Automatic Generation of Activity Labels for Process Fragments

Master Thesis

presented by
Yen-Ting, Wang
Matriculation Number 1714072

submitted to the
Data and Web Science Group
Prof. Dr. Han van der Aa
University of Mannheim

August 2022

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In process mining, creating abstracted views of event logs or process models is often necessary to deliver succinct and impactful information for process stakeholders to make business decisions. Specifically, abstraction techniques enhance the functionalities offered by the process mining tools. For instance, right granularity of event logs can be provided for the users to conduct practical analysis. Additionally, users can generate different abstraction levels of process models dynamically as they interact with the tools. In most scenarios, the value of abstraction highly depends on the quality of its activity labels. These labels have to be comprehensible and representative for one to understand the process. However, it is considered challenging to automatically label process fragments or groups of events [2]. Therefore, we propose a cross-disciplinary approach that integrates state-of-the-art methods from the field of Machine Learning for Natural Language Processing to solve the problem.

1.1 Problem Statement

Event log abstraction addresses problems when events are recorded at a lower-level granularity than the relevant concepts from a business viewpoint. Similarly, process model abstraction helps process stakeholders to grasp quickly what is at hand through obtaining high-level or multi-level abstraction of existing process models. Low-level process steps frequently happen in the areas of customer relationship management (CRM), enterprise resource planning (ERP), health care, or IoT systems. Similar problems also occur when the process intrinsically allows for a high degree of flexibility, which is relatively common in fields like E-commerce, software systems, or SaaS applications [3]. Abstraction is not only essential for process comprehension but also process management. Having high-level activity

labels representing groups of process steps helps detect similar clusters from different active workstreams and subsequently optimize resource planning and usage. Furthermore, for the reusability purpose of process fragments, high-level activity labels from existing process models support users in conducting efficient searching for valuable resources in the database.

Numerous abstraction techniques exist that address this task by detecting sets of low-level process steps that can be grouped into higher-level business activities. However, identifying such groups does not yet lead to useful abstracted process models, since a proper process model needs suitable labels for each of its activities. The current labeling steps involved in abstraction processes rely heavily on expert domain knowledge or some form of a reference model, providing the activity labels with an ideal level of abstraction. There are also earlier works proven to be effective on the topic of automatic label generation, but they are rather rule-based and hard to scale. In general, the manual effort required in current labeling techniques should not be ignored. As a result, we propose a labeling approach that automatically produces suitable labels for business process fragments in a sustainable and scalable way.

1.2 Contribution

It is a complex task to automatically generate meaningful labels given a group of low-level process steps since it requires to infer an overarching activity from the actions, business objects, and application contexts of multiple low-level process steps. Furthermore, how to derive such an overarching label depends on the nature of the underlying process and, thus, differs in each situation. Based on the challenges mentioned and the observations made from the previous work, we defined a set of labeling strategies as pivots of our labeling approach. The approach is a learning mechanism that is guided by the labeling strategies and trained on all available process data. In particular, we utilize machine learning techniques for natural language processing to detect useful patterns and output labels in a data-driven way. Furthermore, the techniques can be continuously trained on new process data as well as extrapolate to unseen process groups.

Given a group of low-level process steps, the outcome of our labeling approach is to automatically select the appropriate labeling strategy for the group and apply it to obtain a high-level activity label. For instance, in Figure 1.1, the proposed labels “manufacture products” and “manage order production” are generated by two variations of our labeling approach to represent the group of process steps “Purchase Raw Materials, Make Production Plan, Manufacture Products”. The labels proposed reflect different labeling strategies selected by the approaches. The

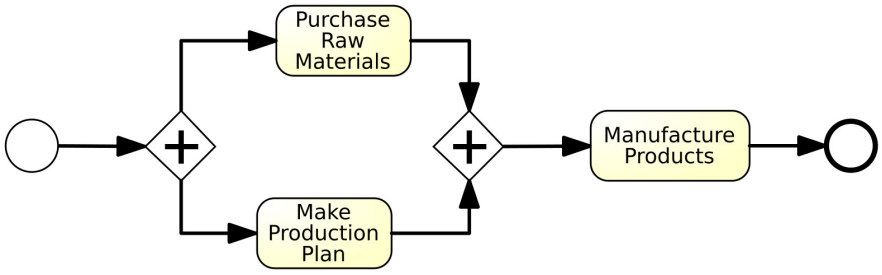


Figure 1.1: Example of Automatic Labeling Generation

former approach extracts the most important task that stands for the main outcome of the process, while the latter one abstracts the underlying process and provides a good overview.

Besides proposing labels for process fragments, our automatic generation approach also provides the basis for labeling the full processes. The approach is applied to different levels of abstraction and validated with suitable evaluation metrics. Furthermore, we conducted a user study that evaluates the generated labels against the human created labels, and the outcome indicates that the overall performance of our labeling approach is comparable to humans.

1.3 Thesis Overview

The remainder of this thesis is structured as follows. In chapter 2, Motivation, we illustrate the motivation behind our labeling approach, namely machine learning for automatic label generation. Next, in chapter 3, Background, we provide fundamental concepts and existing work of three concerning areas: process abstraction, automatic label generation, and machine learning for natural language processing. Then, our labeling approach and its implementation are explained in detail in chapter 4, Labeling Approach. Lastly, the evaluation of the labeling approach and conclusion of this thesis are in chapter 5, Evaluation and chapter 6, Conclusion.

Chapter 2

Motivation

The earlier work of automatic label generation designed a rule-based system that is built upon the observations made from the empirical experiments. Although the rule-based system has proven to be effective, it cannot be easily scaled as the rules defined by the experts are hard-coded. On the contrary, a machine learning system defines its own set of rules through training and produces output based on probabilistic decision making. Such systems constantly evolve and adapt their production in accordance with the new input data or expert feedback. As a result, machine learning systems can be developed in a continuous learning process and, thus, can be easily scaled.

In this chapter, we review some common problems encountered in rule-based methods and how they can be tackled by employing machine learning systems. Through the discussions, we manifest the motivation behind our labeling approach.

2.1 Common Problems in Rule-based Methods

Certain challenges tend to arise regarding rule-based methods. We discuss three specific aspects in automatic label generation: manual effort required, generalization-specialization balance, and treatment of anomalies and unseen data.

Rule-based methods often require more manual updates. For instance, the earlier work suggests constructing a label repository indexed on the dominating element, either main action or business object, as a knowledge base for reference. By identifying a dominating element in the process, this repository can be consulted to find complementing elements that are likely to co-occur with the dominating element, and the returned labels are suggested. The repository is built upon the activities of all available process models, hence, an extensive list of activities constantly being managed and updated. The search time as well as the list of returned

matching labels increase when the size of the repository grows. In addition, such techniques could soon face the issue of efficiency regarding having the most related labels as output.

Besides requiring constant updates and potentially needing more sophisticated algorithms, striking a balance between generalization and specialization can also be a struggle. The earlier work has tried to address the problem of holonymy detection on process model activities by building on WordNet. Yet, while WordNet is useful to detect general-purpose holonymy relations, such as between "finger" and "hand", it fails to detect more complex and domain-specific relations commonly found in process-mining contexts. Furthermore, the list of process elements is reduced to the ones that can be found in the WordNet dictionary. While adding domain-specific ontology as reference can help solve the issue of limited lexical relations retrieved from WordNet taxonomy, it is a non-trivial effort to derive such ontology from domain-specific text corpora and it again requires regular updates and maintenance. In general, the holonym-based labeling strategy using rule-based methods can not be easily scaled.

Another major shortcoming of the rule-based method is that it cannot handle the exceptions or unseen cases well. To detect the main activity of the process means the role of each activity should be determined in the process. The rule-based method utilizes the insights of the earlier work gained in practice, which is that approximately 85% of all main activities are positioned either at the start or at the end of a process. The example of handling goods shipment can support this as such a process would typically include a set of preparatory steps before the last activity eventually asks for the actual shipping. Nevertheless, it only covers the case of the process outcome being the most important task but not for the case of decision point in the process. The earlier work also utilizes the start and end events since they are expected to contain useful information about the process such as the overall goal. However, the start events can also represent a state that was required to trigger the process but not directly concerning the main content of the process. Similar situations can also occur for the end events. Hence, in order to make decisions regarding whether the identified start or end event is useful or not, a mechanism is needed to discern particular signal terms given in the event label. From the example given in the earlier work, for the start events "Asset was found" and "Asset is to be created", we can tell that the term "was" in the first label is more likely to represent a condition for triggering the process, while for the term "is to be" clearly points to an action which is covered by the process. The earlier work derived an extensive classification of such terms from the investigated process model collections. An apparent drawback is that the unseen signal terms existing in the new processes will not be identified and relevant information is therefore left out.

2.2 Machine Learning as Solution

Due to the challenges discussed above, we leveraged techniques from Machine Learning to avoid the limitations faced in rule-based methods while effectively deriving semantic relations between elements within processes. We provide the intuition below why Machine Learning is a natural solution to the problems.

Due to the deterministic nature of rule-based methods, manual updates and maintenance are often required based on the implications of the new data. On the other hand, machine learning systems are adaptive. The ability to learn causes adaptive intelligence, meaning that existing knowledge can be changed or discarded, and new knowledge can be acquired. To illustrate, a machine learning system, such as a deep neural network, automates the process of learning through optimization. Given a defined model and an utility function, the goal of a learning system is to minimize the utility function by adjusting the weights of the model based on the data. The action of optimizing the utility function can be executed continuously whenever there is new data available to be trained on and it can be triggered automatically.

Chapter 3

Background

3.1 Process Abstraction

Process abstraction consists of two main types, event log abstraction and process model abstraction, depending on the input data. As the main goals and fundamental concepts of the two types are similar to each other, we introduce only the event log abstraction for the purpose of conciseness.

3.1.1 The Goal of Event Log Abstraction

Process mining has proven to be useful for the stakeholders to gain insights into processes and hence, improve the performance of the business. Event log and process model are typical input data for process mining techniques. In reality, the mixed or fine granular event data shows limitations in the direct application of process mining techniques. In the context of process discovery, event data is required to be at the appropriate level of granularity in order to produce human-interpretable process model.

As described, problems arise when events are recorded at a lower-level granularity than the concepts that are relevant from a business viewpoint, which happen frequently in the field of customer relationship management (CRM), enterprise resource planning (ERP), health care, or IoT systems. Similar problems also occur when the process intrinsically allows for a high degree of flexibility, which is relatively common in the field of E-commerce, software system, or SaaS applications. Clickstream data of a web application, for example, encounter both aforementioned scenarios. The "ocean" of observed process behavior is discovered automatically when applying process discovery algorithms directly on the raw logged clickstream data. The model discovered is too complex and low-level to interpret and hence, achieve the overall goal of process mining, such as an improved understanding of

the process. Data as such also exhibits a huge behavioral variability causing difficulties to find exact match between the captured events and the modeled activities (in the case where process model is provided).

