, when comparing sentences (2) and (3), we now see two actors, and a very different action — *taking something*. Traditional syntactic dependency trees are not able to capture the *meaning* properly, in fact, the dependency trees for sentences (2) and (3) are identical[5]. In both cases we identify *took* as the verb (think action) that the subject *authors* perform on the object *shower* and *book*, respectively. In a process context, this would lead to a model, where we expect the authors to physically "*take*" the shower (like they would a book). At the same time, sentence (1) and (2) result in very different dependency trees, even though they are the same *semantically*. A meaning representation parser is a system that creates the formal meaning representation of a sentence and is developed using texts from outside the business process domain. In this sense, the use of such a parser and the resulting formal meaning representation shifts some of the complexity of process information extraction — we do not need to learn to interpret the meaning of a sentence, only which parts are relevant to a process.

To extract process information from a sentence, we first transform this sentence to its formal

[5]We used CoreNLP to parse the sentences, see `https://corenlp.run/`, last accessed Feb. 2, 2025.

meaning representation (graph) and enrich nodes corresponding to tokens of the sentence with information about these tokens, e.g., part-of-speech tags or word vectors. The graph is then processed using a graph neural network tasked to assign nodes semantic roles in the process (e.g., actors, activities, data objects). Our approach outperforms a state-of-the-art data-driven approach [61], and achieved competitive results compared to a rule-based system [62] when using the dataset it was designed for, and out-performing it on new, unseen data, which we collected. Working on this publication raised several new research questions following the original question RQ-0. At this point it became clear that the direct application of deep learning is prevented by a lack of useful training data, and as a result these new research questions already reflect the three approaches towards mitigating the lack of data in process information extraction we discussed in Chapter 1 and have shown in Figure 1.2. These are ***E**fficient **A**nnotation* of training data (Option **1**, **EA**), higher ***D**ata **E**fficiency* (Option **2**, **DE**), and the use of ***P**retrained **M**odels* (Option **3**, **PM**). Based on these three options three corresponding research threads arise.

**RQ-EA-1** *How can we collect more data for training process information extraction approaches more efficiently and effectively?*

In our initial publication, it took several weeks to collect and annotate 250 sentences, even for a team of many experts in the field of business process management. Labels often contradict each other, most notably the determiners of actors are not always part of the annotation, e.g., *the authors* vs. just *authors*. Researching proper tool support for data collection would expedite it, as well as potentially open data annotation to non-experts.

**RQ-DE-1** *Can classical machine learning be used to extract relations and resolve coreferences between mentions of process information given a small dataset?*

Previous work on machine learning in process information extraction, as well as our proposed approach is not able to extract interactions, dependencies, and coreferences between entity mentions from multi-sentence process descriptions, but only from fragments [61]. Machine learning approaches towards relation extraction and coreference resolution would result in a fully machine learning based pipeline for process information extraction.

**RQ-PM-1** *Which pretrained models for (general) information extraction are most promising when applied in new domains?*

A common remark of reviewers of our publication was to explore how existing approaches based on pretrained language models perform on the process information extraction task. No empirical study of such models for different datasets existed at the time, but would be an invaluable resource for an informed decision when choosing a model for extracting process information.

These three research questions directly correspond to the three options of mitigating the lack of data in process information extraction research. Contrary to what their order of mention might suggest, we did not investigate them starting with RQ-EA-1, but instead with RQ-PM-1. Learning from our initial publication, we realized that collecting large datasets, even if done efficiently, is a significant undertaking. Instead, investigating the application of pretrained models, or efficient use of existing data promises similar benefits with significantly lower resource requirements.

The paper "*Bridging research fields: an empirical study on joint, neural relation extraction techniques*" [4] addressed the lack of an empirical study of existing, pretrained models for (general, not only process) information extraction. As such it is directly needed to answer RQ-PM-1, which entails an informed decision regarding promising pretrained models for information extraction.

We conducted a systematic literature review, following the PRISMA method [56], finding a total of 1,847 publications, which were manually filtered by reading title and abstract, applying seven exclusion criteria. This resulted in 189 publications, where we read the full text and applied the same seven exclusion criteria, yielding a total of 31 relevant approaches, of which we were able to retrieve and properly set up seven. These approaches were then trained and evaluated on four different datasets, using three variations of the $F_1$ measure. We found four promising approaches, including MARE [39] and SpERT [24], which are able to produce consistent, high-quality predictions of entities and relations. Compared to other approaches, these two approaches were much less affected by adverse data characteristics, such as, the number of relations in a document, the distance between relations, or higher variation in surface forms of entity mention [4]. The proper application of models identified in our work, can be phrased as the following new research question.

**RQ-PM-2** *Can pretrained models achieve state-of-the-art performance in the process information extraction task?*

All the datasets we used in this publication were from information extraction tasks unrelated to business process descriptions. It therefore remains to test the most promising model on a dataset from process information extraction.

We continued work on both RQ-PM-2, as well as RQ-DE-1 in the paper "*Beyond Rule-based Named Entity Recognition and Relation Extraction for Process Model Generation from Natural Language Text*" [53]. Our main contribution in this publication was a fully machine-learning based pipeline for extracting process information from natural language process descriptions. These natural language process descriptions were published by Bellan et al. [8] in the PET dataset. However, they were still missing information about coreferences between mentions of process actors and business objects, which we manually annotated and made publicly accessible. We applied the entity mention extraction technique based on conditional random fields by Bellan et al. [8] without change, and proposed an entity resolution approach based on a pretrained coreference resolution model, as well as a relation extraction approach based on gradient boosting [60]. This pipeline was able to outperform the previously best rule-based approach, and since it was fully based on machine learning, it constitutes a answer to RQ-DE-1. The machine learning methods we used are "classical" approaches and not deep learning approaches, which is why this paper can not be considered an answer to RQ-0.

Additionally, we compared our approach to JEREX [25], an improvement of SpERT, which we identified as promising in our previous study. We could show that even though PET was the currently largest available dataset for process information extraction, it was still too small to be used for fine tuning pretrained models. Therefore, fine tuning pretrained models appeared not feasible, leading us to answer RQ-PM-2 negatively at the time.

Instead, work on a machine learning pipeline lead us to a research question, which corresponds to the second way of mitigating lack of training data in process information extraction from Chapter 1.

**RQ-DE-2** *How can limited amounts of training data be used more efficiently in training process information extractors?*

Even with the publication of the PET dataset, extracting process information from natural language process descriptions remained a *low data* task. It has been shown that similar low data tasks of other domains benefit from data augmentation. Testing data augmentation for process information extraction would bring additional challenges, e.g., the reliance of process descriptions on keywords (*after*, *before*) to describe the control flow of a process.

In our next publication "*Leveraging Data Augmentation for Process Information Extraction*" [54] we applied a total of 19 data augmentation techniques to the PET dataset and used the resulting data to train our process information extraction pipeline from [53]. These data augmentation techniques were selected from related work based on their suitability for improving process information extraction data following four exclusion criteria. We excluded all techniques that (1) are not applicable to the english language, (2) alters the spelling of words, (3) do not work for supervised training data, and (4) use task-, and/or domain-specific databases. Since many of the techniques can be configured in different ways, we searched for optimal configurations using a hyperparameter optimization. Using data augmentation we were thus able to improve the performance of our pipeline by up to 4.5 percentage points in the $F_1$ measure. The scope of this publication did not allow us to answer the following question.

**RQ-DE-3** *How does data augmentation affect the surface form and semantics of process descriptions?*

While many data augmentation techniques had a positive effect in our experiments, we treated them as a black box transformations. This means we could not answer questions about *how* specific data augmentation techniques actually change the surface form and semantics of a process descriptions.

Instead of immediately answering RQ-DE-3, we continued our exploration of the application of pretrained models to process information extraction (RQ-PM-2) in "*A Universal Prompting Strategy for Extracting Process Model Information from Natural Language Text using Large Language Models*" [52]. With the advent of large pretrained language models, researchers showed that it is plausible to use them in business process management tasks [7, 32]. Still, a comprehensive study on how to prompt these models to achieve state-of-the-art extraction results was missing, as previous work only used parts of the available data for evaluation [52]. For this reason we engineered prompts for the tasks entity mention detection, entity resolution, and relation extraction, based on best practices from information extraction research. We then applied these prompts to a total of three different datasets and eight different models, demonstrating that our prompting strategy for process information extraction is universally applicable. Our prompts improved the $F_1$ score by as much as 5 percentage points for the mention detection task on PET, and 17 percentage points compared to previous rule-based approaches to relation extraction. In an ablation study we were able to identify an iterative approach to extracting process information as the main contributing factor to these improvements. In this iterative approach we sequentially extract types process information mentions, feeding already extracted mentions back into the model as additional inputs. First we extract activities from the process description, which are comparatively easy to identify, as they are often verbs. We then extract actors and business objects, which are often in the immediate vicinity of activities. Finally, we

extract the remaining types of information. This approach leads to improvements of 8 percentage points in $F_1$ measure for both the mention detection and relation extraction tasks.

The limitations of extraction approaches based on large language models, such as reliance on cloud service providers, high computational costs, and privacy concerns, lead us to finally pursue answering RQ-EA-1, i.e., how to make data collection for process information extraction datasets more efficient and effective. In "*Assisted Data Annotation for Business Process Information Extraction from Textual Documents*" [51] we measured the cognitive workload of 31 participants during annotation of process relevant entity mentions and their coreferences and relations in process descriptions. We found that assisting the same annotators with automatically generated annotation recommendations resulted in significantly lower cognitive workloads, with reductions as much as $-51.0\%$. At the same time these recommendations improved the average quality of annotations compared to the gold standard by up to $+38.9\%$ measured in $F_1$ score, particularly benefiting non-expert annotators with little business process management experience. Notably, human review of the automatic recommendations improved the $F_1$ measure by up to 0.151, showing that the combination of a human supported with an AI-system is able to perform better, than both of them in isolation. We implemented our findings in a tool for efficient data augmentation, called TeaPie, and describe it in the publication "*TeaPie — A Tool for Efficient Annotation of Process Information Extraction Data*" [55]. There we discuss user feedback regarding the workflow using the tool, which was predominantly positive, with all users agreeing that the workflow was well suited for the annotation task, and allowed for uninterrupted work on it. A few select annotators (one each out of 31) had trouble always understanding what to do in each of the annotation steps, and work on these steps with satisfactory speed. Most notably, we found that many (12 out of 31) annotators did not find it useful to see a visualization of their current annotation as BPMN model, which was surprising given the nature of the annotation task. We plan to continue exploring why this is the case, and how to visualize annotations better for these users.

Finally, we expanded our understanding of data augmentation for process information extraction in "*Repeat, Reorder, Rephrase - Data Augmentation for Process Information Extraction*" (cf. section 12), which is currently under review. In this publication we focused on exploring how the data augmentations we found in [54], as well as seven additional ones, change the text of process descriptions. First, we categorized all data augmentation techniques by their approach into the classes *Repeating* (repeats part of the text or vocabulary), *Reordering* (reorders parts of the text), *Rephrasing* (rewording parts of the text), and *Adding Noise* (random changes to the text). Using a large language model we embed the text of original documents, sentences, mentions, and relations into a high dimensional vector space. Each of the resulting embedding vectors can be understood as a representation of the *meaning* of the original text [58]. We then reduce these high-dimensional vectors to vectors with two dimensions, using the tSNE method [59] and plot them to a scatter plot. Applying the same transformation to all embedding vectors of augmented documents, we can visualize the changes each of the augmentation classes have on process descriptions. We can then easily see, that repeating and reordering augmentations have nearly no impact on the meaning of process descriptions, unless for relations, where control flow relations between activities are often sensitive to their order of appearance in the text. Rephrasing augmentations and those that add noise do change the meaning. Since the latter often break the semantics of process descriptions, the resulting benefits are much lower for

extraction approaches trained with augmented data obtained from these augmentations.

With this we can now give an answer to our initial research question RQ-0. Until large datasets for training are available, pretrained models constitute the easiest way of applying deep learning to the process information extraction task. This notion is also reinforced by related work in the field, which uses pretrained large language models for extracting process information, or generates process models from natural language text inputs [7, 32, 40, 52]. Yet, such approaches have considerable requirements in terms of hardware resources, cost, or privacy concerns, something we will discuss later in Chapter 13. For this reason, supervised deep learning models, specialized on the process information extraction task should be the goal. To this end, tools like TeaPie [55] help with efficient collection of training data, which can then be supplemented with data augmentation techniques for more efficient usage.

# Chapter 5

# Data-driven annotation of textual process descriptions based on formal meaning representations

Ackermann, L., Neuberger, J., Jablonski, S. (2021). Data-Driven Annotation of Textual Process Descriptions Based on Formal Meaning Representations. In: La Rosa, M., Sadiq, S., Teniente, E. (eds) *Advanced Information Systems Engineering. CAiSE 2021.* Lecture Notes in Computer Science(), vol 12751. Springer, Cham. https://doi.org/10.1007/978-3-030-79382-1_5



**Contribution statement.** The concept to use formal meaning representations was developed by Lars Ackermann (LA). Model selection, implementation, hyper-parameter tuning, and feature selection was done by JN. LA and JN planned data collection and annotation. All authors annotated data themselves, and supervised the annotation of external contributors. JN evaluated the approach and created the corresponding figures. LA and JN interpreted the results. Writing and revision of the publication was done by all authors. LA and JN are the corresponding authors.

# Data-Driven Annotation of Textual Process Descriptions Based on Formal Meaning Representations

Lars Ackermann(✉), Julian Neuberger(✉), and Stefan Jablonski

University of Bayreuth, Bayreuth, Germany
{lars.ackermann,julian.neuberger,stefan.jablonski}@uni-bayreuth.de

**Abstract.** Business process management encompasses a variety of tasks that can be solved system-aided but usually require formal process representations, i.e. process models. However, it requires a significant effort to learn a formal process modeling language like, for instance, BPMN. Among others, this is one reason why companies often still stick to informal textual process descriptions. However, in contrast to formal models, information from natural language text usually cannot be automatically processed by algorithms. Hence, recent research also focuses on annotated textual process descriptions to make text machine processable.

While still human-readable, they additionally contain annotations following a formal scheme. Thus, they also enable automated processing by, for instance, formal reasoning and simulation. State-of-the-art techniques for automatically annotating textual process descriptions are either based on hand-crafted rule sets or artificial neural networks. Maintaining complex rule sets requires a significant manual effort and the approaches using neural networks suffer from rather low result quality. In this paper we present an approach based on Semantic Parsing and Graph Convolutional Networks that avoids manually defined rules and provides significantly better results than existing techniques based on neural networks. A comprehensive evaluation using multiple data sets from both academia and industry shows encouraging results and differentiates between several applied text features.

**Keywords:** Process modeling · Text annotation · Semantic parsing · Graph convolutional networks

## 1 Introduction

Business process models are a valuable means serving various purposes in Business Process Management (BPM). Due to their formal foundation they can be formally analyzed and used to configure workflow systems for process execution. Though they are intended to serve as a means of communication between domain experts and software specialists, too, they have to be specified in a predefined Process Modeling Language the stakeholders might not be familiar with.

Thus, the formal and, therefore, unfamiliar foundation of process models hinders their utilization, which among other reasons causes companies to rather rely on *textual process descriptions* in natural language, which can be observed [1–3,14,15,26,27], for instance, in terms of procedure instructions and process manuals. However, textual process descriptions impede the application of tools that operate on process models. Since studies have shown that hand-crafting process models consumes up to 60% of the overall time spent in business process management projects [15], research in the BPM discipline meanwhile considers techniques for transforming textual process descriptions into formal or semi-formal representations [1,2,15,26,27]. One representation type is *annotated textual process description* [23,26,27,31], which is still natural-language text but enriched with schematic information (*annotations*), which foster the derivation of formal process models [1,2,15,26], validate existing models against their descriptions in natural language [3] or against queries for formal reasoning [27,31] and also assist unfamiliar users in the creation of event logs [4,27]. There are only few approaches that automate this task [26,27], avoiding a labor-intensive manual annotation and all of them have drawbacks (see Sect. 3).

In this paper, we propose a technique for automatically annotating textual process descriptions based on semantic parsing [16], which formally describes the semantics of a natural language text, and linguistic features like, for instance, word embeddings [24,25]. Our approach utilizes established techniques but combines them in a novel way to contribute a step towards automated text annotation. We evaluate our approach on datasets from academia and industry [2,15,26,27] as well as a newly created dataset. The evaluation includes a comparison with results from two state-of-the-art approaches [26,27] and is two-fold: *(i)* we calculate metrics widely used in BPM research and *(ii)* we suggest metrics common in text annotation research [11,32] but that are not yet established in the BPM community. Regarding the state-of-the-art solutions our approach contributes to the research field by achieving the following objectives:

***O1*** It applies to different annotation tasks on textual process descriptions,
***O2*** It abstracts from plain syntactic variations and, therefore, reduces the number of linguistic patterns that express the same meaning,
***O3*** It is data-driven, which avoids the effort of manual rule definition but with similar result quality like a recent rule-based approach,
***O4*** It is less data-intensive than currently leading data-driven approaches,
***O5*** while showing a higher quality of the annotation results, and
***O6*** It is open for additional features it uses for solving the annotation tasks.

The achievement of the objectives is discussed in Sect. 6.

## 2 Preliminaries

### 2.1 Natural Language Processing

Subsequently, a selection of Natural language processing (NLP) techniques is introduced that is used in the proposed approach or in related approaches.

***Sentence splitting and Tokenization.*** Splits a text into sequences of basic units (e.g. words and punctuation marks), which are then grouped into sentences.

***Part-of-Speech (POS) Tagging.*** Assigns each token a lexical item category (e.g. VERB and NOUN) that indicates its syntactic properties and functions.

***Dependency Parsing.*** Analyzes the syntactical structure of a sentence by means of a specific syntax tree that describes dependencies between the words of a sentence (*dependency tree*). They are, by nature, sensitive to plain syntactic variations, which is discussed in detail in Sect. 2.2.

***Formal Meaning Representation.*** In contrast to dependency trees a formal meaning representation covers semantic relations between words (see Sect. 2.2).

***Word embedding techniques.*** Map words (or their meanings) from a vocabulary to vectors of real numbers. In our experiments (see Sect. 5), we vary between pre-trained models for word embedding techniques Word2Vec [24] and Glove [25].

***Text annotation.*** Means enriching sentences and tokens with schematic information. For the current paper these are information relevant for process modeling. The text annotation schemes relevant for this paper are explained in Sect. 2.4.

### 2.2 Semantic Parsing, Formal Meaning Representation and UCCA

Semantic parsing precisely transforms natural language utterances into formal meaning representations [17]. In contrast to syntactic representations, formal meaning representations describe the meaning of natural language texts [7] instead of providing insights in the formal constructions used for expressing this meaning [6]. This has three advantages:

1. *Abstraction from plain syntactical variations* in natural language utterances,
2. *Reflection of differences in their meanings*, at the same time, and,
3. *Reduction of their complexity*, and *Disambiguation of their meanings.*

Consequently, semantic parsing can be used as an intermediate step for easing language understanding tasks like the annotation task discussed in this paper. The benefit is illustrated in Fig. 1, which basically describes two situations by means of syntactic dependency trees: authors that took some book (a) and authors that showered (b and c). The phrases “took a book” and “took a shower” are described by exactly the same syntactic structure. Hence, approaches that aim at extracting information like actors, actions and business objects from

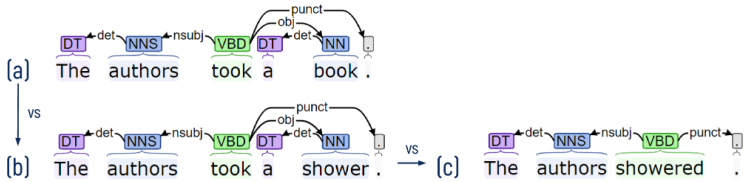



**Fig. 1.** Limitations of syntactical dependency trees

natural language text have to disambiguate whether "shower" and "book" are objects that can be taken in terms of "grasping" them. In formal meaning representations this disambiguation is already covered.

Because of the drawbacks of syntactic representations described above, our approach is based on formal meaning representations. Though there are multiple formal meaning representation schemes available, our approach is based on the graph-based *Unified Conceptual Cognitive Annotation (UCCA)* for the following reasons [6]: *(i)* It forms a cross-linguistically applicable scheme, *(ii)* it is able to describe the semantics of whole paragraphs and not only sentences, and *(iii)* it is based on cognitive categories that bear information, which is also relevant in the domain of business processes. For this paper, we focus on the latter advantage, which is depicted by the UCCA graphs shown in Fig. 2. The shown UCCA graphs differentiate between a procedure of doing something (a) and a procedure of doing something with a particular object (b). In contrast to "shower", "book" is located in a different sub-graph with a different meaning, which directly solves the issue shown in Fig. 1. The meaning of the different sub-graphs are defined by their nodes, which are, in turn, related to each other with cognitive categories defined in [6]. ***(P)****rocess* describes that something evolves over time (e.g. actions or movements). ***(S)****tate* is the opposite since it marks something that does not evolve over time and a *P****(A)****rticipant* is anything that participates in a process or state (e.g. locations or entities). An in-depth discussion of all available cognitive categories can be considered out-of-scope for this paper. We, therefore, refer to [6] for further reading.

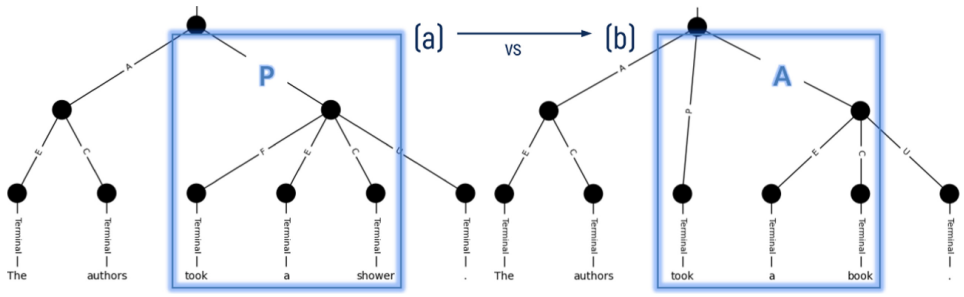



**Fig. 2.** Two UCCA graphs disambiguating a syntactically ambiguous utterance

### 2.3 Artificial Neural Networks for Graphs

Graph convolutional networks (*GCN*) are neural networks able to process graphs. Their core idea is based in graph signal processing [29] and recent advances made them efficient in large scale use [20]. They continuously update a hidden state $h^{(l)}$ for every node, based on its incoming edges. In this work we use *R-GCN* [28], which is able to process graphs with labeled edges.

We define the directed and labeled multi-graph $G$ as a tuple $(V, E, R)$, where $V$ is a set of nodes $v_i \in V$; $E$ a set of labeled and directed edges $(v_i, r, v_j) \in E$

and $R$ a set of edge types, so that $r \in R$. The function for updating hidden state $h_i^{(l)}$ for node $v_i$ at iteration $l$, also called propagation rule is then

$$h_i^{(l+1)} = \sigma(\sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)})$$

$N_i^r$ is the set of indices for nodes connected to node $v_i$ by an edge of type $r$. $W^{(l)}$ is a matrix of learnable weights, $\sigma$ is an element-wise activation function such as the rectified linear unit $ReLU(x) = max(0, x)$ or $softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$. $c_{i,r}$ is a normalization constant, which is usually set to $|N_i^r|$. The output of this propagation rule is then either used as input for the next graph convolutional layer or as the net's final output, see Sect. 4.

### 2.4 Annotation Schemes and Tasks for Textual Process Descriptions

There are several formalisms how textual process descriptions can be annotated with schematic information. In order to evaluate the applicability of our approach, we discuss experiments that are based on two different annotation schemes: *(i)* An important subset of the *Annotated Textual Descriptions of Processes (ATDP)* [27] scheme that has proven to enable formal reasoning [31] and *(ii)* the *multi-grained text-classification (MGTC)* scheme used in [26] to automatically derive procedural business process models from annotated texts.

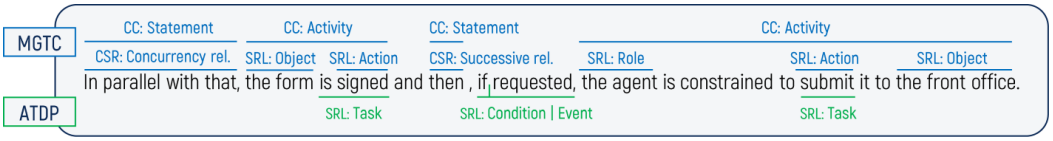



**Fig. 3.** Example of an annotated textual process description

The MGTC scheme defines three annotation tasks[1] on two different levels (clause and token level) and with different annotation types (see Fig. 3).

- *Clause Classification (CC):* Annotate a clause as an *activity* or a *statement*.
- *Clause Semantics Recognition (CSR):* Determines the semantics of a statement clause, i.e. extracts the concrete control flow pattern.
- *Semantic Role Labeling (SRL):* Classifies the tokens of each activity clause.

Activity clauses describe, which *Roles* perform which *Actions* on what *Objects*, which determined by the SRL task. From statement clauses concrete control-flow relations are extracted. These refer to the beginning of a block of

[1] Literature refers to Clause Classification and Clause Semantics Recognition as Sentence Classification and Sentence Semantics Recognition, which suggests processing of whole sentences, though the discussed approach operates on clauses instead.

actions *(block begin)*, the ending of such a block *(block ends)*, a relation organizing activities as a sequence *(successive relation)*, a decision point *(optional relation)* and a parallelization of the control flow *(concurrency relation)*.

In contrast to MGTC, the ATDP scheme does not include annotation tasks like CC or CSR (see Fig. 3). Instead, in an SRL task similar to that in MGTC, it provides annotation types for distinguishing between *activity fragments*:

- *Task* represents the atomic units of work in business processes,
- *Event* is usually part of the process flow but are out-of-scope for the organization responsible for executing the process,
- *Condition* describes circumstances under which other information pertain.

Though the ATDP scheme is far more expressive, we limit the description to the mentioned fragments for two reasons: *(i)* This paper focuses upon the SRL tasks, which means that, for instance, relation extraction is out of scope and *(ii)* several other ATDP classes are not evaluated in [27] making annotation results incomparable. Besides that we consider the CC and CSR tasks according to the MGTC scheme because they are rather similar to the SRL task, except that whole clauses are annotated instead of tokens. Furthermore, the approach that builds upon MGTC [26] and our approach are, in contrast to the approach that uses ATDP [27], both data-driven and, consequently, more comparable.

In addition to the annotation types described above, we introduce a *None* type for the SRL task. This is necessary since our approach treats this task as a token classification problem and, thus, has to reflect tokens that are irrelevant for the SRL task (see Sect. 4). Since both MGTC and ATDP contain an SRL task we refer to the particular tasks as *SRL (MGTC)* and *SRL (ATDP)*.

## 3 Related Work

To clarify the focus of this paper, we concentrate on a discussion of related approaches that explicitly involve or focus on annotating textual process descriptions following a specified scheme and that are state of the art [15,26,27]. Here we distinguish between *rule-based* and *machine-learning-based* techniques. While the former rely on sets of hand-crafted rules to extract annotations, the latter build upon machine-learning techniques like artificial neural networks. We also briefly discuss experiences and similarities with approaches related to information modeling tasks like object-oriented modeling and database modelling.

***Rule-based Approaches.*** Friedrich et al. [15] build upon standard NLP tools that, for instance, extract language features like syntax trees from sentences in order to analyze them using an extensive set of rules. Some of the extracted information are annotations, which are finally processed to generate a BPMN model. Quishpi et al. [27] extract annotations that conform to the ATDP scheme (see Sect. 2.4) and define tree-based rules that analyze dependency trees in order to generate candidates that fit this scheme. Both approaches rely on rule sets that are defined upon syntax trees, which are inherently sensitive to syntactical

changes of the underlying natural language utterance. Consequently, this significantly raises the number of required rules to cover all interesting syntactic patterns. This means that experts for the particular approach have to be mindful of for missing rules and, at the same time, have to avoid ambiguities due to overlaps. Eventually, this usually lowers the portability of rule-based approaches to unseen data (see Sect. 5.2). Furthermore, two natural language utterances might have the same syntactic structure but can likewise have distinct meanings (see Sect. 2.2), which lowers the information content of syntax trees. Another drawback is the limitation to human-interpretable features, i.e. it is hard to hand-craft rules on, for instance, high-dimensional word embedding vectors.

***Machine-learning-based Approaches.*** The approach proposed by Qian et al. [26] extracts annotations conforming to the MGTC scheme (see Sect. 2.4) and, thus, builds upon a multi-grained analysis of textual process descriptions, which consists of three annotation tasks (see Sect. 2.4). For each of the three tasks the approach builds upon a separate artificial-neural-network architecture. Since all of these architectures rely on word embeddings of several hundreds of dimensions (e.g. Word2Vec) the approach depends on the availability of rather huge amounts of data (see Sect. 5.2). Another drawback of the approach discussed in [26] is that it omits the issue of *finding* the correct span of tokens that forms a clause, which can be classified. Instead it requires a manual pre-processing of all input data, which segments natural language utterances. The approach described in [22] extracts and classifies candidate tasks for robotic process automation from textual process descriptions. However, since it focuses on distinguishing between activity types its comparability to our approach is rather low.

***Distantly Related Approaches.*** In [12] 13 approaches are compared, which are tailored to transform textual requirements specifications into UML models and an included study emphasizes the need for automated transformation tools. Several approaches involve text annotation as an intermediate step making them comparable to our proposed approach (e.g. [21]). Other approaches aim at extracting *database models* from requirements specifications [9]. Most of the approaches rely on syntactic features, hand-crafted rules and external knowledge sources (e.g. ontologies). Hence, they suffer from the same drawbacks discussed above and that are in the focus of our research. However, most of them use token-level features like POS tags to disambiguate word senses. Finally, another interesting approach extracts process activities from email logs [18]. While it focuses on issues caused by specific characteristics of emails, it uses word embeddings to derive the semantics of words and sentences. We conclude from this observation that widely used token-level features like POS tags and word embeddings can be valuable for our approach (see Sect. 4), too. Though some approaches overlap with our proposed approach, they cannot be applied directly since they do not achieve the defined objectives: *(i)* According to objective O1 the approach has to solve very different annotation tasks (see Sect. 2.4), *(ii)* according to objective O3 it should be data-driven, and *(iii)* objective O6 requires feature extension to be part of the concept. Finally, the annotation schemes are specific for the process domain and, in contrast to standardized languages like UML in object-

oriented modeling, they are highly variable and bear process-specific challenges (see Sect. 2.4). Other approaches process alternative input types (e.g. controlled natural language), which are easier to be interpreted automatically. But since we focus on already existing textual process descriptions, they are out of scope.

## 4 Approach

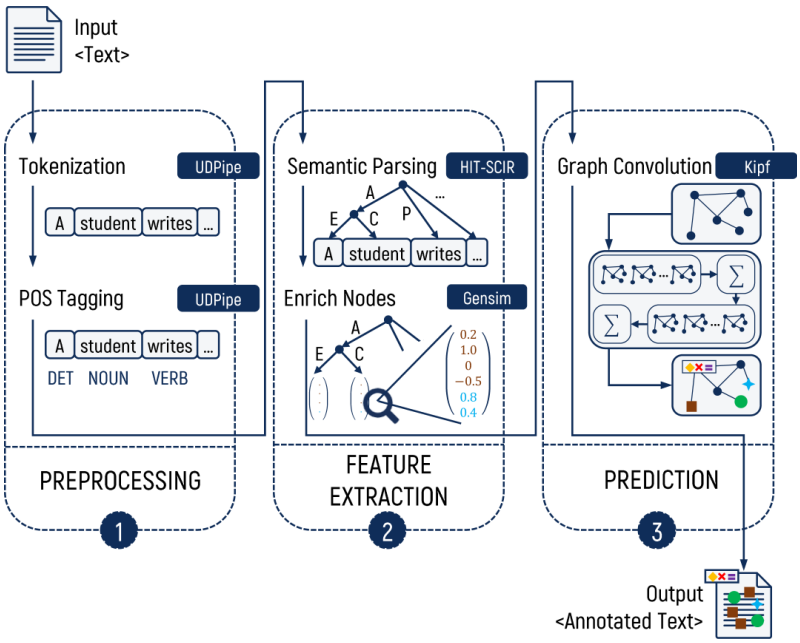



**Fig. 4.** Approach overview (depiction of graph convolution based on [34])

The input for our approach is a textual process description, which is tokenized and segmented into sentences. Each token is associated with its part of speech (POS tagging). We denote this phase as *PREPROCESSING (1)*.

Sequences of tokens prepared in this manner are passed on to the second phase, *FEATURE EXTRACTION (2)*. Here, a semantic parser generates a formal meaning graph, which describes the semantic structure of a sentence. Graph nodes corresponding to tokens are called terminal nodes. These contain features obtained in the previous step, namely POS-tags and token text.

Every node in the graph is then transformed into a numerical representation. Transformations include one of the following:

1. *No Features:* We evaluate the discriminative power of formal meaning representations in isolation, by encoding each node's index to a one-hot-vector. This way subsequent steps have no detailed information about tokens.

2. *Word embedding:* The numerical vector representation for each word from a pre-trained embedding model. Since we use techniques that generate meaningful vector representations, we call this whole step *node enrichment*. Non-terminal nodes, not containing text information are assigned the zero-vector.
3. *POS tag encoding:* POS tags are one-hot-vector encoded. Non-terminal nodes, not containing text information are assigned the zero-vector.

Eventually, terminal nodes are then transformed into class predictions during the *PREDICTION (3)* phase. Here we extract adjacency matrices $A_r$ from the formal meaning graphs. We then generate a node feature matrix $X$ by stacking all node feature vectors obtained in (2). The choice of features influences final accuracy significantly, which is why we analyse different node features in detail in Sect. 5.2. Terminal nodes in UCCA graphs can correspond to several tokens, in which case their feature vectors must be combined, to form a single one. We use averaging in case of word embeddings and addition in case of part-of-speech tags.

Building the adjacency matrices $A_r$ requires some additional steps to fulfill several assumptions made by [20] and [28]:

1. Graph convolutions as proposed by [20] rely on self loop edges to incorporate a node's current information $h^{(l)}$ into its next state $h^{(l+1)}$. Without self loops a node's new state solely relies on its neighbours' information (see Sect. 2.3). Therefore, we extend the set of edges with $\{(v, r_s, v)|\forall v \in V\}$, where $r_s$ is a special self loop relation type added to $R$, so that $r_s \in R$.
2. UCCA graphs are defined as directed acyclic and labeled graphs. To be able to process UCCA graphs, we transform the graph into an undirected and labeled one by extending the set of edges $E$ with $\{(v, r, u)|\forall(u, r, v) \in E\}$.
3. GCNs as proposed by [20] and [28] do not allow for classifying graphs globally, something we need for the CC task. This can be implemented via Global Pooling Layers, Attention Sum or a Global Readout Node [34]. We chose the latter. Therefore, to classify graphs globally we add a new node $v_g$ and edges $\{(v, r_g, v_g)|\forall v \in V\}$, where $r_g$ is a special global relation type added to $R$.

Evaluating the propagation rule for given number of hidden layers $l$, adjacency and node feature matrices, we are left with final node state vectors $h_i^{(l)}$, which represents our model's output. Node classes can now be predicted by masking non-terminal nodes and applying a softmax. Graph-level classes can be predicted by the same process, but instead of masking all non-terminal nodes of the original graph, all nodes except the global node are masked.

## 5 Evaluation

### 5.1 Dataset Description and Experimental Setup

The subsequent quantitative analysis of the proposed approach is based on the datasets shown in Table 1. The datasets COR and MAM stem from [26] and contain textual cooking recipes and maintenance manuals. Though, the two datasets

are of considerable size, they vary comparably little regarding their vocabulary and linguistic structures. We show in Table 2 that our approach achieves a saliently high performance, which causes us to validate it on a dataset of smaller size and higher linguistic variability. Hence, we also evaluate the approach on the dataset stemming from [27] (QCD) and another dataset newly created by us (VBP). Covered domains are, for instance, document management and quality management. For the VBP dataset we intentionally omitted any data curation since we also measure how the approach reacts to noisy data in terms of inconsistent labeling. We consider this, because a common issue in text annotation are ambiguous gold standard labels [11,32] (e.g. including determiners or not).

**Table 1.** Statistics of the datasets used for evaluation

| Domain | COR | MAM | QCD | VBP |
|---|---|---|---|---|
| | Recipe | Maintenance | Mixed | Mixed |
| # Labeled sentences | 17,562 | 14,370 | 203 | 250 |
| # Labeled tokens | 34,439 | 28,174 | 3,581 | 6,600 |
| # Sentence-level categories | 8 | 8 | - | 5 |
| # Word-level categories | 4 | 4 | 5 | 6 |

We implemented our approach using Python 3.6.11 and it consists of R-GC layers proposed by [28]. It is implemented in the Deep Graph Library [33] at version 0.5.2, which allows for different backends, i.e. Tensorflow [5], which we use at version 2.3.1. We build our approach with 2 hidden graph convolutional layers, each consisting of 64 hidden units and train it using the categorical cross-entropy loss [35] and Adam [19] optimizer with learning rate set to 5e−5. For converting text to UCCA graphs we first tokenize and tag them using spaCy (https://spacy.io/. Accessed 5 Dec 2020) and UDPipe [30]. The HIT-SCIR parser [10] creates the graphs, which are then processed as described in Sect. 4.[2]

For each experiment we employ a 5-fold stratified cross validation. Using a stratified cross validation we ensure the same distribution of target classes is present in training and test sets. We remove samples with targets that do not have at least 5 instances across all data sets to guarantee test splits with at least one instance of every class. We report the mean over five folds, with exception of approach by [27], where values are obtained without cross validation, since rule-based methods do not profit from splitting the data into train and test sets.

We intend to capture the ambiguity [32] during labeling with varying degrees of fuzziness. As such, metrics reported are F1 score variants. The *Exact* F1-score is a valid goal but fails to recognize the inherent uncertainty described in Sect. 5.1. F1 is defined as harmonic mean $F1 = 2PR/(P + R)$ of Precision $P = \#ok/\#pred$ and Recall $R = \#ok/\#gold$. $\#pred$ is the number of predicted spans, $\#gold$ is the number of expected spans. Calculation of $\#ok$ follows [11,32].

[2] Our code can be accessed at https://github.com/JulianNeuberger/UCCA4BPM.

***Exact***. *#ok* is increased by 1, if the predicted annotations for a span and its boundaries match those of the gold standard.

***Partial***. *#ok* is increased by 1, if requirements for *Exact* hold; if at least one predicted token annotation matches the gold standard, *#ok* is increased by 0.5.

***Fragment***. works on token-level instead of span-level. All spans are fragmented and then handled like in *Exact* while adjusting *#pred* and *#gold*.

### 5.2 Overall Results and Further Analysis

We compare our approach to a state-of-the-art machine learning approach that outperforms several traditional methods [26]. To address the problems described in Sect. 3, we additionally chose a recent rule-based approach with a much smaller data set [27] as a second baseline. Our approach outperforms [26] on their data sets COR and MAM as well as our smaller and noisier data set. When comparing our approach to the rule-based approach by [27] we perform worse on their data set, while outperforming them on ours, as shown in Table 2. For our dataset MGTC seems to be unable to learn the SRL task and consistently predicts the *None* class, resulting in no exact span matches. Values reported under *Our approach* are for the optimal configuration (see Sect. 5.1). Results marked with "–" are not reported, since they correspond to tasks with a single readout node (see Sect. 4). In the following, we discuss the main aspects impacting our approach's performance.

**Table 2.** Results on datasets by related work and ours.

| Dataset | Public code by [26] | | | Our approach | | |
|---|---|---|---|---|---|---|
| | F1 exact | F1 partial | F1 fragment | F1 exact | F1 partial | F1 fragment |
| COR (CC) | – | – | 66.11 | – | – | **99.94** |
| COR (CSR) | – | – | 60.95 | – | – | **96.77** |
| COR (SRL) | 43.45 | 59.90 | 64.97 | 97.03 | 97.86 | **98.26** |
| MAM (CC) | – | – | 67.57 | – | – | **99.89** |
| MAM (CSR) | – | – | 60.50 | – | – | **97.46** |
| MAM (SRL) | 41.78 | 59.09 | 63.86 | 96.08 | 97.20 | **97.78** |
| Our data set (CC) | – | – | 69.13 | – | – | **87.89** |
| Our data set (CSR) | – | – | 69.24 | – | – | **77.19** |
| Our data set (SRL) | 0.0 | 39.19 | 61.28 | 36.09 | 53.45 | **65.05** |
| | Public code by [27] | | | Our approach | | |
| Dataset from [27] | 54.61 | 68.22 | **91.59** | 45.21 | 63.52 | 88.15 |
| Our dataset (SRL) | 24.07 | 49.54 | 88.40 | 39.35 | 61.98 | **90.66** |

***Node Features.*** Rich features allow the approach to distinguish in harder annotation cases, where the UCCA graph alone is not enough. Highly dimensional feature vectors introduce many new learnable parameters however, which in turn can not be trained with small datasets [8]. The trend in Fig. 5 suggests using no

node features, the UCCA graph itself contains enough information to reach performance comparable to or better than other state-of-the-art approaches. The usefulness of word embeddings depends on the corpus they were trained on. Certain stop words (i.e. "if", "before"), are useful in deciding whether a clause contains Conditions or Tasks. If the word embedding model was trained without those tokens the additional model complexity does not pay off. Fine POS tags seem to strike a good balance between additional information, while condensing it into a fairly small one-hot encoded vector of up to 17 elements[3].

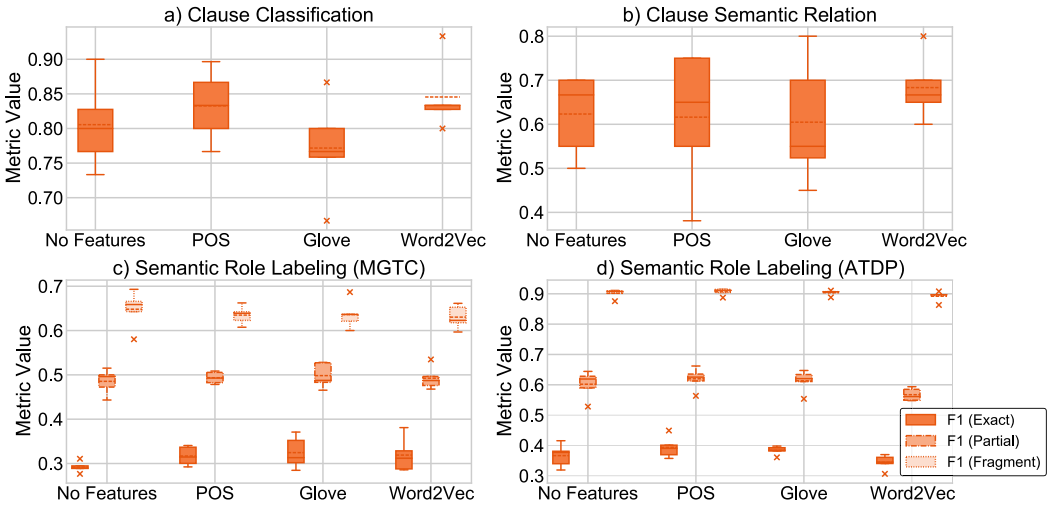



**Fig. 5.** Comparing node features for tasks CC, CSR, SRL(MGTC) and SRL(ATDP)

***Learning Rate.*** A high learning rate will cause the optimizer to change weights by a large delta, therefore, resulting in faster but more unstable training. On the other hand a smaller learning rate will result in more stable training at the cost of longer training times. We trained the network in its optimal configuration only changing learning rate and determined a learning rate of 5e−5 as optimal.

***Size of Hidden Node State.*** Larger hidden node states allow for more complex aggregation of neighbouring node states so the model is able to represent more variance in the data. This comes with the cost of having more parameters, which need fitting, though – unlike with e.g. the number of relations [28] – increasing the number of hidden units in a given layer increases the number of weights linearly [20]. Our experiments indicate an optimal number of hidden units of 64.

***Number of Graph Convolutional Layers.*** Intuitively the number of hidden graph convolutional layers affects the distance one node can collect neighbourhood information from. Initially we suspected a relatively high number of edges need to be traversed to gain the information needed for the text annotation

[3] see https://universaldependencies.org/u/pos/, accessed 2020/12/5.

tasks. But, like Fig. 6 shows, proper node features and the incoming edge[4] alone is enough. We suspect that a model trained on a larger dataset for solving a more complex task would make use of more neighbourhood information.

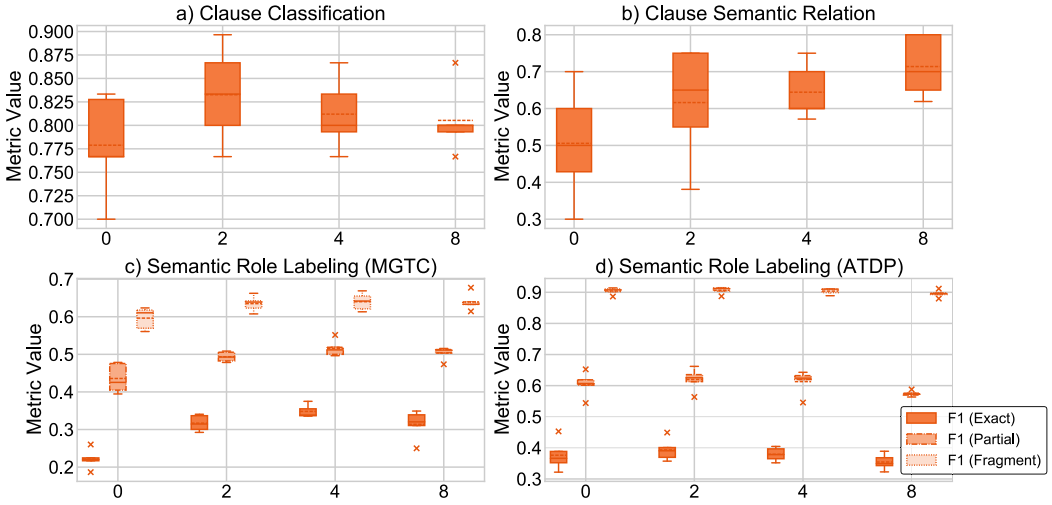



**Fig. 6.** Importance of choosing the right number of hidden layers in our model.

***Assumptions.*** From a conceptual perspective our approach assumes plain text (i.e. no markup language) as input or requires an additional pre-processing step (see Fig. 4). The implementation has a modular structure and further assumptions might arise when adding new or replacing existing features and implementations, like the word-embedding model and the semantic parser. Our implementation involves a semantic parser trained on a mixture of web reviews and Wikipedia articles, while our word embedding models are either trained on Google's news aggregations (Word2Vec) or Twitter tweets (Glove). Hence, the quality of our implementation is to some extent dependent on the differences regarding linguistic structures and vocabularies between the input and the data used for pre-training. The current implementation is language-independent. However, if using the pre-trained Glove model input documents in English are required or one has to retrofit Glove to the intended input language.

## 6 Conclusion and Future Work

In this paper, we propose an annotation approach for textual process descriptions and qualitatively measure its contribution based on a set of objectives (see Sect.

[4] Using token based node features, inner nodes use the zero vector as feature, since they do not have a corresponding token. Therefore, two edges need to be traversed before the incoming edge information is aggregated in a terminal node: The artificial inverse edge "up" the UCCA structure and only then the edge in question, see Sect. 4.

1). It is data-driven, relies on artificial neural networks (O3), and outperforms the currently best-performing data-driven approach (see Sect. 5) (O5) in all defined annotation tasks (see Sect. 2.4) (O1). At the same time we use lower-dimensional word-embedding features, which makes the approach less data-intensive (O4). Though, the state-of-the-art rule-based approach shows better results on one dataset, we outperform it on another dataset with a similar quality showing that our approach is more stable on unseen data. Since the approach proposed in this paper is based on formal meaning representations it abstracts, by nature, from syntactic variations (O2). Finally, we describe how different features can be incorporated in the annotation task (O6).

Currently the approach is not able to model relations between tokens. This drawback presents an exciting avenue of future work, especially since our technical backbone, GCN, is able to perform this task via a technique called *Link Prediction* [28]. Similarly, we are limited to one annotation per token. This can be solved by using a different activation function in the last graph convolutional layer, but it stands to show that training on our small, inter-domain dataset yields comparable results. Finally we would like to advance our experiments with regards to different node features, including, but not limited to, knowledge-graphs like WordNet [13], Universal Features[5] and combinations of different features, e.g. part-of-speech tags and word embeddings.

**Acknowledgements.** We thank Omri Abend (HUJI) and Daniel Hershcovich (UCPH) for their assistance with UCCA, Lluís Padró, Luis Quishpi and Josep Carmona (UPC) for valuable advice regarding their approach, and the DBIS Chair (UBT) for assistance creating the new dataset.

## References

1. van der Aa, H., Carmona, J., Leopold, H., Mendling, J., Padró, L.: Challenges and opportunities of applying natural language processing in business process management. In: Proceedings of COLING. ACL (2018)
2. van der Aa, H., Di Ciccio, C., Leopold, H., Reijers, H.A.: Extracting declarative process models from natural language. In: Giorgini, P., Weber, B. (eds.) CAiSE 2019. LNCS, vol. 11483, pp. 365–382. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21290-2_23
3. van der Aa, H., Leopold, H., Reijers, H.A.: Detecting inconsistencies between process models and textual descriptions. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 90–105. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_6
4. Aalst, W.: Data science in action. Process Mining, pp. 3–23. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4_1
5. Abadi, M., et al.: Tensorflow: a system for large-scale machine learning. In: Proceedings of OSDI (2016)
6. Abend, O., Rappoport, A.: Universal conceptual cognitive annotation (UCCA). In: Proceedings of the ACL. ACL (2013)

[5] https://universaldependencies.org/u/feat/index.html, accessed 2020/12/5.