It is vital for successful application of process mining techniques to have event logs at an appropriate level of abstraction, namely, the right level of granularity of events. After abstracting fine-granular events into higher level concepts, process mining techniques can yield more meaningful results for the process stakeholders to conduct further analysis.

3.1.2 Approaches to Event Log Abstraction

Numerous kinds of event abstraction abstraction techniques are implemented in either supervised or unsupervised manner, each with their own advantages and disadvantages. The following section explains the general implementation logic of the abstraction techniques. It consists of two parts executed in sequence, event grouping or log segmentation and the mapping from low-level event groups to high-level activities.

Event Grouping

This step groups the events that often executed or appear closely together. The grouping method varies in the supervision strategy. It can be done in either a supervised or unsupervised approach. Methods are not limited to the examples below.

Reference Model Components Based on the spatial proximity measured between activities, groups of activities are clustered into a hierarchical structure [4]. The idea behind this measure is that activities which often appear in close proximity to each other form a logical unit with a clear structure. After obtaining the cluster hierarchy, a reference model component is mined for each identified activity set. Reference components are seen as frequently appearing process model building blocks representing high-level activities.

Maximum Likelihood method It is a technique that identifies the potential segment separators and optimize the segmentation likelihood of a trace [2]. The segmentation requires a relatively small set of examples, namely, segmented list, provided by the data owner or expert, annotated with the correct separation between one activity and the next. A standard n-gram analysis can be run on the segmented list and form a digram matrix with each cell representing the frequency of occurrence for the corresponding event digram, hence, the likelihood after divided by the total sum of its row. Based on the matrix, segmentation likelihood can be calculated iteratively with each placement of separator. Once convergence, the segment separators are placed in the flow of low-level events.

Mapping to High-level Activity

This step takes in the outcome of the event grouping step and further maps or projects a sequence of low-level events to the corresponding high-level activity. The result is the high-level process model achieved by the abstracted log transformed from an raw event log.

Alignment-based Technique Alignment technique [5] is usually applied to achieve a high-level event log after activity patterns are obtained at the first step. It is considered as a pattern-based abstraction. The input is an event log and a set of activity patterns. An alignment establishes a mapping between a log trace and a process trace from the activity patterns. Through the use of the alignment technique, we can capture "approximate" executions of activity patterns. The use of alignments is justified by the fact that, in general, event logs are noisy. Hence, not all low-level events are expected be mapped onto high-level activities. Furthermore, the search for an optimal mapping requires considering entire traces. Alignment-based techniques do this by solving optimization problems with user-defined cost function.

Supervised Approach Supervised mapping is usually done with extra information or manual effort provided by the experts or end-users. Most techniques assume that there exist some form of reference model. Some approaches utilize Machine Learning frameworks, such as Support Vector Machine (SVM) and Random Forest (RF), to train classifiers that learn the mapping between low-level event logs and high-level activities. It requires some training data, subset of traces, labeled with high-level target activities by process analysts.

Unsupervised Approach Along with the clustering or grouping technique that group the fine granular events on the basis of close proximity or high similarity, a hierarchical clustering of events is computed, which forms the basis for coarse granular event recognition. Mapping can therefore be executed. Other unsupervised techniques utilize Machine Learning algorithms, such as DBScan and K-means, to execute mapping. It requires segments or sessions to be encoded as vectors to be subsequently clustered. Frequency-based encoding and duration-based encoding are of general applicability. The centroids of the clusters are expected to provide meaningful information for high-level activity identification.

3.2 Automatic Label Generation

3.2.1 Guideline of Labeling Styles

Before looking into how we generate relevant content for the activity labels, grammatical structure of the labels should be examined as it affects the understandability of the process. Simple guidelines based on the user study provided in [6] can help generate fluent and semantically less ambiguous labels for the readers. As the labeling of activities can be seen as a rather arbitrary task in process modeling and undermine the understandability of resulting models, guidelines discussed below can prevent generating counter-intuitive labels to the readers. The authors investigated the labeling styles that are in use to annotate activities in process models and how these styles affect the understandability of such models. Based on their empirical findings, three classes of grammatical context, Verb-object labeling, Action-noun labeling and Rest labeling styles are defined and analyzed.

- **Verb-object labeling:** Activity labels such as "Fulfill Order" and "Sort Invoice" follow this labeling style. In practice, a number of informal guidelines exist that typically suggest a verb-object convention for activity labels. However, confusion might be introduced in the situation of zero derivation in English, meaning the same word can be both a noun and a verb. For the example in the original paper, "Measure Processing" could potentially refer to the processing of a measure or to the measurement of a processing. Yet, it would be clear to the reader to interpret the first term as a verb if the verb-object style is consistently used.

- **Action-noun labeling:** Activity labels such as "Order Fulfillment", "Production Planning Management" follow this labeling style. Some of the action-noun labels can be easily interpreted, but there can also be cases of grammatical ambiguity. For instance, "Notification Printing" with two potential interpretations: a notification is printed, or someone is notified of a printing job. This interpretation is likely in cases where the action noun could also be an object, like Order. Simply extending or changing labels to a verb-object labeling style can avoid the problem of action-object ambiguity.

- **Rest labeling:** Activity labels such as "Customer Data Lifecycle" and "Status Analysis Cash Position" follow this labeling style. Labels of the "rest" category require readers to have context information, otherwise an inference of the action to be performed is a highly problematic task due to the occurrence of verb inference ambiguity, such that the problem of inferring from the context regarding the type of action to be performed as part of the considered process task.

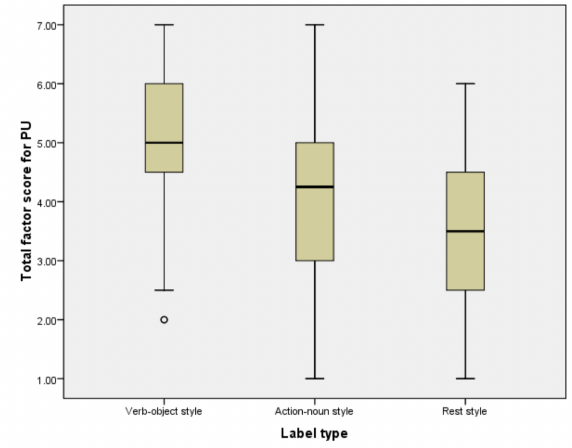


Figure 3.1: Perceived Usefulness of Labeling Styles

As described above, three different classes exhibit different types of ambiguities. Based on the author's grammatical analysis, they theorize that process modelers perceive the verb-object style to be superior to the action-noun and rest labeling style alongside two dimensions, Perceived Ambiguity (PA) and Perceived Usefulness (PU). The former means the degree to which one believes that a label is ambiguous, and the latter means the degree to which one believes that a label is useful for understanding the process modeled. In addition, authors carried out a user study to gather quantitative insights on the hypotheses. Based on the collected results, several analyses are conducted to test the hypotheses. With respect to perceived ambiguity, verb-object labels receive the lowest rank total, which means that this type is least often considered as containing ambiguous labels. Regarding perceived usefulness, the analysis is visualized through a box-plot, Figure 3.1, showing that verb-object labels were found to be best in terms of their perceived usefulness, followed by action-noun labels, and then the rest group. In conclusion, their research suggests that verb-object labeling is an imperative style to apply for modelers to create more understandable process models.

After studying the grammatical structure of labeling and knowing the specific challenges it could bring, we shall incorporate the guideline in the design process of automatic activity label generation.

3.2.2 Earlier Work on Automatic Label Generation

Earlier work around automatic labeling for process data is rather scarce. However, the authors of this work [7] laid a solid foundation in this research area by proposing an automatic labeling approach that builds on the linguistic analysis of process models from industry. In their approach, label suggestions generated for a process model are based on its events and activities, which is applicable to process model fragments as well. The authors discussed the problem of assigning a meaningful label to a process and identified a list of labeling strategies in models from practice. They developed a classification of labeling strategies, including Dominating Element, Main Activity, Start or End Events, Conjunction of Activities, and Semantic Naming. Since this classification has important implications in this paper, they are partially intercepted from the original paper below.

- **Dominating Element:** If one particular business object or action is mentioned more often in activity labels than any other business object or action, this element is considered to be a dominating element.
- **Main Activity:** Processes might contain one particular activity that is of central importance. The remaining activities have the character of side activities supporting, preparing or evaluating the result of this activit
- **Start or End Events:** The state at the beginning or at the end of the process often defines the overall goal of the process, the label of the whole model may be closely related to them.
- **Conjunction of Activities:** If the same action is performed on different business objects or different actions are applied on the same business object, these activities can be easily described in terms of a conjunction.
- **Semantic Naming:** As opposed to other concepts, semantic naming does not refer to one or more model elements, but uses the broader context of the activities for labeling the process model.

Although the labeling strategies are observed and analyzed from processes captured as EPC process models, authors in their continuation of work [8] stated that the strategies and their approach can be generalized to models defined in other process modeling languages beyond EPC, such as BPMN. As the data collection used in this paper is modeled in BPMN 2.0, the sub-process in Figure 3.2 demonstrating the labeling strategies is given in BPMN 2.0.

The sub-process has object coffee as a dominating element. The activity serve coffee has the characteristics of a main activity, while prepare and serve coffee is

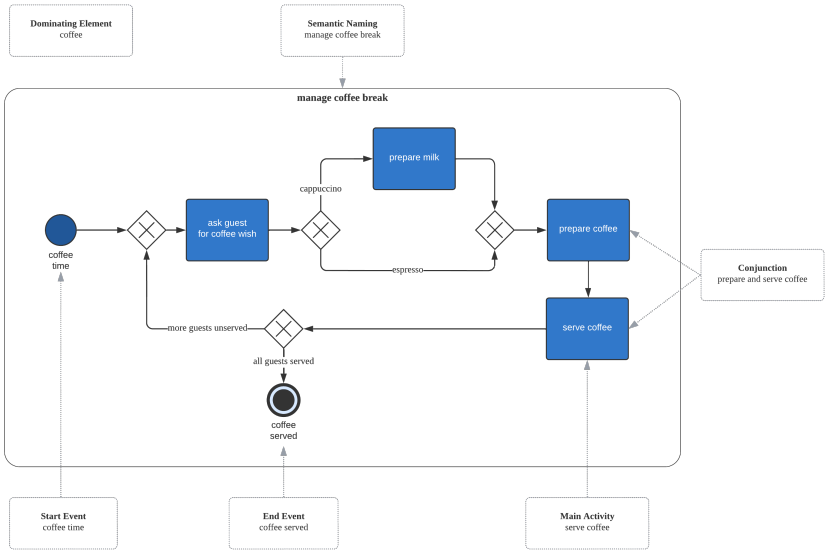


Figure 3.2: Example of Labeling Strategies from the Earlier Work

an example of conjunction of activities. The start event, coffee time, and end event, coffee served, define the overall goal. Lastly, the sub-process label "manage coffee break" exhibits a part-of relationship with activity labels within the sub-process, and therefore, the labeling uses the broader context of activity labels.