7. Abend, O., Rappoport, A.: The state of the art in semantic representation. In: Proceedings of the ACL. ACL (2017)
8. Allen-Zhu, Z., Li, Y., Liang, Y.: Learning and generalization in overparameterized neural networks, going beyond two layers. In: Proceedings of NeurIPS (2019)
9. Btoush, E.S., Hammad, M.M.: Generating ER diagrams from requirement specifications based on natural language processing. In: IJDTA (2015)
10. Che, W., Dou, L., Xu, Y., Wang, Y., Liu, Y., Liu, T.: HIT-SCIR at MRP 2019: a unified pipeline for meaning representation parsing via efficient training and effective encoding. In: Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 CoNLL (2019)
11. Chinchor, N., Sundheim, B.: Muc-5 evaluation metrics. In: Proceedings of MUC. ACL (1993)
12. Dawood, O.S., et al.: From requirements engineering to UML using natural language processing-survey study. In: EJERS (2017)
13. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. Language, Speech, and Communication. MIT Press (1998)
14. Figl, K., Recker, J.: Exploring cognitive style and task-specific preferences for process representations. Requirements Eng. **21**(1), 63–85 (2014). https://doi.org/10.1007/s00766-014-0210-2
15. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 482–496. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21640-4_36
16. Hershcovich, D., Abend, O., Rappoport, A.: A transition-based directed acyclic graph parser for UCCA. In: Proceedings of the ACL. ACL (2017)
17. Jia, R., Liang, P.: Data recombination for neural semantic parsing. In: Proceedings of ACL. ACL (2016)
18. Jlailaty, D., Grigori, D., Belhajjame, K.: Email business activities extraction and annotation. In: Kotzinos, D., Laurent, D., Spyratos, N., Tanaka, Y., Taniguchi, R. (eds.) ISIP 2018. CCIS, vol. 1040, pp. 69–86. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30284-9_5
19. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
20. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of ICLR (2017)
21. Körner, S.J., Landhäußer, M.: Semantic enriching of natural language texts with automatic thematic role annotation. In: Hopfe, C.J., Rezgui, Y., Métais, E., Preece, A., Li, H. (eds.) NLDB 2010. LNCS, vol. 6177, pp. 92–99. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13881-2_9
22. Leopold, H., van der Aa, H., Reijers, H.A.: Identifying candidate tasks for robotic process automation in textual process descriptions. In: Gulden, J., Reinhartz-Berger, I., Schmidt, R., Guerreiro, S., Guédria, W., Bera, P. (eds.) BPMDS/EMMSAD -2018. LNBIP, vol. 318, pp. 67–81. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91704-7_5
23. López, H.A., Debois, S., Hildebrandt, T.T., Marquard, M.: The process highlighter: from texts to declarative processes and back. In: CEUR Workshop Proceedings (2018)
24. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR, Workshop Track Proceedings (2013)
25. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the Conference on EMNLP (2014)

26. Qian, C., et al.: An approach for process model extraction by multi-grained text classification. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds.) CAiSE 2020. LNCS, vol. 12127, pp. 268–282. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49435-3_17
27. Quishpi, L., Carmona, J., Padró, L.: Extracting annotations from textual descriptions of processes. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 184–201. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_11
28. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) Modeling relational data with graph convolutional networks. In: Proc. of ESWC. Springer (2018). LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38
29. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. In: IEEE SPM (2013)
30. Straka, M., Straková, J.: Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In: Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies (2017)
31. Sànchez-Ferreres, J., Burattin, A., Carmona, J., Montali, M., Padró, L.: Formal reasoning on natural language descriptions of processes. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 86–101. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_8
32. Tsai, R.T.H., et al.: Various criteria in the evaluation of biomedical named entity recognition. BMC Bioinform. **7**, 92 (2006)
33. Wang, M., et al.: Deep graph library: a graph-centric, highly-performant package for graph neural networks. arXiv: Learning (2019)
34. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. In: IEEE Transactions on NNLS (2020)
35. Zhang, Z., Sabuncu, M.: Generalized cross entropy loss for training deep neural networks with noisy labels. In: NeurIPS (2018)

# Chapter 6

# Bridging research fields: an empirical study on joint, neural relation extraction techniques

Ackermann, L., Neuberger, J., Käppel, M., Jablonski, S. (2023). Bridging Research Fields: An Empirical Study on Joint, Neural Relation Extraction Techniques. In: Indulska, M., Reinhartz-Berger, I., Cetina, C., Pastor, O. (eds) *Advanced Information Systems Engineering. CAiSE 2023.* Lecture Notes in Computer Science, vol 13901. Springer, Cham. https://doi.org/10.1007/978-3-031-34560-9_28



**Contribution statement.** All authors selected relevant research databases for this literature review. Julian Neuberger (JN) and Martin Käppel (MK) created and ran the search queries. JN, MK, and Lars Ackermann (LA) defined the exclusion criteria. JN and MK filtered the list of sources based on the exclusion criteria. JN created the framework for evaluating the selected pretrained models. JN and MK ran the evaluation. JN created the figures for study results. LA, JN, and MK interpreted the results. Writing and revision of the publication was done by all authors. LA is the corresponding author.

# Bridging Research Fields: An Empirical Study on Joint, Neural Relation Extraction Techniques

Lars Ackermann(✉), Julian Neuberger, Martin Käppel, and Stefan Jablonski

Institute for Computer Science, University of Bayreuth, Bayreuth, Germany
{lars.ackermann,julian.neuberger,martin.kaeppel,
stefan.jablonski}@uni-bayreuth.de

**Abstract.** Information systems that have to deal with natural language text are often equipped with application-specific techniques for solving various Natural Language Processing (NLP) tasks. One of those tasks, extracting entities and their relations from human-readable text, is relevant for downstream tasks like automated model extraction (e.g. UML diagrams, business process models) and question answering (e.g. in chatbots). In NLP the rapidly evolving research field of *Relation Extraction* denotes a family of techniques for solving this task application-independently. Thus, the question arises why scientific publications about information systems often neglect those existing solutions. One supposed reason is that for reliably selecting an appropriate technique, a comprehensive study of the available alternatives is required. However, existing studies *(i)* cannot be considered complete due to irreproducible literature search methods and *(ii)* lack validity, since they compare relevant approaches based on different datasets and different experimental setups. This paper presents an empirical comparative study on domain-independent, open-source deep learning techniques for extracting entities and their relations jointly from texts. Limitations of former studies are overcome *(i)* by a rigorous and well-documented literature search and *(ii)* by evaluating relevant techniques on equal datasets in a unified experimental setup. The results[1] show that a group of approaches form a reliable baseline for developing new techniques or for utilizing them directly in the above mentioned application scenarios([1]Our code and data: https://github.com/JulianNeuberger/re-study-caise.).

**Keywords:** Named Entity Recognition · Relation Extraction · Natural Language Processing · Artificial Intelligence

## 1 Introduction

Due to a dramatic increase in the amount and volume of textual information sources, techniques for automatically extracting and formalizing information from texts play a crucial role in information systems engineering [1,5,15,20,21].

Our work is supported by the Bavarian Research Foundation (grant no. AZ-1390-19).

*Relation extraction (RE)* techniques contribute to solving this task, since they aim at extracting entities and relations among them. Our research group has come into contact with RE from the application perspective of process model extraction from human-readable text sources (e.g. [2,8,21]) written in English (e.g. for identifying resources, activities, data objects, and connections among them) [2]. Further exemplary applications are the extraction of UML diagrams (e.g. [7,25]) or entity-relationship diagrams (e.g. [4]). Each of those applications contains a *specific* solution for extracting relations. This raises the question of whether there is a promising *baseline technique* that could be used "out of the box"[1] to solve this task *without* the effort of developing application-specific solutions, which is the standard approach these days?

Most existing techniques are published in isolation and with an evaluation on few, mostly different datasets and with different experimental setups like different evaluation metrics and different usage of training and validation splits. Hence, it is not possible to compare approaches based on their documented evaluation results [27]. To overcome this issue, *Papers with Code*[2] allows for sharing papers along with code and all resources for reproducing experiment results. However, results can be published by the authors themselves, which neither guarantees a uniform experimental design nor obviates the need for reproducing results in order to ensure correctness. Literature surveys [13,14,16,17,29] do not abolish these issues, since they base their discussions on evaluation results from the original publications, a principle which suffers from the same shortcomings. Empirical comparative studies are intended to provide a unified evaluation but former studies [19,27] are incomplete due to irreproducible literature search methods. Consequently, it is not possible to identify the best-performing relation extraction approach directly from existing literature.

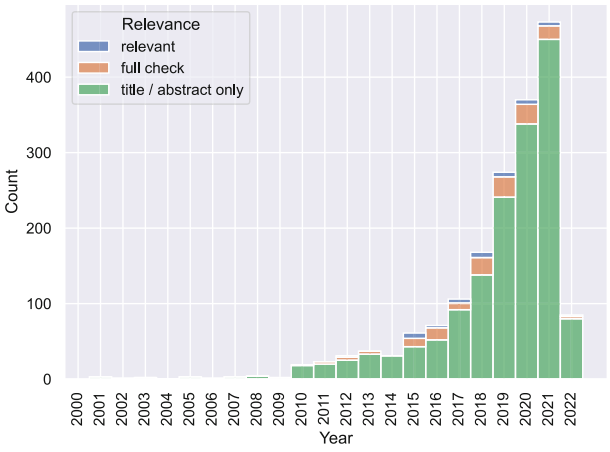



**Fig. 1.** Publications relevant to RE since 2000.

Considering the observations above, the contributions of this paper are the following: ***(C1)*** It comprises a rigorous, reproducible literature search based on well-documented search queries, filter criteria and documented review decisions that is based on PRISMA [18], an established method for literature reviews. ***(C2)*** It provides a unified evaluation of the RE techniques selected in point ***C1***. ***(C3)*** As a basis for contribution ***C2*** all raw predictions on the test sets

[1] A definition for this term is given in Sect. 3.1.
[2] https://paperswithcode.com/.

are (re-)produced by the authors as an independent party. ***(C4)*** It identifies a group of relation extraction techniques that show promising results. ***(C5)*** It uncovers the most likely drivers of performance degradation in neural Joint RE techniques.

The impact of contribution ***C1*** can be inferred from Fig. 1, which shows an exponential increase in RE publications. Our study thus reveals the applicability of the investigated approaches to a given dataset and allows for a well-founded decision in favor of one of these approaches or, alternatively, the development of a specific solution. Due to the reproducible search method, the study can be extended by adding papers published in recent years through running a documented set of search queries[3] and adding a time restriction. The result of the literature search is a selection of RE techniques that fulfills the specified filter criteria (Sect. 2.2). This paper focuses on the subclass of *Neural Joint RE techniques*. *Joint* means that a single model is trained for extracting both entities and relations instead of training two separate models. The term *Neural* refers to approaches that are based on deep learning. The rationale behind these and other restrictions is discussed in Sect. 2.2.

The remainder of this paper is structured as follows: Section. 2 reifies the RE task and provides an overview of related surveys and comparative studies. Furthermore, it describes the criteria used for selecting RE approaches that are evaluated based on an experimental setup described in Sect. 3. This section also presents the evaluation results along with a thorough analysis. Finally, Sect. 4 summarizes core insights of this study and provides impetus for future research.

## 2 Research Scope

Consistent with the idea of bridging research fields, the research scope is tailored to two types of readers: *(i)* Contributors to research domains that require an RE technique to solve a specific downstream task and *(ii)* contributors to the research domain of RE who need to reflect on the state of the art. For both types of readers, this study answers the following leading research question for the subset of RE techniques defined in Sects. 2.1 and 2.2:

> *Are there Joint RE approaches that stably achieve better results on diverse datasets, and what data characteristics cause performance degradation?*

RE is a broad term, therefore Sect. 2.1 defines how we use it in this paper. Section 2.2 defines criteria for including a RE approach in experiments.

### 2.1 Task Description: Relation Extraction

RE in general means to identify entities and relations among them [14,17]. The following describes the understanding of the RE task used in this paper.

[3] See https://github.com/JulianNeuberger/re-study-caise#search-queries.

*Extraction Scope.* RE distinguishes between *mention-level* and *global* [9,11] techniques. *Mention-level* means relation classification for a given sentence and given entities [11]. *Global* means the prediction of relational facts from plain text [9]. This study focuses on global RE (abbreviated with RE in the remainder).

*Input Scope.* The input scope is either *sentence-*, *document-*, or *corpus-level*, depending on whether the input is a single sentence, multiple sentences, or multiple documents containing multiple sentences [14]. In the latter two cases, relations usually cross sentence boundaries. Our experiment scope is sentence-level (Sect. 3) for mainly two reasons: *(i)* It facilitates the relation extraction task and thus better describes the overall potential for extracting relations in a generalist setting and *(ii)* document- and corpus-level Joint RE are emerging fields. The few documented approaches still produce rather poor results, making it hard to apply them in the information-systems domain.

*Output Scope.* Some approaches can only extract exactly one relation per *sample* (corresponds to a sentence according to our input scope), but this study also considers approaches that allow for multiple relations per sample. The *arity* of each relation is two, which means that only binary relations are considered, therefore all experiments described in Sect. 3 expect output relations to be triples of the form $\langle E_1, R, E_2 \rangle$, with $E_1, E_2$ being two entities, and $R$ being the relation among them. Furthermore, all experiments use datasets with *directed* relations [11], meaning the triple $\langle E_1, R, E_2 \rangle$ denotes a relation $R$ with $E_1$ as source (aka *head*) and $E_2$ as target (aka *tail*), and $\langle E_2, R, E_1 \rangle$ denotes the same relation with opposite direction. Finally, in contrast to *Open Relation Extraction* [10], all relation types are represented in both training and test data in experiments conducted in this paper.

### 2.2 Literature Review

To ensure the reproducibility of this study, it is based on a rigorous literature review procedure following the principles of *PRISMA* [18]. In the first stage a set of search queries[4] serves as a coarse-grained filter for selecting neural relation extraction approaches. In the second stage, the title and abstract of all results found by running these queries are manually reviewed for relevance. Finally, the remaining papers are reviewed for relevance using the full text.

*Stage 1: Retrieval Stage.* To search for relevant RE approaches, Google Scholar and the Scopus database are used. Google Scholar offers the advantage of accessing additional scientific databases (e.g. ScienceDirect, DBLP, ACM Digital Library, ACL Anthology, Springer, IEEE, and arXiv). With duplicates removed, the set of queries mentioned above retrieved *1845* publications potentially relevant to the field of Joint RE. For validation, we check whether the results include all articles previously found by the authors in a manual search (*41* articles). Two additional articles are identified in this way, leading to *1847* results overall.

[4] See https://github.com/JulianNeuberger/re-study-caise#search-queries.

*Stage 2: Filtering by title and abstract.* In this stage, the articles found are matched against several exclusion criteria[5] by analyzing the *title and abstract*. To be considered relevant, an article must not meet any of the following exclusion criteria. The first criterion (EXCL 1) excludes all articles not written in English, while the second criterion (EXCL 2) excludes surveys, comparative studies, or domain-specific applications of existing approaches. EXCL 3 filters out articles proposing an approach that cannot handle English text, since they are not compatible with the application scenarios that led to this study (cf. Section 1). Criterion EXCL 4 excludes all articles that do not propose a deep learning approach. We restrict ourselves to RE approaches that use deep learning for entity and relation extraction, since artificial neural networks usually outperform conventional approaches and are known to have better generalization capabilities [14,31]. EXCL 5 requires that the source code of the techniques is publicly available to enable a fair comparison, as re-implementations based on the descriptions in the article would potentially be error-prone. Another requirement is that entity and relation extraction is jointly trained, i.e. a single model is trained to solve both subtasks [22,27,34] (EXCL 6). One reason is that learning to predict entities and relations simultaneously has been shown to bring synergies to both tasks and increases the extraction capabilities of a model [22]. In addition, it avoids the problem of *error propagation* that occurs when misclassified entities are used as input to a separate relation extraction component, potentially lowering the prediction quality of that component even though it is not primarily responsible for these consequential errors. EXCL 7 excludes approaches that rely on domain-specific knowledge bases such as Freebase or Google Knowledge Graph, since this external knowledge needs to be maintained (Freebase, for instance, is offline) and implies a strong dependency on completeness, correctness and availability (e.g. [28]). Furthermore, it may limit transferability to other domains due to application biases (e.g. Google's Knowledge Graph is used to fill info boxes to summarize search results regarding people, places, etc.). After this stage, 189 articles remain for detailed review by reading their full texts.

*Stage 3: Filtering by reading the full text.* In the final phase, irrelevant articles are excluded by reviewing the full text, reusing the criteria used in stage 2. This step is necessary because in many cases the abstract is too vague to seriously evaluate the exclusion criteria. Finally, we obtain 31 relevant approaches.

### 2.3 Related Work

Currently no existing study fully meets our research scope (Sects. 2.1 and 2.2) but there are still several distantly related surveys and comparative studies.

*Empirical Studies.* [27] analyzes 20 approaches regarding flaws frequently occurring in articles related to RE, including a comparison of different RE approaches. The study described in Sect. 3 avoids aforementioned flaws by using a unified

[5] Criteria list: https://github.com/JulianNeuberger/re-study-caise#filtering.

**Table 1.** Detailed characteristics of the approaches considered. Column *Params* lists the approximate total number of weights, trainable or not. Training durations are given for the smallest dataset (ConLL 04) and largest one (NYT 10). Column *Termination* lists the criterion used to stop training. Column *Input features* lists the inputs a model takes. Here *TOK* means a sequence of tokens, *POS* their part-of-speech tags, *CL* char-level input, and *LM*, *WV* pretrained language models and word vectors respectively.

| Approach | Params | Training dur | Machine | Termination | Input features |
|---|---|---|---|---|---|
| RSAN [34] | ≈8M | 1.5 h–6.0 h | Titan RTX | Epochs | TOK,POS,CL |
| Two [30] | ≈7M | 1.5h–3.5 h | Titan RTX | Ep., Steps | TOK,CL,LM/WV |
| CasRel [31] | ≈100M | 2.0 h–5.0 h | RTX 2080 TI | Epochs | TOK,LM |
| JointER [33] | ≈14M | <1 h–4.5 h | Titan RTX | Epochs | TOK |
| PFN [32] | ≈111M | 4.0 h–6.5 h | Titan RTX | Epochs | TOK,LM |
| SpERT [6] | ≈100M | <1 h–10 h | RTX 2080 TI | Epochs | TOK,LM |
| MARE [12] | ≈350M | 1.0 h–16 h | RTX 2080 TI | Patience | TOK,LM |

experimental setup. In [19], the influence of two main information sources is discussed, namely textual context and entity mentions. Both [19,27] tailor their experiments to answer research questions different from those in our study (Sect. 2). Another study, [26], is exclusively focusing on the biomedical application domain and does not perform a rigorous literature search. Furthermore, the evaluation is limited to three basic neural network architectures and omits, for instance, the established transformer models.

*Literature Surveys.* In contrast to the few empirical studies, there are various literature surveys that compare RE techniques mainly on a conceptual level [3, 13,14,16,17,29,36]. All of those articles include a quantitative comparison based on evaluation scores. However, due to the nature of a survey, the evaluation scores are an excerpt from different external sources, such as the original publication of an approach. Therefore, the experimental setups differ significantly – in line with the flaws identified in [27] – lowering the validity of the quantitative analysis. Moreover, none of the aforementioned surveys conducts a reproducible literature search, which challenges the completeness of the literature considered.

## 3 Comparative Study

Addressing the research question posed in Sect. 2, this study evaluates approaches that are able to solve the Joint RE task (Sect. 2.1) and that are selected in the literature review phase (Sect. 2.2). To be able to inspect the performance of approaches from different angles, three different variants of the measures *F1 score*, *precision*, and *recall* are computed. To also vary the application scenario, the approaches are applied to four datasets with diverse characteristics.

### 3.1 Considered Approaches

As mentioned in Sect. 1, this study focuses on approaches that can be used "out of the box". An approach is considered usable "out of the box", iff *(i)*

its executable source code is publicly available, *(ii)* the default hyperparameters are known, *(iii)* hyperparameter optimization is not mandatory, and *(iv)* neither language model re-training nor vocabulary adaptation is required.

To enable consistent evaluation and practical application, all approaches must export their predictions in a structured format. All of these limiting criteria are necessary for an approach to be widely applied without expert knowledge in RE, which is consistent with the intuition of "out of the box".

According to this definition, 31 approaches were classified as relevant. If possible, authors of those approaches as well as authors of this study adapted the code of approaches to the needs of our experiments (Sect. 3.2). However, even with the help of the respective authors, the code of several approaches could not be adapted for various reasons (e.g. incomplete code, lack of reproducibility, missing hyperparameters). Hence, they are discarded. This results in the group of approaches listed in Table 1. Although the above restrictions keep the focus very narrow, approaches that satisfy them have a high practical applicability in terms of availability and documentation for this very reason.

### 3.2 Experimental Setup

This section provides details on the datasets, their splits, and all the evaluation steps performed.

*Datasets.* All approaches are evaluated on four prelabled datasets: SemEval 2010 (task 8) [11], NYT10 [23], FewRel [10], and Conll04 [24]. We have chosen these datasets due to their diversity with respect to several characteristics, such as, for instance, number of samples, distance of entities that have a relation or size of the tagset (see Table 2). Datasets that require an external knowledge graph are ruled out (see *EXCL 7* Sect. 2.1). The train/dev/test splits are created as follows: For SemEval 2010 (task 8) and NYT10 the original training data is split into 80% training and 20% dev data by uniform sampling. For Conll04, the corpus is divided into 70% training data, 20% dev data, and 10% test data by uniform sampling. In the case of the FewRel dataset, the provided training and test data (train_wiki.json and valid_wiki.json respectively) are first merged and a stratified split based on the relation type is made into 70% training data, 20% dev data, and 10% test data. Merging training and test data is necessary because the selected RE approaches are not able to handle relations that are not existent in the training data. However, in FewRel, training and test data are disjoint with respect to the relation types they contain. Global dataset statistics are listed in Table 2. The distribution of relation types is visualized in Fig. 2.

*Experiment details.* For a unified evaluation, the implementations of the approaches are modified in two places[6]. First, each approach is modified so that any dataset matching the required input format can be passed in. Second, parts

[6] the modified implementations can be found at https://github.com/JulianNeuberger/re-study-caise/#considered-approaches.

**Table 2.** Detailed dataset statistics. *# samples* refers to the number of sentences, while *# instances* refers to the number of relation examples. *max dist* and *avg dist* denote the maximum and average number of tokens between relation arguments, measured from the start token of an argument. Average distance is rounded to two decimal places. The minimal distance is 1 for each dataset.

| dataset | # samples | # instances | # rel | max dist | avg dist | vocab | # tokens | # entities | # tags |
|---|---|---|---|---|---|---|---|---|---|
| NYT10 (train) | 56271 | 70247 | 29 | 128 | 10.98 | 75721 | 2142436 | 116297 | 16 |
| NYT10 (valid) | 14068 | 17492 | 28 | 102 | 11.13 | 37338 | 536268 | 29008 | 15 |
| NYT10 (test) | 4006 | 5859 | 29 | 86 | 11.35 | 18001 | 154274 | 8331 | 13 |
| **Total** | 74345 | 93598 | 29 | 128 | 11.03 | 87014 | 2832978 | 153636 | 17 |
| Semeval (train) | 6400 | 5252 | 9 | 26 | 4.73 | 18735 | 123443 | 12800 | 15 |
| Semeval (valid) | 1600 | 1338 | 9 | 35 | 4.84 | 8052 | 30532 | 3200 | 9 |
| Semeval (test) | 2717 | 2263 | 9 | 20 | 4.86 | 11342 | 52500 | 5434 | 9 |
| **Total** | 10717 | 8853 | 9 | 35 | 4.78 | 25194 | 206475 | 21434 | 16 |
| ConLL 04 (train) | 3861 | 1383 | 5 | 68 | 7.77 | 15218 | 98390 | 9928 | 4 |
| ConLL 04 (valid) | 551 | 216 | 5 | 35 | 7.99 | 4602 | 14300 | 1403 | 4 |
| ConLL 04 (test) | 1104 | 449 | 5 | 82 | 8.29 | 7048 | 27490 | 2846 | 4 |
| **Total** | 5516 | 2048 | 5 | 82 | 7.91 | 18401 | 140180 | 14177 | 4 |
| FewRel (train) | 39120 | 39120 | 80 | 34 | 8.90 | 86020 | 975468 | 78240 | 19 |
| FewRel (valid) | 5600 | 5600 | 80 | 33 | 8.84 | 24427 | 139423 | 11200 | 18 |
| FewRel (test) | 11280 | 11280 | 80 | 33 | 8.88 | 38681 | 281523 | 22560 | 18 |
| **Total** | 56000 | 56000 | 80 | 34 | 8.89 | 107253 | 1396414 | 112000 | 19 |

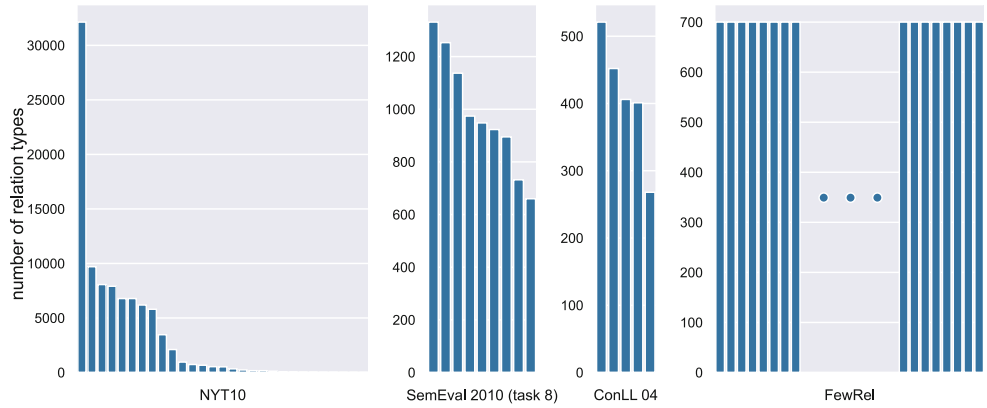



**Fig. 2.** Distribution of relation types per dataset. Visualization of FewRel is abbreviated, as it contains the same number of examples for all relation types.

of the evaluation functionality are modified to output a unified results file that contains all the information needed for scoring with a variety of metrics. Then, this results file is analyzed using an evaluation component implemented by the authors of this study. Moreover, we also corrected serious errors that prevent a fair comparison (e.g. the use of test data for training).

All data is converted into the required format and approach-specific preprocessing is performed if necessary. The model is then trained on the training set and progress is monitored using the dev data. Training is terminated depending on the criteria defined by the approach (Table 1). The study does not perform

hyperparameter optimization, but uses the default parameters specified in the respective approach. Finally, the model is evaluated on the test set. The prediction of the model is stored and later analyzed by the evaluation component. Some approaches limit the input sequence to a maximum size, which is met by filtering out longer sentences. This is treated as an incorrect prediction.

### 3.3 Evaluation Metrics

Before calculating metrics, the predicted relations must be matched with the ground truth relations. In this study, the *exact match* strategy [35] is used, i.e. a predicted triple $(h, r, t)$ is considered to be correct, iff its relation $(r)$ and the boundaries of its head $(h)$ and tail $(t)$ entities are correct. Furthermore, since a sentence might contain an arbitrary number of relations and a model can predict any number of relations, the number of correct predictions $(n_{ok})$ for a sample is determined as the cardinality of the intersection of predictions $(n_{pred})$ and ground truth labels $(n_{gold})$.

Following this, precision $P$ and recall $R$ are defined as $P = n_{ok}/n_{gold}$ and $R = n_{pred}/n_{gold}$. To capture different perspectives on the results of approaches, we use three distinct ways of calculating $n_{ok}$, $n_{pred}$, $n_{gold}$, as well as $P$ and $R$:

1. *micro* metrics are calculated over the entire dataset. It shows how well an approach predicts relations in general, without weighting results by the number of supporting instances of a given relation type. Therefore, its is unsuited for imbalanced datasets, e.g. *NYT10*.
2. $macro_{rel}$ metrics are calculated for each *relation type* separately and averaged. This allows insight into an approach's ability to predict rare relation types correctly, complementing the *micro* scores and their shortcomings.
3. $macro_{doc}$ metrics are calculated for each *document* separately and averaged. Both *micro*, as well as $macro_{rel}$ fail to score the ability of approaches to correctly identify documents with no relation, as they score the entire dataset at once, which $macro_{doc}$ counter-acts.

### 3.4 Results

Table 3 shows the scores for each approach and dataset, which are additionally visualized in Fig. 3. In general, approaches struggle with the large number of unique relation types present in *FewRel* which is evident by low F1 scores. *MARE* and *SpERT* score noticeably higher, reaching 47.21% and 40.36% $F1_{macro_{rel}}$ respectively. Some approaches, like *RSAN* score best in regards to recall (53.14%), identifying the presence of a relation, but fail to properly predict the relation type, resulting in a low precision (9.19%) and overall F1 score (15.67%). Similarly, the long-tail phenomenon, i.e. a large number of relation types with very few examples challenges all approaches, apparent by comparing their high *micro* and low $macro_{rel}$ scores. *CasRel* performs best ($F1 = 40.20\%$) using $macro_{rel}$ and 71.71% using $micro_{rel}$. *Two* ($F1_{macro_{rel}} = 14.72\%$) and *RSAN* ($F1_{macro_{rel}} = 23.29\%$) perform worst, most likely resulting from imbalanced data.

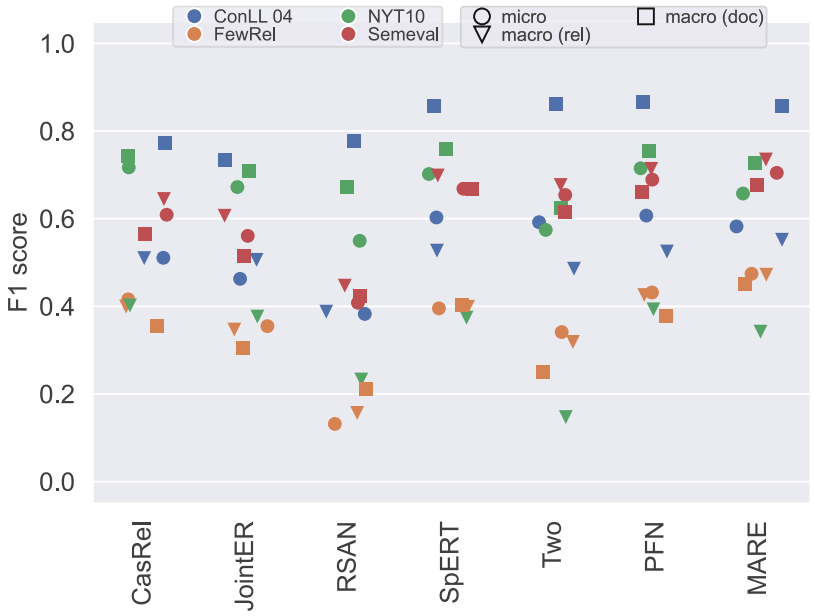



**Fig. 3.** Overview of the test scores for each considered approach. F1 scores are grouped by the averaging strategy used, cf. Section 3.3.

The $macro_{doc}$ scores of approaches on the *ConLL 04* dataset show that all of them are able to model samples, where no relation is present. *PFN* ($F1 = 86.56\%$) is closely followed by *MARE* (85.83%) and *SpERT* (85.82%). Comparing the $macro_{doc}$ scores to *micro* scores reveals its usefulness, as samples containing no relation would otherwise not be scored, resulting in worse perceived performance, e.g. for *PFN* scoring $F1_{micro} = 60.71\%$.

*Semeval* contains a large number of NER tags compared to its size, resulting in high linguistic variability of relation instances. We discuss this characteristic in more detail in Sect. 3.5. High variability presents a challenge to *RSAN*, which only scores a $F1_{micro}$ score of 42.37%. All other approaches are able to cope, with *MARE* performing best (70.47%).

### 3.5 Detailed Analysis

This section analyzes effects of data characteristics on model performance. We selected those by fitting a decision tree to rows of data, consisting of several characteristics present in the dataset and the correctness of the model's prediction results. This allows to only select characteristics with the highest impact.

*Distance between relation arguments.* The number of tokens between a relation's arguments is a hurdle to most approaches, as they have to reason across longer distances. This effect is visualized in Fig. 4. We binned samples of each dataset separately into 15 bins according to the maximum distance of any relation it contains. For each bin we calculated the F1 metrics discussed in Sect. 3.4 for the results of each approach. With the exception of *Two*, which maintains a fairly

**Table 3.** Detailed results of all experiments on the test sets of each dataset. Each variant of F1 score, precision (P) and recall (R) is reported as defined in Sect. 3.3

| | | $macro_{doc}$ | | | $macro_{rel}$ | | | $micro$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | dataset | F1 | P | R | F1 | P | R | F1 | P | R |
| CasRel | ConLL 04 | 77.29% | 76.81% | 77.78% | 51.00% | 43.09% | 62.47% | 51.07% | 44.71% | 59.55% |
| | FewRel | 35.46% | 33.99% | 37.06% | 39.96% | 43.35% | 37.06% | 41.59% | 47.38% | 37.06% |
| | NYT10 | 74.23% | 76.82% | 71.80% | **40.20**% | 34.60% | 47.95% | **71.71**% | 76.59% | 67.42% |
| | Semeval | 56.48% | 56.15% | 56.83% | 64.49% | 71.15% | 58.96% | 60.90% | 62.98% | 58.95% |
| JointER | ConLL 04 | 73.46% | 73.36% | 73.55% | 50.60% | 44.98% | 57.84% | 46.27% | 40.30% | 54.32% |
| | FewRel | 30.48% | 30.18% | 30.78% | 34.69% | 39.75% | 30.78% | 35.48% | 41.87% | 30.78% |
| | NYT10 | 70.81% | 74.03% | 67.86% | 37.66% | 33.22% | 43.45% | 67.22% | 75.19% | 60.78% |
| | Semeval | 51.57% | 50.99% | 52.15% | 60.67% | 64.46% | 57.31% | 56.08% | 54.73% | 57.49% |
| MARE | ConLL 04 | 85.83% | **87.00**% | 84.69% | **55.19**% | **58.25**% | 52.43% | 58.23% | **71.29**% | 49.22% |
| | FewRel | **45.05**% | **45.03**% | **45.07**% | **47.21**% | **49.33**% | 45.27% | **47.43**% | 50.04% | 45.07% |
| | NYT10 | 72.67% | **78.97**% | 67.31% | 34.22% | **35.67**% | 32.88% | 65.74% | **83.59**% | 54.17% |
| | Semeval | **67.78**% | **67.65**% | **67.91**% | **73.51**% | **78.60**% | **69.03**% | **70.47**% | 71.88% | **69.11**% |
| PFN | ConLL 04 | **86.56**% | 86.22% | **86.91**% | 52.51% | 46.35% | 60.55% | **60.71**% | 64.34% | 57.46% |
| | FewRel | 37.83% | 36.94% | 38.76% | 42.58% | 46.87% | 39.00% | 43.16% | 48.69% | 38.76% |
| | NYT10 | 75.36% | 77.13% | 73.67% | 39.31% | 33.04% | 48.53% | 71.50% | 74.44% | 68.78% |
| | Semeval | 66.07% | 65.75% | 66.40% | 71.37% | 77.65% | 66.03% | 68.91% | 71.66% | 66.37% |
| RSAN | ConLL 04 | 77.80% | 77.70% | 77.91% | 38.78% | 36.43% | 41.46% | 38.27% | 38.86% | 37.69% |
| | FewRel | 21.15% | 13.20% | 53.14% | 15.67% | 9.19% | **53.14**% | 13.16% | 7.51% | **53.14**% |
| | NYT10 | 67.34% | 58.40% | **79.51**% | 23.29% | 15.15% | **50.39**% | 54.95% | 43.27% | **75.27**% |
| | Semeval | 42.37% | 39.63% | 45.53% | 44.71% | 40.85% | 49.38% | 40.82% | 34.58% | 49.80% |
| SpERT | ConLL 04 | 85.82% | 85.59% | 86.06% | 52.71% | 46.54% | 60.75% | 60.28% | 63.39% | 57.46% |
| | FewRel | 40.36% | 38.36% | 42.58% | 39.88% | 37.51% | 42.58% | 39.53% | 36.90% | 42.58% |
| | NYT10 | **75.88**% | 76.90% | 74.89% | 37.41% | 30.17% | 49.22% | 70.19% | 70.21% | 70.18% |
| | Semeval | 66.87% | 65.94% | 67.83% | 69.86% | 71.19% | 68.57% | 66.84% | 64.94% | 68.85% |
| Two | ConLL 04 | 86.26% | 85.73% | 86.81% | 48.58% | 38.48% | **65.87**% | 59.20% | 56.34% | **62.36**% |
| | FewRel | 25.03% | 24.54% | 25.53% | 31.87% | 42.39% | 25.53% | 34.12% | **51.42**% | 25.53% |
| | NYT10 | 62.33% | 65.61% | 59.37% | 14.72% | 15.44% | 14.07% | 57.44% | 74.12% | 46.89% |
| | Semeval | 61.53% | 61.24% | 61.83% | 67.68% | 76.94% | 60.41% | 65.41% | 70.84% | 60.76% |

consistent performance, all approaches yield significantly lower performance on relations exhibiting longer distance between arguments.

*Sample length.* Samples with more tokens usually are usually harder to model for mainly two reasons: Firstly more tokens mean more (possible) entities that have to be marked and can participate in relations, secondly longer samples usually exhibit relations where arguments are further apart. For each dataset samples are binned into 15 bins according to the number of tokens they contain. We then calculate the metrics for each bin in the same way as for the *distance between relation arguments*. For the *NYT10* dataset a visualization of the effects of this characteristic can be seen in Fig. 4. In general approaches struggle with longer samples, with the exception of *Two*, which maintains a fairly consistent performance. Results in general are very similar to the *distance between relation*

*arguments* for the aforementioned dependency between a sample's length and the distance between relation arguments it contains.

*Relations per sample.* If a dataset contains samples with multiple relations it can be harder to model. This effect is founded in the fact that additionally to classifying relations correctly, the number of relations has to be predicted as well. We binned samples of each dataset according to the number of relations it contains. Datasets *SemEval* and *FewRel* contain only samples with exactly one relation, so we exempt them from this analysis. We then calculate the metrics for each bin in the same way as for the *distance between relation arguments*. In our experiments the number of relations per sample is detrimental to performance, though much less pronounced compared to effects discussed previously.

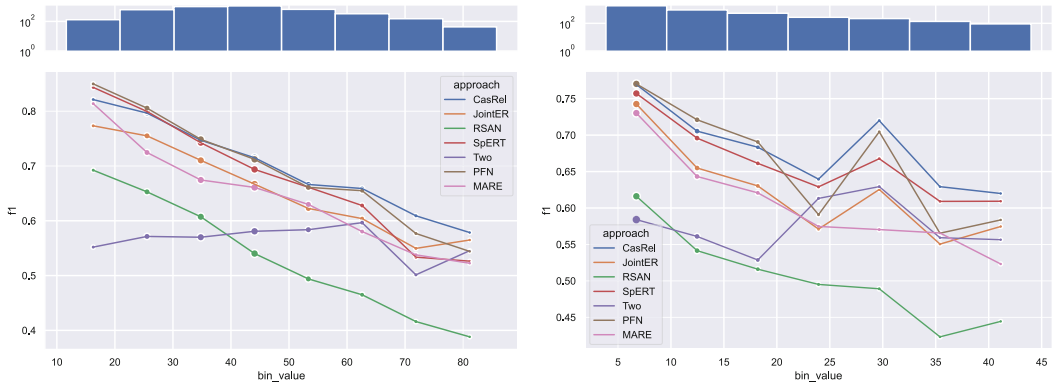



**Fig. 4.** Micro F1 scores of approaches on NYT10 data in relation to distance between relation participants (*left*) and sample length (*right*) in tokens. Samples were sorted into 15 bins, bins containing less than 25 samples are not drawn, as they suffer from the law of small numbers. Marginal log bar plot shows the number of samples in that bin.

*Variability of relation arguments.* It is easy to see that classifying relations containing arguments with little to no variation in constituent tokens is less of a challenge than the other way around. To investigate the effect of this observation on the performance of approaches, we calculated the linguistic variability of relation arguments (i.e. head and tail) by dividing the number of unique tokens by the number of total tokens used in all relation arguments for a specific relation type. This will result in 1.0 at best and $1/\#tokens$ at worst, which we then scale to the range $[0.0, 1.0]$. Using linear regression we calculate the correlation between this linguistic variability factor and $F1_{micro}$. Our null hypothesis is that there is no correlation between linguistic variability and performance, which we reject if $p$ is less than our significance threshold of 5%. All approaches exhibit a statistically significant negative correlation between linguistic variability and performance, with the exception of *Two*, which shows a statistically non-significant weak negative correlation, meaning it is not sure there is a correlation at all. *MARE* has the weakest statistically significant correlation, which

leads us to believe it can handle linguistic variability in relation arguments best. *JointER* has the strongest negative, statistically significant correlation, implying it is worst suited for this specific data characteristic. A visual comparision between the two can be found in Fig. 5. Analysis results for all approaches can be found in Table 4.

**Table 4.** Pearson correlation coefficient ($r$) and statistic significance ($p$). While *Two* exhibits the weakest negative correlation between linguistic variability and performance, we have to reject the null hypothesis, meaning we have to assume there is no correlation. All other approaches feature a clear correlation between linguistic variability and performance. *MARE* is able to cope best, while *JointER* struggles the most. The correlation for those extremes is visualized in Fig. 5.

| Approach | CasRel | JointER | MARE | PFN | RSAN | SpERT | Two |
|---|---|---|---|---|---|---|---|
| r | −0.5496 | −0.6044 | **-0.3117** | −0.4822 | −0.5779 | −0.5242 | −0.1127 |
| strength | strong | strong | **medium** | strong | strong | strong | weak |
| p | ∼ 0.00% | ∼ 0.00% | 0.05% | ∼ 0.00% | ∼ 0.00% | ∼ 0.00% | 22.03% |
| significant | yes | yes | yes | yes | yes | yes | no |

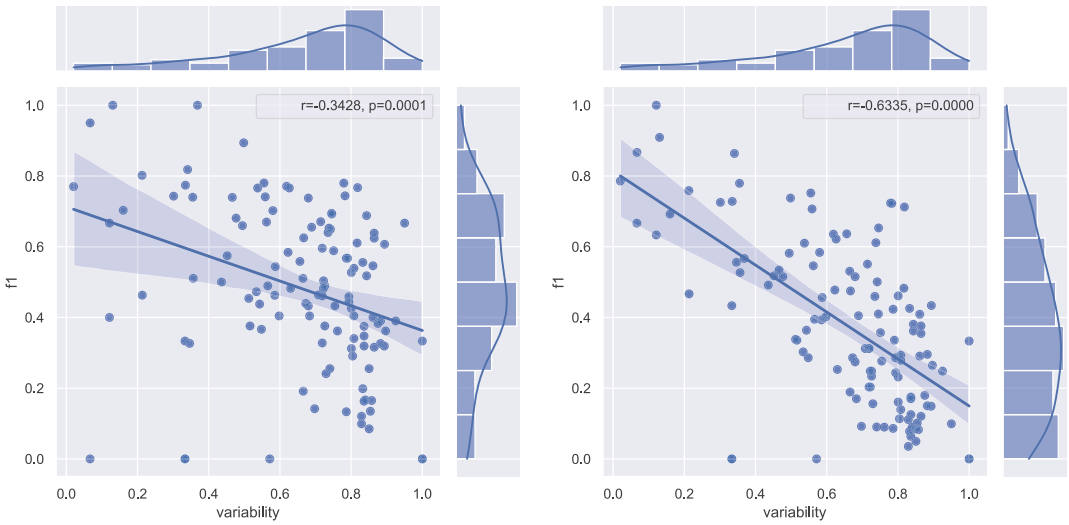



**Fig. 5.** Micro F1 scores of approaches on all datasets in relation to the linguistic variability present in a sample for *MARE* (*left*) and *JointER* (*right*). 1.0 variability corresponds to a sample consisting of unique tokens only, while 0.0 is a sample with only a single unique token. *MARE* is the approach best suited for coping with strongly varying phrasing of relation arguments, while the performance of *JointER* suffered the most. Detailed results are listed in Table 4.

## 4 Conclusion and Future Work

Irreproducible literature search and differing experimental setups invalidate the quantitative comparison of RE approaches directly from literature. From an application perspective, however, there is a strong need to identify approaches that work for a wide range of different domains without making major adjustments. In this paper a rigorous, reproducible literature search is conducted to identify all relevant RE approaches and an empirical comparative study with an unified experimental setup is performed to identify a group of jointly trained neural RE approaches that outperform their competitors.

*MARE* is suitable for most studied datasets, performing best in 21 out of 36 categories and competitively in nearly all other. As such it seems to be applicable to most situations, except datasets with a large number of under-represented relations. Here *CasRel*, *PFN*, and *SpERT* are better choices. *Two* and *RSAN* can be feasible, if the dataset is known to be well balanced. Finally, regarding the research question given in Sect. 2, there are considerations to be made depending on the dataset on hand. Nonetheless, *MARE*, *CasRel*, *PFN*, and *SpERT* form a promising baseline for being fine-tuned to concrete applications.

Currently this study has a few limitations. First, it focuses on neural Joint RE approaches only. Thus, the authors plan a similar unified evaluation of both jointly and separately trained RE approaches. Which approaches are considered in this study largely depends on the selectivity of the search queries used. This potentially causes the systematic literature review to rule out papers that are actually relevant. However, the advantage is that the literature review can be easily extended and thus, missed articles can be added manually. A limitation to the experiments is that datasets without NER tags are artificially enriched using the Stanza Parser, which leads to coarsely labeled entities in some datasets, e.g. SemEval with examples like *"Flowers are carried into the chapel."* and corresponding relation *Entity-Destination*. Thus, automated NER tagging of datasets is potentially error-prone. Another open question is, whether RE approaches could benefit from data augmentation that is used to cope with data quality and quantity issues. Finally, hyperparameters of models have been selected as they were recommended by their authors. This allows us to consider more approaches on multiple datasets, instead of optimizing only a few. Still, this neglects the possibility of achieving a performance boost through a general hyperparameter optimization framework. Due to the long runtime, the authors preceded the hyperparameter optimization with the present study to achieve a smaller preselection of promising approaches.

## References

1. van der Aa, H., Carmona, J., Leopold, H., Mendling, J., Padró, L.: Challenges and opportunities of applying natural language processing in business process management. In: Proceedings of COLING. ACL (2018)
2. Ackermann, L., Neuberger, J., Jablonski, S.: Data-driven annotation of textual process descriptions based on formal meaning representations. In: CAiSE. Springer International Publishing (2021)

3. Alshuwaier, F., Areshey, A., Poon, J.: A comparative study of the current technologies and approaches of relation extraction in biomedical literature using text mining. In: ICETAS (2017)
4. Btoush, E.S., Hammad, M.M.: Generating er diagrams from requirement specifications based on natural language processing. In: IJDTA (2015)
5. Dawood, O.S., et al.: From requirements engineering to uml using natural language processing-survey study. In: EJERS (2017)
6. Eberts, M., Ulges, A.: An end-to-end model for entity-level relation extraction using multi-instance learning. In: EACL ACL (2021)
7. Elallaoui, M., Nafil, K., Touahni, R.: Automatic transformation of user stories into uml use case diagrams using nlp techniques. Procedia Computer Science (2018)
8. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: CAiSE. Springer, Berlin Heidelberg (2011)
9. Han, X., Yu, P., Liu, Z., Sun, M., Li, P.: Hierarchical relation extraction with coarse-to-fine grained attention. In: EMNLP. ACL (2018)
10. Han, X., et al.: FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In: EMNLP. ACL (2018)
11. Hendrickx, I., et al.: SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In: SemEval. ACL (2010)
12. Klöser, L., Kohl, P., Kraft, B., Zündorf, A.: Multi-attribute relation extraction (MARE): simplifying the application of relation extraction. In: DeLTA. SCITEPRESS (2021)
13. Kumar, S.: A survey of deep learning methods for relation extraction. CoRR (2017)
14. Liu, K.: A survey on neural relation extraction. Sci. China Technol, Sciences (2020)
15. López, H.A., Debois, S., Hildebrandt, T.T., Marquard, M.: The process highlighter: From texts to declarative processes and back. In: CEUR Workshop Proc. (2018)
16. Nasar, Z., Jaffry, S.W., Malik, M.K.: Named entity recognition and relation extraction: State-of-the-art. ACM Comput, Surv (2021)
17. Nayak, T., Majumder, N., Goyal, P., Poria, S.: Deep neural approaches to relation triplets extraction: a comprehensive survey. Cogn, Comput (2021)
18. Page, M.J., et al.: The prisma 2020 statement: an updated guideline for reporting systematic reviews. In: Systematic reviews (2021)
19. Peng, H., et al.: Learning from Context or Names? An Empirical Study on Neural Relation Extraction. In: EMNLP. ACL (2020)
20. Qian, K., et al.: Annotation inconsistency and entity bias in MultiWOZ. In: Proceedings of SIGdial. ACL (2021)
21. Quishpi, L., Carmona, J., Padró, L.: Extracting annotations from textual descriptions of processes. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 184–201. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_11
22. Ren, F., et al.: A novel global feature-oriented relational triple extraction model based on table filling. In: EMNLP. ACL (2021)
23. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Proceedings of ECML PKDD. Springer, Berlin Heidelberg (2010)
24. Roth, D., Yih, W.t.: A linear programming formulation for global inference in natural language tasks. In: CoNLL. ACL (2004)
25. Salih Dawood, O., Sahraoui, A.E.K.: From requirements engineering to uml using natural language processing - survey study. EJIE (2017)
26. Saranya, M., Geetha, T.V., Annie, R.A.X.: Comparative analysis of different deep learning techniques for relation extraction from biomedıcal literature. In: Proceedings of ICSADL. Springer Singapore (2022)

27. Taillé, B., Guigue, V., Scoutheeten, G., Gallinari, P.: Let's Stop Incorrect Comparisons in End-to-end Relation Extraction! In: EMNLP. ACL (2020)
28. Vashishth, S., Joshi, R., Prayaga, S.S., Bhattacharyya, C., Talukdar, P.: RESIDE: Improving distantly-supervised neural relation extraction using side information. In: EMNLP. ACL (2018)
29. Wang, H., Qin, K., Zakari, R.Y., Lu, G., Yin, J.: Deep neural network-based relation extraction: an overview. Neural Comput, Appl (2022)
30. Wang, J., Lu, W.: Two are better than one: Joint entity and relation extraction with table-sequence encoders. In: EMNLP. ACL (2020)
31. Wei, Z., Su, J., Wang, Y., Tian, Y., Chang, Y.: A novel cascade binary tagging framework for relational triple extraction. In: ACL (2020)
32. Yan, Z., Zhang, C., Fu, J., Zhang, Q., Wei, Z.: A partition filter network for joint entity and relation extraction. In: EMNLP. ACL (2021)
33. Yu, B., et al.: Joint extraction of entities and relations based on a novel decomposition strategy. In: Proceedings of ECAI (2020)
34. Yuan, Y., Zhou, X., Pan, S., Zhu, Q., Song, Z., Guo, L.: A relation-specific attention network for joint entity and relation extraction. In: IJCAI (2020)
35. Zeng, D., Zhang, H., Liu, Q.: Copymtl: Copy mechanism for joint extraction of entities and relations with multi-task learning. ArXiv (2020)
36. Zhang, X., Dai, Y., Jiang, T.: A survey deep learning based relation extraction. J. Phys.: Conf. Series **1601**, 032029 (2020)

# Chapter 7

# Beyond rule-based named entity recognition and relation extraction for process model generation from natural language text

Neuberger, J., Ackermann, L., Jablonski, S. (2024). Beyond Rule-Based Named Entity Recognition and Relation Extraction for Process Model Generation from Natural Language Text. In: Sellami, M., Vidal, ME., van Dongen, B., Gaaloul, W., Panetto, H. (eds) *Cooperative Information Systems. CoopIS 2023.* Lecture Notes in Computer Science, vol 14353. Springer, Cham. https://doi.org/10.1007/978-3-031-46846-9_10



**Contribution statement.** Concept for the relation extraction and entity resolution methods were developed and implemented by Julian Neuberger (JN). Entity resolution data for the PET dataset was annotated by JN and Lars Ackermann (LA). JN planned and conducted the evaluation of all methods. Results were interpreted by JN. All figures were created by JN. JN and LA wrote the main text of the publication, with additional contributions by Stefan Jablonski (SJ). Revision of the publication was done by all authors. JN is the corresponding author.

# Beyond Rule-Based Named Entity Recognition and Relation Extraction for Process Model Generation from Natural Language Text

Julian Neuberger(✉), Lars Ackermann, and Stefan Jablonski

University of Bayreuth, Bayreuth, Germany
{julian.neuberger,lars.ackermann,stefan.jablonski}@uni-bayreuth.de

**Abstract.** Process-aware information systems offer extensive advantages to companies, facilitating planning, operations, and optimization of day-to-day business activities. However, the time-consuming but required step of designing formal business process models often hampers the potential of these systems. To overcome this challenge, automated generation of business process models from natural language text has emerged as a promising approach to expedite this step. Generally two crucial subtasks have to be solved: extracting process-relevant information from natural language and creating the actual model. Approaches towards the first subtask are rule based methods, highly optimized for specific domains, but hard to adapt to related applications. To solve this issue, we present an extension to an existing pipeline, to make it entirely data driven. We demonstrate the competitiveness of our improved pipeline, which not only eliminates the substantial overhead associated with feature engineering and rule definition, but also enables adaptation to different datasets, entity and relation types, and new domains. Additionally, the largest available dataset (PET) for the first subtask, contains no information about linguistic references between mentions of entities in the process description. Yet, the resolution of these mentions into a single visual element is essential for high quality process models. We propose an extension to the PET dataset that incorporates information about linguistic references and a corresponding method for resolving them. Finally, we provide a detailed analysis of the inherent challenges in the dataset at hand.

**Keywords:** Process-aware Information Systems · Process Extraction · Named Entity Recognition · Relation Extraction · Co-Reference Resolution

## 1 Introduction

Automated generation of formal business process models from natural language process descriptions has become increasingly popular [1,2,7,10,19]. This is motivated, for instance, with the comparatively high time expenditure for manually


**Fig. 1.** Example for differences between information extraction phase with and without resolving process element identities. Resolving process element identity from their mentions (right) allows generation of correct data flow, without (left) data flow is disjointed.

designing said process models. Up to 60% of the total duration in process management projects is spent on the design of process models [10]. Techniques for automated process model generation from natural language text aim to reduce this effort, but have to solve several sub-tasks for this, categorized into two distinct phases: *(i) The information extraction phase* and *(ii) the process model generation phase*. During the information extraction phase, techniques recognize process elements (e.g., activities, actors, data objects), extract relations (e.g., sequence-flow relations between activities), and resolve references (e.g., mentions of the same data object). Building on this information, the process model generation phase creates a concrete process model [10,13,19]. The current state of the art for the information extracting phase exhibits two core issues, which we will briefly discuss in the following.

**Core Issue 1.** Existing approaches are largely rule-based, i.e., approaches use manually crafted rules rooted in domain knowledge [2,10,21]. Rule-based systems usually show remarkable precision and recall for the datasets they are created for. However, they *a)* require significant amounts of labor to capture linguistic subtleties, *b)* require deep technical knowledge, as well as knowledge of the target domain, and *c)* are hard to adapt to even minor changes in the underlying data, which leads to unacceptable expansion in the number of required rules [26]. Using machine learning, these drawbacks can be resolved, especially deep learning methods have been shown to greatly reduce the amount of effort and domain knowledge required [15]. However, deep learning methods usually need considerable amounts of data for stable training [14], something the field of business process modeling research currently can not provide [12]. Using less expressive machine learning models constitute a middle ground to this dilemma, as they can be trained stably with orders of magnitude less data.

**Core Issue 2.** Existing approaches are scoped too narrowly [18]. This includes systems, that do not capture enough information for the generation of complete process models, as well as systems that impose unrealistic assumptions concerning the structure of input text. For high-quality process models, resolving references between mentions of the same process element is crucial. Consider, for instance, the example depicted in Fig. 1. For a human reader it is obvious, that both "*a claim*" and "*it*" refer to the same instance of a *claim*. To automatically extract a process model encoding this knowledge the system needs to resolve

the two mentions "*a claim*" and "*it*" to a single entity. Without this step, at least two problems manifest in the extracted process models: *(i)* Two distinct data objects for *claim* would be created and, thus, the model is not able to correctly express that both the registration and the examination activities process the same data, and *(ii)* one of the created data objects is labeled *it*, because it is unknown that *it* is a reference to *a claim*. Though the *claim* example is solely focusing on the data perspective, entity resolution is also necessary for organizational process elements like, for instance, actors. Here, it is necessary to be able to create process models that contain a single actor type for the two mentions of the *claim officer*, which is expressed as a single swimlane in BPMN, for instance. This issue is rooted in a lack of data. Most notably, the currently largest dataset for the information extraction phase (PET [7]) does not include information about linguistic references between mentions of process elements. In summary, it can be said that entity resolution is what makes it possible in the first place to correctly express relations to data and to actors. Following from these two core issues we state three main research questions.

**RQ1.** Are deep learning methods able to extract process information with precision and recall comparable to rule-based methods given the same dataset?
**RQ2.** If deep learning methods prove inadequate for small datasets, such as PET, can classical machine learning models (e.g., gradient boosting techniques) compete with rule-based methods in terms of precision and recall?
**RQ3.** Can a pre-trained co-reference resolution approach outperform naïve word matching, and can therefore be used as a baseline for resolving linguistic references between process element mentions?

Our work proposes an improved pipeline which tackles both of these issues, which we describe in detail in Sect. 4. We propose a relation extraction approach based on established machine learning methods, while adopting the approach to extracting process elements, as it is based on machine learning already. Future work could investigate other methods and compare them, but this is out of scope for this paper. Additionally, we extend PET with information about the identity of process element mentions, and provide a baseline approach for resolving process element identities from process element mentions. We compare our pipeline to the current state of the art of information extraction on PET and show that we outperform it in five out of six relation types, with an absolute increase of 6% in $F_1$ scores.

The remainder of this paper is structured as follows: In Sect. 2 we formalize the task of process model extraction. In Sect. 3 we discuss differences to work related to this paper. Our thorough investigation of the PET dataset and the extraction approaches in Sect. 6 is based on a rigorous experiment setup introduced in Sect. 5. Short summaries of the answers to our research questions are provided in Sect. 7. Both the source code for our experiments and the extended dataset are publicly available[1], therefore laying the foundation for further focused research.

[1] see https://github.com/JulianNeuberger/pet-baselines.

## 2 Task Description

Natural language processing (*NLP*) is a discipline that aims to exploit natural language input data and spans a wide variety of subfields. One of these subfields is Information Extraction from human-readable texts. In the following, we describe the extraction of process elements and of relations between them as instances of three sub-problems of information extraction, which are *Named Entity Recognition* (NER), *Relation Extraction* (RE), and *Entity Resolution* (ER). We then detail the three subproblems with respect to the extended PET dataset as described in Sect. 5.1. Each task assumes that the input text has already been pre-processed, i.e., *tokenized*. Refer to Fig. 2 for visual examples of input and output of the steps described below.

**Named Entity Recognition (NER).** NER is the task of extracting spans of tokens corresponding to exactly one element from a set of entities [15]. While NER traditionally only considered extraction of proper nouns, the definition now depends on the domain [23]. For the process domain named entities are process relevant facts, such as actors (e.g., *the CEO* vs. *Max*) or activities (e.g., *approve* vs. *the approval*). The PET dataset defines a set of seven process relevant facts, providing a general schema for process model generation from natural language text [6]. Formally the NER task is extracting a set of triples $M$ from a given list of tokens $T$, so that for each triple $m = (i_s, i_e, t_e) \in M$, the indices $i_s$ and $i_e$ denote start and end tokens of the span in $T$ respectively, and $t_e$ refers to the entity type. Throughout this paper, we will refer to the triple $m$ a *mention* of an *entity*. An extracted mention is considered correct, iff its triple has an exact match in the list of ground truth triples given by the dataset.

**Entity Resolution (ER).** During ER techniques extract a set of unique entities from a given set of mentions $M$. This step can be seen as resolving references between mentions of the same process element, which is crucial information for generating useful business process models further down-stream, as shown in Fig. 1. Formally the ER task is defined as finding a set of non-empty *mention clusters* $E$, so that each mention $m \in M$ is assigned to exactly one cluster $e \in E$. These clusters are called *entities*. To disambiguate between the use of entity as in NER, and entity as used in ER, we will call the result of NER *mentions* from now on, and the result of ER *entities*. An entity prediction is considered correct, iff the set of contained mentions is exactly the same as the ground truth defined by the dataset. Entity resolution itself is a super-set of the tasks *Anaphora Resolution*, i.e., back-referencing pronouns, *Coreference Resolution*, i.e., use of synonyms, and *Cataphora Resolution*, i.e., forward-referencing pronouns [24]. While there are subtle differences and overlap between these sub-fields, this work focuses on coreference resolution. The addition of cataphora and anaphora resolution is potentially useful, but is out of scope for our planned baseline. Thus, we refer to coreference resolution, whenever we mention the ER task in later sections. The PET dataset only contains two entity types, where entity identity is relevant: *Actors*, describing a natural person, department, organization, or artificial agent, and *Activity Data*, which are objects or data used by an *Activity* [6]. Further details can be found in Sect. 5.1.

**Relation Extraction (RE).** RE is the task of identifying a set of semantic relations $R$ between pairs of entities. Current literature distinguishes between global and mention level RE [16]. Global RE is the task of extracting a list of entity pairs forming a certain relation from a text, without any additional information. On the other hand, mention level RE methods are given a pair of entity mentions and the sentence containing them, and have to predict the relation between the two. The PET dataset contains relation information on mention level, which allows our approach to learn on local level. There are six relation types defined in the PET dataset, such as *Flow*, which captures the execution order between behavioural elements [6]. Each relation is formally defined by a triple $r = (m_h, m_t, t_r)$, where $m_h$ is the head entity mention or source of the relation, $m_t$ the tail entity mention or target, and $t_r$ the type of the semantic relation. This definition implies relations are directed, that is $(m_h, m_t, t_r) \neq (m_t, m_h, t_r)$ for $m_h \neq m_t$. A predicted relation tuple $r \in R$ is considered correct, iff its triple has an exact match in the list of ground truth triples given by the dataset.

## 3 Related Work

[7] presents the currently largest collection of natural language business processes descriptions with annotations for process relevant facts. They also propose a pipeline for extracting said process relevant facts. The paper at hand is founded on their work and is therefore closely related, as we use, extend, and analyze both data and pipeline. We adopt the approach to NER, and compare our proposed RE approach to their method. They are missing a more in-depth description of their data, especially regarding qualities important for prediction performance, including but not limited to: correlation between a relation's type and its argument types, or the amount of variation in language of their data. Furthermore, the implementation of their pipeline is not publicly available, impeding further research and development.

There are several approaches related to the baselines we present and analyze in this work. An annotation approach based on rule-based pattern matching across the dependency tree representation of a textual process description is presented in [21], which is then used to generate an event log. This allows the extraction of a formal process model via established process mining techniques. While it achieves state-of-the-art results, it uses a tagging schema different from the one used in PET, which makes it unfeasible for use in a direct comparison. [10] presents a pipeline able to extract formal process models in Business Process Model and Notation (BPMN), and therefore is locked into this process notation language. The same limitation holds for the approach presented in [2], which extracts process models utilizing the Declare language. PET follows a different tagging scheme and, thus, a direct comparison is not possible. In [18] a neural method for entity and relation classification is proposed, but assumes that relevant text fragments are already extracted. This is a significantly easier task, since separating relevant process information from redundant, superfluous, and incidental information, appearing in natural language, is a hard task in itself. [3]

presents an efficient deep learning method using formal meaning representations as an intermediary feature. Since they only solve NER, we can not compare their approach with our proposals.

Due to the strong relation between process extraction and the combined NLP task of NER, ER, and RE, there are several approaches potentially able to solve the process extraction task [8,9,11,22]. [4] studies several approaches built for joint NER, ER, and RE on small documents. Applying them to the BPM domain entails fragmenting the larger documents of PET properly, as well as dealing with long distance relations, which is out of scope for this paper. However, we chose Jerex [9], since [8] and [11] predict mentions as their textual representation (*surface forms*) only, meaning the span of text containing them might be ambiguous, and therefore token indices not resolvable. This violates our definition of mentions (Sect. 2) and hampers the evaluation of the predictions.

Using pre-trained large language models seems promising for the task at hand, as shown in [5], which uses in-context learning with GPT3 to extract process relevant facts. These are limited to a small subset of the information extracted in [7] and the paper at hand, though. Furthermore their evaluation only uses a portion of the PET-dataset, i.e., 9 out of a total of 45 documents, without repeated runs, e.g., k-fold cross validation. A direct comparison is therefore impossible.

## 4 Process Information Extraction Approach

In the following we present a short overview of the implementation for the three pipeline steps for *NER*, *ER*, and *RE*. The entire pipeline as we propose it is depicted in Fig. 2. We will refer to this pipeline implementation as *Ours* from now on. We do not detail preprocessing steps, nor the actual generation of a business process model, as both are out of scope for this paper. Our approach extracts the text and location of mentions of process elements (NER), resolves those mentions to unique sets of entities (ER), and extracts the (entity) arguments and type of relations (RE). The types of extracted entities and relations are identical with [7], refer to [6] for an in-depth description.

**The NER step** is identical to the implementation from [7]. The approach is based on Conditional Random Fields (*CRF*), a powerful technique for tagging a sequence of observations, here tokens in a text [25]. Given a sequence of tokens tagged in this way, we then resolve mentions, where each mention contains a set of token indices and the predicted process element type. We did not change or optimize this step, as the main focus of this paper is extracting process relevant facts with machine learning methods, and CRF methods are known to be a strong choice for tagging.

We implemented two modules for **the ER step**, namely a *naive ER method*, and a method based on *pre-trained end-to-end neural coreference resolution*, as described in [13] and implemented in spaCy[2]. The naive ER method, which

[2] See https://explosion.ai/blog/coref for more details.

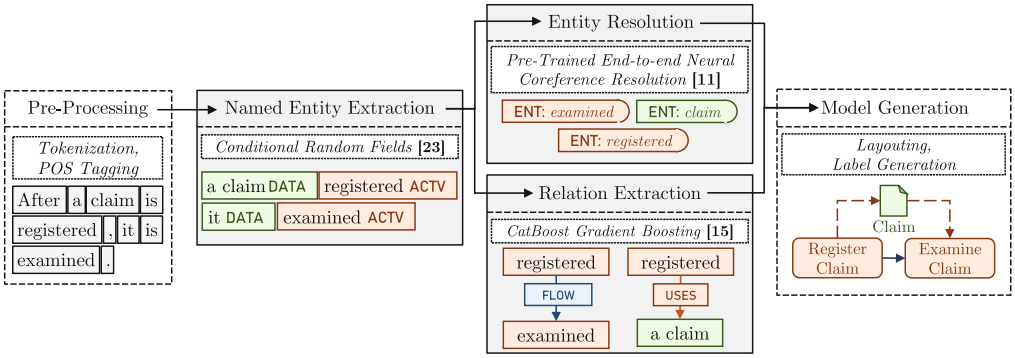



**Fig. 2.** Outline of our proposed extended extraction pipeline.

we will call *naive ER* for short, iteratively selects the best matching mentions with identical NER tags. The match of two mentions is calculated based on the percentage of *overlapping*, i.e., the fraction of shared tokens over the total number of tokens. Ranking mention pairs by this score, the naive ER method merges mentions into clusters. If one of the selected mentions already is part of a cluster, the other mention is added to that cluster as well. If both selected mentions are part of a cluster, the clusters are merged. This is repeated until there are only matches left, which overlap less than some threshold $o$. We ran an optimization to select this overlap optimally and chose $o = 0.5$ The pre-trained end-to-end neural coreference resolution module, which we will call *neural ER* from now on, predicts co-referent spans of text, i.e., spans of text referring to each other. It does so without any domain knowledge, i.e., knowledge about mentions of process elements extracted in prior steps. We then align these predictions with mentions. Here we discard predictions, if **(1)** the corresponding span of text is not a mention at all, **(2)** the corresponding span of text does not overlap with a mention's text by a certain percentage $\alpha_m$, **(3)** the mention corresponding with the predicted span of text was not tagged with the majority tag of other mentions of this entity, or **(4)** not at least a certain portion $\alpha_c$ of predicted text spans was previously accepted. We optimize these parameters using a grid search approach, choosing $\alpha_c = 0.5$ and $\alpha_m = 0.5$. A simple example of this process is shown in Fig. 3

Finally the **RE step** extracts relations between mentions using CatBoost, a gradient boosting technique for classification using numerical, as well as categorical data [17]. We call this module *BoostRelEx* for short in following sections. For each combination of head and tail mention of a relation we build features containing tags, distance in tokens and in sentences between them, and a number $c$ of neighboring mention tags as context. This feature set is then presented to the model, which predicts a class for it. Classes are the set of relation tags and an additional *nothing* tag to enable the model to predict that there is no relation between two mentions. During training we present each of the mention combinations containing a relation to the model exactly once per iteration, as well as a given number of negative examples. These negative examples only consist of

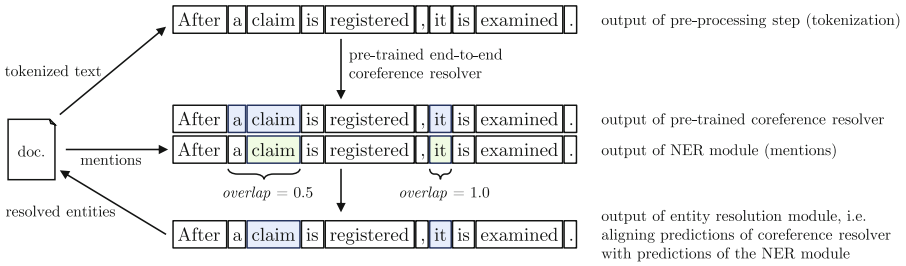



**Fig. 3.** Example for our ER method based on a pretrained end-to-end neural coreference resolver. Predicted coreferent text spans *a claim* and *it* are accepted and resolved to an entity containing the mentions *claim* and *it*, since both text spans overlap at least 50% with the mention's texts.

mention combinations, where corresponding entities do not have a relation. This concept, called negative sampling, is important, as there are many more mention combinations without a relation between them (44,708), as there are ones with one (1,916). Without negative sampling the precision of our relation extraction module would be extremely low, visualized in Fig. 4. For each positive sample we select $r_n$ randomly drawn negative ones. Increasing $r_n$ has a positive impact on the accuracy with which the model predicts the existence of relations between given pairs of mentions, which is called the *precision* $P$. Since the model learns it has to reject some mention combinations, it also inevitably rejects correct combinations. Following directly from this, the model misses more combinations of mentions, where a relation actually would have existed, thus resulting in a lower *recall* $R$. The harmonic mean between the two scores $R$ and $P$ gives us a good idea of the model's performance. We discuss this metric in more detail in Sect. 5.3. We train the *BoostRelEx* module for $i = 1000$ iterations, which is the most computationally intensive step in the whole pipeline, taking about 25 min on an Intel i9-9900K CPU @ 3.60 GHz, using a negative sampling rate of $r_n = 40$ and context size of $c = 2$. A sampling rate $r_n \geq 40$ improves the result quality significantly.

## 5 Experiment Setup

In the following we describe the extension of the original PET dataset accompanied with dataset statistics (Sect. 5.1). To enable empirical evaluation Sect. 5.3 introduces performance measures that are most adequate for the task and the concrete dataset.

### 5.1 Dataset

The PET dataset is presented in detail in [7], in the following we will only discuss aspects of this dataset directly related to our extension and analysis. PET contains a total of 45 documents, with seven entity types, and six relation types. To facilitate the entity resolution task described in Sect. 2, we assign each

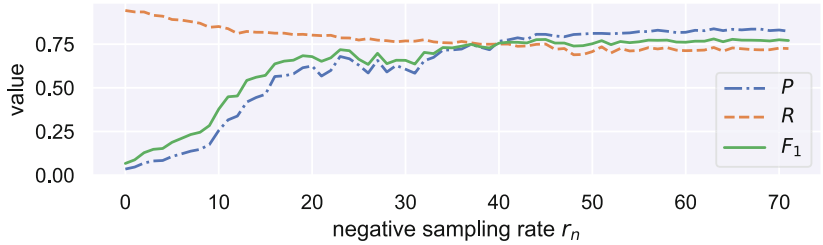


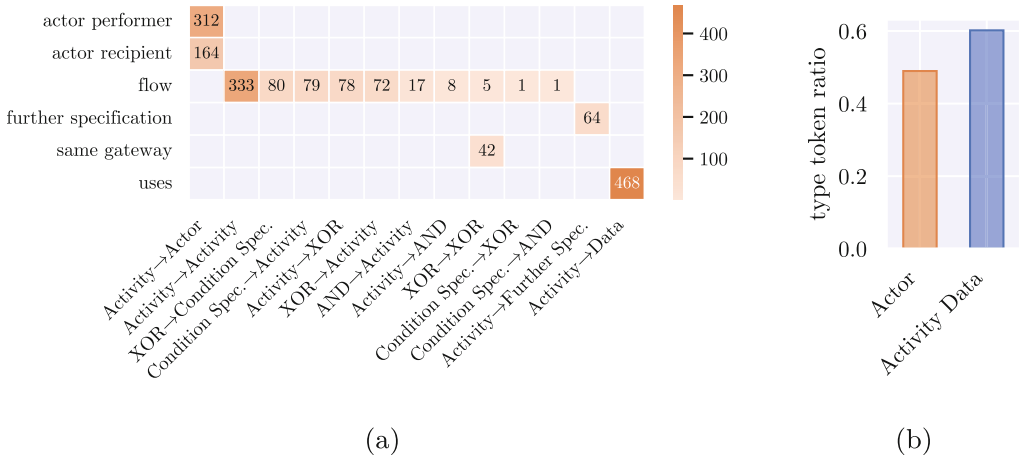

**Fig. 4.** Values of metrics $P$, $R$, and $F_1$ for different negative sampling rates $r_n$.

mention of a process element to a cluster[3]. Thus, each cluster refers to exactly one instance of a process element (e.g., a particular actor) that is mentioned in several places in the text. In Fig. 1 the same claim is mentioned twice, i.e., **the claim** is registered and then **it** is examined. This resulted in a total of 163 clusters with two or more mentions, of which 75 are *Activity Data* mention clusters, and 88 *Actor* mention clusters. All other entity types and the remaining *Activity Data* and *Actor* mentions belong to clusters with only a single mention.

We define the *intra-entity distance* as the maximum of each mention's minimal distance to each other mention in the entity. This gives us the largest span an extraction method has to reason over to detect two mentions as part of the same entity. Averaged over all entities this measure is 31.93 tokens for *Activity Data* elements and 54.84 tokens for *Actors*. Distances between referent mentions are significantly longer for *Actors*, indicating that they possibly are harder to extract. Our experiments seem to support this notion, as shown in Fig. 6c) and d), but further analysis may be required to come to a conclusive rationale.

Intuitively, resolving references between mentions of an entity, is easier, when the texts of those mentions are very similar. Consider, for example, two entities, both made up of two mentions each. One entity has the mentions *"a claim"* and *"the claim"*, while the other has the mentions *"the claimant"* and *"a applicant"*. Resolving the first entity should be much easier, since its mentions share common text. Thus, calculating the lexical diversity of entities of a given type lets us predict how hard it is to extract them without errors. The *type-token ratio* (TTR) can be used to measure the lexical diversity of a given input text [20]. It is calculated as the ratio between unique tokens and total number of tokens. High ratios imply very diverse phrases, while low ratios indicate very uniform text. We select all entities, which contain at least two mentions, and calculate the TTR for each of them. Take for example the entity consisting of the three mentions *"a claim"*, *"the claim"*, and *"it"*. Its TTR would therefore result in $TTR = \frac{4}{5} = 0.8$. We then calculate the mean of these TTR values, split by entity type. On average, *Activity Data* mention clusters exhibit higher type-token ratios compared to *Actors*, as visualized in Fig. 5b. This result is leading us to assume *Actors* should be easier to resolve. Our experiments support this notion, as can be seen in Fig. 6c).

[3] All clusters are defined by two experts, with the help of a third for cases, where their initial annotations differed.




(a) (b)

**Fig. 5.** Statistics of the original dataset. 5a shows the number of relations aggregated by argument types denoted with *head* → *tail*. Only combinations where at least one relation exists are shown. 5b shows the mean type-token ratio for mention clusters with at least two mentions.

Figure 5a shows the distribution of relation types depending on the types of their arguments. For relations of type *Actor Performer*, *Actor Recipient*, *Same Gateway*, and *Flows*, knowing the types of their arguments is no discriminating feature. For these cases, a data driven approach, such as the one we propose in this paper, is very useful, as complex rules are inferred from data automatically, saving a lot of manual effort. In contrast, there are also relation types, where their type can directly be inferred from their argument's types, e.g., all relations that have an *Activity* as head argument, and an *Activity Data* element as tail, are of type *Uses*. This is hardly surprising when factoring in domain knowledge, as in *PET Activity Data* is only used with *Activities*. Predicting relations of those types is therefore more a matter of detecting them (*recall*), rather than correctly classifying them (*precision*).

### 5.2 Compared Approaches

We compare our proposed pipeline to the baseline presented in [7], extended with our ER module. The pipeline looks very similar to ours visualized in Fig. 2, but instead of the *BoostRelEx* module, it uses a rule-based relation extractor, which we will denote *RuleRelEx*. These rules are defined in [7], but have no public implementation, to our knowledge our code is the first executable version available to the community. There are a total of six rules, which are applied to documents in order. This means that rule 1 takes precedence over, e.g., rule 3, which relies on this fact, as it needs information about previously extracted *Flow* relations. We will denote this pipeline with *Bellan + ER* from now on.

Answering RQ1 requires a deep learning approach, which is able to extract mentions, entities, and relations. *Jerex* [9] is suitable for this task, as it is a jointly trained end-to-end deep learning approach, and promises to reduce the

effect of error propagation. Jerex takes raw, untokenized text as input, tokenizes it, and produces predictions for mentions, entities, and relations between them. It is state of the art for the *DocRed* dataset [27], which is a large benchmark dataset for the extraction of mentions, entities, and relations from documents – a task description very similar to the one we gave in Sect. 2. Furthermore, Jerex is able to extract the exact location of mentions inside the input text, unlike competing approaches, which only extract the text of mentions[4]. While this drawback may not be as relevant in applications where only the text of a process element is interesting, for the task of business process generation, i.e., the task of generating human-readable, rich labels for activities in a BPMN process model needs the text surrounding a predicted *Activity* [10].

### 5.3 Evaluation

For performance evaluations of existing baselines, as well as our contributions, we adopted the evaluation strategy from [7]. This means we run a 5-fold cross validation for the entire pipeline and average individual module scores, folds are chosen randomly. This leads to 5 repeated runs, each with 80% of documents used for training, and the remaining 20% for testing. Errors made by modules during prediction are propagated further down the pipeline, potentially even amplifying in severity, as down-stream modules produce errors themselves as a result. To evaluate a given module's performance in isolation, we inject ground-truth data instead of predictions as inputs. This leads to a total of five different scenarios, for which results are discussed in detail in Sect. 6. These scenarios are **(S1)** *entity resolution* using predictions from the *Mention Extraction* module, and **(S2)** using ground-truth mentions. Furthermore, *relation extraction* **(S3)** using entities predicted by the pipeline thus far, **(S4)** using entities predicted during *entity resolution* using ground-truth mentions, and finally **(S5)** using ground-truth entities.

In each case we use the $F_1$ score as a metric, as it reflects the task of finding as many of the expected mentions, entities, and relations as possible (*recall* $R$), without sacrificing *precision* $P$ in type or existence prediction. $F_1$ is then calculated as the harmonic mean of $P$ and $R$, i.e., $F_1 = \frac{2 \cdot P \cdot R}{P+R}$. As there is more than one class within each prediction task, $F_1$, $P$, and $R$ have to be aggregated. Throughout Sect. 6 we use the micro averaging strategy, which calculates $P$ and $R$ regardless of a given prediction's class. This strategy favours classes with many examples, as high scores in those may overshadow bad scores in classes with few examples. Should this be of concern, the macro averaging strategy can be used, where $P$, $R$, and $F_1$ are calculated for each class separately and averaged afterwards. We argue that it is most useful to find as many process elements as possible regardless of their type, i.e., it is better to find 90% of all *Activities* and only 10% of all *AND Gateways*, instead of 50% of all elements, as there are 501 *Activities* and only 8 *AND Gateways* in PET [7]. As such the micro $F_1$ score is better suited to the task. Following the task description in Sect. 2, we use

[4] See for example the discussion https://github.com/Babelscape/rebel/issues/57.

the following matching strategies. We count a *mention* as correctly predicted, iff it contains exactly the same tokens, as the corresponding ground-truth label, and has the same tag. We count an *entity* as correctly predicted, iff it contains exactly the same mentions, as the ground-truth label. Finally, we count a *relation* as correctly predicted, iff both its arguments, and its tag match the ground-truth label. Therefore, e.g., a single missing "*the*" in the mention "*the claim*" would render this mention prediction incorrect, as well as all entities and relations that refer to it. This effect is called *error propagation* and is the reason why we opted for several scenarios that evaluate modules in isolation, or with some degree of ground-truth input, such as in **(S4)**. It may be, that users are fine with slightly less precise predictions, especially if they only miss inconsequential tokens, such as determiners. Surveying how users rank the importance of different levels of precision is out of scope of this paper and part of future work.

## 6 Results

The following section reports results for the experiments defined in the previous Sect. 5. Based on these results, it answers the research questions posed in Sect. 1. In Sect. 6.1 we provide results for the ER step and compare the naive approach to the one based on pretrained end-to-end neural coreference resolution, both for the modules in isolation (scenario **(1)**) and based on predictions of the NER module (scenario **(2)**). Section 6.2 presents the results for experiments with the RE step in the end-to-end pipeline setting (scenario **(3)**), and in isolation (scenario **(5)**). Finally, we discuss factors affecting the quality of RE results in Sect. 6.3, such as error propagation (scenarios **(4)** to **(6)**).

### 6.1 Entity Resolution Performance

We calculate the $F_1$ scores for all mention clusters with at least two mentions, since resolving single mention clusters is trivial. Figure 6d) visualizes the difference between the two approaches. Overall, the naive version reaches $F_1 = 0.26$, while our proposed pretrained method outperforms it significantly and reaches $F_1 = 0.52$. This stark difference is rooted in the fact, that we use exact matching, where a single missing or superfluous mention in a cluster renders the entire prediction incorrect. By design, the naive approach is unable to resolve anaphoras and cataphoras, i.e., back-referencing and forward-referencing pronouns. This means that every entity containing at least one anaphora, or cataphora, will be predicted incorrectly. Using the results from the NER step reduces performance greatly, similar as in the RE step. Based on the results from our experiments we conclude that a naive ER method is not feasible, and significant gains in performance can be achieved by using neural methods. It would be interesting, if fine-tuning the pretrained model would result in improved accuracy. Additionally, using information about mentions extracted in the NER step could be integrated into ER, instead of using a task-agnostic model, as we do currently. These considerations are currently out of scope, as the work on ER in

**Table 1.** a: Overall performance for Jerex, the PET baseline, and our proposed enhanced pipeline. b: Performance of our proposed machine learnt and the rule-based baseline relation extraction modules in isolation.

| | $P$ | $R$ | $F_1$ |
|---|---|---|---|
| Jerex[9] | 0.20 | 0.27 | 0.22 |
| Bellan[7] + ER | 0.32 | 0.29 | 0.30 |
| Ours | **0.34** | 0.29 | **0.31** |

(a)

| | $P$ | $R$ | $F_1$ |
|---|---|---|---|
| Baseline[7] + ER | 0.79 | 0.66 | 0.72 |
| Ours | **0.83** | **0.82** | **0.82** |

(b)

this paper is aimed at bridging the gap between the current state of the art in machine-learning focused data for extracting business process models from natural language text (PET), and the needs of down stream methods. The discussion in this section leads us to answering RQ3: The pretrained coreference resolution approach we presented is able to outperform naive text matching significantly, and is a useful baseline for resolving entities from mentions in the setting of business process model generation from natural language text.

### 6.2 Relation Extraction Performance

Our proposed *BoostRelEx* step clearly beats *RuleRelEx* from [7] by $F_1 = 0.10$, $P = 0.04$, and $R = 0.16$ in our experiments. This is visualized in Fig. 6 (Table 1b lists exact numbers). *BoostRelEx* profits greatly from correct predictions during the *NER step*, as is evidenced by greatly reduced performance when running our proposed pipeline end to end, as well as *Bellan + ER*. While our pipeline is still able to beat *Bellan + ER* in our experiments, the margin is narrowed substantially, with a difference of $F_1 = 0.01$, $R = 0.02$, and equivalent recall. One reason for this drastic performance loss, is the exact matching strategy we employ. A missing, superfluous, or misclassified mention will produce errors during the RE step, as a relation is only considered correct, if all involved mentions are correct (cf. Sect. 5.3). Considering the strong effect error propagation has on *BoostRelEx*, using a jointly trained end-to-end model seems natural. In Sect. 5.2 we presented *Jerex* as a promising candidate. Yet, following from our experiments, *Jerex* is not able to compete, and performs significantly worse, with a difference of $F_1 = 0.11$, $P = 0.14$, and $R = 0.02$, compared to our pipeline. We suspect that this is rooted in PET's small size, as well as the huge number of trainable parameters of Jerex. We therefore have to answer RQ1 with *No*.

Figure 7 breaks down the $F_1$ score by relation type. Following these results we conclude that the dataset PET is not yet suitable to train deep learning models in a supervised manner. The amount of data currently available makes stable training impossible. To alleviate the issue of low data, further research into the use of pretrained models, such as LLMs is warranted. These models make use of large quantities of unlabeled data to learn the structure and makeup of natural language. They are then either employed in a zero-shot setting (never explicitly trained for the task), few-shot setting (fine-tuned on small quantities

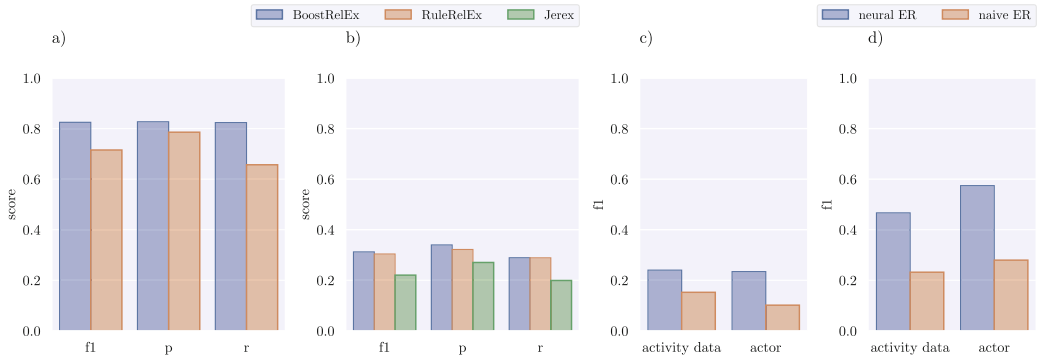



**Fig. 6.** a) shows the comparison between *BoostRelEx* and *RuleRelEx*. b) shows the performance of end-to-end runs of our proposed pipeline *Ours*, and *Bellan + ER*. c) compares the performance of the *naive ER* and *neural ER* using the result of the NER step. d) shows the same comparison as c), but based on ground truth mentions.

of task specific data), or composited into new models (used for extracting useful features from natural language text).

A significant portion of the improvements we present in this work, come from the better extraction of *Actor Recipient* and *Actor Performer*, as well as the *Uses* relations. *BoostRelEx* is clearly outperformed by *RuleRelEx* when extracting the *Same Gateway* relation type. A possible reason for this is how *RuleRelEx* uses information about already extracted *Flow* relations (cf. Sect. 5.2), which is impossible for our machine learnt approach, as it extracts all relations at once. Defining an order of extraction for relation types would defeat the purpose of using our method in the first place: It would be tightly coupled to the dataset and could not be applied easily to others. The overall performance is not affected very much by this, as there are only a handful of examples for the *Same Gateway* relation. Still, further research into useful features for extracting *Same Gateways* is needed, as well as possible training techniques that allow learning more complex rules. Promising features are, e.g., synonyms and hypernyms for key phrases of mentions. Training the model in multiple passes could be useful in predicting relations featuring mutual exclusivity, such as *Same Gateways*.

### 6.3 Performance Analysis

Gradually reducing the quality of inputs to the *BoostRelEx* and *RuleRelEx* steps results in gradually worse performance, a clear indication of error propagation (cf. Sect. 5.3). Using ground-truth mentions from the dataset, but entities predicted by the *neural ER* step, results in a drop in $F_1$ scores of about 0.20 for *BoostRelEx* and 0.12 for *RuleRelEx*. Introducing errors even further upstream, by using the *NER* module, i.e., running the complete pipeline end-to-end results in a drop in $F_1$ of 0.51 for *BoostRelEx* and 0.42 for *RuleRelEx*. Figure 7 visualizes this performance degradation for each relation type individually. Further studies regarding less strict evaluation is warranted, as described in Sect. 5.3, to get a less conservative assessment of prediction quality.

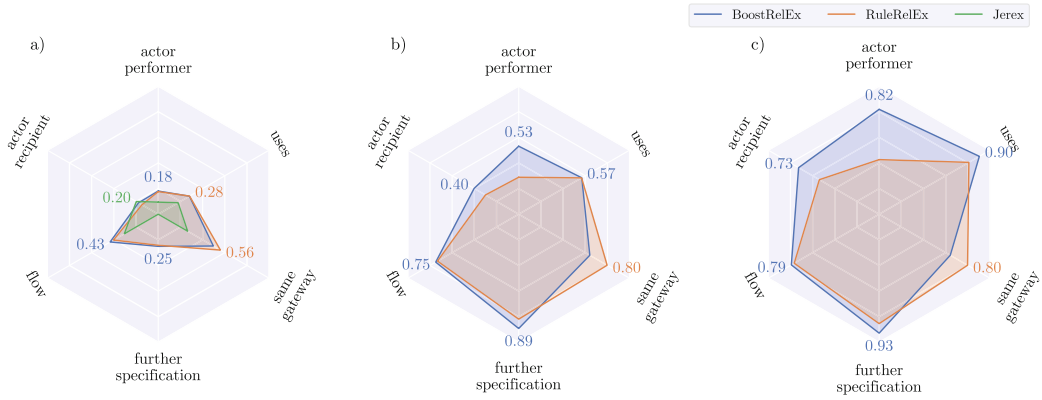



**Fig. 7.** a) Results for relation extraction by relation type for scenario **(4)**, the complete pipeline, b) scenario **(5)**, relation extraction using entities resolved from perfect mentions, c) scenario **(6)**, relation extraction from perfect entities.

Additionally, we found that the distance between a relation's arguments is also negatively correlated with correctness. Longer distance between the head and tail entity of a relation increases the likelihood of misclassifying it, or not detecting it at all. We calculate the distance of a relations arguments as the minimal distance between the two entity's mentions. Examples for this effect are shown in Fig. 8. We created datasets from all predictions of each approach, with tuples of the form $(distance, o)$, where $o = 1$ denotes a correct prediction, and $o = 0$ an incorrect prediction. We then fitted a logistic regression model to these datasets using the *statsmodels*[5] python package. A logistic regression model tries to predict an outcome (response variable) via some input variable (predictor variable). It uses the logistic regression, which is given by $y = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$, and chooses $\beta_0$ and $\beta_1$ in such a way, that the model predicts the observed outcome $y = o$ given an input $x$ as best as possible. We can then use the resulting curve to discuss how well an approach is able to predict certain relation types.

The *Flow* relation can be solved very well for short distances by both *BoostRelEx* and *RuleRelEx*. A very narrow confidence interval indicates a very good fit, leading us to believe, that relations with argument distances upwards of 33 tokens are misclassified by both methods with a significant probability. If this fact is detrimental to the quality of generated business process models is interesting, but out of scope for this paper. The *Same Gateway* relation shows frequent misclassification by the *BoostRelEx* method, something that was already evident in Fig. 7. *BoostRelEx* seems to be very sensitive to the distance between arguments for this relation, more often misclassifying, or outright not recognizing examples, as soon as the distance in tokens exceeds 15 tokens. *RuleRelEx* is significantly more robust in this regard, and able to correctly identify *Same Gateway* relations more often than not, until the distance between their argu-

[5] See https://www.statsmodels.org/stable/generated/statsmodels.discrete.discrete_model.Logit.html.

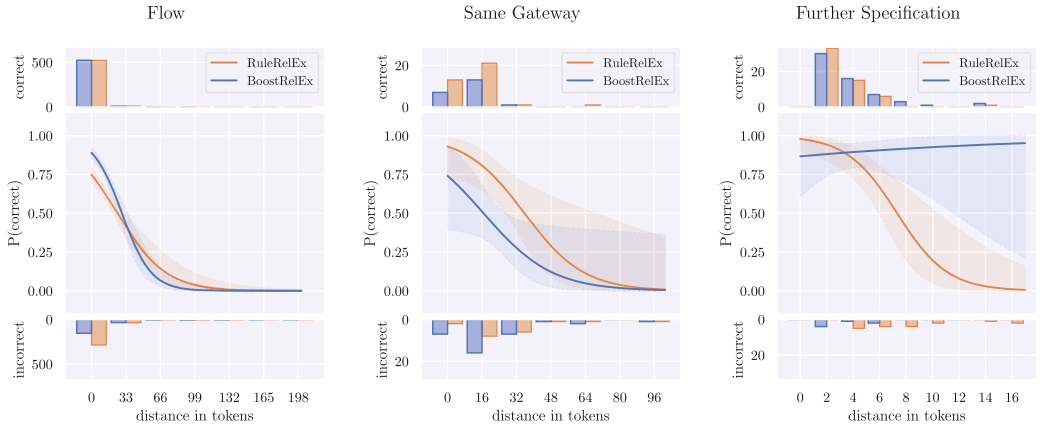



**Fig. 8.** Logistic regression fits for correlation between correctness of a prediction and the distance in tokens between its arguments. Bars show the number of correct (top) and incorrect (bottom) predictions. The main plot shows the fitted logistic regression and the 95% confidence intervals as a transparent channel.

ments exceeds 32 tokens. The fit produces very wide confidence intervals for both approaches, something that could be fixed with more examples for this relation, given a larger dataset. Relations of type *Further Specification* can be extracted by *BoostRelEx* with very high precision and recall. This is already shown in Fig. 7, where the $F_1$ score for *Further Specification* is given as 0.93. The logistic regression fit estimates that there is no correlation between argument distance and correctness. Yet, a very wide confidence interval for distances upwards of 10 tokens leaves open the possibility that there is a correlation given more examples. While *RuleRelEx* predicts more *Further Specification* relations erroneously than *BoostRelEx*, it is able to classify the majority (distances 0 – 6 tokens) correctly. This leads to similar performance overall, as shown in Fig. 7.

In summary, our machine learning based RE method outperforms the rule based RE method, in the best case, and is equivalent in the worst case, we can answer RQ2 with *Yes*. Our in-depth evaluation shows, that *BoostRelEx* robustly extracts long relations, beaten only by *RuleRelEx* on the *Same Gateway* relation, which matters not as much overall, given the small number of examples for this relation.

## 7 Conclusion and Future Work

In this paper we extend the task of business process information extraction by ER. We enrich PET with entity identity information and propose an extraction approach based on pretrained end-to-end neural coreference resolution. Motivated by benefits regarding rapid adaption to new data, domains, or tag sets, we propose a novel gradient boost based approach for the relation extraction

task. We show that our proposed method is able to produce equivalent or better results in the end-to-end setting, and significantly outperform the baseline given higher quality inputs. We show that PET is not yet extensive enough for training a state-of-the-art deep learning approach from the NLP domain, Jerex, even though this approach achieves state-of-the-art results on other, bigger benchmark datasets of a related task. Finally, we discuss traits of the PET dataset that are detrimental to prediction quality, e.g., high linguistic variance, and distance between relation arguments. Our experiments attest to the phenomenon of error propagation, i.e., errors made in early steps are amplified in later ones. Thus, we plan to incorporate joint models for extracting mentions, relations, and for resolving process entities, since they are trained to solve these three tasks simultaneously, and mitigate the error propagation effect. While Jerex did not produce high quality predictions, it, and similar approaches, are predetermined for application in the task of business process generation from natural language text. Therefore, further research into applying deep learning in the low data domain of BPM is needed. We plan to improve performance of the entity resolution module, e.g., by incorporating mention information. Additionally, fine-tuning the pretrained neural coreference resolver on in-domain data is a potential way to improve performance further. Best practises recommend the use of micro $F_1$ scores for judging the quality of predictions in the business process information extraction task. While certainly a useful metric, we suspect it may not capture the needs of down stream tasks and users entirely. We plan to investigate alternative metrics, and their correlation with human expectation. Finally, creating a business process model from the information extracted with our pipeline was out of scope for this paper. More work regarding methods towards solving activity label generation, layouting, and completeness checks is needed.

## 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: Giorgini, P., Weber, B. (eds.) CAiSE 2019. LNCS, vol. 11483, pp. 365–382. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21290-2_23
3. Ackermann, L., Neuberger, J., Jablonski, S.: Data-driven annotation of textual process descriptions based on formal meaning representations. In: La Rosa, M., Sadiq, S., Teniente, E. (eds.) CAiSE 2021. LNCS, vol. 12751, pp. 75–90. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79382-1_5
4. Ackermann, L., Neuberger, J., Käppel, M., Jablonski, S.: Bridging research fields: an empirical study on joint, neural relation extraction techniques. In: Indulska, M., Reinhartz-Berger, I., Cetina, C., Pastor, O. (eds.) CAiSE 2023. LNCS, vol. 13901, pp. 471–486. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-34560-9_28

5. Bellan, P., Dragoni, M., Ghidini, C.: Extracting business process entities and relations from text using pre-trained language models and in-context learning. In: Almeida, J.P.A., Karastoyanova, D., Guizzardi, G., Montali, M., Maggi, F.M., Fonseca, C.M. (eds.) EDOC 2022. LNCS, vol. 13585, pp. 182–199. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17604-3_11
6. Bellan, P., Dragoni, M., Ghidini, C., van der Aa, H., Ponzetto, S.: Guidelines for process model annotation in text (2022)
7. 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: Proceedings of the Sixth Workshop on NL4AI (2022)
8. Cabot, P.L.H., Navigli, R.: Rebel: relation extraction by end-to-end language generation. In: EMNLP (2021)
9. Eberts, M., Ulges, A.: An end-to-end model for entity-level relation extraction using multi-instance learning. In: ACL (2021)
10. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 482–496. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21640-4_36
11. Giorgi, J., Bader, G., Wang, B.: A sequence-to-sequence approach for document-level relation extraction. In: Workshop on Biomedical Language Processing (2022)
12. Käppel, M., Schönig, S., Jablonski, S.: Leveraging small sample learning for business process management. Inf. Softw. Technol. (2021)
13. Lee, K., He, L., Lewis, M., Zettlemoyer, L.: End-to-end neural coreference resolution. In: EMNLP (2017)
14. Li, H.: Deep learning for natural language processing: advantages and challenges. Natl. Sci. Rev. **5**, 24–26 (2018)
15. Li, J., Sun, A., Han, J., Li, C.: A survey on deep learning for named entity recognition. IEEE Trans. Knowl. Data Eng. **34**, 50–70 (2020)
16. Pawar, S., Palshikar, G.K., Bhattacharyya, P.: Relation extraction: a survey. arXiv preprint arXiv:1712.05191 (2017)
17. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: CatBoost: unbiased boosting with categorical features. In: NeurIPS (2018)
18. Qian, C., et al.: An approach for process model extraction by multi-grained text classification. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds.) CAiSE 2020. LNCS, vol. 12127, pp. 268–282. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49435-3_17
19. Quishpi, L., Carmona, J., Padró, L.: Extracting annotations from textual descriptions of processes. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 184–201. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_11
20. Richards, B.: Type/token ratios: what do they really tell us? J. Child Lang. **14**, 201–209 (1987)
21. Sànchez-Ferreres, J., Burattin, A., Carmona, J., Montali, M., Padró, L., Quishpi, L.: Unleashing textual descriptions of business processes. Softw. Syst. Model. **20**(6), 2131–2153 (2021). https://doi.org/10.1007/s10270-021-00886-x
22. Sanh, V., Wolf, T., Ruder, S.: A hierarchical multi-task approach for learning embeddings from semantic tasks. In: Proceedings of the AAAI Conference on Artificial Intelligence (2019)
23. Sharnagat, R.: Named entity recognition: a literature survey. Center For Indian Language Technology (2014)

24. Sukthanker, R., Poria, S., Cambria, E., Thirunavukarasu, R.: Anaphora and coreference resolution: a review. Inf. Fusion **59**, 139–162 (2020)
25. Wallach, H.M.: Conditional random fields: an introduction. Technical reports (CIS) (2004)
26. Waltl, B., Bonczek, G., Matthes, F.: Rule-based information extraction: advantages, limitations, and perspectives. Jusletter IT (02 2018) **4** (2018)
27. Yao, Y., et al.: DocRED: a large-scale document-level relation extraction dataset. arXiv preprint arXiv:1906.06127 (2019)

# Chapter 8

# Leveraging Data Augmentation for Process Information Extraction

Neuberger, J., Doll, L., Engelmann, B., Ackermann, L., Jablonski, S. (2024). Leveraging Data Augmentation for Process Information Extraction. In: van der Aa, H., Bork, D., Schmidt, R., Sturm, A. (eds) *Enterprise, Business-Process and Information Systems Modeling. BPMDS EMMSAD 2024 2024.* Lecture Notes in Business Information Processing, vol 511. Springer, Cham. https://doi.org/10.1007/978-3-031-61007-3_6



**Contribution statement.** The concept to use data augmentation for process descriptions was developed by Julian Neuberger (JN) and Lars Ackermann (LA). The list of possible augmentation techniques was selected by JN. Exclusion criteria for filtering the list of augmentation techniques were defined by JN, Leonie Doll (LD), and Benedikt Engelmann (BE). Filtering the list of augmentations according to the exclusion criteria was done by LD and BE. Adjusting the remaining augmentations was done by JN, LD, and BE. Setting up the experiment, conducting it, and interpreting the results was done by JN. JN mainly wrote the publication text, with additional contributions by LA and Stefan Jablonski (SJ). Revisions were done by all authors. JN is the corresponding author.