In the continuation work of the authors [8], they attentively studied the different theories of meaning and relate each theory to the labeling strategies. Different theories of meaning present relative strengths and weaknesses, and the labeling concepts they stand for directly or partially refer to different textual information captured in the process model labels. Hence, the authors conclude that in order to automatically generate labels for process models, it is necessary to adequately infer the information retrieved from the models. Based on the findings above, they designed and implemented the labeling approach based on the labeling concepts and natural language processing techniques. Although the approach of the authors is rather rule-based, the notions behind the rules resonate with the methods we apply. Furthermore, the main goal of both earlier work and this paper is to generate process labels that reflect the semantics of the model and optimize semantic closeness between a label proposal and the underlying process. Given the reasons stated above, we want to briefly discuss the authors' automatic approach to generate process labels.

As shown in Figure 3.3, the authors organized their method into 3 phases. Phase 1 serves as a preparation step, which automatically annotates all activities and events with their action and business object. Phase 2 represents the main step of the approach, consisting of a set of different techniques to generate label proposals. Finally, in Phase 3, the single best or the k-best label proposals are selected and transformed to the verb-object style in order to present an understandable and unambiguous label.

The label generation techniques from Phase 2 are the endeavors made to capture the semantics in the process model. The goal of Dominating Element Extraction is to identify whether the given process model includes a dominating action or a dominating business object, namely one action or business object has a higher occurrence than do all other elements. If the process model contains such a dominating term, it can be used as input for the Subordinate Element Extraction technique. Otherwise, the approach moves on to the Event Extraction and the Main Activity Extraction techniques, as they do not require the input of a dominating element.

If one type of dominant element was detected, the Subordinate Element Extraction technique identifies those actions or business objects with which the dominant element is connected in the given process model. All activities containing the dominating element are selected and the subordinate elements are derived. Then, two techniques Lexical Conjunction and Logical Conjunction are introduced to construct a process label based on the set of subordinate elements and the dominating element. In case of the Lexical Conjunction, it implies the replacement of the subordinate elements with a newly introduced term derived from the lexical relations among these elements. The authors employed two lexical relations: holonyms (a word representing the whole of a part-of relation) and hypernyms (a word with a broad meaning that more specific words fall under). In particular, the lexical database WordNet is consulted to identify common holonyms and hypernyms of the subordinate elements. In case of the Logical Conjunction, the subordinate elements are simply connected using the logical operators and or or. Another way to make use of the identified dominating elements is through Label Repository, built up using the activity labels of other process models. If a dominating element was identified with the Dominating Element Extraction technique, the repository can be consulted to find a corresponding element which is likely to be connected with the dominating element.

When the condition of having one dominating term is not met, the algorithm focuses directly on the Event Extraction and Main Activity Extraction techniques. The goal of the Event Extraction technique is the generation of process labels from start and end events as they potentially provide information about the model content. As for the Main Activity Extraction technique, it utilizes the insight of the authors' analysis that approximately 85% of the main activities are found either at

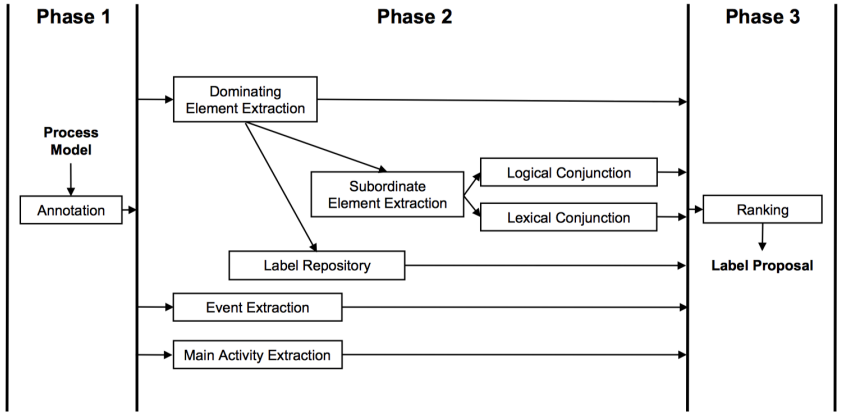


Figure 3: Overview of the Name Generation Approach

Figure 3.3: Rule-based Algorithm

the beginning or at the end of the process. Accordingly, the technique presumes the existence of a main activity in the first or last position and selects the according activity labels as process label proposals.

To conclude, the earlier work provided a basis of automatic generation not only for proposing labels of whole processes, but also for process fragments. The approach proposed by the authors is built on insights from theories of meaning and exploratory research to devise labeling strategies for process models. In addition, the techniques were validated through measuring the semantic closeness between the label proposals and process model defined in the original paper and a case study involving process modelers.

The labeling strategies capture numerous core aspects of the process, indicating the importance of specific parts in the process when it comes to labeling. Furthermore, they provide different reasons that can make a label suitable for the underlying process or process fragment. To be more specific, Main Activity may reflect the most important task of the process, Start or End Events may suggest the main outcome of the process, Conjunction of Activities and Semantic Naming can provide a good overview of the process. As for Dominating Element, whether it is a business object or action, should be intuitively included in the label. From information extraction's point of view, Main Activity and Start or End Events are extracted directly from particular parts of the process, while Conjunction of Activities and Semantic Naming offer more abstractive views of the whole process based on more than one core information exhibited in the process.

3.3 Machine Learning for Natural Language Processing

3.3.1 Natural Language Processing

Natural Language Processing (NLP), is nowadays broadly recognized as the study of interactions between natural language and machines. Specifically, the focus in recent years is how to automate the manipulation of natural language through statistical algorithms programmed in machines, which is known as Statistical NLP [9]. The goal is for the machines to process and extract meaning from human language in order to perform tasks such as Document Classification and Sentiment Analysis [refs]. In particular, Deep Learning, a subfield of Machine Learning, has attracted attention and achieved state-of-the-art results in specific NLP tasks that are considered challenging.

In the book, Neural Network Methods in Natural Language Processing, the author Goldberg (2017) defines NLP as, "Natural language processing (NLP) is a collective term referring to automatic computational processing of human languages. This includes both algorithms that take human-produced text as input, and algorithms that produce natural looking text as outputs" [10]. It delineates the use of deep learning neural networks in Statistical NLP to not only perform inference on specific tasks but also develop robust end-to-end systems. In the following subsection, we discuss how the approaches to NLP have evolved from rule-based to deep learning.

3.3.2 The Evolution of Approaches to Natural Language Processing

Real-world NLP applications are ubiquitous, ranging from simple spelling checks to complex chatbots. The approaches to process raw inputs evolve from heuristically applied rule-based methods to learnable representations using deep learning techniques.

Rule-based Methods Typical tools in the rule-based methods include Regular Expressions for pattern matching and Wordnet that stores semantic relations between words. Humans are heavily involved in designing practical methods that employ the tools based on their experience and expertise. The methods are not guaranteed to be optimal or rational but they are effective as short-term solutions.

Machine Learning Approach While rule-based methods are practical and accurate, they fail at non-deterministic scenarios. In addition, it is a time-consuming task for rule-based methods to detect and adapt to new trends appearing in data. This is where the statistical approach, Machine Learning, becomes active. Machine learning approaches gain insights from existing data by converting the textual form to numeric representation used as input for the model learning. The standard pro-

cess starts from data collection and preprocessing, feature engineering to model building and training. The trained model can be evaluated and deployed. Classic machine learning models include Naïve Bayes [11], Logistic Regression [12] and SVM [13].

Deep Learning Approach In recent years, Deep Learning has gained momentum and outperformed machine learning models in solving challenging artificial intelligence problems in the field of computer vision as well as natural language processing [14]. It is widely adopted for having several advantages. Besides working well with unstructured data, its feature generation is automatic, and, thus, it has self-learning capabilities. In other words, instead of specifying and extracting features from the processed data by linguists or machine learning experts, deep learning models are trained to learn feature representation from raw inputs in an automatic manner. Furthermore, deep learning models are able to exploit nonlinear relationships exhibited in the data and exceed classic linear models in performing complex tasks. However, it may require more data and for this reason, advancements brought by Transfer Learning are crucial.

3.3.3 Neural Language Models

Numerous prominent deep learning techniques in NLP are powered by Neural Language Models [15], including text embedding or dense token representation, text classification, machine translation, question answering, and text summarization. Neural language models are trained through a form of Language Modeling task, which is to predict a token or word in a document based on the previous or surrounding context. The general idea is that the models form a vector representation that encodes the context prior to or surrounding the token. Based on this representation, the models generate the probability distribution of the token conditioned on the context and output the prediction. The prediction itself is a classification task with the number of classes equal to the whole vocabulary set.

N-gram Language Models To illustrate the importance of neural language models, we briefly introduce the classic probabilistic language models, N-gram models [16], and their limitations. N-gram models decompose the probability of a token sequence into conditional probabilities of each token given previous context. Specifically, the models compute the probability of the Nth token conditioned on its history approximated by the N-1 sequence of tokens ahead. Practically speaking, it is the proportion of occurrences or counts of the Nth token following the N-1 token sequence. The limitations of the simple probabilistic models are apparent. Firstly, the models are built upon the probability of words co-occurring, and as N increases, the number of possible token combinations escalates and non-occurring N-grams create a sparsity problem. Secondly, the storage of N-grams leads to

large computation overhead in terms of memory capacity. Hence, N is restricted to a small number by both the issue of sparsity and limited computation power. Lastly, the performance of the simple language models is limited by the lack of model complexity.

Evolution of Neural Language Models To tackle the disadvantages of the N-gram models, neural language models have evolved through different model architectures that entail different breakthroughs. Earliest models such as Word2Vec [17], built in densely connected network structure, ameliorate the sparsity problem encountered in the N-gram models. Instead of storing the n-gram counts, models such as Word2Vec solve the problem by using embedding layers to encode the context tokens. The learned context embeddings facilitate the models to predict the probability distribution of the token. However, such densely connected networks remain the structure of fixed-size windows when it comes to processing the token sequence and each token is processed independently.

To be able to consider the entire history of a token and create dependency between tokens in a sequence, Recurrent Neural Network (RNN) [18] is modeled as a single data point or cell-based network that passes the internal state along the whole sequence. Such model structures iterate over tokens or the timesteps in the sequence through a training loop. When the sequence grows, RNN runs into the issue of long dependency. Therefore, LSTM [19], the next generation of RNN, solves the problem by introducing a gated mechanism that decides what information gets to be delivered to the next internal state in the cell network.

Another main drawback of RNN-based architectures is the long training time due to their sequential nature, and this is where Transformers comes into play. Instead of having the concept of an internal state, the transformers architecture features a self-attention mechanism that processes the whole input sequence at once, allowing parallelization in computation. Furthermore, when processing each token, the self-attention mechanism allows the model to determine which position or context token to attend in the sequence for effective encoding of the token. In simple terms, self-attention determines how tokens impact and associate with each other in the sequence. As a result, transformers outperforms RNN-based models through contextualized token embeddings and computation efficiency. Popular transformers-based language models include BERT [20] and GPT2 [21].