# Leveraging Data Augmentation for Process Information Extraction

Julian Neuberger(✉), Leonie Doll, Benedikt Engelmann, Lars Ackermann, and Stefan Jablonski

University of Bayreuth, Bayreuth, Germany
{julian.neuberger,leonie.doll,benedikt.engelmann,lars.ackermann,
stefan.jablonski}@uni-bayreuth.de

**Abstract.** Business Process Modeling projects often require formal process models as a central component. High costs associated with the creation of such formal process models motivated many different fields of research aimed at automated generation of process models from readily available data. These include process mining on event logs and generating business process models from natural language texts. Research in the latter field is regularly faced with the problem of limited data availability, hindering both evaluation and development of new techniques, especially learning-based ones.

To overcome this data scarcity issue, in this paper we investigate the application of data augmentation for natural language text data. Data augmentation methods are well established in machine learning for creating new, synthetic data without human assistance. We find that many of these methods are applicable to the task of business process information extraction, improving the accuracy of extraction. Our study shows, that data augmentation is an important component in enabling machine learning methods for the task of business process model generation from natural language text, where currently mostly rule-based systems are still state of the art. Simple data augmentation techniques improved the $F_1$ score of mention extraction by 2.9% points, and the $F_1$ of relation extraction by 4.5. To better understand how data augmentation alters human annotated texts, we analyze the resulting text, visualizing and discussing the properties of augmented textual data.

We make all code and experiments results publicly available (Code for our framework can be found at https://github.com/JulianNeuberger/pet-data-augmentation, detailed results for our experiments as MySQL dump can be downloaded from https://zenodo.org/doi/10.5281/zenodo.10941423.).

**Keywords:** Business Process Extraction · Data Augmentation · Natural Language Processing

## 1 Introduction

It has been shown that a major share of time planned for Business Process Management (BPM) projects is spent on the acquisition of formal business process

models [14]. This fact motivated a whole host of work done on automated generation of business process models from varying, readily available sources, such as event logs, or natural language process descriptions. The latter area has seen increasing attention in recent years [1,2,13,14,23,24]. Most of these approaches can be formulated as a two-step process, where first the business process relevant information is extracted from text, and then a concrete model is generated from this information.

Many approaches proposed for the task of business process information extraction (BPIE) from texts are still rule-based. This means they extract the information needed for building a formal business process model by applying rules defined by human experts. These rules are usually optimized for a specific dataset, industry sector, or language. For this reason, such systems usually achieve impressive results for their intended, very limited, application domain, but are difficult to transfer to new ones. Alternative approaches like data driven approaches, often called machine learning (ML) methods, infer rules directly from data, making them a lot easier to be adapted to new datasets, domains, or languages. Ideally, adapting an ML approach involves only training on new data. However, this need for data, both for the initial development, as well as for potential adaptations, is what typically impedes application of ML methods at first. Compared to other disciplines also using machine learning, datasets in BPM are relatively small, especially for the task of generating business process models from natural language text. Data sets for this task are expensive to generate, as time-consuming manual annotation by experts is required, in which raw text data is enriched with the desired results (e.g. process entities) in order to provide the ML methods with a basis for learning. Previous work tried to solve this issue by leveraging out-of-domain data, e.g., via pretrained word embeddings [2], or less expressive models, which are easier to train [23] on small datasets. Other fields of related research, such as computer vision, natural language processing (NLP), or audio analysis, tackle the issue by synthesizing new data samples from the existing dataset. This concept is called data augmentation (DA), injecting controlled perturbations. These perturbations can be structural changes (e.g., rotations), as well as added noise.

Data augmentation has already been shown to be useful in a BPM context, such as for the task of next activity prediction, where it is proposed as a solution for the problem of rare process executions, as well as for extrapolating gradual changes in the way processes are executed [18,19]. The authors show that

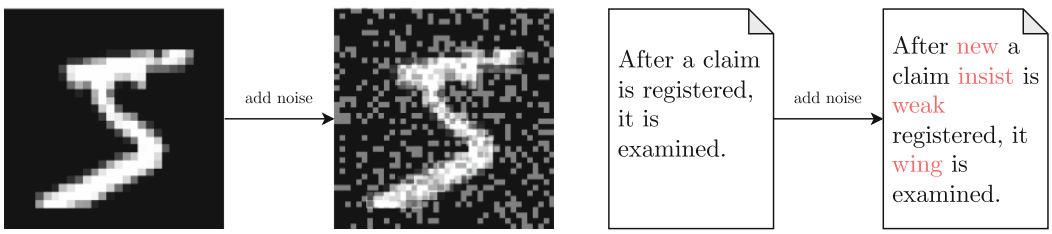



**Fig. 1.** Example for adding noise to an image (left) and to a sentence (right). The image keeps its semantics, while the sentence looses it.

simple data augmentation strategies like swapping, inserting, or deleting events in the record of a process execution can result in significant accuracy gains [18]. Encouraged by those results, we want to analyze how data augmentation can be applied to improve existing methods for generating process models from text. Due to the nature of natural language, introducing noise without changing the semantics of a data sample is much harder, compared to, for example, computer vision tasks. See Fig. 1 for an example, where introducing noise into an image still keeps its semantics (image of a handwritten "5") intact. Introducing noise in the form of random words into a single sentence on the other hand, can change its semantics significantly, to a point, where it is hard to understand even for humans. We will discuss this fact in more detail in Sect. 2 and give examples for data augmentation techniques, which preserve semantics.

To further structure our understanding of applying data augmentation in the context of generating business process models from natural language text, we pose three research questions.

**RQ1** Can simple data augmentation techniques, including swapping, deleting, or randomly inserting words into sentences increase the performance of machine learning methods for BPIE, measured as the harmonic mean of precision and recall?

**RQ2** Does the use of large language models in data augmentation, such as for so called *Back Translation* techniques, provide a significant advantage over simpler, rule-based methods?

**RQ3** What characteristics of the natural language text data are changed by augmentations, and how do they affect different extraction tasks?

The rest of this paper is structured in the following way: Sect. 2 describes the background of data augmentation in a NLP environment. In Sect. 3 we give an overview of work related to our study. Section 4 defines the setup for our experiments. We then present our results in Sect. 5. Finally, in Sect. 6 we discuss limitations, describe future work, and draw a conclusion.

## 2 Background

**Data augmentation** describes a suite of techniques originally popularized in computer vision [25], where simple operations, such as cropping, rotating, or introducing noise into images greatly improved performance of machine learning algorithms used for classification of images. These operations usually preserve the semantics of input data, meaning that an image containing an object will still depict the same object after its data have been augmented, for example, they have been overlaid with noise. This property is called *invariance* [28], and is harder to hold for NLP data [12]. An example for this fact is depicted in Fig. 2. Changing random tokens (e.g., words) in a sentence may alter semantics to a point, where relevant elements or relations between those elements are no longer present after augmentation. Additionally, annotations may be lost, if transformations are applied and afterwards these changes cannot be traced. This

might happen, when, for example, an entire sentence is translated into another language and then is back-translated typically leading to a rephrased version of the original text. Since it is not clear, which parts of the new sample correspond to the original one, annotations of process-relevant elements do not apply to the new sample. For this reason, research on data augmentation techniques has been conducted, which are specifically designed for information extraction tasks in the NLP domain[11,15,21]. These techniques use additional resources, such as pretrained large language models, to augment training samples, while keeping their semantics intact.

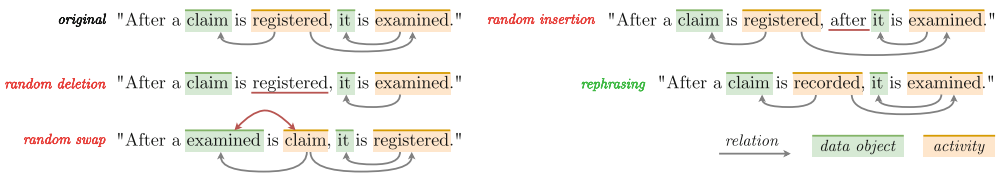



**Fig. 2.** Examples for four different data augmentation techniques. *Random deletion*, *random swap*, *random insertion* (all written in red), are all not preserving the semantics of a sample and its label. *Rephrasing* (green) is an example for a technique that does. (Color figure online)

**Process Relevant Information Extraction** from natural language is a research field immediately relevant for information systems, as business process models are often a central part for process aware information systems. Discovering and creating these process models is an expensive task [14] and a lot of work has been done on extracting them from natural language text directly [1,2,5,13,14]. These texts describe a business process in natural language as technical documentation, maintenance handbooks, or interview transcripts. Sequences of words (spans) in these texts contain information that is relevant to the business process, such as *Actors* (persons or departments involved in the process), *Activities* (tasks that are executed), or *Data Objects* (physical or digital objects involved in the process). Extracting this information is therefore a sequence tagging task, and can be framed as *Mention Detection* (MD). Mentions relate to each other, e.g., defining the order of execution for two Activities, or which Actor executes the Activity. Predicting and classifying these relations is called *Relation Extraction* (RE). Refer to Fig. 3 for an example of this process. It shows a fragment of a larger description of a process from the insurance domain, where insurance claims have to be registered in a system and subsequently examined by an employee. The spans *claim* and *it* are annotated as Data Objects (the claim in question, in green). Activities executed by a process participant are marked in orange. These four spans can now be transformed into business process model elements for a target notation language (here BPMN[1]). How these elements interact with each other can also be extracted from the text

[1] See specification at https://www.bpmn.org/.

fragment, e.g., the *Flow* of activity execution between the mentions *registered* and *examined*, depicted as an orange arrow.

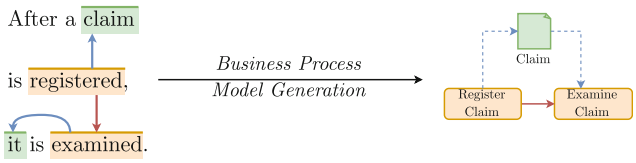



**Fig. 3.** Example for a fragment of a natural language business process description and its corresponding business process model fragment in BPMN. (Color figure online)

Developing approaches towards automated extraction of process relevant information requires data to test performance, and train models, if applicable. The currently largest collection of human-annotated process descriptions is called PET [6]. It contains 45 natural language process descriptions, and is annotated with 7 types of process relevant entities (e.g., Actors, Activities, Data Objects), as well as 6 types of relations between them (e.g., Flow between Activities). In total the dataset contains less than 2,000 examples for both relations and entity mentions. For comparison, typical datasets for related tasks, like Knowledge Graph completion contain more than 200 times as many. For example, the popular *FB15k* dataset comprises more than 500,000 relation examples [8]. Datasets for extraction of named entities and their relations have similar extents, e.g., the DocRed dataset, which contains more than 1,500,000 relation examples [27]. This fact makes PET a prime candidate for data augmentation techniques, in order to make the most out of the limited amount of training examples. We show this in our experiments using PET for the tasks MD and RE in BPIE. To our knowledge our work is the first to attempt applying NLP data augmentation to the BPIE task.

## 3 Related Work

Data augmentation techniques applied in this paper are largely based on the ones available in the *NL-Augmenter* framework [10]. NL-Augmenter provides a list of more than 100 data augmentation techniques, which are suitable for varying tasks like text classification, sentiment analysis, and even tagging. We discuss how we adapted these transformations to the PET data format in more detail in Sect. 4. Not all transformations are relevant for this work, and we have to exclude most of them, as they are not fitting for BPIE. Details of our exclusion criteria can be found in Sect. 4.

In [17] the authors evaluate nine simple data augmentation techniques (e.g., random deletion) on a total of seven event logs, using seven different models. Our paper follows a similar line of thought for BPIE, instead of predictive process monitoring. The transformations we employ differ significantly from theirs in two

core aspects. First, transformations used in this paper are more complex, owing to the more complex character of natural language. While their work focused on reordering events in a log of a process execution, our work uses transformations that are concerned with replacing, extending, or modifying sequences of text, while preserving any annotations present in the data. Second, transformations used in our work often require external resources. These resources can be explicit, i.e., databases like WordNet [22], which contains lexical information such as synonyms, antonyms, or hypernyms of words. They can also be implicit, such as large language models, which contain knowledge about natural language, obtained by unsupervised training on huge amounts of textual data [9].

The techniques we present in our paper mainly benefit work that already exists in the field of BPIE. Therefore, approaches towards BPIE based on machine learning are related to this work. These approaches can be separated into two main fields of research. *(1)* learning approaches, which use the data to train a machine learning models, e.g., a neural network [2], conditional random fields [6], or decision trees [23]. *(2)* prompting based approaches that use the data for engineering input for large language models (e.g., GPT) [16,20], or use the data for so called *in context learning*, by providing examples in the input itself [5].

Automated extraction of information relevant to business processes from natural language text descriptions can be seen as a special case of automated knowledge graph construction or completion [4]. We therefore consider techniques for automated knowledge graph construction and completion as distantly related work, which could still benefit from the augmentation techniques we analyze in this paper. Nonetheless, we focus on methods of BPIE in this paper, as potential solution for this field's small datasets.

## 4 Experiment Setup

The NL Augmenter framework provides a total of 119 data augmentation techniques, but not all of them are applicable to the task at hand. We therefore define four criteria for exclusion: **(EC1)** The technique does not apply to the English language. **(EC2)** The technique alters the spelling of tokens. **(EC3)** The data augmentation technique does not work for supervised data, e.g., it corrupts target annotations. **(EC4)** The technique uses task-, and/or domain-specific resources, such as dictionaries, or databases, which often do not exist for BPM data, and are hard to create given the diversity of BPM application domains. Applying these exclusion criteria results in 19 data augmentation techniques relevant for the task of business process information extraction from natural language text.

### 4.1 Data Augmentation Effects

The data augmentation techniques we selected synthesize samples with three core characteristics.

**(1)** Increased linguistic variability, i.e., augmented text uses a larger vocabulary to describe the same, or at least, a similar business process[2]. The most prominent examples for such techniques are the *Back-Translation* techniques. These use a large language model, e.g., BERT [9] to translate the process description to a different language and subsequently translate it back to the original language – here English. Since data augmentation techniques must not alter the annotations of entities, we only translated spans of text, not the entire document at once. Take for example, the running example *After a claim is registered, it is examined.* Here four spans are annotated as entities – *a claim*, *registered*, *examined*, and *it*. Additionally there are three remaining spans, that do not correspond to entities: *After a*, *is*, *is*. By back-translating these seven spans separately, we obtain variation in their wording (*surface form*), but are still able to preserve annotations. Samples synthesized in this way are especially useful for making methods for the MD task generalize better and more robust.
**(2)** Variations in span length. Many spans of a given entity type, e.g., Actors are very uniform in length across examples. This is a result of several factors, but most apparent actors are often identified by their job title, e.g., *the clerk*, or the department, e.g., *the secretary office*. These titles and departments are very short phrases, and longer ones are abbreviated, reducing their length to two or less tokens, e.g., *the MPOO*. Even though their expanded form may not be known, expanding some of these spans to suitable phrases, e.g., *Manager, Post Office Operations*, creates samples with longer surface forms. This in turn, may improve the robustness of the MD extractors, as well as the generalization capabilities of RE methods.
**(3)** Directionality of relations between mentions. The order of appearance for mentions that form a relation, is very uniform in the current version of PET. This is especially apparent, when looking at the base-line extraction rules defined by the original authors of PET: Here the order of appearance of Activities and Actors is exploited, to form the *Actor Performer* and *Actor Recipient* relations [6]. These relations define the Actor, that performs an Activity, and the Actor, on which an Activity is performed. The Actor left of an Activity is assigned the former, while the Actor right of that Activity is assigned the latter. In this example order uniformity can lead to less robust models, as they rely on this and subsequently make wrong predictions given different linguistic constructs. Synthesizing samples with a different order may encourage models to consider linguistic features (context) rather than just the order of mentions in a sentence during prediction.

### 4.2 Finding Optimal Configurations

Each of the data augmentation techniques we selected can potentially be adjusted by several parameters, which control how augmented samples are synthesized. A typical example for such a parameter is the number of inserted

[2] The augmentation technique might change information in the text, which changes the process overall, e.g., by replacing original actors with new, artificial ones.

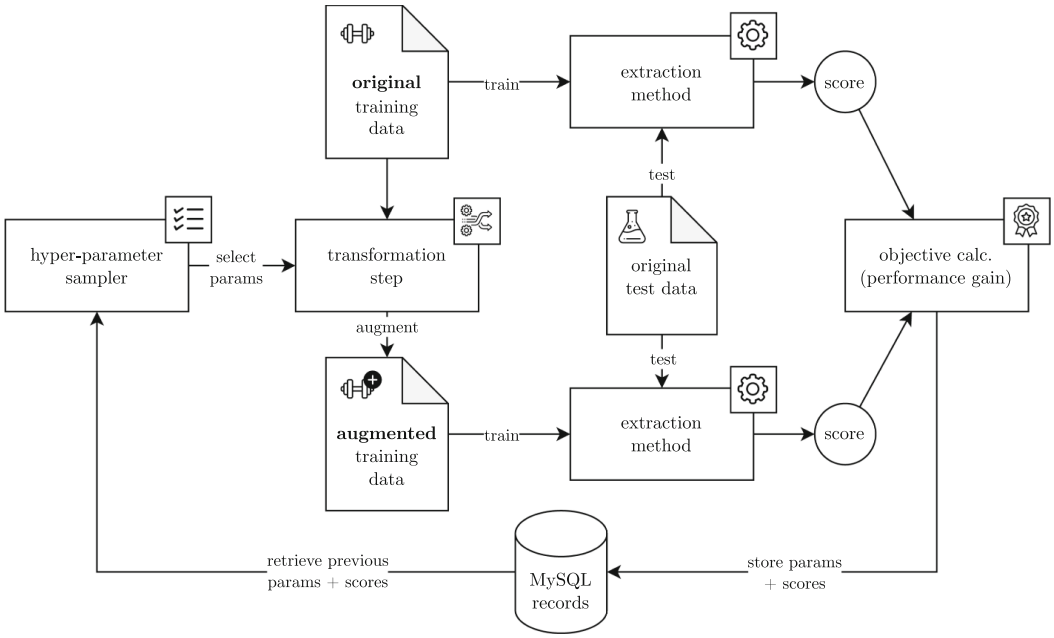



**Fig. 4.** Choosing optimal configurations for data augmentation techniques.

tokens. Increasing this number would result in a sample, which is more perturbed compared to a sample where fewer tokens are inserted. We consider optimally choosing such parameters for a given technique a *hyper-parameter optimization* problem. Hyper-parameter optimization is defined as finding a configuration of parameters so that a given objective (metric to optimize) is minimal or rather maximal, depending on the case. Here, we want to maximize the performance gain that the application of a data augmentation technique has. To that end we run a 5-fold cross-validation of the extraction step (MD, RE) with the original, unaugmented data. We then select a configuration for the given technique and run the same 5-fold cross-validation, but augment the training data of each fold with the data augmentation technique. We define the difference between the scores of these two models on the (unaugmented) test dataset as the *performance gain* and use it as maximization objective for our hyper-parameter optimization. Each data augmentation technique is optimized in 25 runs (*trials*) using Optuna [3] and a Tree-Structured Parzen Estimator for selecting parameter values [7]. We depict this process in Fig. 4.

The most effective data augmentation techniques can then be used to supplement the existing PET dataset, as well as any future datasets used for BPIE. In the following Sect. 5 we present the results of the experiment described here and discuss their implications.

## 5 Results

In this section we will discuss results for the experiments we defined in the previous section. Table 1 lists the differences of all data augmentation techniques

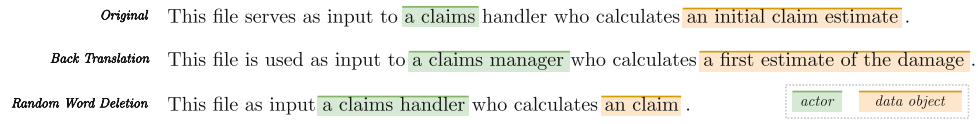



**Fig. 5.** Two examples for the effects of data augmentation techniques.

compared to a run on un-augmented data. All differences are measured as the micro-averaged $F_1$ score. Concluding from our results, we find that the RE task can benefit significantly from most of the data augmentation techniques we selected and tested. Transformations that reorder tokens, like *Shuffle Within Segments*, *Sentence Reordering*, or *English Mention Replacement* seem to be less useful, compared to other techniques. This is most likely rooted in the change in directionality of relations (effect **(3)** from Sect. 4). Since these transformations do not take any context into account, such changes may be breaking semantics of relations in a sentence.

Transformations based on large language models, especially back translation techniques, like *Multilingual Back Translation*, which translate a sentence fragment twice, are very time-intensive. Yet, improvements in relation extraction performance is not significant, when comparing them to more lightweight approaches, e.g., *Synonym Substitution*, which uses WordNet to rephrase text sequences. In our experiments using these large language model based methods is not worth the increase in computing power and time. While the MD task can still benefit from all data augmentation techniques, it does so to a lesser extent when compared to the RE task. This indicates a model, that is already more stable, and generalizes better. Transformations that alter the amount of tokens in mentions, such as *Random Word Deletion*, *Synonym Insertion*, or *Subsequence Substitution for Sequence Tagging*, result in lesser improvements, compared to paraphrasing methods, such as *AntonymsSubstitute*, *BackTranslation*, or *Synonym Substitution* (Fig. 5). Similar to the RE task, the MD task does not benefit significantly more from resource and time intensive, large language model based augmentation techniques for paraphrasing, compared to their simpler counterparts.

Based on our observations, we can answer research question RQ1, with "Yes". Simple data augmentation techniques like swapping tokens, deleting them, or inserting random tokens do have a significant benefit. The RE task benefits more from them, than the MD task does. Since nearly all augmentations have a net positive impact, we argue that the perturbations act as controlled noise, very similar to techniques used for training deep neural networks. There, models that are more robust and generalize better are created, simply by adding noise to the training data [26].

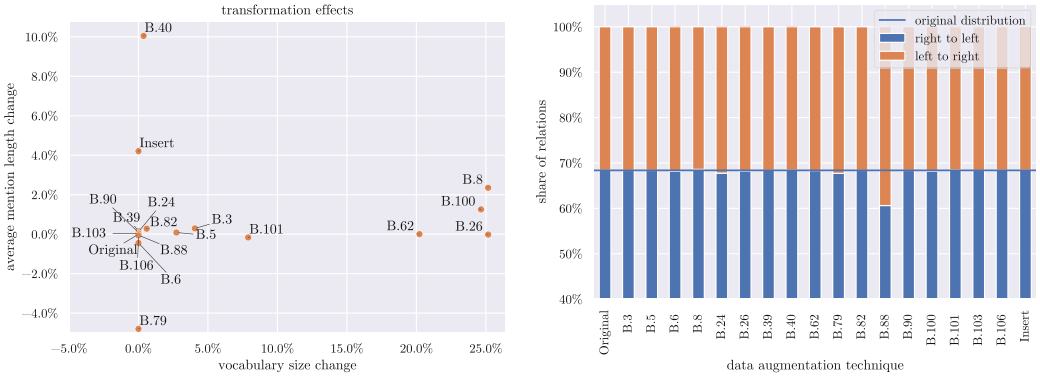
The use of large language models in data augmentation techniques brings with it a significant increase in resources needed, both in terms of computing time, hardware requirements, and power consumption. Based on our experiments, this is not worthwhile for improving the business process information

**Table 1.** Detailed results for all transformation steps, for both the MD and RE task. Column *Id* refers to the identifier defined in [10]. It links to the source code for this technique. Top three results are set in **bold** face. All results are the averages of a 5-fold cross validation on the entire dataset.

| Technique | Id | Description | MD | RE |
|---|---|---|---|---|
| Unaugmented | | results for un-augmented data | 0.695 | 0.759 |
| Adjectives Antonyms Switch | B.3 | use antonyms of adjectives | +0.024 | +**0.045** |
| AntonymsSubstitute (Double Negation) | B.5 | substitute even number of words with antonyms | +**0.025** | +0.042 |
| Auxiliary Negation Removal | B.6 | remove negated auxiliaries | +**0.025** | +0.039 |
| BackTranslation | B.8 | translate to German, then back to English | +**0.025** | +0.036 |
| Concatenate Two Random Sentences | B.24 | remove punctuation between sentences | +0.023 | +0.042 |
| Contextual Meaning Perturbation | B.26 | replace words with use of pretrained language model | +0.004 | +0.040 |
| English Mention Replacement for NER | B.39 | replace mention with one of the same type in document | +0.015 | +0.036 |
| Filler Word Augmentation | B.40 | introduce "uhm", "I think", | +0.021 | +**0.043** |
| Multilingual Back Translation | B.62 | see B.8, language is parameter | +0.022 | +0.041 |
| Random Word Deletion | B.79 | delete random words | +0.011 | +0.034 |
| Replace Abbreviations and Acronyms | B.82 | replace acronyms with full length expression and v.v | +0.020 | +0.042 |
| Sentence Reordering | B.88 | reorder sentences | +0.024 | +0.034 |
| Shuffle Within Segments | B.90 | shuffle tokens in mentions | +0.021 | +0.041 |
| Synonym Insertion | B.100 | insert synonym before word | +0.019 | +**0.043** |
| Synonym Substitution | B.101 | substitute word with synonym | +0.023 | +0.040 |
| Subsequence Substitution for Sequence Tagging | B.103 | replace sequence with another sequence with same POS tags | +0.019 | +0.033 |
| Transformer Fill | B.106 | replace tokens using language model | +0.022 | +0.041 |
| Random Insert | | insert random tokens | +0.020 | +**0.043** |
| Random Swap | | swap position of tokens | +**0.029** | +0.033 |

extraction approaches we used. Back-translation techniques, such as *B.8*, *B.62*, and especially *B.26* do not provide benefits in the MD and RE tasks, that would warrant their additional needs in hardware (GPUs), and runtime, which was several orders of magnitude higher, compared to simpler augmentation techniques. We therefore have to answer RQ2 with "No".

To answer research question RQ3, we defined three characteristics of textual data in Sect. 4 that are changed by the data augmentation techniques we selected. These characteristics are visualized in Figs. 6a and 6b. Figure 6a shows the "landscape" of data augmentation techniques evaluated in this paper. Three groups of techniques emerge. The first one is a group of techniques that only marginally increase the number of tokens in mentions, and keep the size of the




(a) (b)

**Fig. 6.** *(a)*: Effects on vocabulary size and the average length of mentions in tokens. *(b)*: Effects of techniques on relation direction.

vocabulary roughly the same. These techniques mainly change the context (i.e., the text that does not contain immediately process relevant information), or the structure of the text (i.e., modify punctuation, or change the order of tokens). Techniques in the second group do not modify the vocabulary, but have a significant impact on the number of tokens in a given mention. These augmentations can theoretically be useful for the robustness of MD extraction models, but only have a moderate impact in our experiments, using the PET dataset. We count *Random Insertion*, *Filler Word Augmentation*, but also *Random Word Deletion* towards this group, see Fig. 2 for an example taken from the augmented data. The final group of techniques increases the size of the vocabulary, while keeping mention lengths roughly the same. These techniques are paraphrasing, aimed at preserving semantics and the structure of textual data. Techniques using WordNet to insert or substitute synonyms (*B.100*, *B.101*, as well as back translation methods (*B.62*, *B.26*) fall in this group. Figure 2 shows a sentence that is augmented with the back translation method *B.26*.

Figure 6b shows the changes in directionality certain data augmentation techniques have. Most techniques preserve the direction of relation examples in the data, with the exception of techniques *B.88* (Sentence Reordering) and *B.24* (Concatenate Random Sentences). Based on our experiments, this change seems to be less useful than other augmentations. The improvement of B.88 is among the worst ones of all techniques. In future work it could be interesting to investigate, if selectively augmenting only certain types of relations can be helpful. Also, having a more diverse test set, i.e., texts from different sources, like employee notes, handbooks, and interview notes, might change the usefulness of directionality changing data augmentation techniques.

## 6 Conclusion and Future Work

In this paper we evaluated established data augmentation techniques for use in the MD and RE steps of extracting process relevant information from natural language texts for use in the automated generation of business process models. To this end we selected a total of 19 distinct methods from related work, which are suitable for the given data.

We discuss several characteristics these selected data augmentation techniques change in the original data and how they relate to differences in usefulness of certain techniques for either the MD or RE task. We found that many of them are useful for improving the accuracy of the current state of the art machine learning models on the PET dataset for automated business process model generation from natural language text. For the RE model the $F_1$ score could be improved by up to 4.5% points, the MD model was improved by up to 2.9% points. Our findings enable researchers in the field of process model generation from natural language text to make more efficient use of the limited data available to them, enabling more precise and robust machine learning methods for extracting business process relevant information.

In future work, we want to analyze how targeted data augmentation can be used to improve extraction of certain types of mentions or relations, tackling the problem of data imbalance. Additionally we want to explore adaptive data augmentation, where samples are selected for augmentation by their value for model training, e.g., measured by the number of wrong predictions.

## References

1. van der Aa, H., Di Ciccio, C., Leopold, H., Reijers, H.A.: Extracting declarative process models from natural language. In: Giorgini, P., Weber, B. (eds.) CAiSE 2019. LNCS, vol. 11483, pp. 365–382. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21290-2_23
2. Ackermann, L., Neuberger, J., Jablonski, S.: Data-driven annotation of textual process descriptions based on formal meaning representations. In: La Rosa, M., Sadiq, S., Teniente, E. (eds.) CAiSE 2021. LNCS, vol. 12751, pp. 75–90. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79382-1_5
3. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: SIGKDD International Conference on Knowledge Discovery & Data Mining (2019)
4. Bellan, P., Dragoni, M., Ghidini, C.: Assisted process knowledge graph building using pre-trained language models. In: Dovier, A., Montanari, A., Orlandini, A. (eds.) AIxIA 2022. LNCS, vol. 13796, pp. 60–74. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-27181-6_5
5. Bellan, P., Dragoni, M., Ghidini, C.: Extracting business process entities and relations from text using pre-trained language models and in-context learning. In: Almeida, J.P.A., Karastoyanova, D., Guizzardi, G., Montali, M., Maggi, F.M., Fonseca, C.M. (eds.) EDOC 2022. LNCS, vol. 13585, pp. 182–199. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17604-3_11

6. 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)
7. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, vol. 24 (2011)
8. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, vol. 26 (2013)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint (2018)
10. Dhole, K.D., et al.: NL-Augmenter: a framework for task-sensitive natural language augmentation. arXiv preprint (2021)
11. Erdengasileng, A., et al.: Pre-trained models, data augmentation, and ensemble learning for biomedical information extraction and document classification. Database **2022**, baac066 (2022)
12. Feng, S.Y., et al.: A survey of data augmentation approaches for NLP (2021)
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. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 482–496. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21640-4_36
15. Jiang, Z., Han, J., Sisman, B., Dong, X.L.: CoRI: collective relation integration with data augmentation for open information extraction. arXiv preprint (2021)
16. Kampik, T., et al.: Large process models: business process management in the age of generative AI. arXiv preprint arXiv:2309.00900 (2023)
17. Käppel, M., Jablonski, S.: Model-agnostic event log augmentation for predictive process monitoring. In: Indulska, M., Reinhartz-Berger, I., Cetina, C., Pastor, O. (eds.) CAiSE 2023. LNCS, vol. 13901, pp. 381–397. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-34560-9_23
18. Käppel, M., Jablonski, S., Schönig, S.: Evaluating predictive business process monitoring approaches on small event logs. In: Paiva, A.C.R., Cavalli, A.R., Ventura Martins, P., Pérez-Castillo, R. (eds.) QUATIC 2021. CCIS, vol. 1439, pp. 167–182. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85347-1_13
19. Käppel, M., Schönig, S., Jablonski, S.: Leveraging small sample learning for business process management. Inf. Softw. Technol. **132**, 106472 (2021)
20. Klievtsova, N., Benzin, J.V., Kampik, T., Mangler, J., Rinderle-Ma, S.: Conversational process modelling: state of the art, applications, and implications in practice. In: Di Francescomarino, C., Burattin, A., Janiesch, C., Sadiq, S. (eds.) BPM 2023. LNBIP, vol. 490, pp. 319–336. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-41623-1_19
21. Liu, J., Chen, Y., Xu, J.: Machine reading comprehension as data augmentation: a case study on implicit event argument extraction. In: EMNLP (2021)
22. Miller, G.A.: WordNet: a lexical database for English. CACM **38**(11), 39–41 (1995)
23. Neuberger, J., Ackermann, L., Jablonski, S.: Beyond rule-based named entity recognition and relation extraction for process model generation from natural language text. In: Sellami, M., Vidal, M.E., van Dongen, B., Gaaloul, W., Panetto, H. (eds.) CoopIS 2023. LNCS, vol. 14353, pp. 179–197. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-46846-9_10

24. Quishpi, L., Carmona, J., Padró, L.: Extracting annotations from textual descriptions of processes. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 184–201. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_11
25. Shorten, C., Khoshgoftaar, T.M., Furht, B.: Text data augmentation for deep learning. J. Big Data **8**, 1–34 (2021)
26. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
27. Yao, Y., et al.: DocRED: a large-scale document-level relation extraction dataset. arXiv preprint (2019)
28. Zoran, D., Weiss, Y.: Scale invariance and noise in natural images. In: ICCV (2009)

# Chapter 9

# A Universal Prompting Strategy for Extracting Process Model Information from Natural Language Text Using Large Language Models

Neuberger, J., Ackermann, L., van der Aa, H., Jablonski, S. (2025). A Universal Prompting Strategy for Extracting Process Model Information from Natural Language Text Using Large Language Models. In: Maass, W., Han, H., Yasar, H., Multari, N. (eds) *Conceptual Modeling. ER 2024.* Lecture Notes in Computer Science, vol 15238. Springer, Cham. https://doi.org/10.1007/978-3-031-75872-0_3



**Contribution statement.** The concept for a universal prompting strategy was developed by Julian Neuberger (JN) and Lars Ackermann (LA). Reviewing previous work on prompting and integrating best practices was done by LA. JN wrote the study framework. JN and LA conducted the experiments. The main publication text was written by JN, Han van der Aa (HA), and LA, with additional contributions by Stefan Jablonski (SJ). Figures were created by JN and LA. All authors revised the publication. JN is the corresponding author.

# A Universal Prompting Strategy for Extracting Process Model Information from Natural Language Text Using Large Language Models

Julian Neuberger[1(✉)], Lars Ackermann[1], Han van der Aa[2], and Stefan Jablonski[1]

[1] University of Bayreuth, Bayreuth, Germany
{Julian.Neuberger,Lars.Ackermann,Stefan.Jablonski}@uni-bayreuth.de
[2] University of Vienna, Vienna, Austria
han.van.der.aa@univie.ac.at

**Abstract.** Over the past decade, extensive research efforts have been dedicated to the extraction of information from textual process descriptions. Despite the remarkable progress witnessed in natural language processing (NLP), information extraction within the Business Process Management domain remains predominantly reliant on rule-based systems and machine learning methodologies. Data scarcity has so far prevented the successful application of deep learning techniques. However, the rapid progress in generative large language models (LLMs) makes it possible to solve many NLP tasks with very high quality without the need for extensive data. Therefore, we systematically investigate the potential of LLMs for extracting information from textual process descriptions, targeting the detection of process elements such as activities and actors, and relations between them. Based on a novel prompting strategy, we show that LLMs are able to outperform state-of-the-art machine learning approaches with absolute performance improvements of up to 8% $F_1$ score across three different datasets. We evaluate our prompting strategy on eight different LLMs, showing it is universally applicable, while also analyzing the impact of certain prompt parts on extraction quality. The number of example texts, the specificity of definitions, and the rigour of format instructions are identified as key for improving the accuracy of extracted information. Our code, prompts, and data are publicly available at https://github.com/JulianNeuberger/llm-process-generation/tree/er2024.

**Keywords:** Process Information Extraction · Large Language Models · AI-assisted Conceptual Modeling · Business Process Modeling

## 1 Introduction

In the field of Business Process Management (BPM), process models are established tools for designing, implementing, enacting, and analyzing enterprise pro-

cesses [12]. However, the manual creation of these models is very time-consuming and accounts for around 60% of the total time spent on process management [16]. In order to reduce this effort, the automatic creation of these models based on a variety of information sources is a research focus in the field of BPM [5,16]. In this respect, the paper at hand contributes to the extraction of process-relevant information from natural language information sources.

Information on organizational processes is frequently contained in a range of textual documents, such as process descriptions, rules and regulations, and work instructions [1,4]. Recognizing this, a variety of techniques has been developed that aim to automatically extract process information from texts in order to subsequently turn it into process models [2,8,16]. This two-step procedure, in which information is extracted first and turned into a process model second, comes with several advantages in comparison to a direct text-to-model transformation approach: (1) The result quality can be evaluated with established means from the information-extraction domain, (2) extracted information can be transformed in more than one target process modeling language[1], and (3) it is possible to use extracted process information for other purposes such as, for instance, compliance checking, formal reasoning [3,26], and process querying [19].

The goal, scope, and challenges of information extraction depend on the input document type and content, as well as the desired output, i.e., the information to be extracted. Still, the extraction of process information from text generally involves: (1) the identification of textual mentions of process entities, such as activities, process participants, and business objects, and (2) relations between these entities, such as sequential dependencies, exclusivity, and assignments (e.g., who performs which step). As an illustration, Fig. 1 shows a fragment of a textual description and two instantiations of the extraction task, focused on the information necessary for a model in Business Process Model and Notation (BPMN)[2] model [9] (upper part) and for declarative process modeling [2] (lower part). As shown, they involve different entities and relations, which each need to be inferred from the unstructured textual input.

A key problem is that the extraction of process information is still largely rule-based [23]. However, crafting useful rules is complicated, requires an extensive understanding of the process itself, and the rules are hard to transfer across organizations or text sources. To overcome this, recent work proposed the use of machine learning techniques [23], though these are hampered by data scarcity. Work that strives towards using pre-trained generative LLMs, e.g., GPT-3 [8] aims to circumvent this concern. However, the work in [8] only presents a preliminary study, with limitations in terms of analyzed datasets, extracted information, and result discussion. Therefore, this paper aims to provide deeper insights into the usability of LLMs for process information extraction and specifically includes the following core contributions: **(I)** It presents challenges that make the

[1] This is inspired by the paradigm of *interlingua-based machine translation* [27], which reduces the number of translation systems for n languages from $n^2$ to $2n$.

[2] https://www.omg.org/bpmn/, accessed June 2, 2024.

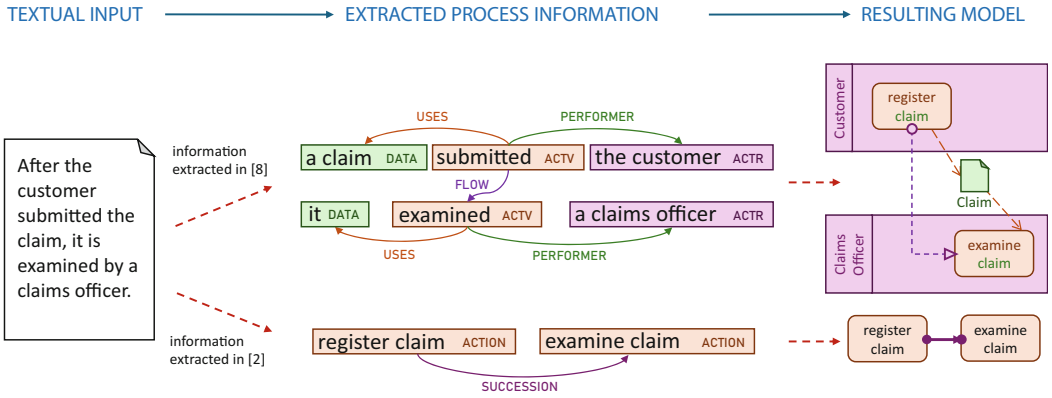



**Fig. 1.** Fragment of a larger text describing a business process of an insurance company. Different methodologies may extract different process relevant information, depending on the target modelling notation or use case.

extraction of process-relevant information in particular a difficult task (Sect. 2). **(II)** As our main contribution, it proposes a novel, task-specific, and rigorously empirically validated prompting strategy for solving the aforementioned information extraction tasks (Sect. 4). **(III)** It provides the currently most comprehensive study of using LLMs for extracting business process relevant information from natural language text (Sects. 5 and 6). To this end we rigorously compare our prompting strategy on multiple datasets with state-of-the-art approaches and achieve up to 7% higher absolute $F_1$ scores compared to machine learning methods and up to 8% compared to rule-based methods. **(IV)** By testing our prompting strategy with eight state-of-the-art LLMs, we empirically demonstrate the generality of both our results and the applicability of our prompting strategy. **(V)** An ablation study (Sect. 6.2) shows that common best practices in prompt engineering are only of limited use for process information extraction. Thus, we also define guidelines for using LLMs for process information extraction (Sect. 6.4).

The rest of this paper is structured as follows. Section 2 describes the information extraction tasks and its challenges in detail. Section 3 summarizes the current state of the art in dealing with these tasks. After that we, describe our prompting strategy, the experiments, and corresponding results (Sects. 4, 5, 6). Section 7 describes limitations and future work.

## 2 Task Descriptions and Challenges

In this section, we describe the three main (sub)tasks of process information extraction (Sect. 2.1), before highlighting a range of challenges associated with such extraction and with the use of LLMs for it (Sects. 2.2, 2.3).

### 2.1 Task Descriptions

Our work focuses on three established subtasks of (process) information extraction from text: Mention Detection (MD), Entity Resolution (ER), and Relation Extraction (RE) [2,7,23,26].

**Mention Detection** (MD) is concerned with finding and extracting text fragments that contain process relevant information, such as activities (or actions), process-relevant objects or data (i.e., business objects), or involved persons and departments (i.e., actors). For instance, in Fig. 1, the upper example shows mentions of *data*, *actions*, and *actor*, whereas the lower one focuses on *activities*. This definition is similar to Named Entity Recognition (NER), though we also extract spans not covered by the traditional definition of NER, e.g., activities.

**Entity Resolution** (ER) aims to recognize when different mentions refer to the same process entity. For example, in Fig. 1, successful ER would identify that the word *it* in "it is examined" corresponds to the *claim* mentioned in the previous phrase. Another common example is using ER to recognize that the same actor (across mentions) performs different steps. ER is a super-set of co-reference resolution and anaphora resolution [29] and is a crucial step when dealing with process-related texts, which frequently involve repeated mentions across sentences or even paragraphs [23].

**Relation Extraction** (RE) is the task of detecting and classifying relations between mentions. Relations are usually directed and have one (unary relation), or two (binary relations) arguments. For instance, the upper example in Fig. 1 shows three kinds of relations: *uses* signals which data objects are used by an activity, *performer* captures which actor performed an activity, and *flow* captures a sequential relation between two activities. RE is crucial when it comes to information extraction in our context, given that processes inherently involve process steps (i.e., activities) that are connected to each other through relations. Note that we regard constraint extraction [2,26] (CE), which relates to declarative process modeling, as an RE problem: constraints have one or two arguments, are directed, and carry type information (e.g., *Succession*, *Init*).

### 2.2 Challenges of Process Information Extraction from Text

Information extraction, a common task in natural language processing (NLP), faces general challenges, which are also well-known in BPM literature [1,15], and often central elements of interest in the design of rule-based and learning-based systems alike [2,23]. Simply using LLMs for process information extraction solves some of these challenges and justifies the investigation of their applicability.

In the context of process-related texts ***Linguistic Variance*** means that the same behavior or process characteristics can be described in a variety of ways such as, for instance, active and passive voice. ***Context Cues*** are a challenge in that single words can fundamentally alter the meaning of a process description (e.g., *"first, a claim is created" and inverted semantics in "finally, a claim is created"*. Processes are typically described in sequential form, although they usually

contain branches (e.g. XOR decision branches). This results in ***Long-distance Relations*** that existing approaches struggle with [23] or cannot handle [2]. ***Implicit and Ambiguous Information*** such as the "examination target" in *"after registering the file in the database, it needs to be examined"* needs to be interpreted [3,15]. Research indicates a negative correlation between correctness of extracted process information and ***Text Length*** [7]. Finally, the application of deep learning is hindered by the fact that the largest available data set contains only 45 human-annotated process descriptions [9] (***Small Datasets***).

LLMs are able to overcome the above challenges [32], which is why this paper analyzes their suitability for process information extraction, as proposed in previous work [6,8]. However, LLMs require great care in the formulation of the input (prompts) [8,22,31–33]. In [22] authors argue: "a good prompt can be worth hundreds of labeled data points". For this reason, the core of the present work lies in the development (Sect. 4) and evaluation (Sects. 5 and 6.2) of suitable prompts for process information extraction.

### 2.3 Challenges of Process Information Extraction Using LLMs

Using LLMs for process information extraction from texts helps with linguistic challenges, but adds itself several additional challenges. We discuss these here and reference them later in Sect. 6.4 to show how we can deal with them.

**(C1) Limited Output Control.** Input and output consist of plain text. Given that the input for inference is raw text, it inherently suits LLMs for our tasks (cf. Section 2.1). However, as the expected output should adhere to a specific schema, it becomes necessary to instruct the LLM to conform to this schema. Moreover, this principle necessitates a robust output parser, as LLMs tend to exhibit variability in their output, which presently cannot be entirely eradicated. Having only limited control over generated output is especially problematic for the BPM domain, where definitions for relevant information often overlap, e.g., *actions* (just predicate) versus *activities* (predicate and object).

**(C2) Input Presentation Dependencies.** Although LLMs provide an interface for natural language input, the quantity, form and level of detail must be carefully matched to the task at hand. The LLM faces the challenge of determining the importance of the input components. Furthermore, while LLMs emulate human reasoning, the interpretation of inputs may diverge significantly from that of human beings, thereby rendering prompt optimization a trial-and-error process. This challenge is further aggravated by some elements of process models, that are complex to explain concisely, e.g., parallel and exclusive workflows.

**(C3) Black-box.** Deep learning methods generally suffer from challenges concerning explainability of predictions [34], which is also true for LLMs. Contrary to classical learning methods, such as decision trees, LLMs offer no fail safe

mechanism to validate extraction rules. This is problematic for business process information extraction in particular, since recent work focuses on "human-in-the-loop" systems [30], where the human must be able to follow system decisions.

**(C4) Data Unawareness.** In contrast to generative AI models trained on task-specific data, an LLM is usually not aware of the particular dataset it is tasked to process. Thus, using LLMs to process a particular dataset requires to form instructions that precisely describe all relevant details of a dataset. The generalizing capabilities of LLMs can be an additional hurdle in this context, especially, when declarative process models are concerned, where a multitude of constraint types exist. The LLM is likely to know of these through the pre-training process, and therefore may extract irrelevant ones for a given dataset.

**(C5) Costly Experiments.** Applying LLMs usually requires usage of commercial APIs (e.g., OpenAI), which come with downsides: (i) a token limit restricting the maximum input and output size and (ii) fees based on the number of tokens processed. In view of the many possible variations in the influencing parameters, conducting experiments can be cost-intensive. This is especially true for the BPM domain, where the density of information in process descriptions is very high, needing many output tokens to extract and encode it.

## 3 Related Work

Related approaches are divided into rule-based, machine learning (ML)-based, deep learning-based, and LLM-based process information extraction. An approach is considered related if it solves at least one of the tasks in Sect. 2.1.

**Rule-Based Approaches.** Rule-based approaches leverage linguistic features to extract information from natural language process descriptions through explicitly coded mapping rules. For instance, Friedrich et al.'s seminal work [16] employs syntax features and word information from a lexical database to identify patterns at both sentence and document levels for BPMN model creation. Other approaches like those in [26,28] adopt similar techniques for automatic text annotation, employing regular expressions for syntactic dependency trees, and part-of-speech tags, showcasing superior performance on novel datasets. Additionally, [2] presents a rule-based technique, currently leading in extracting declarative process models from raw text using syntax parsing and word-level features. Similar advancements are seen in [9,14], the latter integrating ML-based entity MD with subsequent rule-based RE. Furthermore, [21] focuses on extracting Dynamic Condition Response (DCR) graphs. Recent studies suggest that while rule-based approaches can be tailored to specific tasks and data sets, they can hardly deal with ambiguity and linguistic variance. [8,23].

**ML-Based Approaches.** [23] presents a ML extraction pipeline based on [9] and is used as a baseline for our comparative evaluation (Sect. 6). The deep-learning approach presented in [25] classifies text fragments analyzing the input text on several levels of granularity. However, extracting these fragments is not part of the approach, which simplifies the task of MD to mention classification, i.e., locating the information to extract is omitted. Though the work presented in [6] overcomes this limitation and outperforms the approach, it does not support RE. In general, techniques of this paradigm either struggle to deal with linguistic variability and ambiguity, or they require vast amounts of training data, making them particularly unfeasible for small datasets (see Sect. 2.2).

**LLM-Based Approaches.** Bellan et al. [8] utilize pre-trained LLMs to cope with data scarcity, yet their approach exhibits three primary weaknesses: *(i)* It is restricted to a subset of entity types, namely *activities*, *participants*, a *performs* relation, and a *direct-consequences* relation, *(ii)* it lacks strict output formatting, hindering automated result processing, unlike our prompting strategy, and *(iii)* its evaluation is limited to 7 of the 45 process descriptions from the PET dataset, whereas we evaluate our modular prompt on the entire dataset, plus two validation datasets. Thus, direct comparison between [8] and our work is not feasible. Nonetheless, as in [8], our modular prompt also descriptive instructions with input examples accompanied by their expected outputs. Although process models are generated using LLMs in [18], the work is not comparable to ours, as [18] requires human involvement and only supports the extraction of activities and their arrangement in a directed graph (e.g., actors and data are missing).

## 4 LLM-Based Process Model Extraction from Text

Extracting process information with LLMs requires a *prompt design* that addresses the challenges mentioned in Sect. 2.3. Thus, a prompt structure consisting of the three modules *Context*, *Task Description* and *Restrictions* is described below at the example of the Mention Detection task. However, the prompting strategies for the remaining tasks (Sect. 2.1) are analogous.

**High-Level Prompt Structure.** LLMs take freely formulated text as input, which is called the prompt. To this end we base our prompts on an ablation study (Sect. 6.2), which is used to identify beneficial and detrimental prompt components. To do this, we first need a modular prompt design so that we can specifically remove individual components in the study to examine their benefits and ultimately to only keep the advantageous components. Adhering to the best practices outlined in [31], our initial prompt design is structured into three modules (see Fig. 2): *(A)* a context description framing the process information extraction task on a high level, *(B)* a detailed task description, and *(C)* constraints that further restrict the context and the output format, and

contains disambiguation hints. To design potentially relevant components for all three modules we rely on general design patterns [11,22,31,33]. Therefore, in the next subsection, the three modules are specified and discussed in terms of how they address the LLM-specific challenges outlined in Sect. 2.3.

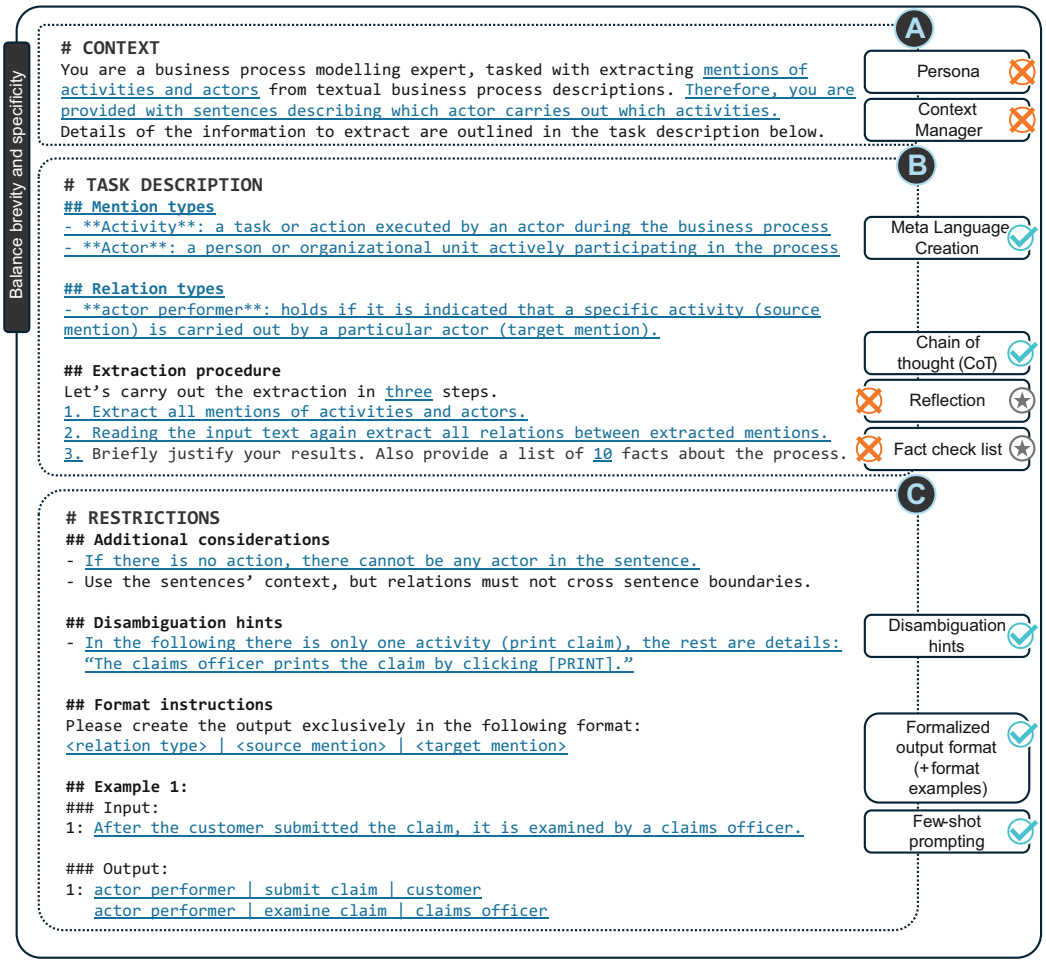



**Fig. 2.** Modular prompt structure (underlined = task-specific content, boxes = design pattern, ✓ = useful, ✗ = non-useful, ★ = use in prompt engineering).

**Context and Task Description.** In module *(A)* we use the *persona* design pattern [33] to control the language style of generation results. We assign it the role of a process modeling expert. This is followed by the *context manager* design pattern [33], which includes a general description of the information extraction task (i.e., objectives and a description of the input specifics). This limits the information basis the LLM may use, mitigating the risk of hallucination.

Module *(B)* is mainly concerned with defining the specifics of the process information extraction task. Its backbone is the *creation of a meta language*

[33], which defines the types of elements to extract from the input text. Figure 2 provides an example that defines activities and actors as mention types and a relation called *actor performer* that associates actions with their performers, following our running example from Fig. 1. Another widespread best practice is known as *chain of thought (CoT)* [31,32], whereby the actual task is broken down into individual steps. Thus, our prompt divides the relation extraction task into two steps that separate the extraction of mentions from the prediction of their relations and a third step, which combines two more best practices, i.e., generating a *list of facts* about the process and *reflection* about the results [33]. These cause the LLM to elaborate both on the input and on its own output, which allows experts to validate the extracted information and has also been shown to have a positive impact on extraction performance [31,32].

**Restrictions.** The last prompt module *(C)* defines expectations towards the LLM's output. *Additional considerations* include rules for the extraction task, such as that an actor can only be described as such if the action it performs is also named (compare Fig. 2). *Disambiguation hints* are particularly useful for information types that are hard to distinguish from other information types or irrelevant information. In Fig. 2 it is intended to guide the LLM what makes up an *Activity*, if the input gives additional, irrelevant specifics. Prompts further include a schematic definition of a *formal output format* [33], which for the exemplary prompt is a tuple of a relation type, a source mention and a target mention, each separated by a pipe symbol. The definition is complemented by an (out-of-domain) example. Finally, *few-shot prompting* [22,31] dynamically adds examples for input and corresponding output. For the current paper few-shot samples are pairs of raw textual process descriptions and a task-dependent set of process-relevant information (e.g., actor mentions). This design pattern and best practice is known to alleviate the issue of *data unawareness* of LLMs [17,20].

## 5 Experiment Setup

In this section we define the experiments we run to evaluate the usefulness of LLMs for process information extraction[3]. It covers an overview of the datasets we use, including the respective baselines, and a definition of metrics we apply.

We use three well-known datasets for evaluating our prompts. One of these (PET) represents the current state of the art, both in terms of size, as well as the process information techniques developed for it. The other datasets feature different characteristics, making them relevant for validation experiments. This lets us gain insights into the robustness of an LLM as a process information extractor, as well as how it behaves when applied to other process modeling languages. We call the best approaches for extracting the information from these datasets *baselines*, and use them in Sect. 6 as comparisons with various LLMs.

[3] Code at https://github.com/JulianNeuberger/llm-process-generation/tree/er2024 .

**PET** [9]: This is the largest dataset currently available. It contains 45 documents with annotations for information especially useful for creating process models in BPMN. These include 7 types of mentions such as activities, actors, data objects, but also 6 relation types. These cover the behavioural process perspective (*Flow*), data perspective (*uses*, and organizational perspective (*actor, performer*). Additionally, this dataset features relations that span multiple sentences. It therefore tests the ability of approaches to reason across wider spans of text. We use an extended version of this dataset, which includes data for the ER task, as presented in [23]. The currently best approach for extracting information is using conditional random fields for MD, a pre-trained neural coreference resolver for ER, and a decision tree ensemble for RE [23]. We use scores as reported in [23], which have corresponding publicly available code and can be reproduced with it.

**DECON** [2]: This collection of 17 textual process descriptions is annotated with a set of 5 Declare [24] Constraint types between business process relevant activities. Additionally, constraints may be negated, as well as unary, i.e., constraining a single action. Annotations are given on a sentence level, and only sentences that describe at least one constraint are contained in the dataset. The expected (ground-truth) activities are already transformed into Declare-conform phrases, i.e., the activity description "*The claim is registered*" should be extracted as "*register claim*". It does not contain an approach for MD in isolation, and only contains mentions of type action. The authors of [2] propose a rule-based approach combining multiple NLP techniques, e.g., *typed dependency relations*.

**ATDP** [26]: This dataset uses 18 textual descriptions, that largely overlap with the ones from [2], but also contains sentences, that describe no constraint. As such, this dataset tests approaches for their ability to judge whether or not sentences contain relevant information, before extracting constraints. Furthermore, the set of constraints was expanded to eight types. Additionally, this dataset also provides annotations of actions, conditions, entities, and events, which we used in an MD setting. Quishpi et al. proposed a rule-based ensemble of patterns for MD and CE on typed dependency structures [26].

We use the well established metrics *Precision* $P$ and *Recall* $R$ for our experiments. $P$ is a measure of how well an extraction approach is able to avoid false positives, i.e., assigning the wrong type to mentions and relations, or extracting them, where they are not expected. $R$ on the other hand measures how much of the expected information (true positives) is found. The two metrics are typically aggregated via their harmonic mean $F_1 = 2\frac{P \cdot R}{P+R}$. Following [26], we use $P = \frac{\#correct}{\#pred}$ and $R = \frac{\#correct}{\#gold}$, with $\#correct$ as the number of correct predictions, $\#pred$ the number of total predictions, and $\#gold$ as the number of expected mentions or relations. For a fair assessment, we count predictions as correct in exactly the way described by the work we compare the LLM to.

## 6 Results

Table 1 shows the results we observed when running the experiments as described in Sect. 5 with an optimized prompt, that follows the recommendations we found in our study of best practices (Sect. 6.2). All results use GPT-4o, the latest version of OpenAI's GPT with *temperature* = 0. *top_p* is unchanged, as per OpenAI's recommendation, when using temperature based sampling.[4]

For the reference dataset PET, our experiments show that GPT-4o is capable of an absolute $F_1$ score improvement of 5% for MD, 22% for ER, and 17% for RE. Remarkably, for RE, GPT-4o is able to match and outperform the machine learnt baseline, which was trained on 36 manually annotated documents [23], without any labeled data (zero-shot). For MD it reaches similar scores, even when not given any examples, compared to the machine learnt baseline, which was trained using 36 manually annotated documents [9]. For real-world application this means that LLMs can be used in business process information extraction scenarios, even if the organization has not a single manually annotated training example. This is an exciting find, as it promises significant speed-up of model creating tasks of practitioners across business domains.

**Table 1.** Results for each dataset and the different extraction stages, compared to baseline results using GPT-4o.

| | Dataset | DECON | | | ATDP | | | PET | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Metric | $P$ | $R$ | $F_1$ | $P$ | $R$ | $F_1$ | $P$ | $R$ | $F_1$ |
| MentionDetection | Baseline | *no baseline* | | | 0.62 | **0.82** | 0.71 | **0.73** | 0.64 | 0.69 |
| | Zero-shot | 0.72 | 0.75 | 0.73 | 0.58 | 0.77 | 0.66 | 0.65 | 0.71 | 0.68 |
| | 1-shot | 0.87 | **0.80** | 0.83 | 0.63 | 0.77 | 0.69 | 0.72 | 0.75 | 0.73 |
| | 3-shot | **0.88** | 0.79 | **0.83** | **0.68** | 0.79 | **0.73** | 0.72 | **0.77** | **0.74** |
| EntityResolution | Baseline | *no data* | | | *no data* | | | 0.55 | **0.51** | 0.52 |
| | Zero-shot | | | | | | | 0.67 | 0.55 | 0.60 |
| | 1-shot | | | | | | | 0.76 | **0.70** | 0.73 |
| | 3-shot | | | | | | | **0.79** | **0.70** | **0.74** |
| RelationExtraction | Baseline | 0.77 | 0.72 | 0.74 | **0.58** | 0.64 | 0.61 | 0.79 | 0.66 | 0.72 |
| | Zero-shot | 0.66 | 0.75 | 0.70 | 0.49 | 0.66 | 0.57 | 0.88 | 0.85 | 0.86 |
| | 1-shot | 0.76 | 0.82 | 0.79 | **0.58** | **0.73** | **0.64** | 0.90 | 0.89 | 0.89 |
| | 3-shot | **0.79** | **0.85** | **0.82** | **0.58** | 0.72 | **0.64** | **0.90** | **0.89** | **0.89** |

When evaluating on the validation datasets (cf. Section 5), we found that GPT-4o is able to match and out-perform the rule-based systems in all cases, most notably improving $F_1$ scores for RE on dataset DECON by an absolute 8%. Our result for MD on dataset DECON has no corresponding baseline, as the

[4] see OpenAI's source code, accessed June 3, 2024.

authors of [2] did not report values for MD in isolation. Errors and ambiguities are common in dataset ATDP hindering machine learning methods in learning valid extraction rules. This also adversely affects the extraction accuracy of GPT-4, when extracting the same types of constraints in the ATDP dataset compared to the DECON dataset. We discuss this further in Sect. 6.4. Since the importance of ER only recently gained attention [23,26], the reference dataset PET is currently the only dataset providing data for evaluation of this task.

### 6.1 Model Comparison

We originally developed our prompts for GPT-4 version *GPT-4-0125-preview*, to assess how well our prompting strategy generalizes to other models we prompted a total of eight models for the MD and RE tasks on PET. We selected models following AlpacaEval[5], which is designed for testing the instruction following capabilities of LMMs [13]. At the time of writing model *YI Large Preview* was not publicly accessible and could not be considered in our comparison, even though it ranked third on AlpacaEval.

Results for the comparison can be found in Table 2. We set the temperature for all models to 0. All GPT models perform on similar levels, with the exception of GPT3.5, which is significantly smaller compared to GPT-4 models. For the zero-shot RE task GPT3.5 even failed to produce responses for most documents, leading to very low recall. Claude3 Opus seems to be as capable as GPT-4, its smaller variant Sonnet performs significantly worse on zero-shot tasks, but is able to produce comparable results given three examples. Llama 3 70B Instruct is an open-weight model and could be run locally, i.e., it is useful for using our prompting strategy in scenarios where sending data to an API is not possible. Llama 3 70B Instruct seems to be nearly as capable as the closed-weights Claude 3 Sonnet and is therefore viable in a few-shot setting.

**Table 2.** Comparison of our prompts across different models. Best results per task and metric are set **bold**.

| Task | PET MD (Zero-shot) | | | PET MD (3-shot) | | | PET RE (Zero-shot) | | | PET RE (3-shot) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | $P$ | $R$ | $F_1$ | $P$ | $R$ | $F_1$ | $P$ | $R$ | $F_1$ | $P$ | $R$ | $F_1$ |
| GPT-4o | 0.58 | 0.69 | 0.63 | 0.68 | 0.77 | 0.72 | **0.88** | **0.85** | **0.86** | 0.90 | 0.89 | 0.89 |
| GPT-4-2024-04-09 | 0.63 | 0.67 | 0.65 | **0.73** | 0.76 | **0.74** | 0.87 | 0.79 | 0.83 | 0.89 | 0.88 | 0.88 |
| GPT-4-0125-preview | **0.65** | 0.71 | **0.68** | 0.72 | 0.77 | **0.74** | 0.87 | **0.85** | **0.86** | 0.89 | 0.87 | 0.88 |
| GPT-3.5-0125 | 0.35 | 0.50 | 0.42 | 0.51 | 0.70 | 0.59 | 0.51 | 0.06 | 0.11 | 0.74 | 0.64 | 0.69 |
| Claude 3 Opus | 0.55 | **0.72** | 0.63 | 0.66 | **0.80** | 0.73 | 0.86 | **0.85** | **0.86** | **0.92** | **0.91** | **0.91** |
| Claude 3 Sonnet | 0.46 | 0.65 | 0.54 | 0.63 | 0.78 | 0.70 | 0.78 | 0.67 | 0.72 | 0.91 | 0.87 | 0.89 |
| Llama 3 70B Instruct | 0.56 | 0.64 | 0.59 | 0.62 | 0.71 | 0.67 | 0.76 | 0.66 | 0.70 | 0.88 | 0.81 | 0.84 |
| Qwen1.5 72B Chat | 0.32 | 0.33 | 0.33 | 0.53 | 0.65 | 0.59 | 0.61 | 0.65 | 0.63 | 0.74 | 0.77 | 0.75 |

[5] See https://tatsu-lab.github.io/alpaca_eval/, last accessed May 30, 2024.

### 6.2 Ablation Study

We conduct an ablation study to assess the usefulness of the best practices presented in Sect. 4 and to measure the impact of the prompt's main components. This study is run on the reference dataset (PET), as it is the largest one and used by recent publications [8,9,23]. To obtain a baseline for the tasks of MD and RE, we use a prompt that implements the best practices as shown in Fig. 2 and run it on the *GPT-4-0125-preview* model. We then purposefully remove specific components from this prompt, namely the *format examples*, the *context manager*, the *persona*, the definition of mention and relation types (*meta language*), the instruction to think in several steps (*chain of thought*), any *disambiguation* hints, and the instruction to generate explanations (*reflection*) and a *fact check list* about the process. Additionally, we also use a prompt with very short descriptions of relations and types (*balancing brevity and specificity*). We run each prompt in the zero-shot setting and record the observed $F_1$ score, as well as the parsing errors that occurred.

Table 3 provides detailed results. Removing examples has a significant negative effect ($-0.22$ for MD and $-0.07$ for RE), mainly rooted in the number of parsing errors that are made (919 for MD), as well as directionality of relations for the RE task (confusing source and target mentions). Removing the context manager and persona only has minor effects ($\pm 0.01$ per task), suggesting lower relevance for process information extraction compared to other NLP settings.

**Table 3.** Changes in $F_1$ score of GPT-4, without specific prompt components given in Fig. 2. Column *relative* $F_1$ shows difference to the baseline prompt, *Useful* shows a ✓, if we recommend this component in prompts for process information extraction and ★ for prompt engineering and data curating only.

| | Mention Detection (MD) | | | Relation Extraction (RE) | | | |
|---|---|---|---|---|---|---|---|
| Experiment | Relative $F_1$ | Absolute $F_1$ | Parsing Errors | Relative $F_1$ | Absolute $F_1$ | Parsing Errors | Useful |
| Baseline | – | 0.59 | 0 | – | 0.77 | 0 | |
| No Format Examples | −0.22 | 0.37 | 919 | −0.07 | 0.70 | 1 | ✓ |
| No Context Manager | +0.01 | 0.60 | 0 | −0.01 | 0.76 | 0 | |
| No Persona | +0.01 | 0.59 | 1 | +0.01 | 0.78 | 0 | |
| No Meta Language | −0.09 | 0.49 | 2 | −0.05 | 0.72 | 0 | ✓ |
| No Chain of Thought | −0.01 | 0.57 | 1 | −0.02 | 0.75 | 0 | ✓ |
| No Disambiguation | −0.03 | 0.55 | 0 | −0.01 | 0.76 | 1 | ✓ |
| No Reflection | +0.04 | 0.63 | 0 | +0.02 | 0.79 | 0 | ★ |
| No Fact Check List | +0.03 | 0.62 | 1 | −0.02 | 0.75 | 0 | ★ |
| Very Short Prompt | −0.04 | 0.54 | 1 | −0.03 | 0.74 | 0 | ✓ |

In addition to removing prompt components, we also tested using GPT in an older, less capable, but much cheaper version, GPT-3.5. Running the baseline prompt, results in a significant drop in extraction quality, with $F_1 = 0.27$ for MD and $F_1 = 0.56$ for RE. Splitting the baseline prompt into multiple prompts, each focusing on only one mention type, lets us prompt GPT-4 repeatedly for the same document. These highly specialized prompts are called "agents", which

pass information between each other. For example, we instruct the first agent to extract *Actions*, which are passed to other agents extracting *Actors* and *Business Objects* respectively. This lets us exploit the inherent dependency between these elements. If no Action is detected in a sentence, then there is likely no relevant Actor or Business Object, even if there are nouns that would qualify from a linguistic standpoint. This way of prompting leads to an absolute improvement of +0.08 in $F_1$ score for both the MD and RE tasks.

### 6.3 Stability of Results

LLMs are notorious for their non-deterministic output [10], which often puts the validity and stability of results into question. To assess the severity of these problems with our prompting strategy, we repeated the extraction of mentions (MD) and relations (RE) on all documents of PET five times. In each iteration we used gpt-4o-2024-05-13 as a model in a 1-shot setting and recorded the micro $F_1$ score. We then calculated mean (0.70 for MD, 0.89 for RE), standard deviation (0.003 for MD, 0.002 for RE), minimum (0.69 for MD, 0.89 for RE), and maximum (0.70 for MD, 0.89 for RE). While there are fluctuations in results, they are so minor that they do not call the validity of our results into question.

### 6.4 Lessons Learned

Using LLMs for extracting process relevant information brings with it a category of challenges, which we already discussed in Sect. 2.3. Solving these is paramount for successful application of LLMs. In this section we discuss how we approached these challenges and what lessons we learned.

**(C1) Limited Output Control.** The expected output format, as well as the form of extracted information, can mainly be influenced by the prompt components *Meta Language* and *Format Examples*. Adding these results in significantly improved $F_1$ scores, (+0.22 and +0.05 respectively for MD on PET). These improvements are explained by less parsing errors (919 less for MD on PET), and better recall and precision in detecting mentions. LLMs also run the risk of being "stochastic parrots", simply synthesizing linguistically correct phrases, based on their training data [10]. In our experiments we observed changes in $F_1$ of maximally −0.02 for rephrased prompts. This indicates robustness of our prompts and suitability of LLMs as an tool for business process information extraction.

**(C2) Black-Box.** A valuable advantage of utilizing LLMs is their ability to reflect, thereby providing justification for their generated results. Figure 3 shows three examples of justifications for extraction results in the ATDP dataset (see Sect. 2.1). Case I shows the ideal outcome where prediction and expected constraint are identical. Note, that the justification even refers to the meta-language provided in the prompt (compare Sect. 4). In case II, the prediction and the gold standard constraint do not match, because of an error in the gold standard data, following [2], which defines completing a process as a *meta action* that can not

be part of any constraint. The dataset creators are alerted of this issue by the LLM, since it plausibly justifies why *send out report* marks the end of a process instance. Finally, in case III the extraction result is controversial, since the sentence is ambiguous. If we consider the term *immediately* to encompass both actions, they are constrained by an *existence* constraint. Alternatively, viewing *check quantity* as a subtask of *process part list* suggests only one action needs modeling. Using such reflective explanations make LLMs useful for "human-in-the-loop" systems, which are already applied in fields like process mining [30].

**(C3) Input Presentation ependencies.** Adding more text to prompts sometimes has an adverse effect, reducing extraction quality (Sect. 2.3). This makes optimizing prompts difficult, since it is not clear, if adding additional disambiguation hints or longer definitions would improve the result. Using partial extraction prompts helps with this issue, as the sections regarding *Meta Language Creation* can be focused on a few types. Depending on the task, there may even be inter-dependency between information, that can be efficiently exploited in this way.

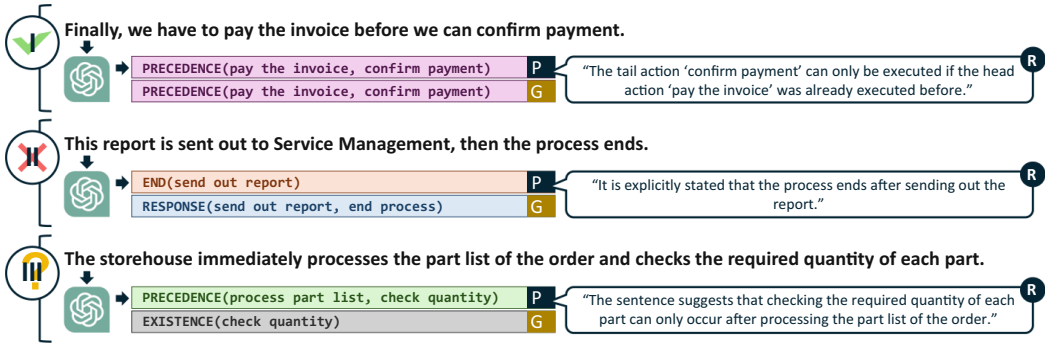



**Fig. 3.** Reflection example with Predicted and Gold standard constraints: (I) perfect match, (II) gold standard error, (III) ambiguous case

**(C4) Data Unawareness.** This issue arises, when LLMs are used in a zero-shot setting. There, the components *Meta Language*, and *Format Examples* are the only ways to "teach" the model how to perform the task. Applying the pattern of *few-shot prompting*, i.e., using labeled data in a few-shot setting was beneficial. This makes the use of an LLM more akin to training a machine learning model, but with significantly lower data requirements. In our experiments, three examples were sufficient to achieve better extraction results than those of machine learning models trained with more than ten times of the data.

**(C5) Costly Experiments.** This is a major drawback of LLM based process information extraction. The most capable LLMs are hosted as cloud-based solutions and are priced per token. We found that limiting the number of examples to 1 resulted in the best cost-value ratio. Additionally, our experiments showed that leaving out the prompt components *Context Manager*, *Persona*, and *Disambiguation* is a valid way to limit the number of tokens sent per request, albeit

with potential minor decreases in extraction accuracy. Prompting LLMs without the request for a *Fact List*, nor *Explanations* for extracted information greatly reduces the amount of tokens as well, especially useful after prompt engineering or data curating (during "*inference*"). Alternatively one can switch to cheaper models, i.e., *LLama 3 72B*, if the drop in performance is acceptable.

## 7 Conclusion

**Summary.** This paper presents an extensive study on the usefulness of LLMs for the extraction of process information from natural language text. We collected linguistic challenges and discuss how LLMs are uniquely fit for solving them. We also discussed challenges that arise through the use of LLMs and show how other communities propose to deal with these (or similar) concerns through prompt engineering. We present experimental results on three process information extraction datasets, which at least match the current state of the art on these datasets and in most cases improve it by as much as 8% in the $F_1$ metric. This shows the suitability of LLMs as a method for extracting business process relevant information from natural language process descriptions. To flesh out this notion, we analyze how well our prompting strategy can be applied to different LLMs without changing them, showing their universal nature. We expect LLMs to be a benchmark in the process information extraction domain for the foreseeable future, as limitations in dataset quality and quantity, combined with the need for complex reasoning make it very hard to train large extraction approaches from scratch. We make all our code, prompts, and LLM answers available, to support further research.

**Limitations.** A limitation of our work is that the list of prompt components we present may not be exhaustive, and they may have interactions that our ablation study does not capture. Additionally, some models suffer from hallucinations, especially Qwen1.5 and Llama 3, which hallucinate non-existent entity and relation types – 20 and 37 instances in the worst cases respectively. However, the severity of this problem diminishes in the few-shot setting (0 and 4 instances respectively in the worst case). We plan on analyzing how our prompts could be enhanced to improve instruction following for these models. Lastly, the current pricing models prohibit large-scale application of the most capable model to process information extraction. Alternatives (e.g., Llama) avoid this issue, but require more labeled examples to reach comparable levels of performance. This limitation may change in the near future, as more cost-efficient models and specialized hardware reduce costs to acceptable levels. Alternatively, very capable—and therefore expensive—LLMs could be used to create and curate training data for smaller local models, leveraging the reasoning capabilities of LLMs indirectly.

**Future Work.** In future work we aim to use LLMs as tools to support labeling of new data. Current datasets are limited in origin, i.e., they usually describe processes from municipalities or small service providers. We plan to analyze the

ability of LLMs to generalize beyond the domains with available labeled data and highlight the promising flexibility observed in our current experiments. Additionally, using GPT-4o's image processing and generation capabilities could be a promising line of research for direct text to model transformation. Finally, our results show very limited improvement in extraction quality, when the prompt includes a role the LLM is restricted to (persona). A slight variation of this idea is to describe the target audience of extraction results in the prompt, to further improve the quality of extracted process information.

## 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.: Checking process compliance against natural language specifications using behavioral spaces. IS (2018)
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. van der Aalst, W.: Process Mining. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
6. Ackermann, L., Neuberger, J., Jablonski, S.: Data-driven annotation of textual process descriptions based on formal meaning representations. In: La Rosa, M., Sadiq, S., Teniente, E. (eds.) Advanced Information Systems Engineering: 33rd International Conference, CAiSE 2021, Melbourne, VIC, Australia, June 28 – July 2, 2021, Proceedings, pp. 75–90. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-79382-1_5
7. Ackermann, L., Neuberger, J., Käppel, M., Jablonski, S.: Bridging research fields: An empirical study on joint, neural relation extraction techniques. In: CAiSE (2023)
8. 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)
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. Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: On the dangers of stochastic parrots: Can language models be too big? In: ACM FAccT (2021)
11. Cui, L., Wu, Y., Liu, J., Yang, S., Zhang, Y.: Template-based named entity recognition using bart. arXiv preprint arXiv:2106.01760 (2021)
12. Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How do practitioners use conceptual modeling in practice? Data Knowl. Eng. **58**(3), 358–380 (2006)
13. Dubois, Y., et al.: Alpacafarm: A simulation framework for methods that learn from human feedback. Adv. Neural Inform. Process. Syst. **36** (2024)
14. 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)

15. Franceschetti, M., Seiger, R., López, H.A., Burattin, A., García-Bañuelos, L., Weber, B.: A characterisation of ambiguity in BPM. In: Almeida, J.P.A., Borbinha, J., Guizzardi, G., Link, S., Zdravkovic, J. (eds.) Conceptual Modeling: 42nd International Conference, ER 2023, Lisbon, Portugal, November 6–9, 2023, Proceedings, pp. 277–295. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-47262-6_15
16. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: CAiSE (2011)
17. Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., McHardy, R.: Challenges and applications of large language models. arXiv preprint (2023)
18. Kourani, H., Berti, A., Schuster, D., van der Aalst, W.M.: Process modeling with large language models. arXiv preprint arXiv:2403.07541 (2024)
19. Leopold, H., van der Aa, H., Pittke, F., Raffel, M., Mendling, J., Reijers, H.A.: Searching textual and model-based process descriptions based on a unified data format. SoSym **18**, 1179–1194 (2019)
20. Lewis, P., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks: Adv. Neural. Inf. Process. Syst. **33**, 9459–9474 (2020)
21. López-Acosta, H.A., Hildebrandt, T., Debois, S., Marquard, M.: The process highlighter: From texts to declarative processes and back. In: CEUR Workshop Proceedings, pp. 66–70. CEUR Workshop Proceedings (2018)
22. Min, B., et al.: Recent advances in natural language processing via large pre-trained language models: a survey. ACM Comput. Surv. **56**(2), 1–40 (2023)
23. 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)
24. 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)
25. Qian, C., et al.: An approach for process model extraction by multi-grained text classification. In: CAiSE (2020)
26. Quishpi, L., Carmona, J., Padró, L.: Extracting annotations from textual descriptions of processes. In: BPM 2020 (2020)
27. Richens, R.H.: Interlingual machine translation. Comput. J. **1**(3), 144–147 (1958)
28. Sànchez-Ferreres, J., Burattin, A., Carmona, J., Montali, M., Padró, L., Quishpi, L.: Unleashing textual descriptions of business processes. In: SoSyM (2021)
29. Sukthanker, R., Poria, S., Cambria, E., Thirunavukarasu, R.: Anaphora and coreference resolution: A review. Information Fusion (2020)
30. Ter Hofstede, A.H., et al.: Process-data quality: The true frontier of process mining. In: ACM JDIQ (2023)
31. Törnberg, P.: Best practices for text annotation with large language models. arXiv preprint arXiv:2402.05129 (2024)
32. Wei, J., et al.: Chain-of-thought prompting elicits reasoning in large language models. In: NIPS (2022)
33. White, J., et al.: A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv:2302.11382 (2023)
34. Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., Zhu, J.: Explainable ai: brief survey on history, research areas, approaches and challenges. In: NLPCC (2019)

# Chapter 10

# Assisted Data Annotation for Business Process Information Extraction from Textual Documents

Neuberger, J., van der Aa, H., Ackermann, L., Buschek, D., Herrmann, J., Jablonski, S. (2025). Assisted Data Annotation for Business Process Information Extraction from Textual Documents. In: Comuzzi, M., Grigori, D., Sellami, M., Zhou, Z. (eds) *Cooperative Information Systems. CoopIS 2024.* Lecture Notes in Computer Science, vol 15506. Springer, Cham. https://doi.org/10.1007/978-3-031-81375-7_11



**Contribution statement.** The concept for an efficient and effective data annotation tool for process information was developed by Julian Neuberger (JN). Jannic Herrmann (JH) implemented the research prototype. JN, Lars Ackermann (LA), and Daniel Buschek (DB) designed the user study for evaluating the research prototype. JH conducted the user study, JN supervised the user study. JN evaluated the data collected during the user study and interpreted the results. Figures were created by JN. JN and Han van der Aa (HA) wrote the main body of the publication, with additional contributions by JH, LA, and Stefan Jablonski (SJ). The publication was revised by all authors. JN is the corresponding author.

# Assisted Data Annotation for Business Process Information Extraction from Textual Documents

Julian Neuberger[1(✉)], Han van der Aa[2], Lars Ackermann[1], Daniel Buschek[1], Jannic Herrmann[1], and Stefan Jablonski[1]

[1] University of Bayreuth, Bayreuth, Germany
{julian.neuberger,lars.ackermann,daniel.buschek,
jannic.herrmann,stefan.jablonski}@uni-bayreuth.de
[2] University of Vienna, Vienna, Austria
han.van.der.aa@univie.ac.at

**Abstract.** Machine-learning based generation of process models from natural language text process descriptions provides a solution for the time-intensive and expensive process discovery phase. Many organizations have to carry out this phase, before they can utilize business process management and its benefits. Yet, research towards this is severely restrained by an apparent lack of large and high-quality datasets. This lack of data can be attributed to, among other things, an absence of proper tool assistance for business process information extraction dataset creation, resulting in high workloads and inferior data quality. We explore two assistance features to support dataset creation, a recommendation system for identifying process information in the text and visualization of the current state of already identified process information as a graphical business process model. A controlled user study with 31 participants shows that assisting dataset creators with recommendations lowers all aspects of workload, up to $-51.0\%$, and significantly improves annotation quality, up to $+38.9\%$ in $F_1$ score. We make all data and code available to encourage further research on additional novel assistance strategies.

**Keywords:** Business process management · Process information extraction · Natural language processing · Human computer interaction

## 1 Introduction

Business process management (BPM) can provide organizations with many benefits by improving their regular operating procedures. Organizations looking to utilize these benefits first need to discover and model their business processes, which is a very time consuming, and therefore expensive task [13]. To alleviate this, researchers in the BPM community use the information contained in natural language process descriptions from sources like quality management handbooks, documentation of standard operating procedures, or employee notes to automatically generate formal process models [4]. While this area is actively researched [1,5,8,11,13,28], new and innovative approaches are quite rare. One reason is the limited availability of data to develop, train, and assess approaches in BPMN contexts [18]. Recent initiatives aim to mitigate this issue, providing a gold-standard dataset for the process information extraction task—PET [9]. With

this dataset, systems for extracting process information can be developed, e.g., machine learning models for extraction are trained, and subsequently evaluated. Extracted information is then the basis for automated model generation methods, allowing fully automated process model generation from process descriptions. Still, PET contains only 45 process descriptions, which is not enough to train deep neural networks [28], although they have been shown to be well suited for similar tasks in other areas [6]. Even techniques based on pretrained large language models are affected by the lack of data, as rigorous evaluation on many different data sources is essential to assess their practicality, especially in light of the large variation in terms of the structure, style, and contents of textual documents that contain process information [2].

The lack of suitable data for process information extraction tasks can in part be attributed to the effort required to establish gold standard annotations. Such annotations are a critical requirement for both training and evaluation of information extraction approaches. However, manually annotating process information in textual process descriptions involves elaborate guidelines [9] and considerable ambiguity [3,12], making it time consuming and mentally taxing. Figure 1 shows two sentences of a process description from the PET dataset, fully annotated with the gold-standard process information. Note, that annotating these sentences requires identifying 14 process-relevant elements, and 16 dependencies between them, in just these two sentences, where the average description in PET has 9.27 sentences [9]. We discuss the task in detail in Sect. 3.1 and how to circumvent this complexity in Sect. 3.2. Additionally, depending on the annotation schema, some of these annotations are not intuitive, e.g., "*decides*", which would intuitively be annotated as an *activity*, underlining the need for annotation guidelines mentioned above.

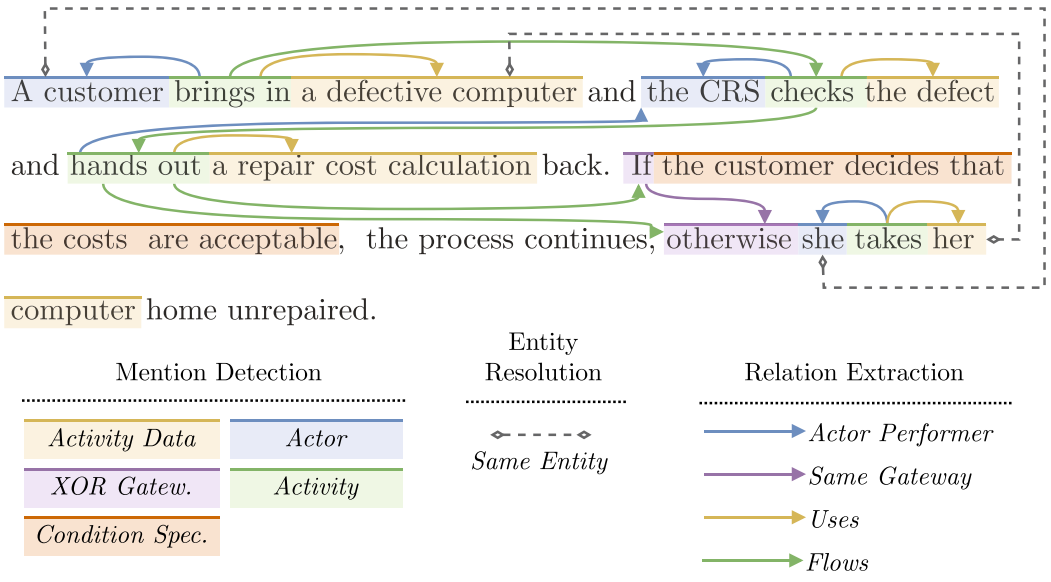



**Fig. 1.** The first two sentences of document *doc-1.2* in the PET dataset, fully annotated with entity mentions, entity references, and relations.

Recognizing this issue, we explore how dataset creators (*annotators*) can be assisted in their data annotation task, so that their workload is lightened, while simultaneously improving the quality of their extractions. Therefore, in this paper, we propose and evaluate the benefits of two assistance features that can support human annotators: (1) AI-based recommendations, which allow annotators to quickly tackle trivial parts of the annotation task—as well as receive suggestions for less trivial aspects—and (2) the use of a visualizations of the currently annotated information through a graphical process model, which allows annotators to observe the process that they have so far captured. Note that, although the task of text annotation is generic, the assistance features are tailored to the specifics of text annotations for process information extraction.

We implement both assistance features in a prototypical annotation tool that we use as a basis for a rigorous user study with 31 participants, ranging from modeling novices to experts, to assess the effectiveness and efficiency of the proposed features. Code[1] and data[2] are made publicly available to the research community, to allow others to efficiently and effectively annotate their own datasets, but also encourages exploration of additional assistance features. The main insights from this study are as follows.

1. Assisting annotators with suggestions made by artificial intelligence systems is observed to make annotating process relevant information in textual process descriptions significantly easier. This results in a significant reduction of key workload metrics by more than one half ($-51.0\%$). At the same time, assistance improves the quality of extracted information measured in $F_1$ score by up to $+0.224$ ($+38.9\%$).
2. The use of assistance features is recognized to considerably reduce the gap between novice and experienced process modelers in annotation tasks. Specifically, complete beginners can reach annotation quality comparable to expert annotators, speeding up the training process for new data annotators considerably. This insight shows that annotation features can help assembling larger data annotation teams, and speeds up the creation of new datasets in the space of business process model extraction from natural language text.

The rest of this paper is structured as follows. In Sect. 2, we discuss related work on process information extraction, relevant user studies, and annotation tools. In Sect. 3, we present our concept behind a tool built specially for annotating textual process descriptions, its implementation in a research prototype, and the assistance features. In Sect. 4 we describe the design and execution of the user study. We present results for this study in Sect. 5. We conclude the paper in Sect. 6, summarizing, discussing limitations, and describing future work.

## 2 Related Work

Work related to this paper can be roughly categorized into three sections.

**Business Process Information Extraction.** The last decades have seen various approaches to the task of extracting process relevant information from natural language text, including systems based on expert-defined rules [1,8,11,30], data-driven

[1] see https://github.com/JulianNeuberger/assisted-process-annotation.
[2] see https://zenodo.org/doi/10.5281/zenodo.12770686.

ones [5,9,28], and systems based on pretrained generative large language models [8,21,27]. We use approaches from [28] and [9] in our work to implement annotation recommendations. Many of the works mentioned also propose data annotation schemata tailored towards specific modeling languages, such as [1,30] for declarative process modeling, towards different task descriptions, such as [29] for information extraction from process relevant sentence fragments, or towards other stages in the process life cycle, e.g., process redesign [25].

We focus on PET, as it is heavily biased towards the current industry standard, BPMN, and the to date largest available dataset. PET was extended with the notion of entity identities [28], i.e., the task of resolving multiple mentions of the same process element across the textual description to a single one. This is important for properly modeling business objects and process participants, which would otherwise be duplicated in the generated model. In this paper we use this extended version of PET.

**User Studies.** Schützenmeier et al. [32] present a user study on the cognitive effort of understanding declarative process models, though they do not consider data annotation, but process simulation and verification. Rosa et al. [31] develop and evaluate a tool for business process modelling which assists users by identifying core BPMN 2.0[3] elements and highlighting them in the process description. Our work, in contrast, aims to be a step towards alleviating the data scarcity problem in business process model generation from text, by making data annotation easier. In a study of similar size to ours, the authors evaluate the usefulness of the *BPMN Sketch Miner* for process modelling based on textual descriptions with visual representations of process elements [16]. While their study mainly focuses on usability and subjective values, ours also considers objective measures.

**Annotation Tools.** Both our concept and implementation for assisted process-relevant information annotation are related to a number of annotation tools. These can usually be used to annotate text for use in Named Entity Recognition (NER), Entity Matching and Resolution (ER), or Relation Extraction (RE), tasks which are similar to business process information extraction (compare definition in Sect. 3). Still, these tools are not designed with characteristics of business process descriptions in mind, including but not limited to, the high information density present in such descriptions, its inherent ambiguity (see Sect. 1), and the target down-stream task, i.e., generation of a formal and graphical process model. Additionally, unlike in the NLP community, data annotators for process information extraction are often times experts in BPM, but not in NLP, and therefore can benefit from purposeful simplifications in the annotation tool. In the following we will describe several notable examples of multi-purpose Natural Language Processing (NLP) data annotation tools, from which we drew inspiration and how our proposed concept differs from them.

Doccano [26] provides features useful for collaborative data annotation, creating datasets in multiple languages, and comparing annotations between annotators. Label Studio [34] supports more machine learning domains, e.g., computer vision, and audio processing. This makes the tool even more multi-purpose and less bespoke, compared to our research prototype. The authors already have experience annotating textual busi-

[3] https://www.omg.org/bpmn/, accessed July 4, 2024.

ness process descriptions using Label Studio [5], which is integrated into our concept for assisted annotation (Sect. 3). Finally, INCEpTION [20] uses *recommenders* to make suggestions for new annotations, which would fit our requirement for AI-based annotation recommendations, but to the best of the author's knowledge can not be extended to show the current state of annotation as a BPMN model. The authors of [20] did not investigate the effectiveness of recommendations for text annotation, and while a positive effect seems plausible, we are interested in proving and quantifying this effect.

## 3 Concept for Assisted Annotation

This section outlines our concept for assisted data annotation. First, we define the task of annotators in Sect. 3.1. Based on this we motivate the need for more efficient and effective data annotation and derive assistance features in Sect. 3.3. Finally, we describe our research prototype implementation in Sect. 3.4.

### 3.1 The Process Information Extraction Task

Ultimately, human annotators have to complete the process information extraction task to annotate process descriptions with process-relevant information. Therefore, we define this task in the following. Consider, for example, document *doc-1.2* from the PET dataset describing the process of a computer repair. Figure 1 shows this document fully annotated with all process relevant information, which consists of three major categories. First, ***Mentions*** of process relevant entities in PET are continuous sequences of text with a given type, for example, *Actors* (process participants, "a customer"), *Activity Data* (business objects, "a computer"), or *XOR Gateways* (decision points, often indicated by "if", "otherwise"). The last example illustrates, why we call detecting and extracting such mentions *Entity Mention Detection* (MD) and not *Named Entity Recognition* (NER). Named Entities are defined by either proper names (e.g., persons, locations) or natural kind terms (e.g., enzymes, species) [22]. "If" or "otherwise" do not fall into this definition, which is why we use the more relaxed definition of (non-named) entities and the detection of their mentions within the text [35]. Mentions are then resolved to ***Entities***, i.e., clustered, allowing subsequent model generation steps to only render a single process element, instead of multiple (one for each of its mentions). This task is called *Entity Resolution* (ER) and is closely related with co-reference and anaphora resolution [33]. ***Relations*** between mentions define how these elements interact with each other. PET defines, for example, *Flow* (order of task execution), *Uses* (association between a task and the business object it uses), or *Actor Performer* (assigning a process participant as executor of a given task).

### 3.2 Annotation Workflow

As we discussed in Sect. 1, annotating the process relevant information contained in textual process descriptions is a complex task and very demanding for the human performing the annotation, as it requires attention to three sub-tasks, as outlined in the previous Sect. 3.1. We therefore split the task into its sub-tasks MD, ER, and RE. While this partially alleviates the issue of complexity, it will also allow us to assist annotators

in these sub-tasks differently, and analyze how assistance features help during a specific sub-task. Figure 2 depicts the resulting workflow. After the annotator submits a natural language process description, they are then asked to select mentions (MD), resolve entities (ER), and define relations between mentions (RE), in three separate steps. Finally, all information is shown again, so that the annotator may reconcile any errors.

While this workflow reduces the complexity of process information annotation by splitting it up into smaller tasks, the overall complexity remains high. High density of information makes annotating very confusing, especially for beginners. The example in Fig. 1 contains a total of 40 words, of which only eight are not part of one of the 14 entity mentions (20%), while also containing 14 relations between them. From previous annotation experience in other tools (see Sect. 2), we know that this can be partially mitigated by splitting the task into smaller sub-tasks, e.g., focusing on a subset of entity and relation types, or by annotating the categories from above one after the other. Based on this experience we defined a *workflow*, which we describe in Sect. 3.2. High information density and the resulting complexity of displaying this information also motivates us to find ways to visualize the information better, and help the user focus on information they potentially would miss otherwise. This results in two assistance features, *visualization* and *recommendation*, which we describe in Sect. 3.3.

### 3.3 Assistance Features

In one of our preliminary studies two assistance features were identified as promising candidates for improving the efficiency, quality, and user experience of annotation documents for the process information extraction task.

**AI-Based Annotation Recommendations.** Building on the progress that has already been made in the development of automated information extraction approaches for mentions, entities, and relations, we can present the user with recommendations for these elements. Interviews with BPM experts during the preliminary study and our review of related work (Sect. 2) suggested that recommendations can be a powerful tool for speeding up annotation in trivial cases and provide useful ideas in non-trivial ones. We used an approach based on conditional random fields for extracting mentions, as presented by Bellan et al. [9], with code from [28], a pretrained neural co-reference resolver for entities [28], and a relation extraction approach based on gradient boosting on decision trees [28]. All trainable approaches were trained with 80% of the available data (36 documents) and the rest was held out for use during the user study. Recommendations are shown during the appropriate workflow steps and can be confirmed, discarded, edited, or marked for later review. The extraction approaches we use are limited in their understanding of business processes, and as such have no real world knowledge that could help during extracting process information from unseen descriptions. This means recommendations can be flawed, but are enough to effectively support data annotation (Sect. 5).

**Visual Result Representation.** Second, the information that a human annotator marks in a textual process description is always process relevant, i.e., a perfect annotation results in a model that perfectly reflects the process description. This shows how a human annotator may benefit from a graphical process model as a visualization of the currently annotated information, as any missing information is reflected in the (therefore incomplete)

graphical process model. Visualizing the current state of annotation involves three major stages. First, in the ***Consolidation*** stage, we assign conditions to their respective paths in the process, merge mentions of entities, and find the closest actor in the text left of activities that are not explicitly assigned one. In the second stage, the ***Vertex*** stage, we create process elements for all mentions, e.g., *Tasks*, *Data Objects*, *Swimlanes*, etc. The final ***Linking*** stage connects related elements, e.g., successive tasks and gateways with *Sequence Flows*, if they are located in the same Pool, or *Message Flows* between them. We also create *Data Associations* between Data Objects and Tasks, adding the label of the Data Object to the label of the Task, for labels like "*send a mortgage offer*". In this way the graphical process model is generated and layouted automatically, and as such has limitations that might affect its usefulness, which we discuss in Sect. 6.

### 3.4 Implementation

We have implemented our concept in a usable research prototype. It consists of a user-facing web application, implemented in JavaScript, using React[4].

A backend server provides NLP pre-processing functionality, such as tokenization and the information extraction approaches for the recommendation assistance feature. It is implemented in Python 3.11, using spaCy[5] for pre-processing. When the user inputs a textual process description, it is first sent to this server to pre-process the text. The result is then displayed in the web application. In each of the annotation sub-tasks defined in Sect. 3.2 the relevant information is extracted by the backend server and presented to the annotator as recommendations. The backend server is also responsible for storing annotation results. A second backend server is used for visualizing the current annotations by generating a formal process model in BPMN and its graphical representation. We implement this using Java 17 and the Camunda Model API[6]. Both back-end servers expose any functionality using REST interfaces, which the user-facing web application can query. Figure 2 shows the workflow in the web application, as well as the communication between servers.

## 4 Study Design

We conducted a user study with 31 participants to assess the effectiveness of the two assistance features, based on several measures in scenarios of varying assistance. Both measures and scenarios are defined later in this section. We focus our efforts on answering the following three key research questions.

**RQ1.** Which annotation assistance features or combinations lower the workload of annotating process information?

**RQ2.** Which annotation assistance features or combinations improve the quality of annotations?

[4] https://react.dev/, last accessed July 11, 2024.

[5] https://spacy.io/, last accessed July 11, 2024.

[6] https://docs.camunda.org/manual/7.21/user-guide/model-api/bpmn-model-api/, last accessed July 11, 2024.

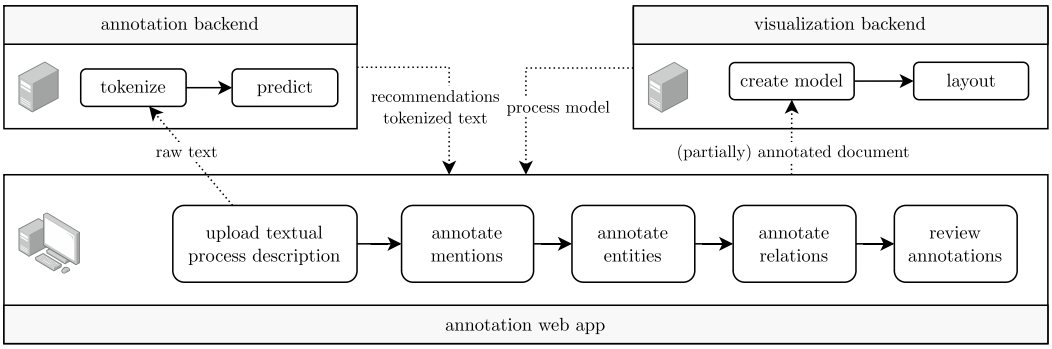



**Fig. 2.** Visualization of the workflow and general architecture of our implementation.

**RQ3.** Which annotation assistance bridge the gap in annotation quality between beginner annotators and those with BPMN experience?

The general setup of this study is as follows. All supplementary material, such as the questionnaires and resulting data can be found online, see Sect. 1.