3.3.4 Text Summarization

The objective of text summarization is to extract useful knowledge out of a document for the users to reduce reading time and process information efficiently. Summarization task condenses a piece of text into a shorter version that preserves key informational components representing the core content of the original text.

Text summarization can be utilized to automatically produce short descriptions of a product, sum up or recap a news article, facilitate document classification, ranking and retrieval.

There are mainly two kinds of text summarization: Extractive [22] and Abstractive Summarization [23]. Extractive summarization, as the name suggests, extracts the top-ranked sentences from the original text to form a summary. Therefore, the information extracted has an identical counterpart in the original text. On the other hand, abstractive summarization generates a unique summary paraphrasing the core message of the original text based on the learned internal representation.

Extractive Summarization ranks the sentences in descending order and selects top k sentences that best represent a document with k as a user input. Well-known extractive summarizers such as LexRank and TextRank are unsupervised graph-based algorithms. The graphs are constructed based on co-occurrences or similarity measures between two words or sentences. Then, graph ranking models such as Google's PageRank algorithm can be applied to extract top-scoring key phrases or sentences. Besides the graph-based algorithms, neural language models such as BERT and GPT2 are also employed to perform the task. On the basis of sentence embeddings generated by the language models, there are several ways to conduct the summary extraction. One way is to apply sentence ranking as aforementioned by ordering the predicted probabilities and selecting the top k. Another is to apply clustering techniques and extract the centroid sentence of each cluster. In general, extractive summarization can be seen as binary classification problems where the objective is to identify the sentences belonging to the summary.

Abstractive Summarization reproduces the key information of the document in the model's own language. Besides detecting the most representative sentences, abstractive summarization forms its own understanding and generates the summary. Neural language model is a natural choice for the task due to its ability to produce semantic representations of the document as well as its generative nature. The employed models are in Sequence-to-Sequence form with the document as input and abstractive summary as output. Specifically, the transformers models such as T5 [24] and Pegasus [25] are developed and widely adopted for performance reasons. They typically follow an encoder-decoder architecture. The encoder is in charge of encoding the source document into a hidden representation for a compact message delivery, while the decoder is responsible for generating the summary as output based on the encoded representation.

Chapter 4

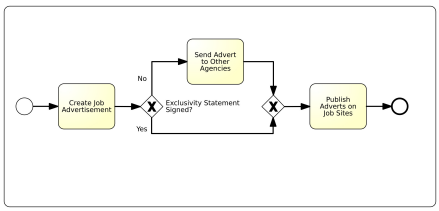
Labeling Approach

In the previous chapter Motivation, we provided numerous reasons regarding why we value Machine Learning as a solution for our task. High-level activity label generation resembles text summarization or title generation tasks in the field of NLP. By translating a group of low-level process steps into a piece of text, namely spelling out activity labels in sequence following the control flow, text summarization techniques, either extractive or abstractive approach, can be applied to achieve high-level representation.

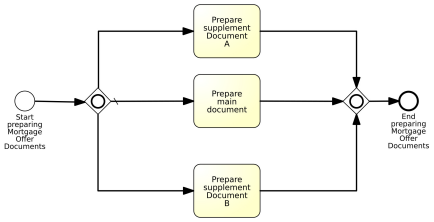
This chapter is structured as follows, we first introduce the data collection utilized in this thesis. Then, we outline the defined labeling strategies and labeling approach. The objectives of the labeling approach is to select and reflect the best suited labeling strategies. Lastly, we present the realization and optimization process of the labeling approach.

4.1 Data Collection

We collected the data from the Model Collection of the Business Process Management Academic Initiative (BPMAI) [26]. Specifically, we conducted all the experiments with the BPMN 2.0 process models provided in the collection as our input data. All models retrieved are used for self-supervised learning and the named sub-processes of the models are utilized as labeled data for supervised learning. There are several drawbacks regarding this model collection. The data quality issues such as wrong model configurations and misspellings affect process understanding, process text parsing and model training. In addition, there are only a small number of sub-processes available for supervised training. Nevertheless, it is the largest collection that offers abundant textual features and a decent level of content variety in process models.



(a) Example 1 of Outcome-based Labeling



(b) Example 2 of Outcome-based Labeling

Figure 4.1: Outcome-based Labeling Strategies

4.2 Labeling Strategies

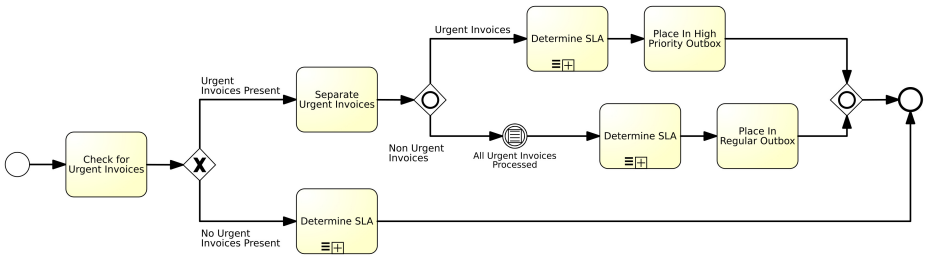
Three specific labeling strategies: outcome-oriented labeling, decision-based labeling and holonym-based labeling, are defined based on the behavioral analysis of the grouped process steps as well as the observations made from the earlier work on automatic label generation for process models.

4.2.1 Outcome-oriented Labeling Strategy

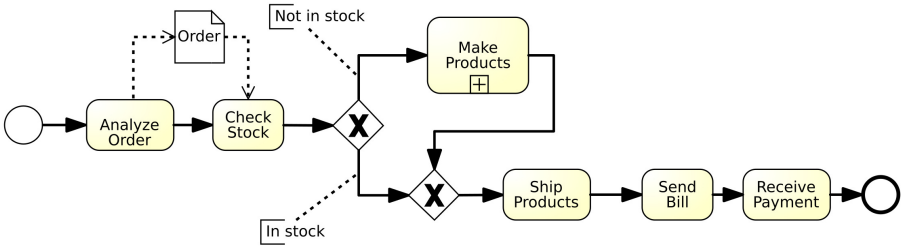
The outcome-oriented labeling strategy can be applied if the considered group of process steps has a clear result that can be used to describe the previous steps. An example in Figure 4.1a from the data collection is the “Publish Adverts on Job Sites” activity label for activities “Create Job Advertisement”, “Send Advert to Other Agencies” in the previous steps. Another example in Figure 4.1b shows that this time start/end event label instead of end activity label reflects the outcome, which is about “preparing Mortgage Offer Documents”.

4.2.2 Decision-based Labeling Strategy

The decision-based labeling strategy builds on the fact that many processes are driven by key decisions being made, such as “Decide about customer offer” activity that decides to reject or accept an offer from a customer. Broadly speaking, central split-off points in the processes can also stand for such decisions. For example in Figure 4.2a, after “Check for Urgent Invoices” activity, the process is split into different work streams depending on the urgency of the invoices. If a considered group of process steps encompasses such a decision point, this can be used as the main information captured in a label.



(a) Decision-based Labeling Strategy



(b) Holonym-based Labeling Strategy

Figure 4.2: Decision-based and Holonym-based Labeling Strategy

4.2.3 Holonym-based Labeling Strategy

Finally, the holonym-based labeling strategy is applied to situations where low-level process steps jointly contribute to a higher-level activity. A holonym is a term in linguistics that has a part-of relation with a number of meronyms, e.g., a “finger” is a meronym of the holonym “hand”. By applying the notion of holonym to process steps, we aim to recognize actions in combination with business objects that follow the logic of the meronyms, allowing us to detect higher-level activity that stands for the broader context. For example in Figure 4.2b, “Fulfill Order” presents a more general term placed on top of its low-level activities, “Analyze Order”, “Check Stock”, “Make Products”, “Ship Products”, “Send Bill”, and “Receive Payment”.

4.3 Labeling Approach

Judging from the perspective of process information retrieval, we categorized the labeling strategies into two distinct methods, namely extractive and abstractive

methods, in order to output the ideal labels driven by each strategy. We made use of machine learning algorithms to not only implement the two methods but also select the labeling strategies to achieve the final results, high-level activity labels.

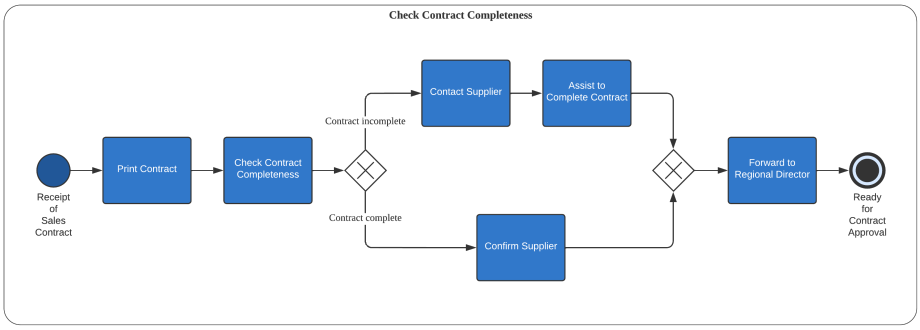
4.3.1 Extractive and Abstractive Methods

Given a group of low-level process steps, which may contain information including actions, business objects, and application contexts, our objective is to represent them with either an extractive or abstractive method. The extractive method discerns and outputs the important process step representing the outcome or decision point that stands for the primary information of the group. As a result, labeling strategies outcome-oriented labeling and decision-based labeling fall into this category as they target specific process steps in the group and purely extract them as labels. In contrast to the extractive method, the abstractive method is utilized to provide a holistic view of information gathered from more than one equally important process steps within the group. Naturally, labeling strategy holonym-based labeling belongs to this category as it aims to provide labels that exemplify the broader context of the group.

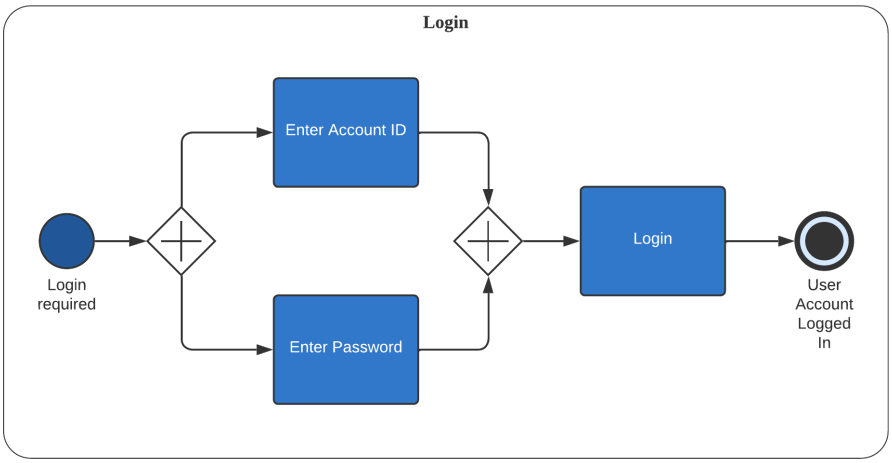
4.3.2 Labeling Strategy Selection

Different labeling strategies can result in different labels for the same group of process steps. Depending on the nature of each group, a labeling strategy should be chosen to generate the most representative abstraction. Therefore, aside from realizing the labeling strategies, a decision mechanism that selects the most appropriate strategy for a specific group is equally important to be developed.