**Study Procedure.** Each participation within our user study, is structured into three blocks. First, general demographic information is collected, and the task and annotation tool are explained to the participant, which involved giving a brief tutorial and a small guided annotation task, without any assistance features enabled. This task is only done for training purposes and is not evaluated later. Next, the participant has to complete four annotation scenarios, which we describe later in this section. After each scenario a short questionnaire is conducted, aimed at collecting user opinion, sentiment, and feedback concerning the scenario they just completed. The last block involves a questionnaire to gather general feedback and data regarding overall user preferences.

**Measures.** We measure the effectiveness of assistance features using several metrics aimed at *objective* and *subjective* values. For objective values we measure the time a user takes to annotate a document and the quality of mention, entity, and relation annotations, each measured with the $F_1$ score. Subjective values are derived from the NASA Task Load Index (TLX), which is widely used for measuring the workload during or right after performing a task [15]. The NASA-TLX can be used in many different contexts, and was also already used to evaluate information systems [7]. It defines a total of six dimensions that measure different aspects of workload. We used the four relevant to our study.

*mental demand*: How much the annotator has to focus on the task
*uncertainty*: How uncertain the annotator is of their annotations
*effort*: How much work is needed to complete the task
*frustration*: How frustrated the annotator is with the task

We excluded physical and temporal demands, to focus on the subjective metrics most relevant to our research questions. While *physical* demand is not completely irrelevant (think of mouse movements), it is far less informative than the other measures. Regarding *temporal* demands we refer to our objective measure of task completion time. Note that compared to the original definition of the NASA-TLX, we rephrase

*performance* to measure the *uncertainty* of an annotator with their annotation results. Additionally to the NASA-TLX, we also asked users to share their experiences with the tool and assistance features in a questionnaire using 5-point Likert items [17].

**Annotation Scenarios.** We assess the efficiency and effectiveness of annotators in four scenarios. Scenario *(A)* entails no assistance, besides the workflow defined in Sect. 3.2 and serves as a baseline. Scenario *(B)* visualizes the current state of annotation, and *(C)* gives recommendations for annotations made by an artificial intelligence system. Finally, scenario *(D)* combines both assistance features.

Documents in PET contain 168 words on average, which took experts in a preliminary experiment as much as 25 min to annotate. We therefore decided to instead only use fragments of documents, containing two sentences. These fragments were carefully selected by measuring the number of mentions, relations, as well as their types. We selected fragments from documents *doc-1.2*, *doc-3.6*, *doc-8.3*, and *doc-9.2*. Document fragments are part of the supplementary material for this paper and available in the repositories mentioned in Sect. 1.

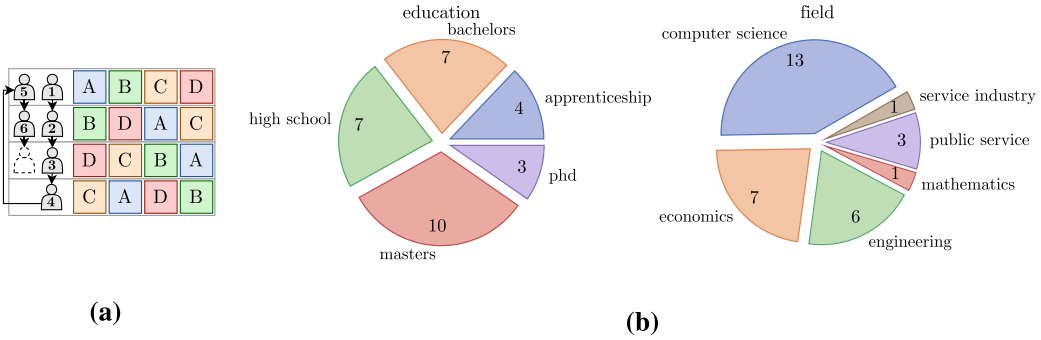
To avoid carry-over effects, i.e., confounding variables such as familiarity with the task after completing a scenario and therefore performing better in the next one, we use the Balanced Latin Square method [19]. This method systematically produces sequences of the scenarios described above, so that each scenario appears as the first one in the sequence equally often, as well as two scenarios preceding or succeeding one another equally often. Users are assigned a sequence of scenarios in a round-robin fashion, compare Fig. 3a. This setup minimizes the number of scenarios each user has to perform while addressing carry-over effects between scenarios, such as increasing familiarization with the annotation task.

## 5 Results

In this section we will describe our observations during the experiments described in Sect. 4, starting with an overview of study participants. We had respondents of various age, education, and field of work. 39% have not obtained a university degree, or did not pursue higher education, while 39% completed either Masters or PhD studies. The majority (71%) of participants work in a technical field, i.e., computer science, engineering, or mathematics. Figure 3b shows a detailed break-down of demographic characteristics of participants.

### 5.1 Subjective Measures

As described in Sect. 4, we measure four subjective sub-metrics of the NASA-TLX—mental demand, uncertainty, effort, and frustration—across four different assistance scenarios. We then used a repeated measure ANOVA [10] to find if there are statistically significant differences in the four assistance scenarios defined in Sect. 4. A repeated measure ANOVA can be used to test if two ore more non-independent samples (measurements) are from the same distribution, measured by $p \in [0, 1]$. In our case, we test for differences in workload between annotation assistance scenarios. We reject the Null hypothesis (no difference) and accept the alternative one (difference exists) when $p < 0.05$.




**(a)** **(b)**

**Fig. 3.** Assigning annotators to a sequence of scenarios based on a balanced Latin square (left), and demographic information about user study participants (right).

Repeated measures ANOVA assumes sphericity in data, i.e., the difference in metrics for all combinations of two scenarios have the same variance. This assumption can be tested with Mauchly's test for sphericity [24]. Data for three out of four metrics violated the assumption of sphericity ($p < 0.05$). We use the Greenhouse-Geisser correction [14] to account for this. Even then, our observations show that each one of the four workload metrics are affected by changing how annotators are assisted by our annotation tool and the differences are statistically significant with $p < 0.001$.

Since the repeated measures ANOVA indicated a difference in the NASA-TLX metrics when using different assistance features, we ran six post-hoc tests, looking for the differences between each combination of two features, e.g., measurements for non-assisted annotation (scenario A in contrast to only recommendations (scenario C). We corrected all $p$ values using Bonferroni's method [23] for running multiple tests. Intuitively, running many tests increases the likelihood of finding statistically significant differences in one of them, even though there is none. This correction multiplies the P-value with the number of tests, to account for this increased likelihood. Table 4 in Sect. A reports details.

In summary, no assistance feature at all (scenario A) is statistically significantly worse than either only recommendations (scenario C) or both assistance features combined D). Surprisingly, assisting annotators with a visualization of the information they found in the text (i.e., the generated graphical process model, scenario B) was not found to help with reducing the workload.

Compared to no assistance, assisting the annotator with recommendations reduced mental demand by 24.7 ($-34.6\%$), effort by 22.4 ($-34.2\%$), and frustration by 20.5 ($-51.0\%$). Uncertainty is best lowered by combining recommendations with visualizations, which reduces it by 24.4 ($-44.8\%$), according to our observations. Note, that we found no statistically significant effect on any sub-metric when comparing recommendations (scenario C) to the combination of recommendations and visualizations (scenario D). Similarly, we could not observe a difference between non-assisted annotation (scenario A) and just visualization of annotated information (scenario B). This indicates that only visualizing the currently annotated process-relevant information is not enough to reduce the workload of the annotation task. Contrary, recommendations are a way to reduce it by up to to nearly $50\%$. Some limitations apply to our findings concerning the quality of the graphical process representation, which we discuss in Sect. 6.

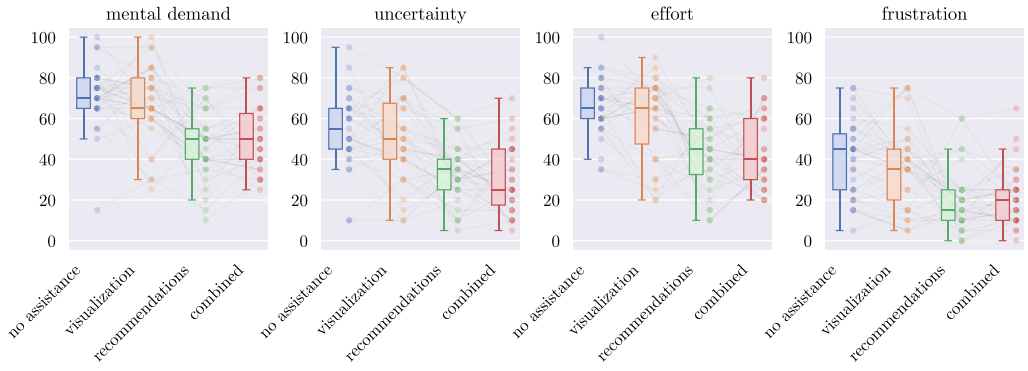



**Fig. 4.** Subjective measures for each of the four scenarios from Sect. 4.

Figure 4 aggregates our data for each sub-metric into plots, showing values for each participant and scenario as strip plots, where the values of a given participant are connected by lines. Additionally the data points are aggregated into box plots, showing the data mean, $25^{th}$ and $75^{th}$ percentiles as box, and the rest of the distribution as whiskers, excluding outliers. The plots mirror the general observations we drew from Table 4, and shows that recommendations and the combination of both assistance features help best with reducing the workload of data annotators. Overall, the ordering of assistance features in terms of reducing the workload is obvious from the plots. No assistance (scenario A) and visualizations only (scenario B) share the spot for least useful, while recommendations and the combination of features seem to be equally useful in lowering the workload of process information annotation, thus answering research question **RQ1**.

### 5.2 Objective Measures

As discussed in Sect. 1, our goal with assisting data annotators is twofold. The previous Sect. 5.1 discussed metrics that are subjective, i.e., are based on the experiences of a data annotator. On the other hand, assisting annotators also affects the quality of annotations. We measured a total of four objective metrics, which we presented in detail in Sect. 4. These are the $F_1$ scores for annotated mentions, entities, and relations, as well as the total time a given annotator needed to complete annotating a document fragment. An aggregate of the data we obtained is shown in Fig. 5 as a plot, similar to the one we showed and explained in Sect. 5.1. Detailed results are listed in Table 2 in Appendix A.

Again, using a repeated measure ANOVA we found significant effects on the annotation quality measured in $F_1$ when using different assistance features during the annotation of mentions ($p < 0.001$) and relations ($p < 0.001$). The annotation quality of entities was not affected ($p = 0.450$), which may be caused by the low number of entities[7], as well as the fact that we count an entity only as correct, if contains all expected mentions. This means errors by the annotator during MD propagate to the ER task.

[7] On average, fragments used in the user study only contained one entity that needed resolution, i.e., there are at least two entity mentions referring to the same entity.

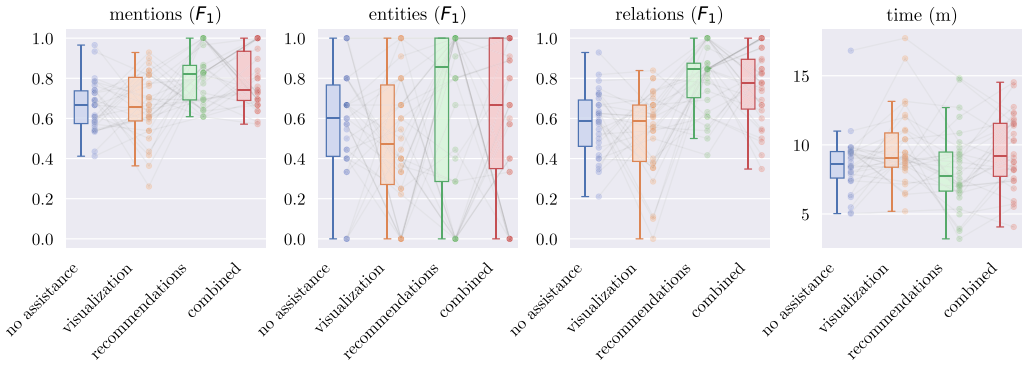



**Fig. 5.** Objective measures for each of the four scenarios from Sect. 4.

Furthermore, we did observe a statistically significant difference in the time an annotator needs to annotate a document ($p = 0.011$), but during post hoc tests we could only explain this with a statistically significant difference between the assistance features *visualization* (B) and *recommendations* (scenario C), where the latter speeds up completion times by about 1.5 min (see Table 3 in the Appendix A).

Several participants remarked during the user study, that identifying relations and classifying them correctly is a very challenging task. These participants were mostly inexperienced with BPM and BPMN and thus greatly benefited from the recommendation assistance feature. We can observe this across all participants, as the post hoc tests for relation annotation quality show. Comparing scenario A (no assistance) against scenario C (recommendations only), we see a statistically significant ($p = 0.001$) increase in $F_1$ of 0.224 (+38.9%). Using the visualization does not seem to have a significant effect compared to no assistance at all ($p = 1.000$), but is significantly worse than using recommendations ($-0.259$, $p < 0.001$) or using both assistance features ($-0.204$, $p < 0.001$).

The same analysis can be made for the task of mention detection. Participants of our user study seem to benefit most from recommendations, when compared to no assistance at all ($p < 0.001$), with an improvement of 0.141 (+21.4%). Visualization has no statistically significant effect ($p = 1.000$) compared to no assistance at all. Using only visualization has an adverse effect compared to just recommendations ($p < 0.001$) with a decrease in $F_1$ of 0.141. Similar to Sect. 5.1, this effect can be attributed to limitations in our graphical process model, which we discuss in Sect. 6, or a user's familiarity with BPMN (Sect. 5.3).

This also answers research question **RQ2.**, as recommendations seem to be the best choice for improving the quality of annotations. Notably, for all three tasks the annotation recommendations themselves are of lower quality than the average annotations by a human annotator assisted by recommendations (scenario C). Human review improved the $F_1$ score of annotations by +0.100 for MD, +0.181 for ER, and +0.151% for RE, showing how humans assisted by AI-based systems can perform better than each part in isolation.

**Table 1.** Independent Samples T-Test for the hypothesis that objective scores for annotations by *novices* are lower than those by *experts*.

| | | t | df | $p^a$ |
|---|---|---|---|---|
| mentions | no assistance | −1.950 | 20.691 | 0.032 * |
| | recommendations | 0.023 | 23.041 | 0.509 |
| | visualization | −1.166 | 23.984 | 0.128 |
| | combined | −1.463 | 24.491 | 0.078 |
| relations | no assistance | −1.800 | 25.072 | 0.042 * |
| | recommendations | 0.664 | 21.132 | 0.743 |
| | visualization | −1.182 | 28.000 | 0.124 |
| | combined | −0.590 | 20.711 | 0.281 |

*, **, *** statistically significant results of increasing degrees.
[a]P-value following Welch's test.

### 5.3 Effects of Annotator Experience

We asked participants for their experience with BPMN, measured in years. With this information, we now investigate if a user's experience with BPMN influences how much they can benefit from assistance. To this end we split the data into two groups—*experts*, which we define for the purposes of this analysis as participants with at least one year of BPMN modeling experience, and *novices*, which are the remaining study participants. This split results in 10 experts and 18 novices. We hypothesize that annotations by *novices* are worse in terms of $F_1$ score compared to those by *experts*. Table 1 lists results for an independent samples T-Test.

We can confirm our assumption that BPMN experience improves the quality of mention (MD) and relation annotations (RE), for un-assisted annotation (scenario A). In all assisted scenarios (B, C, D) we have to reject our hypothesis, i.e., *novices* no longer produce worse annotations than *experts*, from which we infer that the two assistance features can indeed bridge the gap in annotation quality caused by differences in experience with BPMN. We therefore answer **RQ3.** with *annotation recommendations*, *visualizations*, and *combined assistance*.

## 6 Conclusion

In this section we will reiterate the core contribution of this paper, the limitations of the user study we conducted, and we describe our plans for future work.

**Core Contributions.** This paper presents an in-depth exploration on the usefulness of two features for assisting data annotators in the domain of business process information extraction. A user study with 31 participants shows that annotation recommendations reduce certain workload aspects by up to $-51.0\%$ (**RQ1.**). We find that recommendations obtained by a system based on machine learning improve annotation quality as much as $+38.9\%$ (**RQ2.**). The same recommendations bridge the gap in annotation

quality between beginner and expert annotators, promising easier assembly of annotation teams by means of shorter training times (**RQ3.**). We make all data and code publicly available.

**Limitations.** First, we focused our study on two assistance features, to ensure its feasibility, while also guaranteeing methodological correctness. Investigating more assistance features would either increase participation times, or limit each participation to a subset of scenarios. Next, while we could not observe statistically significant effects of any assistance features on the quality of entity resolution annotations, we cannot eliminate the possibility that this caused by errors propagated from the MD task. Our automated method used for generating and layouting a graphical process model from the process information (annotations) used for the visualization assistance feature has limitations in terms of structure, accuracy stemming from the employed heuristics, and clarity of generated labels. This may affect its usefulness, as these limitations may make the graphical model harder to understand, especially for untrained annotators. Finally, we only present and analyze a sub-set of the data collected during the user study. For example, we recorded all user interaction with the tool, such as when a recommended annotation is discarded, or a new annotation is created. These logs constitute valuable data for improving the workflow for annotating process relevant data in textual process descriptions.

**Future Work.** Our future work is mainly concerned with eliminating the limitations we discussed in the previous Sect. 6. As such we plan to improve the implementation of our annotation tool, e.g., improve the way relations are displayed. We also plan to extend our analysis of the data we already obtained during this user study, e.g., by evaluating the interaction logs. This data can be very valuable to learn how annotators interact with the annotation tool, and give indications on how to improve the workflow, or which parts of the interface are still unintuitive. Furthermore, we want to explore better annotation recommendation methods, as this feature seems to have a consistently positive effect. We plan to evaluate integrating incremental training, as soon as annotators have submitted a document. Finally we would like to extend the user study to new assistance features, in addition to comparing different workflows and user interface options. The initial findings regarding how experts benefit in different ways from assistance features, compared to novice users, motivate us to conduct a targeted study to find ways to properly assist users of different experience levels.

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

# A Appendix

**Table 2.** Post hoc comparisons of assistance features on objective metrics. Largest statistically significant absolute difference to unassisted annotation is set in **bold**. We abbreviate mean difference with MD and standard error with SE.

| | | | MD | SE | t | $p_{bonf}{}^{a}$ |
|---|---|---|---|---|---|---|
| mentions | no assistance | recommendations | $-$**0.141** | 0.037 | $-3.774$ | 0.002 ** |
| | | visualization | $-0.000$ | 0.037 | $-0.008$ | 1.000 |
| | | both | $-0.132$ | 0.037 | $-3.529$ | 0.014 ** |
| | recommendations | visualization | 0.141 | 0.037 | 3.766 | 0.005 ** |
| | | both | 0.009 | 0.037 | 0.244 | 1.000 |
| | visualization | both | $-0.132$ | 0.037 | $-3.522$ | 0.004 ** |
| entities | no assistance | recommendations | $-0.068$ | 0.097 | $-0.706$ | 1.000 |
| | | visualization | 0.080 | 0.097 | 0.825 | 1.000 |
| | | both | $-0.038$ | 0.097 | $-0.398$ | 1.000 |
| | recommendations | visualization | 0.148 | 0.097 | 1.532 | 0.775 |
| | | both | 0.030 | 0.097 | 0.309 | 1.000 |
| | visualization | both | $-0.118$ | 0.097 | $-1.223$ | 1.000 |
| relations | no assistance | recommendations | $-$**0.224** | 0.049 | $-4.524$ | $<.001$ *** |
| | | visualization | 0.025 | 0.049 | 0.842 | 1.000 |
| | | both | $-0.179$ | 0.049 | $-3.691$ | 0.002 ** |
| | recommendations | visualization | 0.259 | 0.049 | 5.367 | $<.001$ *** |
| | | both | 0.055 | 0.049 | 0.834 | 1.000 |
| | visualization | both | $-0.204$ | 0.049 | $-4.533$ | $<.001$ *** |

*, **, *** statistically significant results of increasing degrees.
[a]P-value adjusted for comparing a family of six using Bonferroni correction.

**Table 3.** Post hoc comparisons of assistance features on completion time. We abbreviate mean difference with MD and standard error with SE.

| | | | MD | SE | t | $p_{bonf}{}^{a}$ |
|---|---|---|---|---|---|---|
| time (s) | no assistance | recommendations | 23.778 | 30.528 | 0.779 | 1.000 |
| | | visualization | $-67.621$ | 30.528 | $-2.215$ | 0.176 |
| | | both | $-56.082$ | 30.528 | $-1.837$ | 0.418 |
| | recommendations | visualization | $-91.399$ | 30.528 | $-2.994$ | 0.022 * |
| | | both | $-79.860$ | 30.528 | $-2.616$ | 0.063 |
| | visualization | both | 11.539 | 30.528 | 0.378 | 1.000 |

*, **, *** statistically significant results of increasing degrees.
[a]P-value adjusted for comparing a family of six using Bonferroni correction.

**Table 4.** Post hoc comparisons of assistance features on subjective metrics. Largest statistical significant absolute difference to unassisted annotation for a given metric is set in **bold**. We abbreviate mean difference with MD and standard error with SE.

| | | | MD | SE | t | $p_{bonf}{}^{a}$ |
|---|---|---|---|---|---|---|
| mentaldemand | no assistance | recommendations | **24.677** | 3.521 | 7.009 | $<.001$ *** |
| | | visualization | 4.194 | 3.521 | 1.191 | 1.000 |
| | | both | 21.290 | 3.521 | 6.047 | $<.001$ *** |
| | recommendations | visualization | $-20.484$ | 3.521 | $-5.818$ | $<.001$ *** |
| | | both | $-3.387$ | 3.521 | $-0.962$ | 1.000 |
| | visualization | both | 17.097 | 3.521 | 4.856 | $<.001$ *** |
| uncertainty | no assistance | recommendations | 21.129 | 3.558 | 5.938 | $<.001$ *** |
| | | visualization | 4.677 | 3.558 | 1.314 | 1.000 |
| | | both | **24.355** | 3.558 | 6.844 | $<.001$ *** |
| | recommendations | visualization | $-16.452$ | 3.558 | $-4.623$ | $<0.001$ *** |
| | | both | 3.226 | 3.558 | 0.907 | 1.000 |
| | visualization | both | 19.677 | 3.558 | 5.530 | $<.001$ *** |
| effort | no assistance | recommendations | **22.419** | 3.710 | 6.043 | $<.001$ *** |
| | | visualization | 5.000 | 3.710 | 1.348 | 1.000 |
| | | both | 21.129 | 3.710 | 5.695 | $<.001$ *** |
| | recommendations | visualization | $-17.419$ | 3.710 | $-4.695$ | $<.001$ *** |
| | | both | $-1.290$ | 3.710 | $-0.348$ | 1.000 |
| | visualization | both | 16.129 | 3.710 | 4.347 | $<.001$ *** |
| frustration | no assistance | recommendations | **20.484** | 3.655 | 5.604 | $<.001$ *** |
| | | visualization | 3.548 | 3.655 | 0.971 | 1.000 |
| | | both | 18.548 | 3.655 | 5.074 | $<.001$ *** |
| | recommendations | visualization | $-16.935$ | 3.655 | $-4.633$ | $<.001$ *** |
| | | both | $-1.935$ | 3.655 | $-0.530$ | 1.000 |
| | visualization | both | 15.000 | 3.655 | 4.104 | $<.001$ *** |

*, **, *** statistically significant results of increasing degrees.
[a] P-value adjusted for comparing a family of six using Bonferroni correction.

## References


1. van der Aa, H., Di Ciccio, C., Leopold, H., Reijers, H.A.: Extracting declarative process models from natural language. In: CAiSE (2019)
2. van der Aa, H., Leopold, H., Mannhardt, F., Reijers, H.A.: On the fragmentation of process information: challenges, solutions, and outlook. In: International Workshop on Business Process Modeling, Development and Support (2015)
3. Van der Aa, H., Leopold, H., Reijers, H.A.: Checking process compliance against natural language specifications using behavioral spaces. IS (2018)
4. Ackermann, L., et al.: Recent advances in data-driven business process management (2024)
5. Ackermann, L., Neuberger, J., Jablonski, S.: Data-driven annotation of textual process descriptions based on formal meaning representations. In: CAiSE (2021)

6. Ackermann, L., Neuberger, J., Käppel, M., Jablonski, S.: Bridging research fields: an empirical study on joint, neural relation extraction techniques. In: CAiSE (2023)
7. Bagozi, A., Bianchini, D., De Antonellis, V., Garda, M., Melchiori, M.: Personalised exploration graphs on semantic data lakes. In: On the Move to Meaningful Internet Systems: OTM 2019 Conferences: Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, 21–25 October 2019, Rhodes, Greece (2019)
8. 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)
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. Bergh, D.D.: Problems with repeated measures analysis: demonstration with a study of the diversification and performance relationship. Acad. Manage. J. (1995)
11. 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)
12. Franceschetti, M., Seiger, R., López, H.A., Burattin, A., García-Bañuelos, L., Weber, B.: A characterisation of ambiguity in BPM. In: International Conference on Conceptual Modeling (2023)
13. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: CAiSE (2011)
14. Greenhouse, S.W., Geisser, S.: On methods in the analysis of profile data. Psychometrika (1959)
15. Hart, S.G.: Nasa-task load index (NASA-TLX); 20 years later. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting (2006)
16. Ivanchikj, A., Serbout, S., Pautasso, C.: From text to visual BPMN process models: design and evaluation. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (2020)
17. Joshi, A., Kale, S., Chandel, S., Pal, D.K.: Likert scale: Explored and explained. Br. J. Appl. Sci. Technol. (2015)
18. Käppel, M., Schönig, S., Jablonski, S.: Leveraging small sample learning for business process management. Inf. Softw. Technol. (2021)
19. Kim, B.G., Stein, H.H.: A spreadsheet program for making a balanced Latin square design. Rev. Colombiana Ciencias Pecuarias (2009)
20. Klie, J.C., Bugert, M., Boullosa, B., de Castilho, R.E., Gurevych, I.: The inception platform: machine-assisted and knowledge-oriented interactive annotation. In: Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations (2018)
21. Kourani, H., Berti, A., Schuster, D., van der Aalst, W.M.: Process modeling with large language models. arXiv preprint arXiv:2403.07541 (2024)
22. Li, J., Sun, A., Han, J., Li, C.: A survey on deep learning for named entity recognition. IEEE Trans. Knowl. Data Eng. (2020)
23. Ludbrook, J.: Multiple comparison procedures updated. Clin. Exp. Pharmacol. Physiol. (1998)
24. Mauchly, J.W.: Significance test for sphericity of a normal N-variate distribution. Ann. Math. Stat. (1940)
25. Mustansir, A., Shahzad, K., Malik, M.K.: AutoEPRS-20: extracting business process redesign suggestions from natural language text. In: ASE (2020)
26. Nakayama, H., Kubo, T., Kamura, J., Taniguchi, Y., Liang, X.: doccano: text annotation tool for human (2018). https://github.com/doccano/doccano
27. 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 (2024). https://arxiv.org/abs/2407.18540

28. 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)
29. Qian, C., et al.: An approach for process model extraction by multi-grained text classification. In: CAiSE (2020)
30. Quishpi, L., Carmona, J., Padró, L.: Extracting annotations from textual descriptions of processes. In: BPM 2020 (2020)
31. Rosa, L.S., Silva, T.S., Fantinato, M., Thom, L.H.: A visual approach for identification and annotation of business process elements in process descriptions. Comput. Stand. Interfaces (2022)
32. Schützenmeier, N., Käppel, M., Fichtner, M., Jablonski, S.: Scenario-based model checking of declarative process models. In: ICEIS (2023)
33. Sukthanker, R., Poria, S., Cambria, E., Thirunavukarasu, R.: Anaphora and coreference resolution: a review. Inf. Fusion (2020)
34. Tkachenko, M., Malyuk, M., Holmanyuk, A., Liubimov, N.: Label studio: data labeling software (2020-2022). https://github.com/heartexlabs/label-studio
35. Xu, M., Jiang, H., Watcharawittayakul, S.: A local detection approach for named entity recognition and mention detection. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2017)

# Chapter 11

# TeaPie — A Tool for Efficient Annotation of Process Information Extraction Data

Neuberger, J., Herrmann, J., Käppel, M., van der Aa, H., Jablonski, S. (2025). TeaPie: A Tool for Efficient Annotation of Process Information Extraction Data. In: Comuzzi, M., Grigori, D., Sellami, M., Zhou, Z. (eds) *Cooperative Information Systems. CoopIS 2024.* Lecture Notes in Computer Science, vol 15506. Springer, Cham. https://doi.org/10.1007/978-3-031-81375-7_28



**Contribution statement.** Jannic Herrmann (JH) wrote the code for the TeaPie tool, Julian Neuberger (JN) adjusted the code for deployment and made it available for public use. JN evaluated and interpreted user study data and created corresponding figures. All authors wrote and revised the publication text. JN is the corresponding author.

# TeaPie: A Tool for Efficient Annotation of Process Information Extraction Data

Julian Neuberger[1(✉)], Jannic Herrmann[1], Martin Käppel[1], Han van der Aa[2], and Stefan Jablonski[1]

[1] University of Bayreuth, Universitätsstraße 30, 95444 Bayreuth, Germany
{julian.neuberger,jannic.herrmann,martin.kaeppel,
stefan.jablonski}@uni-bayreuth.de
[2] University of Vienna, Universitätsring 1, 1010 Vienna, Austria
han.van.der.aa@univie.ac.at

**Abstract.** Machine-learning based generation of process models from natural language text process descriptions is severely restrained by a lack of datasets. This lack of data can be attributed to, among other things, an absence of proper tool assistance for dataset creation, resulting in high workloads and inferior data quality. We address these shortcomings with a tool for annotating textual process descriptions. Compared to other, existing data annotation tools, ours implements a multi-step workflow specifically designed for extracting process information, including supporting features that have been shown to reduce workloads and improve data quality.

**Keywords:** Process information extraction · Text annotation · Business process management

## 1 Introduction

Organizations looking to utilize the benefits of Business Process Management (BPM) initially have to model their internal business processes. These so-called as-is process models are expensive to create, as it is a time consuming task, usually performed by BPM experts together with process experts of the organization [3]. To accelerate this initial step, approaches using Natural Language Processing (NLP) have been proposed. These extract the process-relevant information contained in textual process descriptions of various sources, such as quality management handbooks, standard operating procedures, or employee notes [2]. In a subsequent step, this information is transformed into formal models, e.g., in the BPMN modeling standard (see https://www.omg.org/bpmn/).

While approaches based on machine learning became more common in recent years [5,13], Process Information Extraction (PIE) still has not adopted the state-of-the-art machine learning techniques and architectures used in other fields of information extraction, even though the tasks share many similarities [4]. These approaches need vast amounts of annotated training data, which is not yet available in BPM in general [9], and especially for PIE [13], where the currently largest available dataset (PET [6]) contains just 45 process descriptions. Approaches based on Large Language

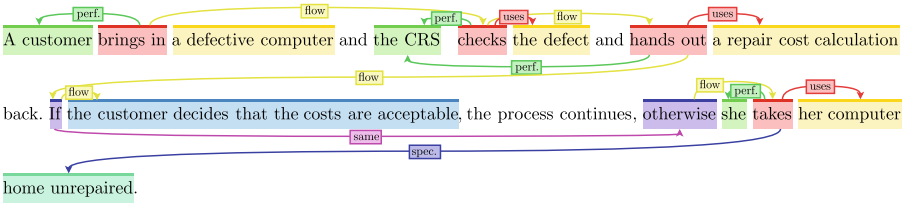



**Fig. 1.** Example for the high information density of PIE data. Of 40 tokens total, only six (15%) are not directly relevant for the process. The text is a fragment of *doc-1.2* of the PET dataset.

Models (LLM) circumvent this issue, as they are pretrained on out-of-domain data, and only need marginal amounts of data for in-context learning [5]. However, they are hard to optimize for this task, cause considerable costs, and have a poor ecological footprint, making them a suboptimal solution. Accordingly, approaches specifically trained for PIE are preferable. To provide appropriate training data for this, it is necessary to annotate large amounts of natural language text. Although annotating text is a common task in machine learning research and is supported by various tools to enhance efficiency and productivity, most existing tools are not suited for annotating *process description* text. This is primarily due to three reasons: First, process descriptions have a very high density of information (cf. Fig. 1). As a result, identifying, annotating, and displaying information quickly becomes confusing, which hampers completeness and correctness. Second, annotation of process information is very susceptible to errors, which invalidate the resulting process entirely. Such errors include, but are not limited to accidentally reversing control-flow, disjointed process models, or missing decision points in the process (*XOR-Gates*). We argue, supporting the user with proper visualizations while they annotate yields more complete and correct process models. Third, annotation of process descriptions is often ambiguous. This means that there is more than one arguably correct set of annotations, which makes annotating process descriptions mentally demanding, as many possibilities have to be considered at any given time. The validity of these issues is underlined by other work in the same context, such as the tool Model Judge [8], which facilitates the training of novice modelers in the text-to-model task. To this end, the user's model is compared to a gold standard model and discrepancies are highlighted. While the means of Model Judge are very similar, the ends differ fundamentally—most notably, there is no gold standard during data annotation to which an annotation could be compared to. To address these issues, we present TeaPie, a tool for efficient process information annotation[1].

## 2 System Overview

To facilitate the extraction of process information from textual descriptions, we developed a modular annotation tool comprising three main components: the front-end web application, the annotation backend and the visualization server. Fig. 2 shows a high-level overview of the architecture of TeaPie.

[1] See https://github.com/JulianNeuberger/assisted-process-annotation for code, video, and live demo. Credentials: *coopis* (user), *processes2024* (password).

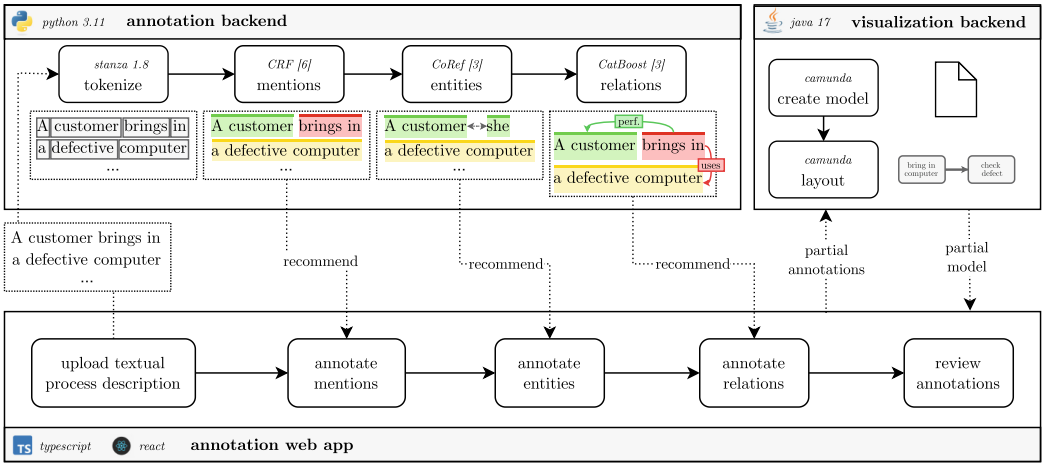



**Fig. 2.** Overview of the modular architecture of TeaPie.

**The front-end** web application is implemented using TypeScript and React[2]. It serves as the primary interface for annotators to interact with the system, guiding them through the annotation workflow.

**The annotation backend** server is built with Python 3.11 and handles various NLP tasks required for generating annotation suggestions. After a user submits a process description, it is tokenized using the Stanza package[3]. The resulting tokens are then fed into three prediction models, to generate annotation recommendations for the user. First a Conditional Random Fields model from Bellan et al. [5] identifies and extracts mentions of process-relevant entities. Following this, the pre-trained neural co-reference resolution model presented in [13] clusters mentions referring to the same entity throughout the process description. Finally, we use the CatBoost model presented in [13] to extract relation. Models are trained on 80% of the PET dataset, while the remaining 9 documents were set aside for the user study.

**The visualization server** is developed in Java 17 and utilizes the Camunda Model API[4] to generate graphical models for the front-end. The graphical model updates whenever annotations are modified.

## 3 Key Innovations

One of TeaPie's three core innovation lies in its six-step workflow, which is specifically designed to reduce the complexity of PIE, both for expert, as well as for beginner annotators. As such, for each piece of process information (mentions, entities, relations) annotations are first recommended by TeaPie, reviewed by the annotator, and subsequently amended with missing annotations. During early prototyping iterations, we found that annotators often would recognize additional mentions during annotation of

[2] See https://www.typescriptlang.org/ and https://react.dev/ respectively.
[3] See https://stanfordnlp.github.io/stanza/pipeline.html.
[4] See https://docs.camunda.org/manual/7.21/user-guide/model-api/.

entities and relations, which is why we added a final review step, where all information is presented at once and annotations can be rectified before finalizing.

The second innovation that sets TeaPie apart from other annotation tools, is the visualization of the current state of annotated process information. This visualization is generated as soon as the first process relevant entity mention is annotated, and regenerated, whenever the annotations change. This gives users immediate feedback on the information they annotated so far, which helps them understand the impact of certain annotations on the over all process model. This was especially helpful for annotators familiar with BPMN, who compared the graphical process model with their expectations.

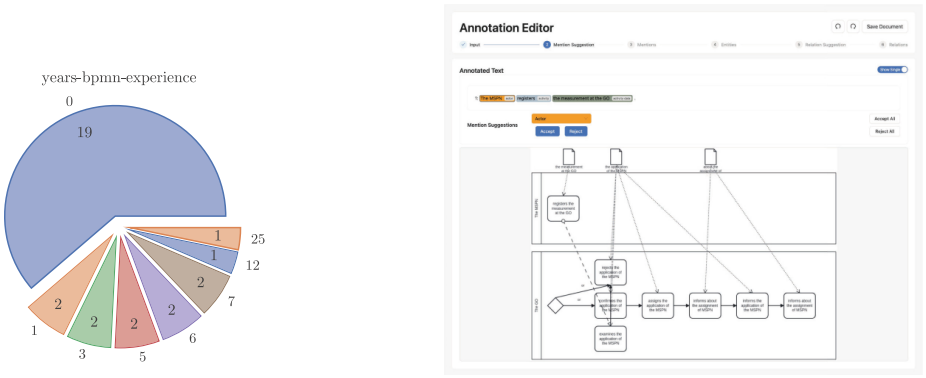
The third innovation of TeaPie are machine learning based recommendations. While other text annotation tools support recommendations of annotations, TeaPie generates recommendations for all three types of PIE data end-to-end. These recommendations have been shown to improve the quality of annotations beyond what either humans or recommendation system in isolation could achieve, while at the same time making the annotation process cognitively less taxing [11]. Furthermore, recommendations help to bridge the experience gap between annotators, which makes it easier to assemble teams [11].

## 4 Maturity

We evaluated TeaPie in a controlled user study where we asked 31 participants to annotate fragments of textual process descriptions and recorded their feedback regarding TeaPie's practicality. Note that 19 of the 31 participants (61.3%) had no prior experience in BPMN. These users are potential data annotators, currently unable to contribute to PIE annotation projects, due to their inherent complexity and ambiguity [1]. For this reason their feedback is particularly valuable to us. In the following we present a brief analysis focused on usability. A detailed analysis of metrics like annotation accuracy, mental workload, or time per document can be found in the full paper of our user study [11]

We found that the workflow we implemented was well suited to how most users extract process relevant information from text. Fig. 4 shows the how much users agreed with statements regarding certain aspects of the workflow implemented in TeaPie. Most users felt the speed with which they were able to complete their tasks was satisfactory (Fig. 4a), understood their task in each extraction step (Fig. 4b), and agreed with the order of extraction steps (Fig. 4c). All users were satisfied with the way the workflow was implemented (Fig. 4d,e). This is especially encouraging, as first time BPMN users agree with experts in this matter, leading us to believe the workflow provides good guidelines for novice users, while not being overly restrictive for experienced ones.

**Limitations.** We currently see two main limitations with TeaPie. First, the process generation algorithm we use is a very rough prototype and sometimes results in confusing or incomplete process models. This results in many users rating the visualization as less useful, preferring annotation recommendations over the visualization of currently extracted information (see Fig. 5c). We plan to use an improved visualization algorithm to further improve the usefulness of the process model visualization. Second,




(a) (b)

**Fig. 3.** Years of experience with BPMN of user study participants (left), and a screenshot of TeaPie (right).

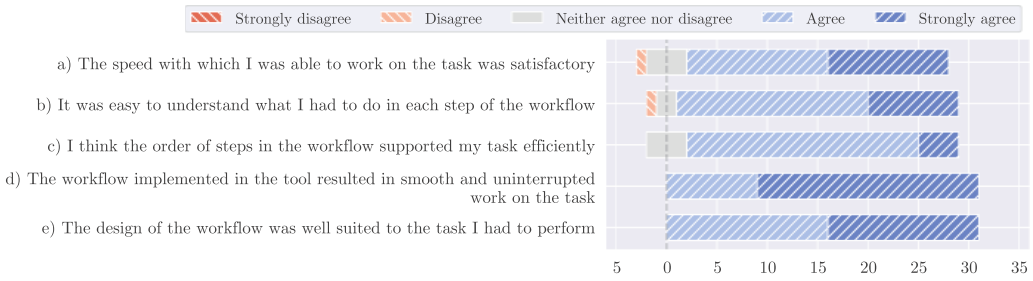



**Fig. 4.** User feedback regarding the workflow implemented in TeaPie.

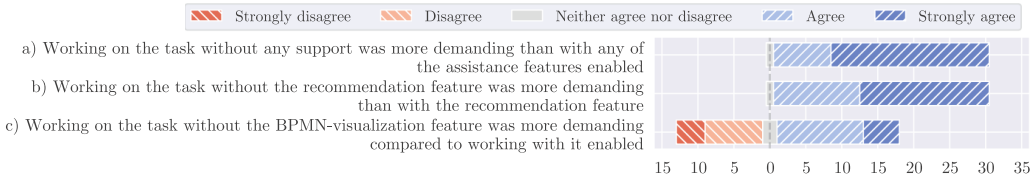



**Fig. 5.** User preferences regarding the supporting features of TeaPie.

TeaPie only supports the PET data annotation schema. We are actively working on the dynamic definition of annotation schemas in a graphical user interface integrated into TeaPie. This will make TeaPie useful in data annotation projects for various modelling languages, e.g., DCR graphs [10], as well as different paradigms, e.g., Object-Centric modelling.

**Future Work.** Besides the future work mentioned during our discussion of current limitations, we plan to extend TeaPie with additional features for large-scale data annotation projects. First, we want to integrate features to support the collaboration of multiple annotators. These features include, among others, the automatic calculation of inter-annotator agreement, i.e., how well the annotations of two or more annotators align. Process descriptions where annotators disagree, will be assigned to a referee anno-

tator. This concept proved useful for other text annotation projects [7] and results in higher data quality. Next, TeaPie will provide annotation statistics, such as linguistic variability of process elements, preliminary results of training extraction models, or the percentage of process relevant text in documents. Such statistics are often included in articles presenting new datasets (cf. PET [6]). Additionally, providing automatic conversion of other modalities to text, such as image-to-text, or audio-to-text, could enable new applications of TeaPie. Furthermore, we plan to experiment with different approaches towards generating annotation recommendations. The current approach uses very few learnt parameters, which makes it efficient, but less effective compared to LLMs, which outperform shallow machine learning approaches [12]. Finally, we plan to provide a publicly accessible instance of TeaPie for use in annotation projects of the BPM research community.

## References

1. van der Aa, H., Leopold, H., Mannhardt, F., Reijers, H.A.: On the fragmentation of process information: challenges, solutions, and outlook. In: BPMDS (2015)
2. Ackermann, L., et al.: Recent advances in data-driven business process management (2024)
3. Ackermann, L., Neuberger, J., Jablonski, S.: Data-driven annotation of textual process descriptions based on formal meaning representations. In: CAiSE (2021)
4. Ackermann, L., Neuberger, J., Käppel, M., Jablonski, S.: Bridging research fields: an empirical study on joint, neural relation extraction techniques. In: CAiSE (2023)
5. 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)
6. 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)
7. Brants, T.: Inter-annotator agreement for a German newspaper corpus. In: LREC, Citeseer (2000)
8. Delicado Alcántara, L., Sanchez-Ferreres, J., Carmona Vargas, J., Padró, L.: The model judge: a tool for supporting novices in learning process modeling. In: BPMTracks. CEUR-WS. org (2018)
9. Käppel, M., Schönig, S., Jablonski, S.: Leveraging small sample learning for business process management. Inf. Softw. Technol. **132**, 106472 (2021)
10. López, H.A., Strømsted, R., Niyodusenga, J.M., Marquard, M.: Declarative process discovery: linking process and textual views. In: CAiSE (2021)
11. 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 (2024)
12. 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. arXiv preprint arXiv:2407.18540 (2024)
13. 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)

# Chapter 12

# Repeat, Reorder, Rephrase — Data Augmentation for Process Information Extraction

Neuberger, J., Ackermann, L., Jablonski, S. (2025). Repeat, Reorder, Rephrase — Data Augmentation for Process Information Extraction. Submitted for publication.

**Contribution statement.** Julian Neuberger (JN) extended the list of data augmentation techniques compared to the conference publication [54]. JN adjusted the experiment setup, conducted the experiments, and interpreted the results. JN developed the idea of analyzing the changes in meaning caused by data augmentation, implemented it, and interpreted the resulting plots. JN wrote the majority of new content for this publication. All authors revised the final version of the publication.

# Repeat, Reorder, Rephrase — Data Augmentation for Process Information Extraction

**Julian Neuberger**[1], **Lars Ackermann**[1], **Stefan Jablonski**[1]

University of Bayreuth, Universitätsstraße 30, Bayreuth, Germany

Received: date / Revised version: date

**Abstract** Automatic retrieval of formal business process models from their natural language descriptions is a well established way to facilitate the time and cost intensive modeling procedure. Yet, a lack of data usable for developing and training new retrieval methods is impeding progress in this field of research.

This issue can be overcome by either using methods less reliant on high quality data, such as large language models, or by creating bigger datasets. The latter is often preferable in the context of business process modeling, especially when internal workflows of organizations have to be treated confidentially. It is the more data-intensive solution, though, which is costly. Data augmentation techniques aim to improve both quality and quantity of existing datasets, by deliberate perturbations resulting in new, synthetic data.

In this article we present a collection of simple data augmentation techniques, which are specifically selected for the task of improving data quality in the context of process information extraction. We show why data augmentation techniques from the wider field of natural language processing are often not applicable to process information extraction, and how the resulting data differs in terms of linguistic variety, structure, and feature space coverage. In our experiments, data augmentation results in an absolute improvement in the $F_1$ measure of 5.7% for extracting process relevant entities from text, and 4.5% for extracting relations between those entities.

We make all code available at `https://github.com/JulianNeuberger/pet-data-augmentation`, and detailed results for our experiments at `https://zenodo.org/doi/10.5281/zenodo.10941423`.

## 1 Introduction

In recent years, many systems for extracting process relevant information from natural language process descriptions have been proposed to expedite process discovery, i.e., the initial modeling of an as-is process [16,41,7,30, 29,37,2,33]. Interest in this research topic is founded in the fact that discovering such an as-is process is known to require up to 60% of time planned for new business process management projects [17]. Generally, generating process models from natural language descriptions is done in two phases. First, process relevant information is extracted from the text, which is then used to synthesize a formal business process model. Note, that there are direct transformation approaches, as well as conversational agents to generate process models from text [25,13,27].

While there are unsupervised approaches to the first phase [6,18,26,34], many recent approaches to business process information extraction are still based on supervised machine learning, which requires tuples of input (process descriptions) and expected output (e.g., process relevant actors, activities, etc.). Such tuples are expensive to create and require deep knowledge about processes, as well as, natural language, which is why even the largest collection of data for process information extraction is still comparatively small, containing just 2,000 examples of process relevant entities [7], while datasets for other information extraction tasks contain multiple orders of magnitude more examples. Datasets for extraction of named entities and their relations, for example the DocRed dataset, contain more than 1,500,000 examples of entities and relations between them [42].

Other domains of research use data augmentation (DA) to improve both quality and quantity of training data. This entails creating new data tuples, by applying targeted perturbations on the input, while preserving the underlying semantics, and therefore maintaining the validity of the expected output, e.g., extracted information. One of the earliest adopters of data augmentation was the field of computer vision, i.e., image classification. There images are perturbed by operations including, but not limited to, rotation, translation, or addition of noise. Fig. 1 shows these operations applied to an image of a handwritten *9*. Note how the intensity of

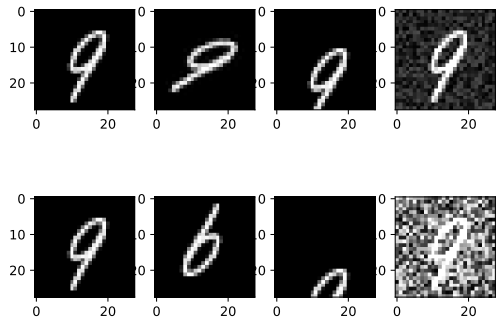


Fig. 1: Motivating example for data augmentation in computer vision.

operations dictates the usefulness of data augmentation in this example. Rotating the image by a few degrees keeps its *semantics*, i.e., it is still an image of the digit 9, but rotating it by 180 degrees yields an image of a 9. Similarly, translating the image too much makes the image ambiguous, and introducing too much noise makes it hard to decipher, even for humans. Still, when configured properly, data augmentation enables more efficient use of valuable training data, creation of more robust models, and better generalization capabilities of those [39, 40].

Despite its popularity and success in other fields of research, data augmentation is used in very few areas of business process management [1, 23, 22]. In this article, we apply 26 data augmentation techniques, specifically designed for information extraction tasks, to process data. We analyze how these data augmentation techniques change the data on a linguistic level, how the resulting dataset covers the feature space, and present configurations of augmentations, which results in data that allows us to train machine learning process information extraction approaches with a total, absolute increase in performance of up to 5.7%. We find that simple to use and computationally cheap perturbations rival the use of large language models in terms of extraction quality improvement, for the current state of the art in process information extraction approaches.

This article is an extension of our previous work in [35], where we used a black-box evaluation to determine the feasibility of data augmentation in a process information extraction context. This work expands on this in two major ways. *(1)* We adjust all augmentation techniques used in the experiment as well as the experiment itself, to improve their applicability to the process information extraction task, We describe these modifications in more detail in Sect. 4.4. Furthermore, we investigate five additional augmentation techniques and two oversampling techniques. *(2)* Sect. 5 discusses the effects of data augmentation on the data itself. This includes changes in the process description text (Sect. 5.1), common errors introduced by data augmentation (Sect. 5.2), and visualizations of changes in meaning through data augmentation (Sect. 5.3).

The rest of this article is structured as follows. Sect. 2 defines important notions of data augmentation and the process information extraction task. Sect. 3 covers work related to this article. We then describe our experiment setup in Sect. 4, starting with our leading research questions (Sect. 4.1), the selection of data augmentation techniques (Sect. 4.2), a classification of these (Sect. 4.3), how we adjusted some techniques for the use with process information extraction data (Sect. 4.4), and finally our strategy for finding optimal configurations for data augmentation techniques (Sect. 4.5). Sect. 5 analyzes the effects of data augmentation on the text of process descriptions (Sect. 5.1), common errors introduced by data augmentation (Sect. 5.2), visualizations of the changes in meaning (Sect. 5.3), and vocabulary and relation direction (Sect. 5.4). Sect. 6 discusses the results of a black-box evaluation using data augmented with the selected techniques and answers our research questions. We conclude the article in Sect. 7.

## 2 Background

**Data augmentation** describes a suite of techniques originally popularized in computer vision [40], where simple operations, such as cropping, rotating, or introducing noise into images greatly improved performance of machine learning algorithms used for classification of images. These operations usually preserve the semantics of input data, meaning that an image containing an object will still depict the same object after its data have been augmented, for example, they have been overlaid with noise. This property is called *invariance* [43], and is harder to hold for natural language data [15]. An example for this fact is depicted in Figure 2. Changing random tokens (e.g., words) in a sentence may alter semantics to a point, where relevant elements or relations between those elements are no longer present after augmentation. Additionally, annotations may be lost, if techniques are applied and afterwards these changes cannot be traced. This might happen, when, for example, an entire sentence is translated into another language and then is back-translated typically leading to a rephrased version of the original text. Since it is not clear, which parts of the new sample correspond to the original one, annotations of process-relevant elements do not apply to the new sample. For this reason, research on data augmentation techniques has been conducted, which are specifically designed for information extraction tasks [20, 28, 14]. These techniques use additional resources, such as pretrained large language models, to augment training samples, while keeping their semantics intact.

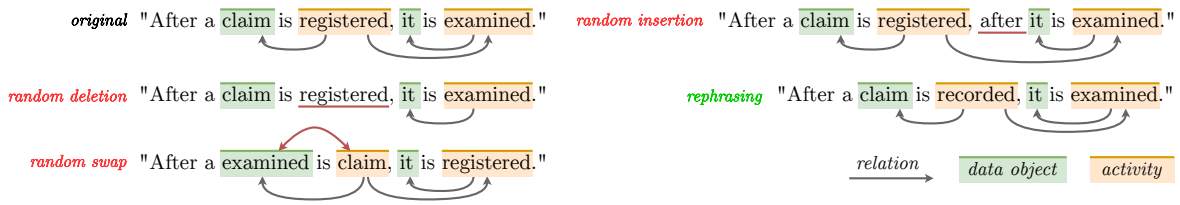



Fig. 2: Examples for four different data augmentation techniques. *Random deletion*, *random swap*, *random insertion* (all written in red), are all not preserving the semantics of a sample and its label. *Rephrasing* (green) is an example for a technique that does.

**Process Relevant Information Extraction** from natural language is a research field immediately relevant for information systems, as business process models are often a central part for process aware information systems. Discovering and creating these process models is an expensive task [17] and a lot of work has been done on extracting them from natural language text directly [17, 41, 16, 2, 6]. These texts describe a business process in natural language as technical documentation, maintenance handbooks, or interview transcripts. Sequences of words (spans) in these texts contain information that is relevant to the business process, such as *Actors* (persons or departments involved in the process), *Activities* (tasks that are executed), or *Data Objects* (physical or digital objects involved in the process). Extracting this information is therefore a sequence tagging task, and can be framed as *Mention Detection* (MD). Mentions relate to each other, e.g., defining the order of execution for two Activities, or which Actor executes the Activity. Predicting and classifying these relations is called *Relation Extraction* (RE). Refer to Figure 3 for an example of this process. It shows a fragment of a larger description of a process from the insurance domain, where insurance claims have to be registered in a system and subsequently examined by an employee. The spans *claim* and *it* are annotated as Data Objects (the claim in question, in green). Activities executed by a process participant are marked in orange. These four spans can now be transformed into business process model elements for a target notation language (here BPMN[1]). How these elements interact with each other can also be extracted from the text fragment, e.g., the *Flow* of activity execution between the mentions *registered* and *examined*, depicted as an orange arrow.

Developing approaches towards automated extraction of process relevant information requires data to test performance, and train models, if applicable. The currently largest collection of human-annotated process descriptions is called PET [7]. It contains 45 natural language process descriptions, and is annotated with 7 types of process relevant entities (e.g., Actors, Activities, Data Objects), as well as 6 types of relations between them

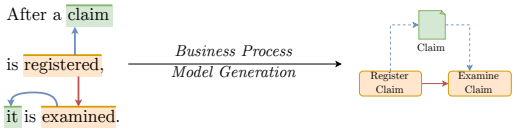



Fig. 3: Example for a fragment of a natural language business process description and its corresponding business process model fragment in BPMN.

(e.g., Flow between Activities). In total the dataset contains less than 2,000 examples for both relations and entity mentions. For comparison, typical datasets for related tasks, like Knowledge Graph completion contain more than 200 times as many. For example, the popular *FB15k* dataset comprises more than 500,000 relation examples [10]. Datasets for extraction of named entities and their relations have similar extents, e.g., the DocRed dataset, which contains more than 1,500,000 relation examples [42]. This fact makes PET a prime candidate for data augmentation techniques, in order to make the most out of the limited amount of training examples. We show this in our experiments using PET for the tasks MD and RE in process information extraction. To our knowledge our work is the first to attempt applying NLP data augmentation to the process information extraction task.

## 3 Related Work

Data augmentation techniques applied in this paper are largely based on the ones available in the *NL-Augmenter* framework [12]. NL-Augmenter provides a list of more than 100 data augmentation techniques, which are suitable for varying tasks like text classification, sentiment analysis, and even tagging. We discuss how we adapted these techniques to the PET data format in more detail in Section 4. Not all techniques are relevant for this work, and we have to exclude most of them, as they are not fitting for process information extraction. Details of our exclusion criteria can be found in Section 4.

In [22] the authors evaluate nine simple data augmentation techniques (e.g., random deletion) on a total of seven event logs, using seven different models. Our

[1] See specification at `https://www.bpmn.org/`.

paper follows a similar line of thought for process information extraction, instead of predictive process monitoring. The techniques we employ differ significantly from theirs in two core aspects. First, techniques used in this paper are more complex, owing to the more complex character of natural language. While their work focused on reordering events in a log of a process execution, our work uses techniques that are concerned with replacing, extending, or modifying sequences of text, while preserving any annotations present in the data. Second, techniques used in our work often require external resources. These resources can be explicit, i.e., databases like WordNet [31], which contains lexical information such as synonyms, antonyms, or hypernyms of words. They can also be implicit, such as large language models, which contain knowledge about natural language, obtained by unsupervised training on huge amounts of textual data [11].

The techniques we present in our paper mainly benefit work that already exists in the field of process information extraction. Therefore, approaches based on machine learning are related to this work. These approaches can be separated into two main fields of research. *(1)* learning approaches, which use the data to train a machine learning models, e.g., a neural network [2], conditional random fields [7], or decision trees [33]. *(2)* prompting based approaches that use the data for engineering input for large language models (e.g., GPT) [21,24,34], or use the data for so called *in context learning*, by providing examples in the input itself [6].

Automated extraction of information relevant to business processes from natural language text descriptions can be seen as a special case of automated knowledge graph construction or completion [5]. We therefore consider techniques for automated knowledge graph construction and completion as distantly related work, which could still benefit from the augmentation techniques we analyze in this paper. Nonetheless, we focus on methods of process information extraction in this paper, as potential solution for this field's small datasets.

## 4 Experiment Setup

This article answers four leading research questions, which we define in Sect. 4.1. To this end we use data augmentation techniques from the domain of natural language processing. The NL Augmenter framework [12] provides a total of 119 of such data augmentation techniques, but not all of them are applicable to the task at hand. We therefore define four criteria for exclusion in Sect. 4.2. We group the remaining, selected data augmentation techniques into five classes in Sect. 4.3 and discuss how we modify them to be better applicable to process descriptions in Sect. 4.4. Finally, in Sect. 4.5 we discuss our approach towards finding suitable configurations for the parameters of data augmentation techniques.

### *4.1 Research Questions*

Following the intuition from Sect. 1, our first two research questions **RQ1** and **RQ2** are focused on answering how much knowledge about natural language and process information extraction is needed to augment data for the process information extraction task.

While *simple data augmentation*, i.e., randomly deleting, swapping, or inserting words into a text, are easy to implement and require no additional knowledge about natural language or process information extraction, they are also prone to breaking semantics of process descriptions, and introducing unnatural noise. Still, related work in business process management have been shown to benefit from such simple data augmentation techniques, e.g., predictive business process monitoring [22,23]. These considerations lead us to raising research question **RQ1**.

On the other hand, using large scale language modeling, e.g., through the use of pretrained language models, such as BERT [11], or GPT [38], is highly demanding in terms of resources (hardware), and time. We therefore are interested in how useful these methods are compared to smaller, rule-based methods, i.e., if the investment of hardware and time is worth it through significantly higher data quality, measured by the performance gain of models trained with it. We therefore pose research question **RQ2**.

One popular technique to improve classification of rare classes in a classification task is called *oversampling* [32]. Here, samples containing rare classes are shown multiple times during training. Preliminary experiments showed that the RE method presented in [33] can already benefit from this technique. Transformations that yield improvements lower than oversampling could even be considered detrimental to model performance. For this reason, a third question, **RQ3**, is focused on finding data augmentation techniques, that are better than simply repeating data.

The fourth and final one question, **RQ4**, is aimed at the *apparent* effects of data augmentation on the text of process descriptions. This question is aimed at understanding how process descriptions change as a result of applying data augmentation. Since this is a highly qualitative question, we aim to answer it by exploring process descriptions altered by data augmentation, finding common errors, and visualizing changes in characteristics.

In summary, our research questions are as follows.

**RQ1** Can simple data augmentation techniques, including swapping, deleting, or randomly inserting words into sentences increase the performance of machine learning methods for process information extraction, measured as the harmonic mean of precision and recall?

**RQ2** Does the use of deep learning models, especially (large) neural language models, in data augmentation, provide a significant advantage over simpler, rule-based methods?

**RQ3** Does data augmentation outperform oversampling of documents during training?
**RQ4** What characteristics of the natural language text data are changed by augmentations?

### *4.2 Selection of Techniques*

The NL Augmenter framework provides a total of 119 data augmentation techniques, but not all of them are applicable to the task at hand. We therefore define four criteria for exclusion.

**EC1** Language: The technique does not apply to the English language, i.e., we exclude techniques targeted at all other languages. The dataset we use for our experiments, PET, is in English, techniques targeting other languages are therefore not relevant for this paper.
**EC2** Spelling: The data augmentation technique alters the spelling of tokens, i.e., misspelling perturbations, which allows for evaluating robustness to spelling mistakes. This issue is not present in the PET dataset, but may be relevant for future work, if less perfect data sources (e.g., notes taken by employees) are analyzed.
**EC3** Supervised Training Data: The data augmentation technique does not work for supervised data, i.e., perturbations would corrupt labels present in the PET dataset and we can not adjust it to preserve labels.
**EC4** External Resources: The technique uses task-, and/or domain-specific resources, such as dictionaries, or databases, which do not exist for processes represented in PET and prevent the technique to be task-agnostic.

Applying these criteria results in 20 data augmentation techniques relevant for the task of generating business process models from natural language text. We have listed the exact number of data augmentation techniques excluded by each criterion in Figure 4. Two of the 20 relevant techniques had errors in their original code, which we fixed. Note that one of the 20 selected techniques had two modes of operation, which we split into two separate techniques, resulting in a total of 21 data augmentation techniques at this stage. We then we added five more data augmentation techniques, specifically designed to help us answering our four research questions, which are as follows.

**Random Swap.** Randomly selects two tokens in the document and swaps them. **Random Insert.** Uniformly samples tokens from the dataset vocabulary and inserts them at random positions in the document. Together with augmentation technique B.79 (Random Deletion) from the NL Augmenter framework, Random Swap, and Random Insert are the three techniques are the *simple data augmentation techniques* referenced in research question **RQ1**.

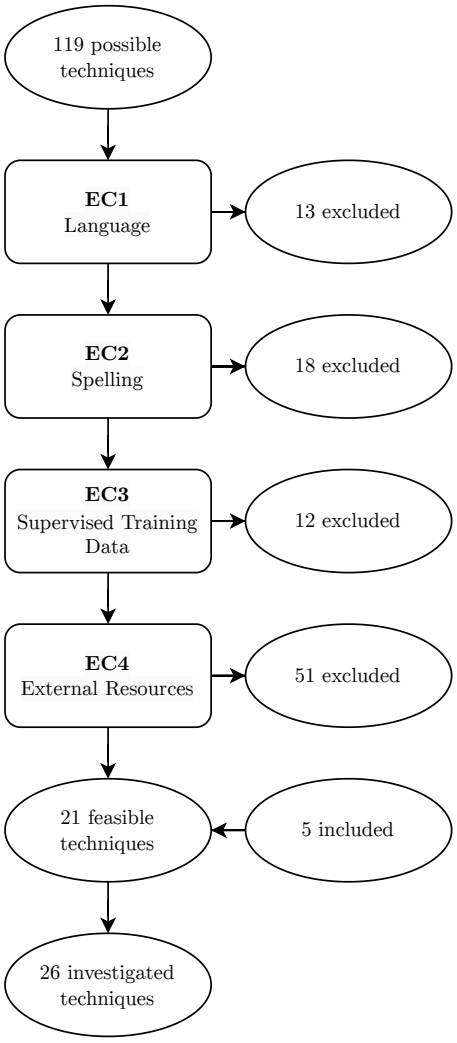



Fig. 4: Application of exclusion criteria, and inclusion of five additional techniques, leading to 26 investigated data augmentation and oversampling techniques.

**Inverse Type Frequency Oversampling.** Oversample, i.e., repeat documents in the dataset according to the rarity of their contained entity and relation types.
**Uniform Oversampling.** Randomly oversample documents from the dataset. These two oversampling techniques serve as a baseline of improvement, to answer research question **RQ3**.

**Large Language Model Rephrasing.** Rephrase text segments via a Large Language Model (LLM). We utilize a local *Llama 3.1* model with 70B parameters for this augmentation technique and use the results to supplement those of other, existing techniques in answering research question **RQ2**. This augmentation technique is the most time-intensive one by far (see Sect. 2), but should help us get a good idea of the usefulness of generative artificial intelligence for data augmentation.

*4.3 Classifying Data Augmentation Techniques*

Many data augmentation techniques are similar in their inner workings, i.e., they use similar approaches towards perturbing documents. In this section we present the five major categories of data augmentations we identified.

**C1** *Rephrasing.* This class of data augmentation techniques targets spans of one or more tokens and replaces them with synonymous wording. Techniques in this class are not guaranteed to keep syntactic and semantic integrity, but do so more often than techniques in other classes.
**C2** *Reordering.* Techniques that change the order of tokens, without introducing new tokens fall into this class. The scope of such techniques varies, and ranges from reordering the tokens of a single mention, reordering sentences (without reordering tokens inside these sentences), or randomly swapping tokens in a document. Techniques in this class regularly break linguistic syntax.
**C3** *Repeating.* When techniques use existing data, we categorize them as repetition techniques. These include data augmentation that concatenate documents, repeat whole documents (oversampling), or randomly insert token spans sampled from the dataset.
**C4** *Adding Noise.* Data augmentation techniques in this class randomly insert, delete, or replace tokens in the data. Inserted and replaced tokens are not part of the original dataset and include speaker phrases ("*uhm*", "*er*") or inserting (not substituting) synonyms of words.

*4.4 Adjustment for Process Information Extraction*

We have to adjust some of the augmentation techniques slightly, to make them applicable to the process information extraction task. Our adjustments fall into the following categories.
**Label Preservation.** This adjustment applies mainly to augmentations that replace spans of tokens, such as rephrasing techniques. If the technique would rephrase an entire sentence, such as "*After* ***a claim*** *is registered, it is examined.*", it could result in "*Following registration of a claim, a review follows.*". Even though the rephrased sentence has the same number of tokens, naively assuming the positions of labels have not changed would result in ***registration of*** as an extraction target for training extraction models, impacting the performance of models trained with such data. Instead, we segment the text into sequences with the same corresponding entity type, i.e., *After*, *a claim*, ,, *is registered*, *it*, *is examined*, and .. We then rephrase the segments in isolation and replace the entire original sequence with the resulting rephrased sequence. This way, we can preserve labels, but may break the semantics of the sample (see Sect. 5.2).
**Runtime Optimizations.** Some augmentation techniques had easy to fix bottlenecks adversely affecting runtime, such as loading language models multiple times throughout their lifetime. We fixed these cases, to make running augmentations many times feasible, which becomes important for Sect. 4.5.
**Forcing Variety.** When we augment a single document with an augmentation factor larger than 1, an augmentation technique potentially has to produce more than one augmented document, e.g., two augmented documents for the augmentation factor 2. Naively implementing this by re-running the augmentation may lead to identical augmented documents, especially for deterministic techniques. Were possible, we changed existing techniques in such a way, that they return a list of augmented documents that are different from each other.

*4.5 Finding Optimal Configurations*

Each of the data augmentation techniques we selected can potentially be adjusted by several parameters, which control how augmented samples are synthesized. A typical example for such a parameter is the number of inserted tokens. Increasing this number would result in a sample, which is more perturbed compared to a sample where fewer tokens are inserted. We consider optimally choosing such parameters for a given technique a *hyper-parameter optimization* problem. Hyper-parameter optimization is defined as finding a configuration of parameters so that a given objective (metric to optimize) is minimal or rather maximal, depending on the case. Here, we want to maximize the performance gain that the application of a data augmentation technique has. To that end we run a 5-fold cross-validation of the extraction step (MD, RE) with the original, unaugmented data. We then select a configuration for the given technique and run the same 5-fold cross-validation, but augment the training data of each fold with the data augmentation technique. We define the difference between the scores of these two models on the (unaugmented) test dataset as the *performance gain* and use it as maximization objective for our hyper-parameter optimization. Each data augmentation technique is optimized in 25 runs (*trials*) using Optuna [4] and a Tree-Structured Parzen Estimator for selecting parameter values [9]. We depict this process in Figure 5.

We use the best parameter configurations of a given data augmentation technique as its default configuration in later sections and experiments. You can find values for the parameters in each augmentation technique in Appendix A.

## 5 Data Augmentation Effects

In this section we will describe the classes of data augmentations. Tab. 1 shows a compact overview of all data

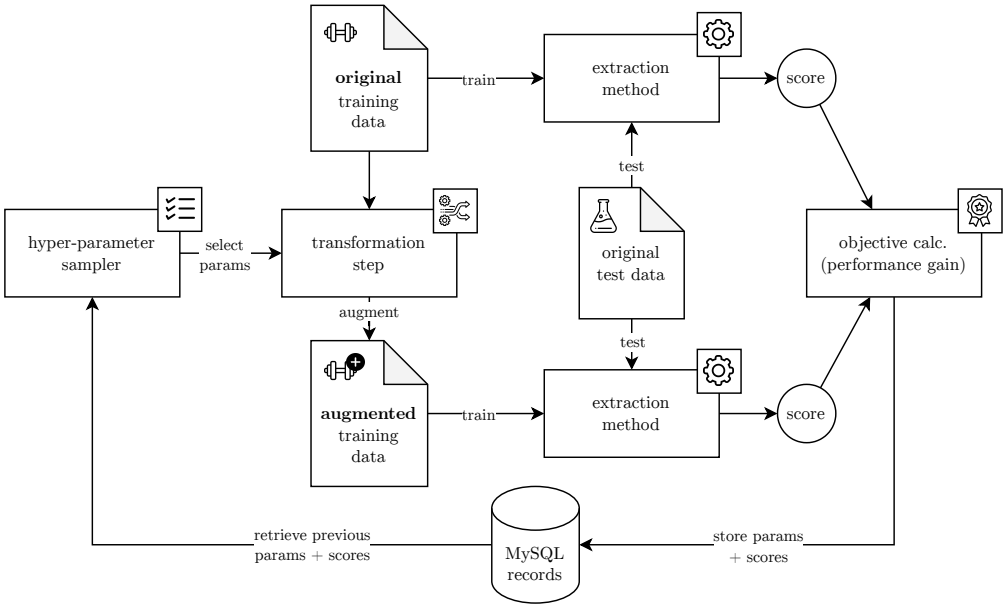



Fig. 5: Choosing optimal configurations for data augmentation techniques.

augmentation techniques. It also summarizes the information found in this section, which is structured as follows. First, we start by analyzing the process descriptions themselves, i.e., how the surface form text is changed by select data augmentation techniques in Sect. 5.1. We derive common classes of errors from this analysis and discuss how these errors impact the quality of synthesized data in Sect. 5.2. Next, to generalize the analysis of how augmentation changes the text, and more importantly, meaning of process descriptions, we visualize the text of whole process descriptions, sentences, and process entity mentions as scatter plots in Sect. 5.3. Finally, we discuss three more characteristics of textual process descriptions, linguistic variability, vocabulary size, and relation direction, in Sect. 5.4.

### *5.1 Surface Form Changes*

In this section, we will discuss how data augmentation changes the wording (surface form) of process descriptions. We will also discuss how these changes affect the annotations (labels) of data, such as mentions and relations. Throughout this section, we will show examples of augmented process descriptions base on the running example shown in Fig. 6. To improve clarity, we omitted all relations with the exception of the *Flow* relations, which we will use in some augmented examples.

**Rephrasing Augmentations.** Consider, for example, the technique Synonym Substitution (B.101) as representative for augmentations that rephrase text segments in process descriptions, with results as shown in Fig. 7.

Synonym Substitution changes the form of verbs heavily. While it properly selects synonyms, it fails to inflect

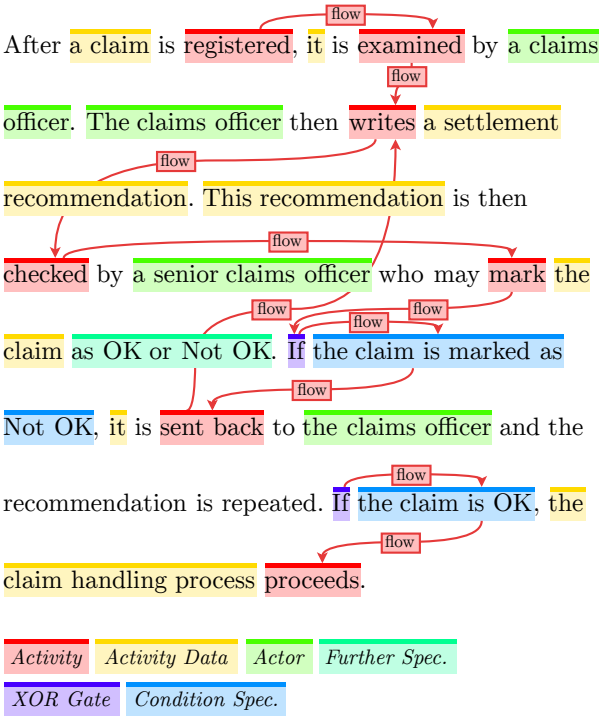



Fig. 6: Original surface form of document *doc-3.3* of the PET dataset.

them according to the original verb. Synonym substitution also fails to take context into account, and as such uses synonyms that are contextually wrong, e.g., replacing *settlement* with *colony* in *a settlement recom-*

| Technique | Id | Description | Category | Errors |
|---|---|---|---|---|
| Adjectives Antonyms Switch | B.3 | use antonyms of adjectives | Noise | SEM |
| AntonymsSubstitute (Double Negation) | B.5 | substitute even number of words with antonyms | Rephrasing | SEM |
| Auxiliary Negation Removal | B.6 | remove negated auxiliaries | Noise | SEM |
| BackTranslation | B.8 | translate to German, then back to English | Rephrasing | SEM, SYN, PUN |
| Concatenate Two Random Sentences | B.24 | remove PUN between sentences | Repeating | PUN |
| Contextual Meaning Perturbation | B.26 | replace words with use of pretrained language model | Rephrasing | SYN, SEM, GL |
| Contractions and Expansions Perturbation | B.27 | Contract phrases where common contractions exist, e.g., *"I am"* to *"I'm"* and vice versa | Rephrasing | — |
| English Mention Replacement for NER | B.39 | replace mention with one of the same type in document | Repeating | SYN, SEM |
| Filler Word Augmentation | B.40 | introduce "uhm", "I think", ... | Noise | GL |
| Lost in Translation | B.58 | repeatedly translate text segment into other languages and finally translate it back | Rephrasing | SYN, SEM, PUN, GL |
| Multilingual Back Translation | B.62 | see B.8, language is parameter | Rephrasing | SEM, SYN, PUN |
| Random Word Deletion | B.79 | delete random words | Noise | AN, SYN, SEM |
| Replace Abbreviations and Acronyms | B.82 | replace acronyms with full length expression and v.v. | Rephrasing | SEM |
| Hypernym Replacement | B.86 | replace token with its hypernym (super term), e.g., Mountain Bike with vehicle | Rephrasing | SEM |
| Hyponym Replacement | B.86 | replace token with its hyponym (super term), e.g., Mountain Bike with Bicycle | Rephrasing | SEM |
| Sentence Reordering | B.88 | reorder sentences | Reordering | SEM |
| Shuffle Within Segments | B.90 | shuffle tokens in mentions | Reordering | SYN, SEM, PUN |
| Synonym Insertion | B.100 | insert synonym before word | Noise | SYN, SEM |
| Synonym Substitution | B.101 | substitute word with synonym | Rephrasing | SEM |
| Subsequence Substitution for Sequence Tagging | B.103 | replace sequence with another sequence with same POS tags | Repeating | SEM |
| Transformer Fill | B.106 | replace tokens using language model | Rephrasing | SYN, SEM |
| Random Insert | | insert random tokens | Noise | SYN, SEM, GL, PUN |
| Random Swap | | swap position of tokens | Reordering | SYN, SEM, GL, PUN |
| Large Language Model Rephrasing | | use an LLM to rephrase text segments, while considering the entire surrounding process description as context | Rephrasing | SYN |
| Inverse Type Frequency Oversampling | | oversample documents containing rare mention / relation types | Repeating | — |
| Uniform Oversampling | | uniformly oversample documents | Repeating | — |

Table 1: Overview of augmentation techniques considered in this article. Column *Id* refers to identifier used in [12], column *category* is one of the categories defined in Sect. 4.3, while column *Errors* references one of the common error classes defined in Sect. 5.2, i.e., Semantics (SEM), Syntax (SYN), Punctuation (PUN), violations of Annotation Guidelines (GL), and deleted Annotations (AN).

After a title be registered, it is examined by a claims officer. The claims policeman then writes a settlement recommendation. This recommendation be then checked by a senior claims officer who may mark the claim as all right or Not OK. If the claim embody marked as Not OK, it is sent back to the title officer and the recommendation embody repeated. If the title is OK, the title handling process proceed.

Fig. 7: Effects of synonym substitution on surface form. Relations are unchanged by this augmentation technique.

*mendation*, when settlement is used in a financial context. These problems can be subsumed as breaking both syntax, as well as semantics to some degree. Rephrasing augmentations, that only replace a single token at a time did not break annotations in our experiments, i.e., they did not invalidate training targets of the dataset.

**Reordering Augmentations.** For reordering augmentations we use the Sentence Reordering (B.88) technique as an example, shown in Fig. 8.

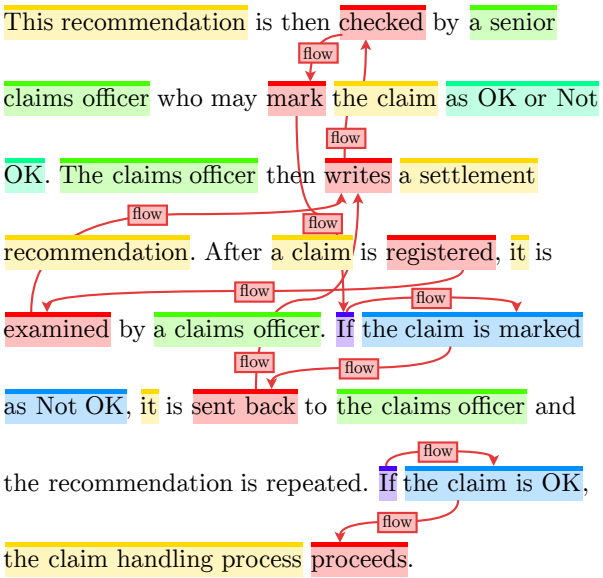



Fig. 8: Effects of reordering sentences on the direction of relations. Note that many relations are now wrong, as the text actually describes the order of execution differently.

This augmentation does not change the surface form of mentions, but is one of the few augmentations, that changes the direction of relations. Depending on the role of a relation, this augmentation is problematic for process information data. Take, for example, the *flow* between *mark* and *writes* in the original data shown in Fig. 6 and the augmented one in Fig. 8. Reordering sentences leads to a different flow when following the description in the text, but the *flow* relation is not updated to reflect that.

Other augmentations in this class change the surface form more dramatically, such as the *Shuffle Within Segements* augmentation, which splits the process descriptions into segments based on mention types (or lack thereof), and shuffles the tokens of randomly sampled segments. Fig. 9 shows this, where for example the phrase *a recommendation settlement* becomes *recommendation a settlement*.

After claim a is registered, it is examined by officer a claims. The claims officer then writes recommendation a settlement. recommendation This is then checked by a claims officer senior who may mark the claim as OK or OK Not. If the claim is marked OK Not as, it is sent back to the claims officer and the recommendation repeated is. If the claim is OK, the claim handling process proceeds.

Fig. 9: Effects of shuffling tokens within segments. This augmentation does not affect the direction of relations.

Note, that this class of augmentation breaks syntax regularly and may even border on breaking semantics, e.g., when a phrase like *as OK or Not OK* becomes *as OK or OK Not*, like in Fig. 9. Since the punctuation is also subject to shuffling, this augmentation techniques in this class sometimes produce invalid punctuation.

**Repeating Augmentations.** Augmentations that repeat entire documents, such as the oversampling strategies *Inverse Mention Frequency Sampler*, or *Uniform Repeat*, do not change the surface form of documents. Similarly, the *Merge Documents* augmentation strategy merges two random documents to produce a new one, which technically changes the surface form, but not in any conceptually interesting way.

For this reason, we will focus on augmentations that insert and replace parts of a document with repeated phrases from other documents, such as "*English Mention Replacement for NER*". Fig. 10 shows an example for this augmentation technique. Since the replacements are sampled randomly, this technique and others like it tend to break both syntax and semantics of process descriptions. Still, since it respects the type of the replaced entity mention, it is potentially useful for the mention detection and relation extraction tasks, as we show in our experiments in Sect. 6.

After a claim is registered, it is registered by a claims officer. The claims officer then writes a settlement recommendation. This recommendation is then checked by a senior claims officer who may reviews the claim to begin preparing the food. If the claim is marked as Not OK, it is sent back to the claims officer and the recommendation is repeated. If the claim is OK, the claim handling process proceeds.

Fig. 10: Effects of replacing entity mentions in the process description with those of the same type, e.g., *examined (activity)* with *registered (activity)* from the same document, or *as OK or Not OK (further specification)* with *to begin preparing the food (further specification)*, from a different document. This augmentation does not affect the direction of relations.

**Augmentations Adding Noise.** Augmentations that add random tokens into the text, can help to make models more robust [12]. These augmentation techniques are very likely to break both semantics and syntax of textual data. Still, they can be useful to simulate noisy input, which becomes increasingly useful, e.g., in the environment of chat bots, especially when the input is a transcript of human speech.

Figure 11 shows the effects of inserting so called speaker phrases into the process description. Speaker phrases are utterances of uncertainty, filler phrases, or other exclamations that do not contain any information, such as "*err*", or "*uhm*".

After I guess a claim is registered, it is I feel examined by a claims officer. The I think claims officer then writes a settlement recommendation. This recommendation is then checked by a senior claims officer who may mark that is the claim I feel as OK or Not OK. If the claim is marked as Not OK, it is sent I feel back to the claims officer and the recommendation is repeated. If the claim is OK, the claim handling process proceeds.

Fig. 11: Effects of inserting filler words, simulating uncertainty in a speaker.

### *5.2 Common Errors*

We identified five types of errors, reoccurring in many of the data augmentation techniques. In this section we present these errors and discuss their implications.

**Violating Annotation Guidelines.** When inserting tokens into the process description, e.g., filler words (Fig. 11), or tokens sampled from the remaining dataset (Fig. 10), the boundaries of mentions may be expanded. This is the case, when a token is inserted in the middle or directly after the span of tokens making up a mention. For example, inserting "*I think*" into the second sentence in Fig. 11 expands the mention "*The claims officer*". A better way of annotation would be using non-continuous spans of text (only *The* and *claims officer*), but the annotation guidelines of PET are not designed for this, and instead would ask annotators to only annotate "*claims officer*". When we automatically augment data, we can not decide which parts of a mention to keep, and which to discard, should an inserted token bisect a mention.

**Removing Annotations.** Randomly deleting tokens may remove mentions completely, if its only token is deleted. This has a cascading effect, as we also have to delete relations that use that mention as an argument. Compare, for example, the text that results after we ran the *Random Deletion* augmentation technique, shown in Fig. 12, with the original shown in Fig. 6. Many *activity* mentions are now missing, and as a result the corresponding *flow* relations as well.

After a claim, is examined by a claims officer. The claims officer then a settlement recommendation. This is then checked by a claims officer who may mark the claim as OK or Not OK If the claim is marked as Not , it is sent the officer and the recommendation is If the claim is OK, the claim handling process proceeds.

Fig. 12: Random deletion augmentation deleting many tokens, mentions and relations.

**Breaking Syntax.** It can be argued that breaking syntax to some degree can be useful for improving the robustness of information extraction approaches. This becomes clear, when a human reads the sentence "*After a claim registered is, [...]*". The intention can still be transported, even though there are grammatical inaccuracies. Yet, this intuition can also show, how syntax can be broken beyond a point, where compensation is possible. Randomly ordering the tokens from the previous example leads to "*Is after claim a registered, [...]*", which becomes hard to understand, or even ambiguous (is the claim registered after something, or is something happening after a claim is registered). Reordering is not the only operation that broke syntax in our experiments, we also observed this when replacing or inserting tokens.

**Breaking Semantics.** Unlike syntax, linguistic semantics are harder for extraction approaches to exploit, as it requires modeling language [8]. In recent years, large pretrained language models have made this easier, but the

amount of data available for developing process information extraction approaches is impeding their use [33]. In turn, this means while breaking semantics appears egregious for humans, it is less important for the shallow learning approaches we use in our experiments. As a result, approaches that use large amounts of compute, to preserve semantics, are only marginally (if at all) better in our black box evaluation. Nonetheless, we argue that avoiding semantic-breaking errors will become more and more important, when larger models are used for process information extraction. Fig. 7 preserves syntax properly, but changes the semantics of "The claim officer" so much that it is even hard for human readers to recognize its other mentions (*claims policeman* and *title officer*).

**Invalid Punctuation.** When augmentations remove or insert tokens, they may remove or insert punctuation. The latter is especially the case for all back-translating augmentations, as they translate only small, incomplete fragments of text and often end the translated fragment with punctuation. Invalid punctuation is generally not an issue for state-of-the-art extraction methods, but can be a problem for the relation extraction method used in this article, as it uses the sentence index as a feature for predicting relations between arguments [33].

### *5.3 Variations in Meaning*

The aim of this section is to give a high level overview of the changes in meaning introduced by augmentation. To this end we embedded both the original and augmented documents, sentences, mentions, and relations using the Jina Embedding model [19]. This embedding model is based on a BERT architecture [11], i.e., a transformer model that can embed sequences of up to 8192 input tokens and is therefore very well suited to embed even long texts, such as process descriptions. Since relations are not pure text, and can not directly be embedded using a text embedding model, we formatted them as "*text of head mention –> text of tail mention*" and embedded the resulting string. The resulting embedding vectors have 768 dimensions, which can not be visualized directly. Instead, we use openTSNE [36], a dimension reducing transformation based on TSNE, which allows us to learn this transformation on the vectors of the original documents, and then apply the same transformation on the vectors of the augmented documents.

Plotting the result as a scatter plot lets us visualize how a given class of augmentation changes the meaning of entire process descriptions (documents), sentences, mentions, or relations. Fig. 17 in Appendix B shows such as scatter plot for the entire text of documents in the PET dataset. Augmentations that do not change the semantics, such as *reordering*, or *adding noise*, are hardly visible in these scatter plots, as their embeddings barely change. *Repeating* augmentations on the other hand change them enough, so that they cover previously empty space in the visualization. Similarly, rephrasing augmentations introduce small variations in the text, which are minute, but visible in our visualizations.

Visualizing how the content of sentences changes (Fig. 18 in Appendix B) results in similar patterns, *Reordering* and *Adding Noise* barely changes the vectors of sentences, i.e., their overall content and meaning. *Repeating* and *Rephrasing* augmentations produce more varied text. Visualizations of sentences already show a property that is intensified, when we visualize mention texts in Fig. 13 — since sentences are smaller, text variations have a larger effect on their overall embedding. This leads to reordering and noise augmentations being visible in the visualization, compared to visualizations on document level, where they are invisible.

Mentions are usually quite short, and only comprised of a few tokens. Similarly to sentences, even small alterations of their text have effects on the resulting embedding vectors and their position in the scatter plot. Still, rephrasing has the most pronounced effect on the text (see Fig. 13).

Embedding and visualizing relations shows similar results as sentences and mentions. Rephrasing shows the most pronounced effect on embeddings. Surprisingly, the relation embeddings of augmented documents are quite close to their originals, as shown in Fig. 14. We suspect this might be a limitation with our approach of embedding relations, and less so with the augmentations themselves. In future work one could try to embed relations differently, e.g., by including their type, or building natural language sentences from them, such as "*registered uses a claim*" from the *uses* relation *registered (Activity) → a claim (Activity Data)*.

### *5.4 Linguistic Variability, Mention Length, and Relation Direction*

To supplement the visualizations from Sect. 5.3, we define three additional characteristics of textual process descriptions that are changed by the data augmentation techniques we selected. These characteristics are **(1)** linguistic variability, i.e., the number of unique tokens (words) used in a process description, **(2)** the length of entity mentions, i.e., how many tokens make up a single entity mention on average, and **(3)** the direction of relations.

**(1)** Increased linguistic variability, i.e., augmented text uses a larger vocabulary to describe the same, or at least, a similar business process[2]. The most prominent examples for such techniques are the *Back-Translation* techniques. These use a large language model, e.g., BERT [11] to translate the process description to a different language and subsequently translate it back to the original

[2] The augmentation technique might change information in the text, which changes the process overall, e.g., by replacing original actors with new, artificial ones.

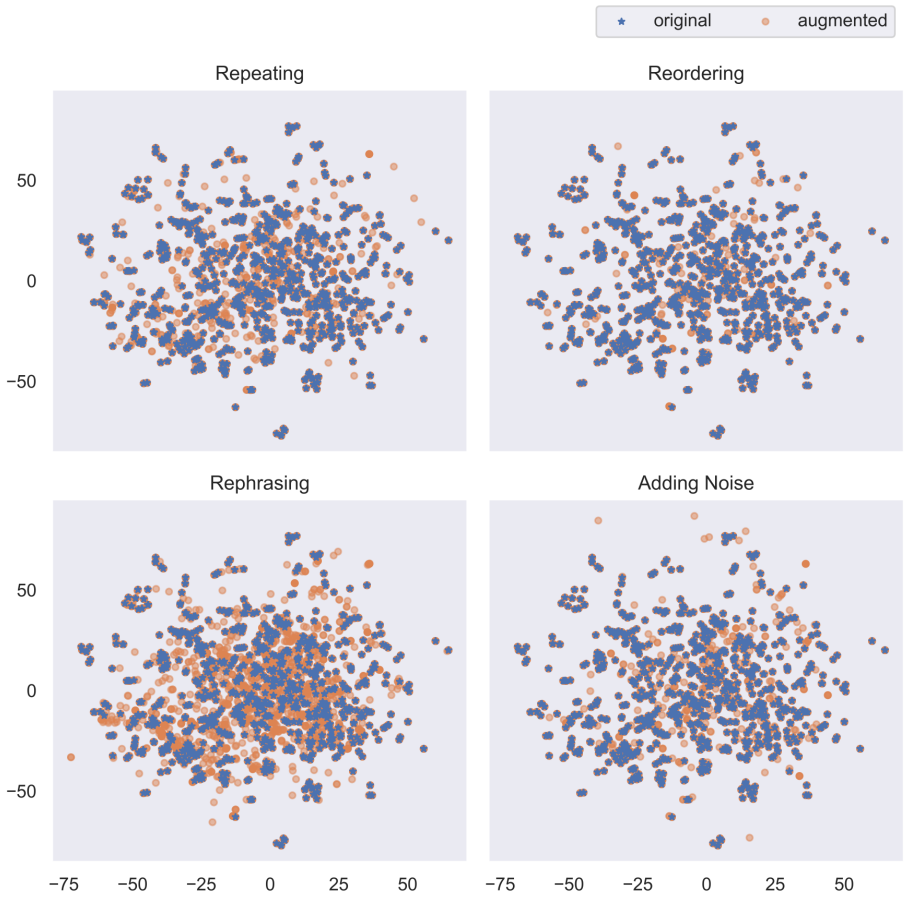



Fig. 13: Mentions

language – here English. Since data augmentation techniques must not alter the annotations of entities, we only translated spans of text, not the entire document at once. Take for example, the running example *After a claim is registered, it is examined.* Here four spans are annotated as entities – *a claim*, *registered*, *examined*, and *it*. Additionally there are three remaining spans, that do not correspond to entities: *After a*, *is*, *is*. By back-translating these seven spans separately, we obtain variation in their wording (*surface form*), but are still able to preserve annotations. Samples synthesized in this way are especially useful for making methods for the MD task generalize better and more robust.

**(2)** Variations in span length. Many spans of a given entity type, e.g., Actors are very uniform in length across examples. This is a result of several factors, but most apparent actors are often identified by their job title, e.g., *the clerk*, or the department, e.g., *the secretary office*. These titles and departments are very short phrases, and longer ones are abbreviated, reducing their length to two or less tokens, e.g., *the MPOO*. Even though their expanded form may not be known, expanding some of these spans to suitable phrases, e.g., *Manager, Post Office Operations*, creates samples with longer surface forms. This, in turn, may improve the robustness of the MD extractors, as well as the generalization capabilities of RE methods.

**(3)** Direction of relations between mentions. The order of appearance for mentions that form a relation, is very uniform in the current version of PET. This is especially apparent, when looking at the base-line extraction rules defined by the original authors of PET: Here the order of appearance of Activities and Actors is exploited, to form the *Actor Performer* and *Actor Recipient* relations [7]. These relations define the Actor, that performs an Activity, and the Actor, on which an Activity is performed. The Actor left of an Activity is assigned the former, while the Actor right of that Activity is assigned the latter. In this example order uniformity can lead to less robust models, as they rely on this and subsequently make wrong predictions given different linguistic constructs. Synthesizing samples with a different order may encourage models to consider linguistic fea-


Fig. 14: Relations

tures (context) rather than just the order of mentions in a sentence during prediction.

These characteristics are visualized in Figures 16 and 15. Figure 16 shows the "landscape" of data augmentation techniques evaluated in this paper. Three groups of techniques emerge. The first one is a group of techniques that only marginally increase the number of tokens in mentions, and keep the size of the vocabulary roughly the same. These techniques mainly change the context (i.e., the text that does not contain immediately process relevant information), or the structure of the text (i.e., modify punctuation, or change the order of tokens). Techniques in the second group do not modify the vocabulary, but have a significant impact on the number of tokens in a given mention. These augmentations can theoretically be useful for the robustness of MD extraction models, but only have a moderate impact in our experiments, using the PET dataset. We count *Random Insertion*, *Filler Word Augmentation*, but also *Random Word Deletion* towards this group, see Figure 2 for an example taken from the augmented data. The final group of techniques increases the size of the vocabulary, while keeping mention lengths roughly the same. These techniques are paraphrasing, aimed at preserving semantics and the structure of textual data. Techniques using WordNet to insert or substitute synonyms (*B.100*, *B.101*, as well as back translation methods (*B.62*, *B.26*) fall in this group.

## 6 Results

In this section we will discuss results for our experiments and answer the research questions from Sect. 4.1. Table 3 lists the differences of all data augmentation techniques compared to a run on un-augmented data. All differences are measured as the micro-averaged $F_1$ score. Concluding from our results, we find that both the MD and RE tasks significantly benefit from some of the data augmentation techniques we selected and tested. Data augmentation results in improvements of up to +5.7% (absolute percentage points) in mention detection performance, and up to +4.5% (absolute percentage points) in relation extraction performance.

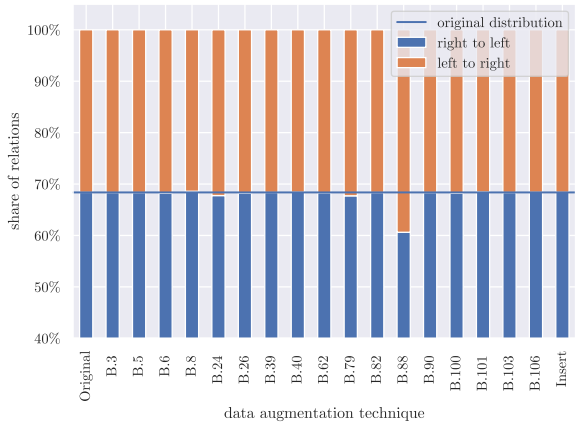



Fig. 15: Effects of techniques on relation direction. Oversampling techniques are omitted, as they can not change direction of relations.

**RQ1 — Simple data augmentation.**. With the exception of *Random Word Deletion*, simple data augmentation techniques, i.e., *Random Insert*, and *Random Swap*, are not useful for improving mention detection results, meaning they are worse than or equal to oversampling documents. We expand on the notion of usefulness later in this section, when answering research question **RQ3**. This observation is even clearer for the relation extraction task, where none of the simple data augmentation techniques are able to outperform oversampling. We therefore answer our leading research question **RQ1** with *No*. While it seems that simple data augmentation techniques improve the performance of both the MD and RE tasks, we attribute that improvement to the implicit oversampling instead of higher data quality.

**RQ2 — Complex data augmentation.** Looking at the remaining, more complex data augmentation techniques, only some of them outperform oversampling in both mention detection and relation extraction. For mention detection, we find that especially *Rephrasing* (e.g., *Antonyms Substitute (Double Negation)*) and *Repeating* (e.g., *Subsequence Substitution for Sequence Tagging*) techniques are most useful. Notably, techniques focused on acronyms, i.e., *Replace Abbreviations and Acronyms* and *Contractions and Expansions Perturbation* are worse than oversampling. This can be explained by their reliance on lists of possible acronyms, which are from many different domains, and not always applicable to the domain of a process description. For example, document *doc-10.6* of the pet dataset contains sentences like "*The MSPN sents a dismissal to the MSPO.*". These acronyms are undefined in the lists of the acronym-focused data augmentation techniques. In these cases they operate like *Uniform Oversampling*, explaining their poor performance. Many of the based on large language models, especially back translation techniques, like *Multilingual Back Translation*, which translate a sentence fragment twice, and rephrasing techniques based on LLMs, are very time-intensive, as Tab. 2 shows. Yet, improvements in relation extraction performance is not significant, when comparing them to more lightweight approaches, e.g., *Synonym Substitution*, which uses WordNet to rephrase text sequences, and runs several orders of magnitude faster. Juding from the observations made during our experiments, using these large language model based methods is not worth the increase in computing power and time. While the MD task can still benefit from all data augmentation techniques, it does so to a lesser extent when compared to the RE task. This indicates a model, that is already more stable, and generalizes better. Transformations that alter the amount of tokens in mentions, such as *Random Word Deletion*, *Synonym Insertion*, or *Subsequence Substitution for Sequence Tagging*, result in lesser improvements, compared to paraphrasing methods, such as *AntonymsSubstitute*, *BackTranslation*, or *Synonym Substitution*. Similar to the RE task, the MD task does not benefit significantly more from resource and time intensive, large language model based augmentation techniques for paraphrasing, compared to their simpler counterparts, answering our second research question **RQ2**. Based on our observations we do not recommend using complex data augmentation techniques with classical supervised approaches

| **Technique** | **Time** |
|---|---|
| Adjectives Antonyms Switch | 0.008s |
| Antonyms Substitute (Double Negation) | 1.374s |
| Auxiliary Negation Removal | 0.096s |
| BackTranslation | 13.93s |
| Concatenate Two Random Sentences | 0.009s |
| Contextual Meaning Perturbation | 6.040s |
| Contractions and Expansions Perturbation | 0.011s |
| English Mention Replacement for NER | 0.031s |
| Filler Word Augmentation | 0.078s |
| Hypernym Replacement | 2.514s |
| Hyponym Replacement | 3.406s |
| Inverse Type Frequency Oversampling | 0.011s |
| Large Language Model Rephrasing | 214.2s |
| Lost in Translation | 63.28s |
| Multilingual Back Translation | 62.53s |
| Random Insert | 0.012s |
| Random Swap | 0.002s |
| Random Word Deletion | 0.016s |
| Replace Abbreviations and Acronyms | 0.018s |
| Sentence Reordering | 0.004s |
| Shuffle Withing Segments | 0.046s |
| Synonym Insertion | 0.055s |
| Synonym Substitution | 0.057s |
| Subsequence Substitution for Sequence Tagging | 0.385s |
| Transformer Fill | 1.958s |
| Uniform Oversampling | 0.002s |

Table 2: Time it takes each augmentation technique to augment a single document ten times.

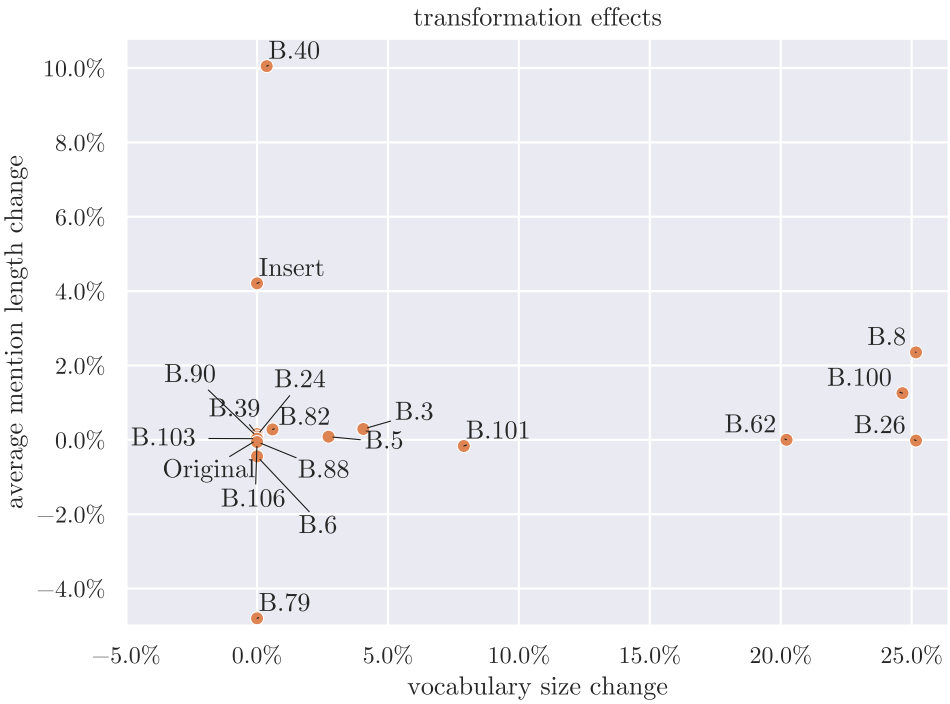



Fig. 16: Effects on vocabulary size and the average length of mentions in tokens. Oversampling techniques are omitted, as they neither change the vocabulary size, nor the length of mentions.

to process information extraction. If they are useful for more complex extraction models (e.g., deep learning), needs further research in future work.