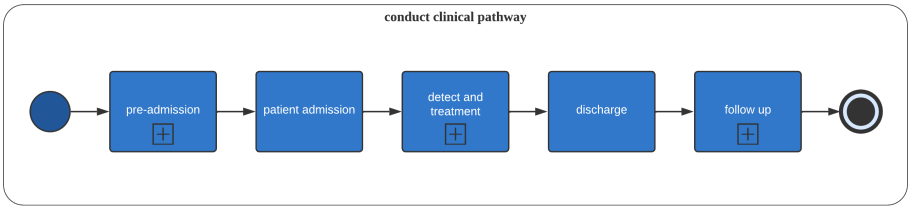
Through proper training, Machine Learning algorithms can select strategy through learning behavioral as well as semantic relations of the low-level process steps. Based on different training objectives, the algorithm is trained to pick up the different signals exhibited in the processes. For instance, by noticing when low-level actions have clearly opposite meanings such as reject versus accept, or through detecting mutually exclusive events followed by a decision point in the process, the algorithm decides to employ decision-based strategy. In Figure 4.3a, “Check Contract Completeness” is a clear demonstration of the activity followed by mutually exclusive process streams. Similarly, the outcome-oriented strategy shall be employed if the group of events represents a synchronization point that concludes the paths in front, such as a login step in Figure 4.3b. Finally, the holonym-based strategy shall be reserved for situations where such holonym relations can be identified for the low-level process steps, such as the example of “conduct clinical pathway” provided in Figure 4.3c.



(a) Example 1 of Labeling Strategy Selection



(b) Example 2 of Labeling Strategy Selection



(c) Example 3 of Labeling Strategy Selection

Figure 4.3: Labeling Strategy Selection

4.4 Realization Process of Labeling Approach

We accomplished the main goal of the labeling approach through realizing subgoals in stages. The main goal is to generate representative high-level activity labels as output with groups of process steps as input, and text summarization techniques are the fundamental approach. The first subgoal is to simply generate representative summary from text using existing tools or off-the-shelf methods, which is a process-agnostic stage. Next, we defined the ideal form of process summary, namely high-level activity label, and expected shape of process text. Under the predefined process input and output, the second subgoal is to generate high-level activity labels from process text through adopting the existing approach into a domain specific approach, which is a process-specific stage. Lastly, based on the observation made from the second stage, the third subgoal is to improve the performance by optimizing the domain specific approach. Through executing the tasks associated with each subgoal, we recognized and tackled the remaining challenges in the next stage.

4.4.1 Process-agnostic Stage

Off-the-shelf Methods

With the goal of producing high-level activity labels through process summarization in mind, we theorized two primary hypotheses that assume representative labels of the underlying processes can be produced through different text summarization techniques, namely the extractive and abstractive approach.

Hypothesis 1 *The extractive approach extracts the sentences that are important in the text as summaries. Therefore, key activities that represent the outcome or important decision points in the group of process steps are extracted as labels through the approach.*

Hypothesis 2 *The abstractive approach paraphrases important messages exhibited in the text as summaries. Therefore, a more abstract and holistic view of information gathered from the group of process steps is provided to produce labels through the approach.*

We conducted a preliminary experiment to test the two primary hypotheses with off-the-shelf methods. Thanks to the AI and NLP communities, open-source libraries such as the HuggingFace transformers package [27] provide pre-trained models that are useful for a variety of NLP tasks, including text summarization. These pre-trained models can be employed directly off-the-shelf to produce results

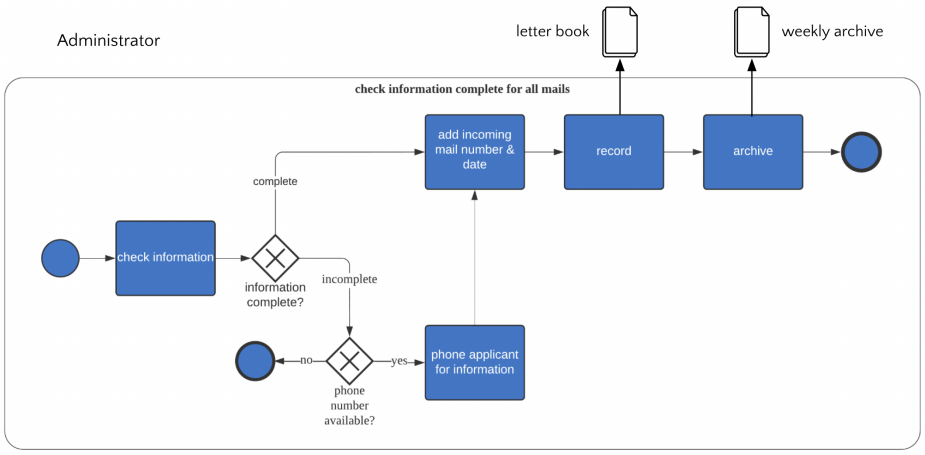
based on the input data. Through utilizing these models, namely the summarizers, we obtained some insights into our assumptions.

To examine H1, extractive summarizers BERT and GPT2 were employed, while to examine H2, abstractive summarizers Pegasus and T5 were applied. In order to generate a smoother process text as input data for the summarizers, we utilized all available information provided by BPMN models, including resources, data objects, gateway and edge labels. The rationale behind this is that the input data should be as similar as the original training data of the summarizers, which is the usual syntactically structured paragraphs. We provide two examples below to explain how we conducted the experiment and discuss the observations and initial conclusion of the labeling approach made through this experiment.

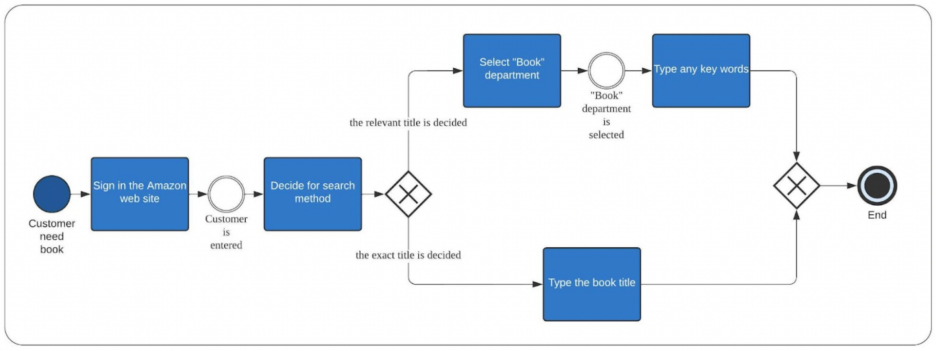
Before diving in the examples, we briefly discuss the parsing logic used to generate the process text. We interpreted gateways and their following process streams depending on their characteristic and whether the labels of gateways and following edges exist. For instance, the work streams that follow exclusive gateways are translated into "if", "if-else", or "either-or" statements. The rest of the labels were parsed in the original form even if they might contain misspellings or be in informal language structure. Punctuation marks were added after each parsed label. Based on the process elements of the labels and whether it follows the gateway statements, different punctuation marks were placed. For instance, the activity "add incoming mail number & date" in the example 1, Figure 4.4a, is followed by a comma instead of a period as it is in the gateway statement.

Example 1

- **Process Text:**
  "Administrator check information. Is information complete? If complete, add incoming mail number & date, record to letter book, archive to weekly archive. If incomplete, is phone number available? If yes, phone applicant for information."
- **Summaries of Extractive Summarizers:**
  - **BERT** "Administrator check information. If yes, phone applicant for information."
  - **GPT2** "Administrator check information. If incomplete, is phone number available?"
- **Summaries of Abstractive Summarizers:**
  - **Pegasus** "All information is copyrighted."
  - **T5** "administrator check information. is information complete?"



(a) Example 1 of Off-the-self Methods



(b) Example 2 of Off-the-self Methods

Figure 4.4: Off-the-self Methods

Following the logic described above, process text was produced and fed into the summarizers. In example 1, Figure 4.4a, longer summaries were produced by extractive summarizers. From the content perspective, we would argue that the GPT2 summarizer performed the best as it extracted the activity that stands for the decision point in the process "Administrator check information." and provided the solution of incomplete cases "If incomplete, is phone number available?". BERT, GPT2, and T5 summarizers captured relevant content as the summaries, while Pegasus summarizer produced rather irrelevant, abstract but more concise summary or label.

Example 2

- **Process Text:**
  "Customer need book. Sign in the Amazon web site. Customer is entered. Decide for search method. Either the relevant title is decided, select "Book" department, "Book" department is selected, Type any keywords, or the exact title is decided, Type the book title."
- **Summaries of Extractive Summarizers:**
  - **BERT** "Customer need book."
  - **GPT2** "Customer need book. Decide for search method."
- **Summaries of Abstractive Summarizers:**
  - **Pegasus** "How to find a book on Amazon?"
  - **T5** "customer need book. Sign in the amazon web"

Again following the text parsing logic, process text was produced and fed into the summarizers. For example 2 in Figure 4.4b, we would once more argue that the GPT2 summarizer performed the best as it extracted the start event "Customer need book." that triggers the whole process and the activity "Decide for search method." that stands for the decision point in the process. All summarizers captured relevant content as the summaries in this case. Most of them centered around the start event, while Pegasus summarizer again produced a more abstract and concise summary or label.

In the light of this experiment, we are able to gain some initial insights into our labeling approach and draw conclusions concerning the two primary hypotheses, H1 and H2. We observed that the extractive summarizer GPT2 outperformed the rest in terms of capturing important tasks within the process. Nevertheless, it is restricted to output only what it has seen in the process text and tends to generate longer summaries that appear to be unsuitable for high-level activity labels.

Through this observation, we can confirm our assumption in H1 to be correct despite the results not yet in the ideal form for labels. On the other hand, abstractive summarizer Pegasus produced content that is less relevant to the underlying process, yet it is able to produce summaries that are comparably abstract and concise. By further optimizing Pegasus with our domain-specific process data, there is a high chance that it can generate summaries that are representative, such as the subprocess name of example 1, "check information complete for all mails". Nevertheless, the off-the-shelf abstractive summarizers can not yet output summaries that are both abstract and appropriate for activity labels at this point, and therefore, we can not yet verify our assumption in H2 with high confidence. In the later subsection Process-specific Stage, we further explain how we utilize the insights to develop a domain-targeted training method to achieve our goal and verify both H1 and H2 assumptions.

Definition of Process Summary and Process Text

Before going into details about how we can improve our labeling approach to achieve the goal, we need to discuss our general expectations towards activity labels and process text. First and foremost, the labels are expected to reflect one of the three labeling strategies in order to represent the groups of process steps. However, we expect the activity labels to be not only precise in content but also short in length. To realize that, the core information of the underlying process has to be either extracted in a shorter form or abstracted into a concise and high-level manner. As for the process text, instead of using the rule-based parsing logic shown in the previous experiment, where lots of exceptions have to be dealt with, we proposed to parse the process in the simplest automated fashion. Following the control flow, we parsed only the labels of events and activities and converted all parsed labels into lower-case for normalization purposes (normalization was applied after obtaining benchmark results). For example in Figure 4.5, the parsed process text consists of all activity labels extracted in order: "sort invoice per vendor, sort invoice per amount, place in batch $< \$2500$, place in batch $\geq \$2500$".

It's self-evident that the process text with newly defined parsing logic is far from the usual syntactically structured paragraph. However, this simple parsing strategy is straightforward in implementation and it allows us to easily generalize the input type to both event logs and process models.

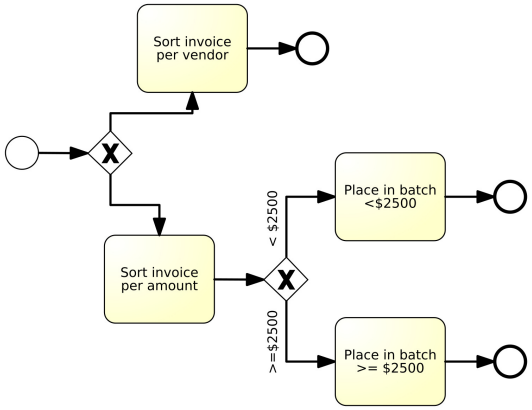


Figure 4.5: Example of Process Text Definition

4.4.2 Process-specific Stage

Domain-Targeted Training for Labeling Optimization

As we expected to generate high-level activity labels that are representative, short and precise while adapting to the newly defined parsing logic for process text, we needed to develop a method that is specific to our process domain. As a result, we suggested fine-tuning a selected summarizer as our next step for optimization. Pegasus was chosen to achieve representative domain-specific abstraction. Besides the characteristics we mentioned in the earlier experiment, Pegasus comes with several other good qualities described below. Furthermore, abstractive summarizers can be trained or fine-tuned to output both extractive and abstractive results that stand for three labeling strategies as shown in Evaluation chapter.

Pegasus (Pre-training with Extracted Gap-sentences for Abstractive Summarization) [25] is a sequence-to-sequence summarisation model and it utilized Gap-Sentences Generation (GSG) as self-supervised learning. The GSG technique masks not only the random tokens but also important sentences from the input document for the encoder and reproduces them at the decoder output. The decoder in this case acts as an autoregressive model. The GSG technique functions similar to an extractive summarization and it works well as a pre-training objective for the downstream summarization tasks. Furthermore, after pre-training through GSG, Pegasus performs well on low-resource summarization tasks (around 1000 examples).

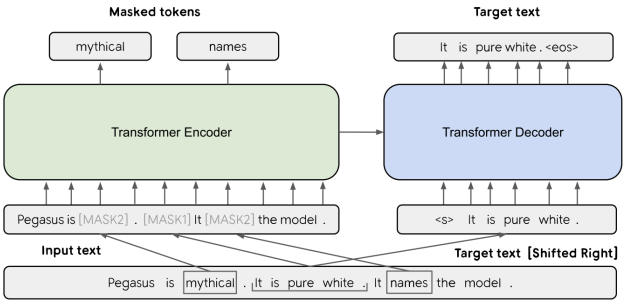


Figure 4.6: Pegasus

Benchmark Model

In order to accommodate our limited labeled data while utilizing all available data, two-step training on pre-trained Pegasus was proposed to produce domain-specific benchmark results. Two-step training aims to first pre-train the model further with self-supervised learning then fine-tune the model with supervised learning.

We further pre-trained the model with the same pre-training technique, namely GSG, using the whole process text collection parsed from BPMN 2.0 process models. Important sentences should be masked to carry out the self-supervised learning. In our case, important sentences are the essential process steps in the process models. Based on the empirical analysis and previous work on labeling generation, we could get a few hints on where the important process steps naturally lie within the process models. Oftentimes, the task in front of the gateway stands for the decision point of the process, start event or the first task usually stands for the state or action that triggers the beginning of the process, and end event or the end task could mean the state or action that concludes the process or contain overall information of the process. We considered all process steps mentioned above to contain important information with a higher probability. Therefore, we masked them automatically based on their position in the process model. During the self-supervised training, the model was given the masked process text as input and expected to reproduce the masked sentences at output. We split the data into 1000 training examples and around 100 test examples for validation.

Once pre-training was done, we fine-tuned the model for the summarization downstream task. To carry out this step, labeled data is needed for supervised learning. The sub-processes existing within the BPMN 2.0 process model collection are retrieved as labeled data since they usually come with sub-process names that represent the underlying processes. However, the amount of sub-processes we

could get is very limited and it is also one of the main reasons why we decided for a 2-step training. There are only around 120 sub-processes in total and we again split the data into 100 training examples and around 20 test examples for validation. During the supervised training, the model was given the complete sub-process, a group of low-level activities, as source and sub-process name, the human created label, as target.

We expected the model to generate summaries that have the characteristics of high-level activity labels after this domain-targeted two-step training. Besides outputting labels in an ideal form for the processes, the model was also trained to decide whether to use an extractive or abstractive approach through the labeled training examples. In other words, the labeled data consists of both extractive and abstractive labels that stand for three labeling strategies.

Benchmark Examples

We validated our training method introduced above using the test set from both steps of the training. Two examples are given below to illustrate the results of step-1, and following that, another three examples are presented for step-2. In the end, we discuss the findings of the intermediate results generated from the benchmark model and the challenges faced in this stage.

Example 1 of step-1: Task masked in front of the gateway

- **Process Text:**
  "Apple ID Details Required, [mask_1], Date of Birth, Email, Password, Verify Password, Security Question, Answer, Details Entered"
- **Model Result:** "Enter Required Information"
- **Human Label:** "Enter Apple ID Details"

First example of step-1 results is demonstrated, Figure 4.7. As shown in the process text, the second sentence was masked, and in this case, it is the task in front of the parallel gateway that functions as the split-off point in the process and naturally includes important information. When it comes to masking, we tried to avoid situations where two consecutive sentences are masked and kept the masking ratio between 10% to 20% as empirical study might suggest. In this example, labels of the start event and task in front of the gateway both exist and in consecutive order. Our masking scheme prefers to mask the decision points or split-off points over the rest as they cover important information more consistently in our data collection judging by experience. After the first-step training, the model output "Enter

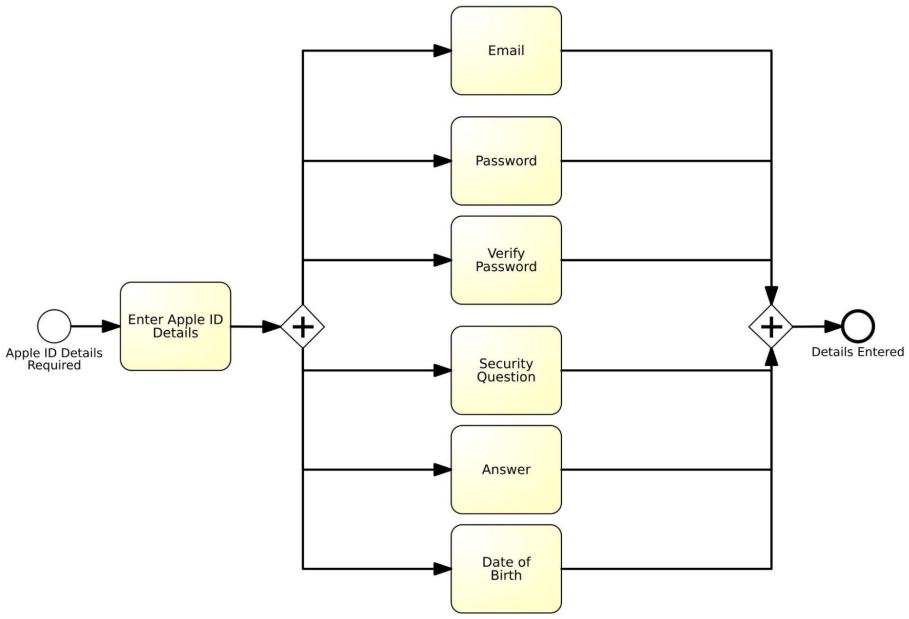


Figure 4.7: Example 1 of step-1 in Benchmark Model

Required Information" that echoes the start event "Apple ID Details Required" and fits well with the context, however, one can argue that compared with the original label "Enter Apple ID Details", the model result is less precise contextually.

Example 2 of step-1: First task masked (also in front of the gateway)

- **Process Text:**
  "[mask_1], Select one way, Select return, Select departure city, Select destination city, Select class, Select seat, Enter type and No of Pax, Continue to flight selection"
- **Model Result:** "Select flight"
- **Human Label:** "Identify trip requirements"

Second example of step-1, Figure 4.8, is masked with the first sentence, which is the first task and at the same time the task in front of the gateway in the process. The model output "Select flight" that appears to fit well with the context and can stand for the action that triggers the rest of the process, however, it is not the best suited label when the end task "Continue to flight selection" hints that the actual

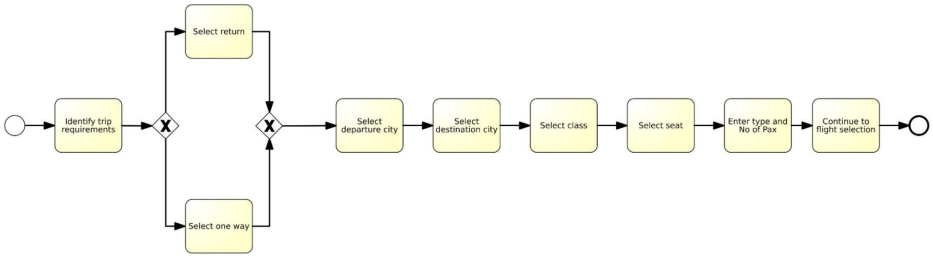


Figure 4.8: Example 2 of step-1 in Benchmark Model

process of flight selection comes after the current process. Therefore, the original label "Identify trip requirements" demonstrates a more appropriate option.

Although not all test examples in step-1 are with well-behaved results, the output labels serve the purposes of summarizing or echoing the surrounding process steps in general. The act of reproducing important sentences or replicating essential messages lies within the processes is what we are after at this training step.

Example 1 of step-2: Provide good overview of the process

- **Process Text:**
  "Symptoms of heart failure, Confirm diagnosis, Monitor patient, Refer for evaluation for heart transplant"
- **Model Result:** "Assess patient"
- **Human Label:** "General guideline for evaluation and care of patients with heart failure"

The first example of step-2, Figure 4.9, demonstrates utilizing holonym-based labeling strategy in order to provide the most suitable label for this sub-process. This labeling strategy offers a good overview over the group of low-level process steps shown in the process text. One thing to note here is that our parsing algorithm unfortunately cannot adapt to all possible scenarios in the data files. Hence, it is usual that not all labels can be successfully parsed as shown in this example. Nevertheless, the model proves itself to be robust to the incomplete data or in general data quality issues, which is discussed in more detail in Evaluation Section. The model output "Assess patient" which concludes the underlying process quite well despite not having all labels available. On the other hand, the original sub-process name is more specific into the medical domain and much longer in length.