**RQ3 — Oversampling.** Many of the data augmentation techniques we selected outperform the two oversampling strategies we investigated. For mention detection most of these techniques fall into the *Rephrasing* category, while *Adding Noise* does not seem to be useful. For the relation extraction task, a lot less data augmentation techniques are better than oversampling. We suspect this is due to a relation extraction model with suboptimal hyper-parameter configuration. Since we kept this configuration fixed throughout our experiments and followed previous work [7,33], oversampling may be compensating for a learning rate that is set too low. The experiment described in Sect. 4.5 could be expanded in future work, to consider the hyper-parameters of mention detection and relation extraction models, but was out of scope for this article.

**RQ4 — Characteristics.** Based on Sect. 5, which presents a deep dive into the effects of data augmentation techniques on the text of process descriptions, we see three major areas affected by different data augmentation techniques. **(1)** The embedding of entity mentions and relations between them is affected by *Rephrasing* and *Reordering* data augmentation techniques the most. This implies that the meaning of the corresponding process elements changes the most, if techniques from these categories are used. These changes sometimes also break the syntax and semantics of process descriptions, which does not appear to be a problem for the models used in our experiments, as the improvements are still outperforming oversampling strategies. **(2)** *Reordering* techniques are the only category of data augmentation that change the directions of relations. Fig. 15 visualizes this fact, with technique *B.88* (*Sentence Reordering*) clearly standing out. While reordering sentences often breaks the semantics of the descriptions of the order of activity execution ("*Register claim. Then examine it*" vs. "*Then examine it. Register claim.*"), this does not seem to be an issue, as this data augmentation technique is one of the best ones for improving the performance of relation extraction. **(3)** The last characteristic changed by our selection of data augmentation techniques, is the linguistic variability, i.e., the number of unique tokens (words) used to describe process elements. We observed an increase of up to 25% (Fig. 16). We can not directly conclude from our experiments, if this is a beneficial effect or not, since the data augmentation techniques that increase the linguistic variability the most, are the ones based on large language models, which do not have clear advantages compared to other techniques. Quantifying the effects of increased vocabulary in process descriptions would require further research, especially using larger deep learn-

| Technique | MD | RE |
|---|---|---|
| Unaugmented | 69.5% | 75.9% |
| Adjectives Antonyms Switch | +1.8% | +3.3% |
| AntonymsSubstitute (Double Negation) | +5.7% | +2.4% |
| Auxiliary Negation Removal | +2.2% | +3.5% |
| BackTranslation | +2.9% | +3.6% |
| Concatenate Two Random Sentences | +2.3% | +4.5% |
| Contextual Meaning Perturbation | +2.5% | +4.1% |
| Contractions and Expansions Perturbation | +0.6% | +1.9% |
| English Mention Replacement for NER | +2.0% | +3.8% |
| Filler Word Augmentation | +1.6% | +3.3% |
| Hypernym Replacement | +3.0% | +3.8% |
| Hyponym Replacement | +2.2% | +4.0% |
| Inverse Type Frequency Oversampling | +1.4% | +3.3% |
| Large Language Model Rephrasing | +3.4% | +3.1% |
| Lost in Translation | +1.0% | +1.3% |
| Multilingual Back Translation | +1.7% | +4.1% |
| Random Insert | +1.3% | +3.3% |
| Random Swap | +2.2% | +2.6% |
| Random Word Deletion | +3.0% | +2.7% |
| Replace Abbreviations and Acronyms | +1.4% | +2.9% |
| Sentence Reordering | +2.4% | +4.0% |
| Shuffle Within Segments | +2.1% | +4.1% |
| Synonym Insertion | +1.6% | +4.4% |
| Synonym Substitution | +3.3% | +3.2% |
| Subsequence Substitution for Sequence Tagging | +5.3% | +3.6% |
| Transformer Fill | +2.1% | +3.3% |
| Uniform Oversampling | +2.2% | +3.5% |

Table 3: Detailed results for all augmentation techniques, for both the mention detection (MD) and relation extraction (RE) task. All results are the averages of a 5-fold cross validation on the entire PET dataset, reported as the absolute increase in $F_1$ measure compared to a run on unaugmented training data.

ing models, and not just classical machine learning methods.

## 7 Conclusion and Future Work

This section will conclude this article by summarizing its key insights (Sect. 7.1), discuss limitations (Sect.7.2), and provide directions for future work (Sect.7.3).

### *7.1 Summary*

In this article we evaluated established data augmentation techniques for use in the mention detection and relation extraction steps of extracting process relevant information from natural language texts for use in the automated generation of business process models. To this end we selected a total of 21 suitable methods from the NLAugmenter framework [12]. We complemented this selection with two oversampling techniques and two simple augmentation techniques, to answer how much data augmentation can improve the quality of mentions and relations extracted from textual business process descriptions. We find that while not all data augmentation techniques are useful, i.e., improve performance more than repeating unaltered data (oversampling), many are and can improve the performance of detecting process relevant entity mentions by up to +5.7% (absolute percentage points), and up to +4.5% (absolute percentage points) in relation extraction performance.

Additionally, we discuss several characteristics of the data that are changed by the investigated data augmentation techniques, including the change in vocabulary size of process descriptions, the direction of relations between process elements, and the change of meaning in the descriptions of processes. We complement this analysis with an extensive exploration of the surface form changes in process description texts and derive four common error classes, which are introduced by data augmentation techniques.

### *7.2 Limitations*

We judged the usefulness of data augmentation techniques using only one extraction pipeline based on classical (shallow) machine learning models. This mainly originates from the lack of supervised machine learning methods for the process information extraction methods. While there is previous work using deep learning models for extracting process information, the authors only in-

vestigated entity mention detection, not relation extraction [2]. It would also be interesting to investigate, if data augmentation can solve the problem of small datasets preventing the training of general purpose information extraction methods from natural language process [3]. Though, for this article this investigation was considered out of scope.

Furthermore, our exploration of the changes in surface form of process descriptions is highly qualitative and focused on finding noteworthy occurrences of errors in the augmented data. Deeper analysis on a linguistic level could lead to more insights where data augmentation techniques introduce unwanted and unexpected perturbations, potentially detrimental to the process information extraction task.

Finally, we did not investigate the effects of data augmentation on models trained with augmented data. While we did do a black-box evaluation to judge how much certain techniques improve the extraction quality, this is merely descriptive, and does not explain these improvements. In future work we plan to analyze models trained with augmented data in terms of their robustness, resilience against adversarial examples, and generalization capabilities, compared to models trained with unaugmented data only.

### *7.3 Future Work*

In this section we describe several avenues of further research. First, some future work is already described in our discussion of limitations (Sect. 7.2), including testing augmented data for training deep learning models, analyzing augmented data on a linguistic level, and investigating why augmented data changes an extraction model's capabilities.

Additionally, we want to analyze how targeted data augmentation can be used to improve extraction of certain types of mentions or relations, tackling the problem of data imbalance. We also want to explore adaptive data augmentation, where samples are selected for augmentation by their value for model training, e.g., measured by the number of wrong predictions the cause during evaluation.

## References


1. Ackermann, L., Käppel, M., Marcus, L., Moder, L., Dunzer, S., Hornsteiner, M., Liessmann, A., Zisgen, Y., Empl, P., Herm, L.-V., et al. Recent advances in data-driven business process management. *arXiv preprint arXiv:2406.01786* (2024).
2. Ackermann, L., Neuberger, J., and Jablonski, S. Data-driven annotation of textual process descriptions based on formal meaning representations. In *CAiSE* (2021).
3. Ackermann, L., Neuberger, J., Käppel, M., and Jablonski, S. Bridging research fields: An empirical study on joint, neural relation extraction techniques. In *CAiSE* (2023).
4. Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (2019), pp. 2623–2631.
5. Bellan, P., Dragoni, M., and Ghidini, C. Assisted process knowledge graph building using pre-trained language models. In *Proceedings of AIxIA 2022 - Advances in Artificial Intelligence* (2022).
6. Bellan, P., Dragoni, M., and Ghidini, C. Extracting business process entities and relations from text using pre-trained language models and in-context learning. In *EDOC* (2022).
7. Bellan, P., Ghidini, C., Dragoni, M., Ponzetto, S. P., and van der Aa, H. Process extraction from natural language text: the PET dataset and annotation guidelines. In *NL4AI* (2022).
8. Bellegarda, J. R. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE 88*, 8 (2000), 1279–1296.
9. Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems 24* (2011).
10. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems 26* (2013).
11. Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
12. Dhole, K. D., Gangal, V., Gehrmann, S., Gupta, A., Li, Z., Mahamood, S., Mahendiran, A., Mille, S., Shrivastava, A., Tan, S., et al. Nl-augmenter: A framework for task-sensitive natural language augmentation. *arXiv preprint arXiv:2112.02721* (2021).
13. Eldin, A. N., Assy, N., Anesini, O., Dalmas, B., and Gaaloul, W. A decomposed hybrid approach to business process modeling with llms.
14. Erdengasileng, A., Han, Q., Zhao, T., Tian, S., Sui, X., Li, K., Wang, W., Wang, J., Hu, T., Pan, F., et al. Pre-trained models, data augmentation, and ensemble learning for biomedical information extraction and document classification. *Database 2022* (2022), baac066.
15. Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., and Hovy, E. A survey of data augmentation approaches for NLP.
16. Ferreira., R. C. B., Thom., L. H., and Fantinato., M. A semi-automatic approach to identify business process elements in natural language texts. In *ICEIS* (2017).
17. Friedrich, F., Mendling, J., and Puhlmann, F. Process model generation from natural language text. In *CAiSE* (2011).
18. Grohs, M., Abb, L., Elsayed, N., and Rehse, J.-R. Large language models can accomplish business process management tasks. In *International Conference on Business Process Management* (2023), Springer, pp. 453–465.

19. Günther, M., Ong, J., Mohr, I., Abdessalem, A., Abel, T., Akram, M. K., Guzman, S., Mastrapas, G., Sturua, S., Wang, B., Werk, M., Wang, N., and Xiao, H. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents, 2023.
20. Jiang, Z., Han, J., Sisman, B., and Dong, X. L. Cori: Collective relation integration with data augmentation for open information extraction. *arXiv preprint arXiv:2106.00793* (2021).
21. Kampik, T., Warmuth, C., Rebmann, A., Agam, R., Egger, L. N., Gerber, A., Hoffart, J., Kolk, J., Herzig, P., Decker, G., et al. Large process models: Business process management in the age of generative ai. *arXiv preprint arXiv:2309.00900* (2023).
22. Käppel, M., and Jablonski, S. Model-agnostic event log augmentation for predictive process monitoring. In *International Conference on Advanced Information Systems Engineering* (2023), Springer, pp. 381–397.
23. Käppel, M., Schönig, S., and Jablonski, S. Leveraging small sample learning for business process management. *Information and Software Technology* (2021).
24. Klievtsova, N., Benzin, J.-V., Kampik, T., Mangler, J., and Rinderle-Ma, S. Conversational process modelling: state of the art, applications, and implications in practice. In *International Conference on Business Process Management* (2023), Springer, pp. 319–336.
25. Köpke, J., and Safan, A. Introducing the bpmn-chatbot for efficient llm-based process modeling.
26. Kourani, H., Berti, A., Schuster, D., and van der Aalst, W. M. Process modeling with large language models. *arXiv preprint arXiv:2403.07541* (2024).
27. Kourani, H., Berti, A., Schuster, D., and van der Aalst, W. M. Promoai: Process modeling with generative ai. *arXiv preprint arXiv:2403.04327* (2024).
28. Liu, J., Chen, Y., and Xu, J. Machine reading comprehension as data augmentation: A case study on implicit event argument extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021), pp. 2716–2725.
29. López, H. A., Strømsted, R., Niyodusenga, J.-M., and Marquard, M. Declarative process discovery: Linking process and textual views. In *International Conference on Advanced Information Systems Engineering* (2021).
30. López-Acosta, H.-A., Hildebrandt, T., Debois, S., and Marquard, M. The process highlighter: From texts to declarative processes and back. In *CEUR Workshop Proceedings* (2018), CEUR Workshop Proceedings, pp. 66–70.
31. Miller, G. A. Wordnet: a lexical database for english. *Communications of the ACM 38*, 11 (1995), 39–41.
32. Mohammed, R., Rawashdeh, J., and Abdullah, M. Machine learning with oversampling and undersampling techniques: overview study and experimental results. In *2020 11th international conference on information and communication systems (ICICS)* (2020), IEEE, pp. 243–248.
33. Neuberger, J., Ackermann, L., and Jablonski, S. Beyond rule-based named entity recognition and relation extraction for process model generation from natural language text. In *CoopIS* (2023).
34. Neuberger, J., Ackermann, L., van der Aa, H., and Jablonski, S. A universal prompting strategy for extracting process model information from natural language text using large language models. In *International Conference on Conceptual Modeling* (2024), Springer, pp. 38–55.
35. Neuberger, J., Doll, L., Engelmann, B., Ackermann, L., and Jablonski, S. Leveraging data augmentation for process information extraction. In *International Conference on Business Process Modeling, Development and Support* (2024), Springer, pp. 57–70.
36. Poličar, P. G., Stražar, M., and Zupan, B. opentsne: A modular python library for t-sne dimensionality reduction and embedding. *Journal of Statistical Software 109*, 3 (2024), 1–30.
37. Quishpi, L., Carmona, J., and Padró, L. Extracting annotations from textual descriptions of processes. In *BPM 2020* (2020).
38. Radford, A. Improving language understanding by generative pre-training.
39. Shorten, C., and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of big data 6*, 1 (2019), 1–48.
40. Shorten, C., Khoshgoftaar, T. M., and Furht, B. Text data augmentation for deep learning. *Journal of big Data* (2021).
41. van der Aa, H., Di Ciccio, C., Leopold, H., and Reijers, H. A. Extracting declarative process models from natural language. In *CAiSE* (2019).
42. Yao, Y., Ye, D., Li, P., Han, X., Lin, Y., Liu, Z., Liu, Z., Huang, L., Zhou, J., and Sun, M. Docred: A large-scale document-level relation extraction dataset. *arXiv preprint arXiv:1906.06127* (2019).
43. Zoran, D., and Weiss, Y. Scale invariance and noise in natural images. In *2009 IEEE 12th International Conference on Computer Vision* (2009), IEEE, pp. 2209–2216.

## Appendix A Parameters of Data Augmentation Techniques

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 4.70 | 3.60 |
| replace probability | Probability to replace an adjective with its antonym | 93.1% | 89.8% |

Table 4: Parameters for augmentation technique *Adjective Antonyms Switch* (*B.3*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 0.98 | 1.52 |

Table 5: Parameters for augmentation technique *Antonyms Substitute (Double Negation)* (*B.5*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 1.56 | 4.77 |

Table 6: Parameters for augmentation technique *Auxiliary Negation Removal* (*B.6*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 3.13 | 5.63 |
| replace probability | Probability to replace a text segment with its back-translation | 98.5% | 63.2% |
| segment length | Minimum length of segment to be considered for translation | 3 | 3 |

Table 7: Parameters for augmentation technique *Back Translation* (*B.8*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 6.52 | 9.80 |

Table 8: Parameters for augmentation technique *Concatenate Two Random Sentences* (*B.24*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 6.52 | 7.94 |
| replace probability | Probability to replace a text segment with its back-translation | 31.4% | 3.4% |
| part of speech tag groups | Groups of part of speech tags to consider for back-translation, e.g., nouns, verbs, adjectives, ... | Nouns Adj. Adv. | Nouns Adj. Adv. |

Table 9: Parameters for augmentation technique *Contextual Meaning Perturbation* (*B.26*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 0.47 | 2.24 |
| replace probability | Probability to replace an abbreviation with its long form and vice versa | 97.3% | 91.9% |

Table 10: Parameters for augmentation technique *Contractions and Expansions Perturbation* (*B.27*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 8.81 | 6.51 |
| replace probability | Probability to replace an entity mention with one from another process description | 10.7% | 73.9% |

Table 11: Parameters for augmentation technique *English Mention Replacement for NER* (*B.39*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 5.13 | 7.63 |
| insert probability | Probability to insert any of the filler phrases | 6.3% | 9.1% |
| insert filler phrases | Should phrases like *err*, *uhm*, *ahh* be inserted? | No | Yes |
| insert speaker phrases | Should phrases like *I think*, *I mean*, *I would say* be inserted? | Yes | Yes |
| insert uncertainty phrases | Should phrases like *maybe*, *probably*, *possibly* be inserted? | No | No |

Table 12: Parameters for augmentation technique *Filler Word Augmentation* (*B.40*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 3.47 | 4.24 |
| replace probability | Probability to replace an entity mention with its back-translation | 22.3% | 18.9% |
| pivot language | Language to translate the segment to and back | lo | ja |

Table 13: Parameters for augmentation technique *Multilingual Back Translation* (*B.62*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 3.58 | 6.72 |
| delete probability | Probability to delete a token | 12.2% | 0.3% |

Table 14: Parameters for augmentation technique *Random Word Deletion* (*B.79*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 1.41 | 0.09 |
| replace probability | Probability to replace an abbreviation with its long form and vice versa | 88.6% | 99.6% |

Table 15: Parameters for augmentation technique *Replace Abbreviations and Acronyms* (*B.82*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 4.30 | 6.01 |
| replace probability | Probability to replace a word with its hypernym | 38.7% | 98.0% |

Table 16: Parameters for augmentation technique *Hypernym Replacement* (*B.86*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 3.20 | 6.87 |
| replace probability | Probability to replace a word with its hyponym | 47.4% | 91.7% |

Table 17: Parameters for augmentation technique *Hyponym Replacement* (*B.86*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 2.67 | 4.18 |

Table 18: Parameters for augmentation technique *Sentence Reordering* (*B.88*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 7.90 | 4.23 |
| replace probability | Probability to replace a segment with its shuffled version | 46.2% | 7.2% |

Table 19: Parameters for augmentation technique *Shuffle Within Segments* (*B.90*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 7.90 | 6.23 |
| insert probability | Probability to insert the synonym of a word before that word | 14.7% | 25.7% |

Table 20: Parameters for augmentation technique *Synonym Insertion* (*B.100*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 2.14 | 4.54 |
| replace probability | Probability to replace a word with its synonym | 30.2% | 10.6% |

Table 21: Parameters for augmentation technique *Synonym Substitution* (*B.101*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 5.04 | 6.67 |
| replace probability | Probability to replace a segment with a segment with identical part-of-speech tags from a different document | 30.2% | 40.7% |
| min length | Minimum length of segment to replace | 1 | 4 |
| max length | Maximum length of segment to replace | 5 | 8 |

Table 22: Parameters for augmentation technique *Subsequence Substitution for Sequence Tagging* (*B.103*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 3.37 | 5.56 |
| replace probability | Probability to mask a word and fill it using a transformer model (BERT) | 30.4% | 38.4% |
| part of speech tags | Groups of part-of-speech tags that are considered for masking and filling | Nouns | Nouns, Adj. |

Table 23: Parameters for augmentation technique *Transformer Fill* (*B.101*). Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 6.04 | 20.3 |
| insert probability | Probability to insert a random token from a different document at each original token | 2.4% | 0.6% |

Table 24: Parameters for augmentation technique *Random Insert*. Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 2.81 | 2.03 |
| swap probability | Probability to swap any two tokens in the process description | 0.7% | 1.5% |

Table 25: Parameters for augmentation technique *Random Swap*. Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| augmentation rate | Number of augmented process descriptions created per one original | 4.51 | 6.19 |
| replace probability | Probability to replace a text segment with a rephrased version | 27.7% | 30.6% |

Table 26: Parameters for augmentation technique *Large Language Model Rephrasing*. Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| oversampling rate | Number of uniform randomly sampled original documents | 1.46 | 5.08 |

Table 27: Parameters for augmentation technique *Uniform Oversampling*. Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

| Name | Description | MD | RE |
|---|---|---|---|
| oversampling rate | Number of randomly sampled original documents, weighed by the rarity of contained types | 4.09 | 5.05 |

Table 28: Parameters for augmentation technique *Inverse Type Oversampling*. Columns *MD* and *RE* refer to the parameter value for the mention detection (MD) and relation extraction task (RE).

## Appendix B Additional Figures for Changes in Meaning

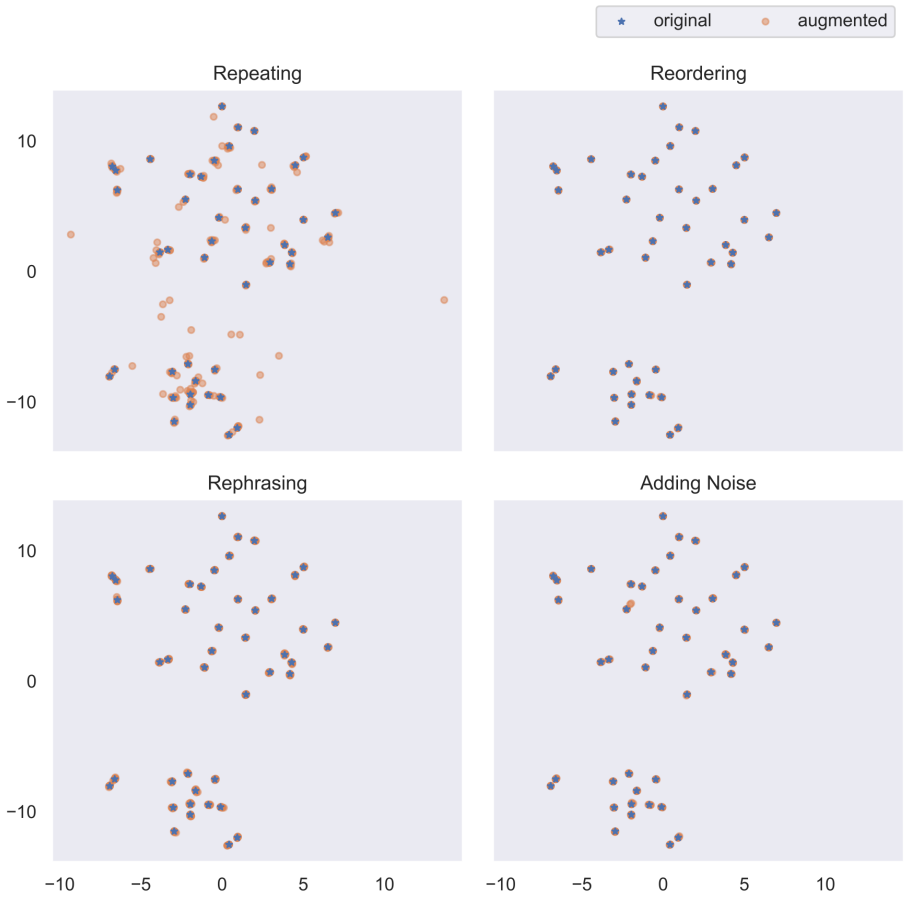



Fig. 17: Scatter plot of the embedding vectors for the texts of original and augmented documents.

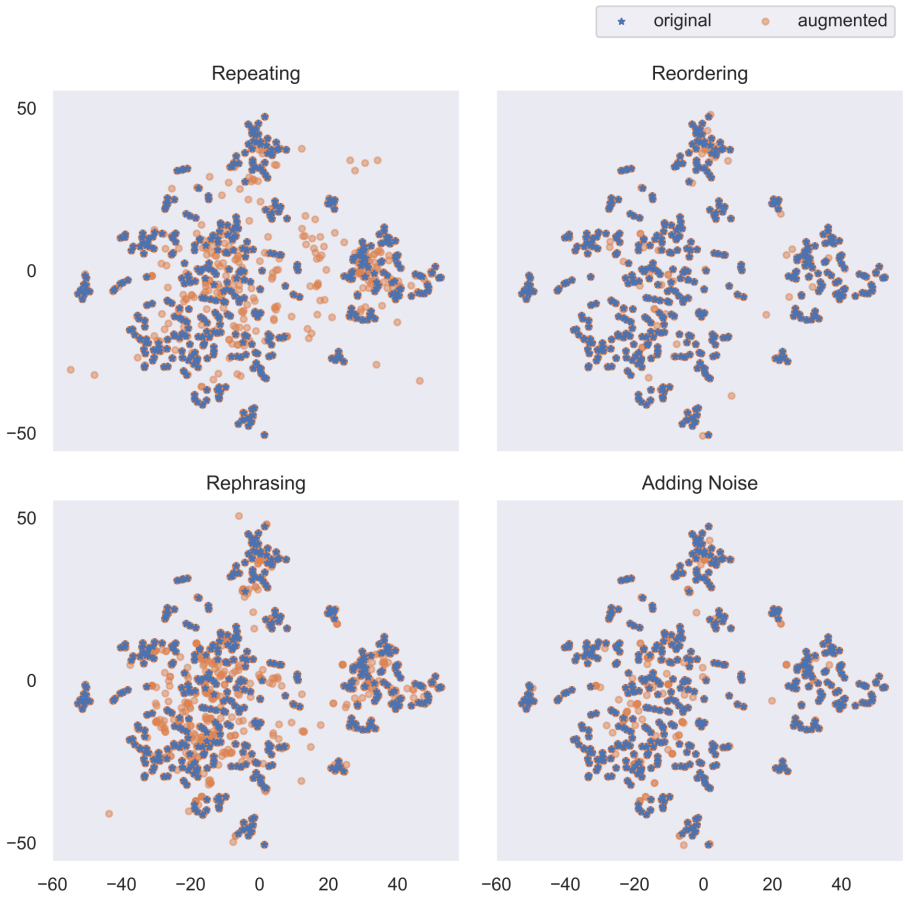



Fig. 18: Scatter plot of the embedding vectors for the sentences of original and augmented documents.

# Chapter 13

# Discussion and Future Work

In this thesis we presented three approaches to applying machine learning to the task of extracting process information from natural language process descriptions. We found that directly applying existing models from the wider field of information extraction is not feasible due to a lack of training data. Using pretrained language models, especially large, generative decoder-only transformer architectures and in-context learning seems to be possible, but comes with downsides that must be considered. These include, foremost, concerns about privacy when using external providers in the cloud, or major hardware requirements, when using large language models locally [52].

Instead, smaller machine learning models might be used right now, but could require retraining, if the application domain differs to much from the data contained in academic datasets, such as PET [53]. To alleviate this, data augmentation seems to be useful, allowing users to utilize existing data more efficiently. Annotating new data is made easier through our work on assisted data annotation for process information extraction data and the corresponding annotation tool TeaPie.

In summary, academics or practitioners interested in extracting process information and generating formal process models from it, should start with applying our work on large language model based process information extraction, due to its low barrier of entry [52]. These early experiments should start with non-critical processes, and could potentially be run on cloud providers of large language models. Alternatively, local models could be used, if the drop in predictive performance is acceptable. To proceed towards a solution for use beyond these early experiments, a custom dataset should be created to supplement the PET dataset, preferably with proper tool support, such as TeaPie [51, 55]. This dataset can then be used to train machine learning models small enough to run on local hardware [3, 53].

In the following sections we will discuss some points of consideration when it comes to applying the findings presented in this thesis. The remainder of this chapter will therefore serve as a summary of existing limitations and future work resulting from these limitations. Section 13.1 highlights some limitations in the flexibility of machine learning methods for process information extraction, when the target process modeling language is changed. Section 13.2 details several challenges in the synthesis of formal models from extracted process information. Section 13.3 discusses limitations and future work related to detecting and dealing with incomplete process descriptions. Finally, we conclude this thesis in Section 13.4.

## 13.1 Switching Modeling Languages

One of the major advantages of using machine learning for process information extraction is the flexibility it offers, when applying the same method to texts of different domains, e.g., legal texts, manufacturing, or public service, or applying them to different natural languages, e.g., German or English. These texts may use very different vocabulary and grammar, but they still describe the same type of information, i.e., the styles of description remain similar. This is less the case when switching modeling languages, especially when switching from imperative modeling languages like BPMN to declarative ones like DCR. While we showed that methods are able to adapt to different description styles either during training [3] or during inference [52], the discussion how humans describe workflow constraints in natural language remains part of future work. Related work indicates text describing law and medical procedures can be transformed particularly well to declarative process models [1, 46, 45]. Yet, it is still unclear if there is a fundamental difference between texts describing imperative processes and those describing declarative ones, and if humans prefer *thinking* in one way or the other. If there is a fundamental difference, reliably detecting the most suitable process modeling language would be a task that warrants further research.

## 13.2 From Process Information to Formal Models

Since there is often a trivial mapping between business process elements and the information extracted, we used process information extraction and model generation synonymously throughout this thesis. For PET mappings are, for example, *entity of type Activity → action in BPMN task label*, and *entity of type Actor → label of BPMN pool / lane*. Still, there are several challenges and issues left, when transforming extracted process information to formal process models. In this section we will discuss a selection of them to illustrate the importance of this transformation step, without claiming exhaustiveness.

**Label Generation.** This is the task of creating an easy to understand, concise, and correct phrase, that describes process elements, such as tasks, gateway paths, or lanes. There are guidelines for generating labels that satisfy these criteria [50], which a model generation approach should follow. Most importantly, these guidelines recommend avoiding noun-actions, e.g., *Order shipping*, and labels that do not contain any explicit action, e.g., *Order procedure*, in favor of explicit verb-object labels, e.g., *Ship the order*. Therefore, generating useful activity (task) labels involves transforming actions from their surface form as extracted from the natural language process description to a useful label [75, 1]. Many of the modern approaches do not consider this step [9, 3, 62].

**Layouting.** While creating elements in a formal business process model, arranging them is often disregarded in current approaches, or solved using simple heuristics. This can be problematic, since the ordering of tasks in an imperative process model (e.g., in BPMN), implies temporal ordering of process elements, especially the control flow of tasks. This ordering is commonly recommended as left-to-right [27], but other directions only have a small and statistically insignificant effect on the overall process comprehension of study participants [28]. Recent research instead highlights the importance of consistent use of workflow patterns [19], i.e., the ordering on a smaller scale [26]. Lübke et al. [47] found that most (87.92%) of publicly available

BPMN models on GitHub follow a strictly linear layout, i.e., either vertical or horizontal without any "bends". This is still a challenging field of research, which becomes clear, when considering that placing elements and routing edges between them becomes more complex, the more elements there are in a model. Even in moderately sized models placing, e.g., a data object element and routing the data association to all tasks that use said data object, without overlapping existing elements, becomes complex. Dedicated approaches for layouting process models implement ideas from graph layouting [37], but this limitation holds even for those.

**Inappropriate Granularity.** Extracting entities that are to coarse-grained can prohibit proper use of model elements. For example, most current annotation schemes do not consider the relation between different actors (e.g., *is part of*, or *is supervisor*) [9, 3, 62, 61, 1], which prevents proper use of swimlanes and pools, and in extension, modeling the hierarchical relationship between actors. Similarly, only the dataset by Quishpi et al. [62] differentiates verbs into actions and events. Increasing the granularity would allow model generation approaches to select appropriate elements, e.g., system or user tasks in BPMN. Extraction approaches should consider multi-grained entity types, like in Few-NERD [22], which contains 8 coarse-grained (e.g., *Person*) and 66 fine-grained entity types (e.g., *Actor*, *Artist*, *Athlete*).

**Relation Directionality.** The PET dataset used for evaluation of methods in this thesis does not fully exploit the semantics the direction of a relation provides, i.e., additional information valuable for generating a process model. Consider relations, which connect data objects to the activity in which they are used. In PET this relation is always originating in an activity and ending in the data object [9]. Thus, distinguishing an activity that produces a data object from one consuming it is impossible. Considering the directionality of relations in both the annotation schema and extracted information would solve this issue, but was out of scope for this thesis, and thus, is part of future work.

**Non-Continuous Entity Mentions.** If process relevant information is interrupted by tokens, that do not belong to a certain entity (process element), the methods of this thesis will only extract the information partially. This issue arises very often with prepositional verbs, e.g., *notify of*, *agree about*, or *send to*, if verb and preposition are separated by the sentence's object. Consider the sentence "*The INQ **notifies** the IP **of** the change*" from document *doc-10.12* of PET. Here, the complete action should be *notify of*, so that the activity label *notify of the change* can be generated. However, since most state-of-the-art data annotation schemes used for business process information extraction require continuous spans [9, 62, 3], only *notifies* is extracted. This would usually lead to a task with the label *notify the change*, which is misleading. Datasets and methods may resolve this issue by allowing non-continuous spans, or by including the transformation to an action in the extraction task, i.e., relaxing the alignment of extracted information with the original text [1].

## 13.3 Description Completeness

All of the methods we developed for this thesis assume completeness in the descriptions of business processes. This includes explicit statements of all relevant process elements, such as tasks, actors, or data objects, and also explicitness in statements of decisions. Humans tend to convey much information implicitly, e.g., the sentence "*If the bike passes quality assurance, it is*

*sent to the customer.*" implies that the bike may not pass quality assurance and needs rework. A human reader easily understands this fact and might even include it in the corresponding process model, while a machine might not.

Description completeness also refers to issues with the precision of natural language, or the lack thereof. Natural language statements such as "The customer fills out the form and submits it to the agency, who registers and reviews it. In parallel to these steps, [...]" leave room for interpretation, as to which steps the text refers to.

Checking natural language process descriptions for completeness was out of scope for this thesis, but is a stream of research that becomes more and more popular, especially with the advent of process modeling chatbots [40]. These chatbots can ask for additional information, if an incomplete description is detected.

## 13.4 Future Work and Conclusion

In this thesis we developed three approaches for applying deep learning in the task of process information extraction from natural language process descriptions. In the earlier Sections 13.1, 13.2, and 13.3, we discussed several limitations. These limitations represent promising avenues of future work beyond the scope of this thesis.

Additionally, we plan to create a cloud-based platform for synthesizing process models from natural language process descriptions, based on the work presented in this thesis. We plan to achieve this by implementing the pipeline from Figure 1.2 in a web application, and integrating the options **1**– **3** for mitigating a lack of training data. This means, the platform should have an improved version of our efficient annotation tool *TeaPie* (Chapters 10 and 11), which allows users to create their own training data. This data will then be automatically augmented with the perturbation techniques we identified as useful in the corresponding publications (Chapters 8 and 12). Finally, we will let users use a variety of pretrained models, that are either fine-tuned with their training data (Chapters 5, 6, and 7), or use in-context learning (Chapter 9). An implementation of a research prototype for this platform is already in active development, and a extensive study on its feasibility with representatives from industry and academia is planned in the future.

In conclusion, the application of deep learning in process information extraction has the potential of facilitating the creation of formal process models, and support academics and practitioners in domains, where no rule-based text-to-model methods exist. The methods we developed in this thesis enable the application of deep learning, even if no (or not enough) training data exists, which is crucial for small and medium sized organizations interested in the automatic generation of process models from their internal data. With this, we make a contribution towards lowering the barrier of entry for applied business process management.

# Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet habe. Weiterhin erkläre ich, dass ich die Hilfe von gewerblichen Promotionsberatern bzw. -vermittlern oder ähnlichen Dienstleistern weder bisher in Anspruch genommen habe, noch künftig in Anspruch nehmen werde. Zusätzlich erkläre ich hiermit, dass ich keinerlei frühere Promotionsversuche unternommen habe.

Bayreuth, den 25.02.2025

# List of Figures

# Bibliography

[1] Han van der Aa, Claudio Di Ciccio, Henrik Leopold, and Hajo A Reijers. "Extracting declarative process models from natural language". In: *CAiSE*. 2019.

[2] Han van der Aa, Henrik Leopold, Felix Mannhardt, and Hajo A Reijers. "On the fragmentation of process information: Challenges, solutions, and outlook". In: *International Workshop on Business Process Modeling, Development and Support*. Springer. 2015, pp. 3–18.

[3] Lars Ackermann, Julian Neuberger, and Stefan Jablonski. "Data-Driven Annotation of Textual Process Descriptions Based on Formal Meaning Representations". In: *CAiSE*. 2021.

[4] Lars Ackermann, Julian Neuberger, Martin Käppel, and Stefan Jablonski. "Bridging Research Fields: An Empirical Study On Joint, Neural Relation Extraction Techniques". In: *CAiSE*. 2023.

[5] Henry S Baird. "Document image defect models". In: *Structured Document Image Analysis*. Springer, 1992, pp. 546–556.

[6] Patrizio Bellan and Mauro Dragoni. "PET Annotation Visualizer: A Tool to Visualize the Process Model Extraction from Text (PET) Dataset". In: *International Conference on Applications of Natural Language to Information Systems*. Springer. 2024, pp. 79–84.

[7] Patrizio Bellan, Mauro Dragoni, and Chiara Ghidini. "Extracting Business Process Entities And Relations From Text Using Pre-Trained Language Models And In-Context Learning". In: *EDOC*. 2022.

[8] Patrizio Bellan, Mauro Dragoni, Chiara Ghidini, Han van der Aa, and Simone Paolo Ponzetto. "Process Extraction from Text: Benchmarking the State of the Art and Paving the Way for Future Challenges". In: *arXiv preprint arXiv:2110.03754* (2021).

[9] Patrizio Bellan, Chiara Ghidini, Mauro Dragoni, Simone Paolo Ponzetto, and Han van der Aa. "Process extraction from natural language text: the PET dataset and annotation guidelines". In: *NL4AI*. 2022.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[11] Timm Caporale. "A Tool for Natural Language Oriented Business Process Modeling." In: *ZEUS*. 2016, pp. 49–52.

[12] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]". In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 542–542.

[13] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. "Pre-trained image processing transformer". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 12299–12310.

[14] Alebachew Chiche and Betselot Yitagesu. "Part of speech tagging: a systematic review of deep learning and machine learning approaches". In: *Journal of Big Data* 9.1 (2022), p. 10.

[15] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[16] Nicolas Daclin, Sihem Mallek-Daclin, and Gregory Zacharewicz. "Generative AI for Business Model Generation (GAI4BM): from Textual Description to Business Process Model". In: *the 10th International Food Operations and Processing Simulation Workshop*. CAL-TEK srl. 2024.

[17] Luis Delicado Alcántara, Josep Sanchez-Ferreres, Josep Carmona Vargas, and Lluís Padró. "The model judge: a tool for supporting novices in learning process modeling". In: *BPMTracks 2018: BPM 2018 Dissertation Award, Demonstration, and Industrial Track: proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018), Sydney, Australia, September 9-14, 2018*. CEUR-WS. org. 2018, pp. 91–95.

[18] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society: series B (methodological)* 39.1 (1977), pp. 1–22.

[19] Wil MP van Der Aalst, Arthur HM Ter Hofstede, Bartek Kiepuszewski, and Alistair P Barros. "Workflow patterns". In: *Distributed and parallel databases* 14 (2003), pp. 5–51.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[21] Kaustubh D Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahamood, Abinaya Mahendiran, Simon Mille, Ashish Shrivastava, Samson Tan, et al. "Nl-augmenter: A framework for task-sensitive natural language augmentation". In: *arXiv preprint arXiv:2112.02721* (2021).

[22] Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Hai-Tao Zheng, and Zhiyuan Liu. "Few-nerd: A few-shot named entity recognition dataset". In: *arXiv preprint arXiv:2105.07464* (2021).

[23] Marlon Dumas, L Marcello Rosa, Jan Mendling, and A Hajo Reijers. *Fundamentals of business process management*. Springer, 2018.

[24] Markus Eberts and Adrian Ulges. "Span-based joint entity and relation extraction with transformer pre-training". In: *ECAI 2020*. IOS Press, 2020, pp. 2006–2013.

[25] Markus Eberts and Adrian Ulges. “An End-to-end Model for Entity-level Relation Extraction using Multi-instance Learning”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021.

[26] Kathrin Figl, Pnina Soffer, and Barbara Weber. “Guiding attention in flow-based conceptual models through consistent flow and pattern visibility”. In: *Decision Support Systems* 185 (2024), p. 114292.

[27] Kathrin Figl and Mark Strembeck. “On the importance of flow direction in business process models”. In: *2014 9th International Conference on Software Engineering and Applications (ICSOFT-EA)*. IEEE. 2014, pp. 132–136.

[28] Kathrin Figl and Mark Strembeck. “Findings from an experiment on flow direction of business process models”. In: *International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA)*. Gesellschaft für Informatik, Bonn. 2015, pp. 59–73.

[29] Fabian Friedrich, Jan Mendling, and Frank Puhlmann. “Process Model Generation from Natural Language Text”. In: *CAiSE*. 2011.

[30] Ian Goodfellow. *Deep learning*. 2016.

[31] Daniel Grathwol, Han van der Aa, and Hugo A López. “Automating Pathway Extraction from Clinical Guidelines: A Conceptual Model, Datasets and Initial Experiments”. In: *International Conference on Cooperative Information Systems*. Springer. 2024, pp. 296–312.

[32] Michael Grohs, Luka Abb, Nourhan Elsayed, and Jana-Rebecca Rehse. “Large language models can accomplish business process management tasks”. In: *International Conference on Business Process Management*. Springer. 2023, pp. 453–465.

[33] Joachim Herbst and DIMITRIS Karagiannis. “An inductive approach to the acquisition and adaptation of workflow models”. In: *Proceedings of the IJCAI*. Vol. 99. Citeseer. 1999, pp. 52–57.

[34] Siyu Huo, Kushal Mukherjee, Jayachandu Bandlamudi, Vatche Isahagian, Vinod Muthusamy, and Yara Rizk. “Accelerating the support of conversational interfaces for RPAs through APIs”. In: *International Conference on Business Process Management*. Springer. 2023, pp. 165–180.

[35] Ana Ivanchikj, Souhaila Serbout, and Cesare Pautasso. “From text to visual BPMN process models: Design and evaluation”. In: *Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems*. 2020, pp. 229–239.

[36] Martin Käppel and Stefan Jablonski. “Model-Agnostic Event Log Augmentation for Predictive Process Monitoring”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 2023, pp. 381–397.

[37] Ingo Kitzmann, Christoph König, Daniel Lübke, and Leif Singer. “A simple algorithm for automatic layout of bpmn processes”. In: *2009 IEEE Conference on Commerce and Enterprise Computing*. IEEE. 2009, pp. 391–398.

[38] Nataliia Klievtsova, Janik-Vasily Benzin, Juergen Mangler, Timotheus Kampik, and Stefanie Rinderle-Ma. “Process Modeler vs. Chatbot: Is Generative AI Taking Over Process Modeling?” In: ().

[39] Lars Klöser, Philipp Kohl, Bodo Kraft, and Albert Zündorf. "Multi-Attribute Relation Extraction (MARE)–Simplifying the Application of Relation Extraction". In: *arXiv preprint arXiv:2111.09035* (2021).

[40] Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil MP van der Aalst. "Process Modeling With Large Language Models". In: *arXiv preprint arXiv:2403.07541* (2024).

[41] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[42] Bohan Li, Yutai Hou, and Wanxiang Che. "Data augmentation approaches in natural language processing: A survey". In: *Ai Open* 3 (2022), pp. 71–90.

[43] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. "A survey on deep learning for named entity recognition". In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[44] Josip Tomo Licardo, Nikola Tanković, and Darko Etinger. "A Method for Extracting BPMN Models from Textual Descriptions Using Natural Language Processing". In: *Procedia computer science* 239 (2024), pp. 483–490.

[45] Hugo A López, Rasmus Strømsted, Jean-Marie Niyodusenga, and Morten Marquard. "Declarative process discovery: Linking process and textual views". In: *International Conference on Advanced Information Systems Engineering*. 2021.

[46] Hugo-Andrés López-Acosta, Thomas Hildebrandt, Søren Debois, and Morten Marquard. "The process highlighter: From texts to declarative processes and back". In: *CEUR Workshop Proceedings*. CEUR Workshop Proceedings. 2018, pp. 66–70.

[47] Daniel Lübke, Maike Ahrens, and Kurt Schneider. "Influence of diagram layout and scrolling on understandability of BPMN processes: an eye tracking experiment with BPMN diagrams". In: *Information Technology and Management* 22.2 (2021), pp. 99–131.

[48] J MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*. 1967.

[49] Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. "Building a large annotated corpus of English: The Penn Treebank". In: *Computational linguistics* 19.2 (1993), pp. 313–330.

[50] Jan Mendling, Hajo A Reijers, and Wil MP van der Aalst. "Seven process modeling guidelines (7PMG)". In: *Information and software technology* 52.2 (2010), pp. 127–136.

[51] Julian Neuberger, Han van der Aa, Lars Ackermann, Daniel Buschek, Jannic Herrmann, and Stefan Jablonski. "Assisted Data Annotation for Business Process Information Extraction from Textual Documents". In: (2024).

[52] Julian Neuberger, Lars Ackermann, Han van der Aa, and Stefan Jablonski. "A Universal Prompting Strategy for Extracting Process Model Information from Natural Language Text Using Large Language Models". In: *International Conference on Conceptual Modeling*. Springer. 2024, pp. 38–55.

[53] Julian Neuberger, Lars Ackermann, and Stefan Jablonski. "Beyond Rule-Based Named Entity Recognition and Relation Extraction for Process Model Generation from Natural Language Text". In: *CoopIS*. 2023.

[54] Julian Neuberger, Leonie Doll, Benedikt Engelmann, Lars Ackermann, and Stefan Jablonski. "Leveraging Data Augmentation for Process Information Extraction". In: *International Conference on Business Process Modeling, Development and Support*. Springer. 2024, pp. 57–70.

[55] Julian Neuberger, Jannic Herrmann, Martin Käppel, Han van der Aa, and Stefan Jablonski. "TeaPie—A Tool for Efficient Annotation of Process Information Extraction Data". In: (2024).

[56] Matthew J Page, Joanne E McKenzie, Patrick M Bossuyt, Isabelle Boutron, Tammy C Hoffmann, Cynthia D Mulrow, Larissa Shamseer, Jennifer M Tetzlaff, Elie A Akl, Sue E Brennan, et al. "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews". In: *bmj* 372 (2021).

[57] Lucas Francisco Amaral Orosco Pellicer, Taynan Maier Ferreira, and Anna Helena Reali Costa. "Data augmentation techniques in natural language processing". In: *Applied Soft Computing* 132 (2023), p. 109803.

[58] Mohammad Taher Pilehvar and Jose Camacho-Collados. *Embeddings in natural language processing: Theory and advances in vector representations of meaning*. Morgan & Claypool Publishers, 2020.

[59] Pavlin G. Poličar, Martin Stražar, and Blaž Zupan. "openTSNE: A Modular Python Library for t-SNE Dimensionality Reduction and Embedding". In: *Journal of Statistical Software* 109.3 (2024), pp. 1–30. DOI: 10.18637/jss.v109.i03. URL: https://www.jstatsoft.org/index.php/jss/article/view/v109i03.

[60] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. "CatBoost: unbiased boosting with categorical features". In: *NeurIPS* (2018).

[61] Chen Qian, Lijie Wen, Akhil Kumar, Leilei Lin, Li Lin, Zan Zong, Shu'ang Li, and Jianmin Wang. "An Approach for Process Model Extraction by Multi-grained Text Classification". In: *CAiSE*. 2020.

[62] Luis Quishpi, Josep Carmona, and Lluís Padró. "Extracting annotations from textual descriptions of processes". In: *BPM 2020*. 2020.

[63] Alec Radford. "Improving language understanding by generative pre-training". In: (2018).

[64] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[65] Josep Sànchez-Ferreres, Andrea Burattin, Josep Carmona, Marco Montali, and Lluís Padró. "Formal reasoning on natural language descriptions of processes". In: *Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17*. Springer. 2019, pp. 86–101.

[66] Josep Sànchez-Ferreres, Andrea Burattin, Josep Carmona, Marco Montali, Lluís Padró, and Luís Quishpi. "Unleashing textual descriptions of business processes". In: *SoSyM* (2021).

[67] John Searle. *Minds, Brains, and Programs*. 1980.

[68] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1 (2019), pp. 1–48.

[69] Konstantinos Sintoris and Kostas Vergidis. "Extracting business process models using natural language processing (NLP) techniques". In: *2017 IEEE 19th conference on business informatics (CBI)*. 2017.

[70] Tijs Slaats, Raghava Rao Mukkamala, Thomas Hildebrandt, and Morten Marquard. "Exformatics declarative case management workflows as DCR graphs". In: *Business Process Management: 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings*. Springer. 2013, pp. 339–354.

[71] Konstantin Sokolov, Dimitri Timofeev, and Alexander Samochadin. "Process extraction from texts using semantic unification". In: *International Conference on Knowledge Management and Information Sharing*. Vol. 2. SCITEPRESS. 2015, pp. 254–259.

[72] Riad Sonbol, Ghaida Rebdawi, and Nada Ghneim. "A Machine Translation Like Approach to Generate Business Process Model from Textual Description". In: *SN Computer Science* 4.3 (2023), p. 291.

[73] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[74] Martin A Tanner and Wing Hung Wong. "The calculation of posterior distributions by data augmentation". In: *Journal of the American statistical Association* 82.398 (1987), pp. 528–540.

[75] Han Van der Aa, Josep Carmona Vargas, Henrik Leopold, Jan Mendling, and Lluís Padró. "Challenges and opportunities of applying natural language processing in business process management". In: *COLING*. 2018.

[76] Han Van der Aa, Henrik Leopold, and Hajo A Reijers. "Checking process compliance against natural language specifications using behavioral spaces". In: *IS* (2018).

[77] A Vaswani. "Attention is all you need". In: *Advances in Neural Information Processing Systems* (2017).

[78] Anli Wang, Linyi Li, Xuehong Wu, Jianping Zhu, Shanshan Yu, Xi Chen, Jianhua Li, and Hongtao Zhu. "Entity relation extraction in the medical domain: based on data augmentation". In: *Annals of Translational Medicine* 10.19 (2022).

[79] Jason Wang, Luis Perez, et al. "The effectiveness of data augmentation in image classification using deep learning". In: *Convolutional Neural Networks Vis. Recognit* 11.2017 (2017), pp. 1–8.

[80] Patrick Henry Winston. *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[81] Wei Xiang and Bang Wang. “A survey of event extraction from text”. In: *IEEE Access* 7 (2019), pp. 173111–173137.

[82] Shuoheng Yang, Yuxin Wang, and Xiaowen Chu. “A survey of deep learning techniques for neural machine translation”. In: *arXiv preprint arXiv:2002.07526* (2020).