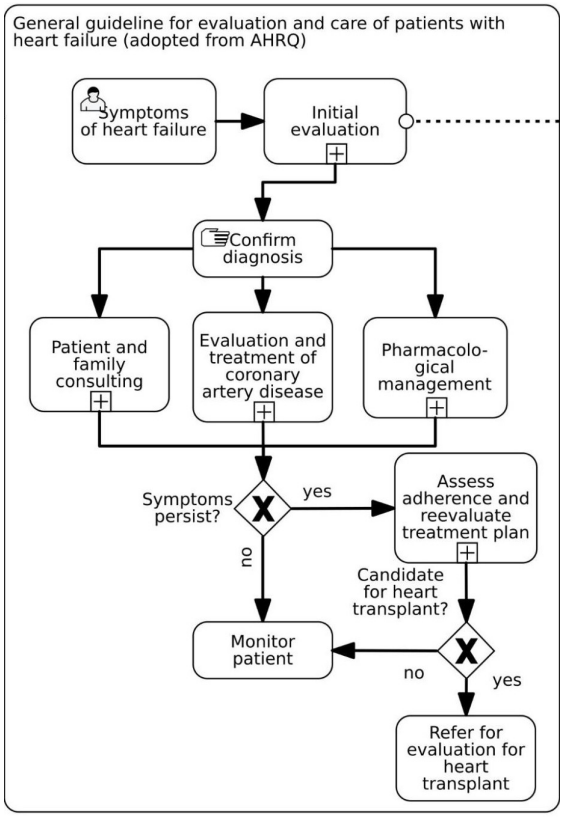


Figure 4.9: Example 1 of step-2 in Benchmark Model

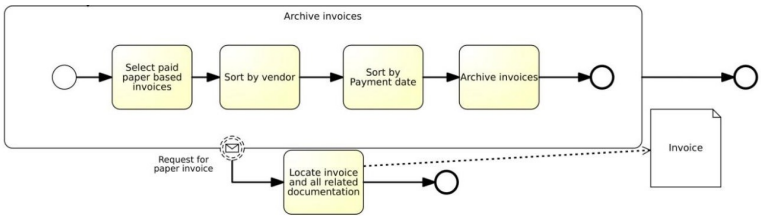


Figure 4.10: Example 2 of step-2 in Benchmark Model

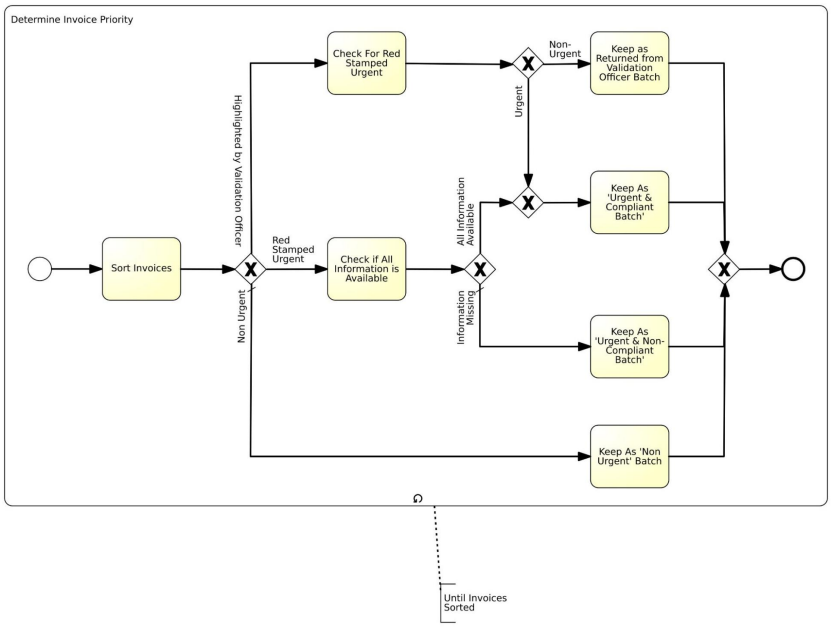


Figure 4.11: Example 3 of step-2 in Benchmark Model

Example 2 of step-2: Main outcome of the process

- **Process Text:**
  "Select paid paper based invoices, Sort by vendor, Sort by Payment date, Archive invoices"
- **Model Result:** "Invoices validation"
- **Human Label:** "Archive invoices"

The second example of step-2, Figure 4.10, shows a simpler sub-process with the main focus on the outcome of process "Archive invoices", which is also the original sub-process name. This example is the typical scenario in outcome-oriented labeling strategy featuring the end task as the outcome. Our model output "Invoices validation" which correctly included the key object invoices but incorrectly extrapolated the main action validation. Since a tiny dataset could easily introduce selection bias [28], it is not unexpected that the model overfit to the bias in the training set. In this example, the training set might contain most invoice-related processes emphasizing the action validation.

Example 3 of step-2: High-level abstraction of the most important task

- **Process Text:**
  "Sort Invoices, Check if All Information is Available, Keep As 'Urgent & Compliant Batch', Keep As 'Urgent & Non-Compliant Batch', Keep As 'Non Urgent' Batch, Check For Red Stamped Urgent, Keep as Returned from Validation Officer Batch"
- **Model Result:** "Batch Validation"
- **Human Label:** "Determine Invoice Priority"

The third example of step-2, Figure 4.11, features the combination of holonym-based and decision-based labeling strategies surrounding the main decision point of the process "Sort Invoices". The original sub-process name "Determine Invoice Priority" high-level represents the information gathered from the decision point and the following parallel work streams around the urgency of the invoice. The model output "Batch Validation" which seems rather unrelated. Although batch appears a few times in the process text, it is not the key object of the process. Similarly, validation with its one-time appearance in the process text should not be the focal point of the process and hence the situation exhibited in the last example also exists here. In this case, the model missed to pick up the information presented in the central split-off point and its following work streams.

Overall, the benchmark model presents decent performance in some cases such as the first example of step-2 . It has shown the ability to generate proper process-label-like summaries that give high-level abstractions of the underlying processes. However, it is obvious that there were some imminent problems that needed to be addressed as shown in the second and third examples of step-2. One of them is the limited data issue. Besides having a small amount of labeled data, its data quality is also very poor. To deal with it, manual labeling is inevitable. Another potential issue is that the behavioral information following the control flow of the underlying process is not captured by the process text. In other words, not all process steps are in directly-follows or (undirected) directly-associated relations even when they appear so in the process text. For example in , process steps "Keep As 'Non Urgent' Batch" and "Check For Red Stamped Urgent" are in consecutive order in the process text even though they are in parallel relations following an exclusive gateway in the process. We classified the relations between a group of the process steps into three types: directly-follows or directly-associated relation, parallel relation, and no-direct-association relation.

In the following subsection, we explain how we addressed the issues mentioned above through implementing manual labeling, data augmentation and control-flow relation learning.

4.4.3 Optimization Stage

Manual Labeling

In order to address the limited labeled data, we decided to conduct manual labeling. To reduce the labeling bias introduced by us, we chose to utilize smaller process models that are attached with usable process names. Similar to sub-processes, process models can also be seen as groups of low-level process steps closely related to each other. The size of the chosen process models ranges from 3 to around 20 process steps, so that the model can learn to offer high-level abstractions for a decent range of processes. By filtering out the process models with unfitting process names, we obtained a collection of around 450 process models. Since most process names are used as file names such as "exercise_123_process_name", they cannot be properly used without manual transformation since there were no obvious patterns in the file naming. To ensure the quality of the labels, we also applied the verb-object labeling style to each of them. As mentioned in the Background section, verb-object labeling style improves process understandability for users and is suggested to be applied consistently for the process modelers.

Data Augmentation

Apart from the rather time-consuming manual labeling, we also employed data augmentation techniques to further increase the sample size. It is proven to be particularly useful in low-resource scenarios [29]. Data augmentation techniques can come in handy for situations where only limited data can be accessed, especially in certain domain areas, including ours. However, not all techniques are suitable for augmenting process labels as the content should be fact-driven and controlled by the underlying processes. As a result, easy text augmentation techniques such as random insertions and deletions are not ideal even when they are frequently used for tasks such as text classification [29]. Due to the reasons stated above, round-trip translation technique was selected for our purposes.

Round-trip translation [30] paraphrases the data by translating it to another language and back to the original language to generate augmented data. We performed the task by exploiting the pre-trained English-German and German-English translation models. In text summarization, either both text and summary are augmented or only text is augmented and summary is kept as original. Although previous work has found that the former approach creates diverse paraphrases for augmentation while preserving semantic meaning [31], there are experiments conducted with both approaches. Therefore, we decided to experiment with both variations. For each data sample, we translated it into German and back to English with top 10 beam hypotheses from beam search as output. As a result, we obtained 10 aug-

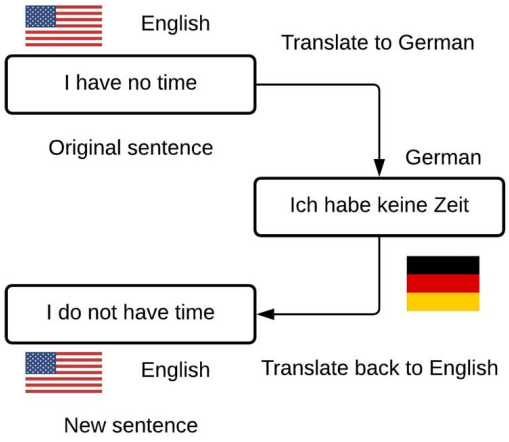


Figure 4.12: Round-trip Translation [1]

mented data for each data sample. While the amount of data was largely boosted, the verb-object labeling style was compromised in scenarios where summary was also augmented. Nevertheless, this effect provides generalization and fits in the reality as the labels are not restricted in verb-object form.

Control-Flow Relation Learning

As the behavioral information of control flow typically exists in the processes, it is crucial for us to incorporate it and observe its potential impact regarding labeling. In order for the summarizer to learn this behavior that exists implicitly in the process text, we assigned an additional training objective as a second pre-training task to Pegasus, which is entailment-relation learning. Textual entailment which is also known as Natural Language Inference (NLI) [32] is a task of determining whether the given "hypothesis" and "premise" logically follow (entailment), unfollow (contradiction), or are undetermined (neutral) to each other. That is to say, the hypotheses are classified into three classes based on their respective premise: entailment, contradiction, and neutral. In our case, the three entailment relations roughly represent directly-follows or directly-associated, parallel, and no-direct-association relations between process steps.

This idea was inspired by the paper KEPLER [33], where the authors aimed to incorporate the factual relations in the Knowledge Graph into the pre-trained Language Models such as BERT and RoBERTa. Besides the original training objec-

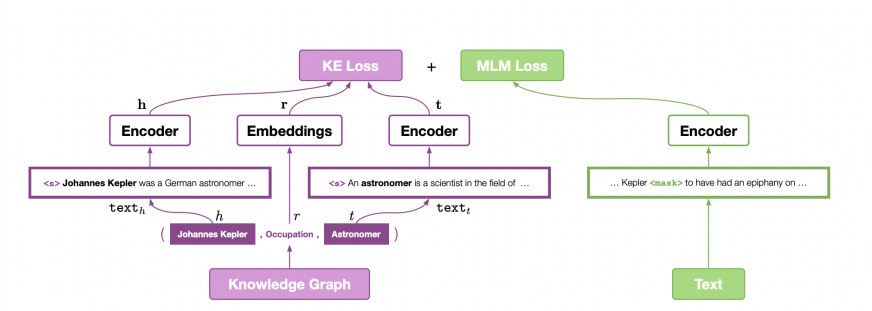


Figure 4.13: KELPLER

tive, which is often achieved by minimizing the loss function of masked language modeling (MLM), the language model is asked to minimize the loss function of knowledge embedding (KE) simultaneously. Through this joint training approach, the language model is encouraged to learn different factual relations between text entities and maintain the language understanding as training objectives. The authors stated that the KEPLER model can better understand fact-related text and extract knowledge from it while storing factual knowledge.

After seeing the successful example, we decided to apply similar training logic to Pegasus. We adapted contrastive learning to detect process entailment, more specifically, the widely used Triplet Margin Loss [34] was implemented. As the name suggests, triplet margin loss consists of three distinct points, namely anchor, positive and negative samples. In simple terms, the loss function encourages the distance between dissimilar pairs, the anchor-negative pairs, to be larger than any similar pairs, the anchor-positive pairs, by a user-defined margin value. The distance can be calculated by Cosine distance, Euclidean distance or any other distance metric.

To learn process entailment, we define dissimilar as either a parallel or no-direct-association relation and similar as directly-follows or directly-associated relation between any two process steps. One of the crucial steps to make triplet margin loss work well in practice is Hard Negative Sampling [35]. This sampling strategy means at each training step, we sample triplets that our network fails to discriminate or is not able to discriminate with high confidence as model input. To illustrate, we look at the process in Figure 4.14 where the parsed process text is “Check Application for Basic Completeness, Sort Applications, Disregard Application, Add to Rejection Notification Que”. An easy set of positive and negative pairs for the models would be to set the directly-follows process steps, “Check Application for Basic Completeness” and “Disregard Application”, as the anchor-positive pair and no-direct-association process steps, “Check Application for Basic

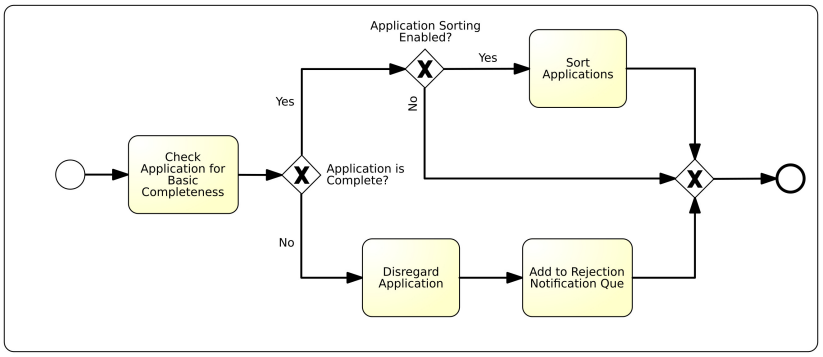


Figure 4.14: Example used for Triplets

Completeness" and "Add to Rejection Notification Que", as the anchor-negative pair. On the other hand, a hard negative set would be to set the (undirected) directly-associated process steps, "Sort Applications" and "Check Application for Basic Completeness", as the anchor-positive pair, and parallel process steps, "Sort Applications" and "Disregard Application", as a hard anchor-negative pair. The negative pair is considered hard since it is harder for the models to find dissimilarity between the actions Sort and Disregard with respect to the object Application. Furthermore, it appears in the parsed process text as directly-follows relation instead of parallel relation.

In essence, contrastive learning was incorporated into Pegasus as an additional training objective besides Gap-Sentences Generation (GSG) or important sentence masking. We named this model version as Pegasus-TML.

In total, there are 6 different models trained to conduct high-level activity labeling tasks. All 6 models were built upon the 2-step training strategy as foundation and trained with new labeled data obtained through manual labeling. Pegasus and Pegasus-TML are the base models of 4 others trained with two different augmented strategies, with or without summary augmented. Pegasus-Aug and Pegasus-TML-Aug are with both text and summary augmented, while Pegasus-Aug-Doc and Pegasus-TML-Aug-Doc are with only text or document (Doc) augmented.

Chapter 5

Evaluation

This chapter consists of three main sections: evaluation setup, evaluation results, and in-depth analysis and discussion. In the evaluation setup, the train-test split is utilized to create train and test sets, and the split is examined to ensure the validity of the evaluation, and the full evaluation is made up of automatic evaluation and human evaluation. In the section of evaluation results, the model performance results of automatic evaluation and human evaluation are presented. Finally, in-depth case study and discussion on evaluation are provided in the last section.

5.1 Evaluation Setup

5.1.1 Performance Evaluation through Train-Test Split

Train-test split is fundamental in evaluating the performance of prediction-based algorithms. A random split of 80/20 resulted in having 90 test examples out of a total of 450 after the manual labeling. During the process of manual labeling, we were aware that our process model collection contains a wide range of diversity in terms of process content and model complexity. To ensure a valid evaluation, we conducted a few experiments to validate that the train and test set follow the same distribution with respect to the two aspects.

Process Content Distribution

One common way to explore the content distributions of the train and test sets is through visualizing the document embeddings of their process text. Doc2Vec [36] is a popular algorithm that produces numeric representations for a collection of documents, and in our case, the document is the process text. By visualizing the embeddings with the dimension reduction technique TSNE [37], it allows us to

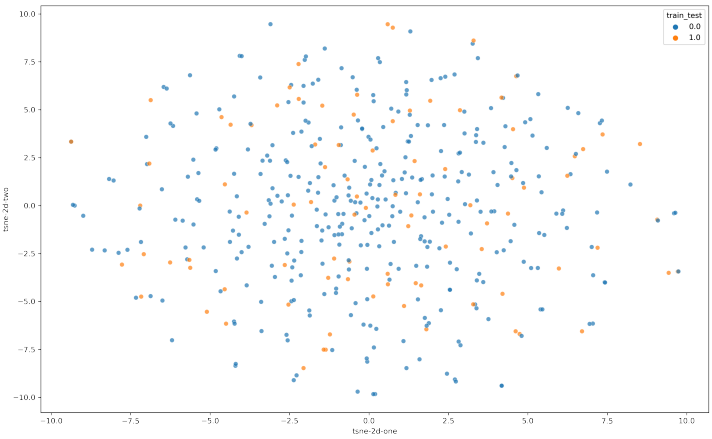
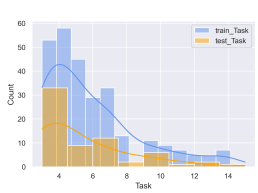
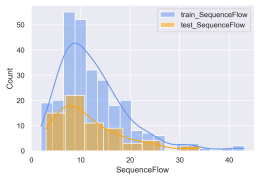


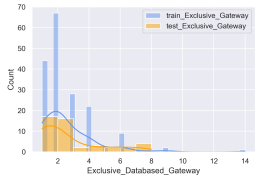
Figure 5.1: Process Content Distribution



(a) Task Distribution



(b) Sequence Flow Distribution



(c) Exclusive Gateway Distribution

Figure 5.2: Process Model Complexity Distribution

check the distribution of our data collection. From the Figure 5.1, we can see that there are no obvious clusters where similar process texts gathered, which demonstrates the diverse content of the data collection. Furthermore, the train and test sets are equally distributed in the space and hence, we can confirm that there is no imbalanced distribution of process samples regarding the content, including the corner-case samples.

Process Model Complexity Distribution

To study the data distribution of process model complexity, there are few aspects to consider such as the number of tasks or activities, sequence flows, and gateways [38]. We can simply examine the train and test set distributions of these measure-

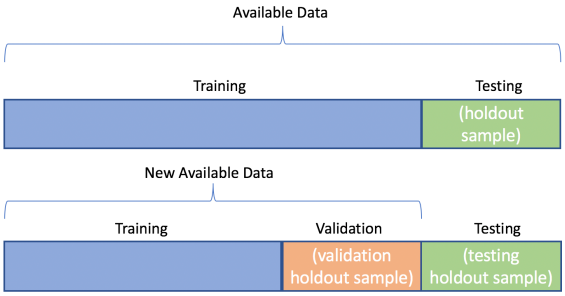


Figure 5.3: Train-Test-Validation Split from Stack Exchange

ments by plotting them through histograms. In Figure 5.2, we can see that the histogram distributions of train and test sets roughly follow the same curvelines for all three measurements. As a result, we can assume that the train and test sets share the same distribution concerning process model complexity.

Model Validation during Training Process

Other than using a test set for the final evaluation, model performance can also be monitored during training with a holdout validation set from the train set. To prevent the model overfit to the train set, we applied early stopping in the training process. Early stopping [39] approach terminates the training loop once the model performance drops consistently on the validation set. This is a simple but effective method widely used to avoid poor performance on the test set and improve the generalization of deep neural networks. The use of early stopping requires the selection of a performance metric to monitor and function as a trigger to stop training. We explain the selected metric BERTScore in the following subsection, Automatic Evaluation Metrics. In our case, the validation set was split from the train set again with 80/20 ratio after the first train-test split, Figure 5.3. This resulted in a validation set of 72 examples and a final train set of 288 examples.

5.1.2 Automatic Evaluation

Motivation

Automatically applied evaluation metrics are often used to be an indicator of quantitative model assessment in terms of performance. By measuring the similarities between the model generated results and the references created by humans, we are able to obtain an indication of how well each model performs and further compare the performance between the models. However, the information obtained is limited

since the performance between model generated and human created results cannot be directly assessed and compared. In an automatic evaluation setting, human created references are considered as gold standards.

Automatic Evaluation Metrics

Our research into the evaluation metrics is based on the frequently applied ones for text generation and summarization tasks. After considering a series of different metrics, BERTScore [40] was selected. We explain the reasoning behind our selection by first introducing the classic metric ROUGE [41] and compare it to BERTScore.

ROUGE measures the number of overlapping n-grams or word sequences between the automatically produced summary and a set of reference summaries, typically human-produced. It has been proven to be effective in various Document Understanding Conference (DUC) tasks. Moreover, it correlates with human judgements in both multiple-reference and single-reference settings. ROUGE is essentially a set of metrics that consider three important aspects, precision, recall, and F1 measure.

Simply put, recall refers to how much of the reference summary does the generated summary cover or capture, while precision measures how much of the generated summary is in fact relevant compared against the reference summary. F1 measure takes both recall and precision into account. Recall is important in summarization tasks as all essential information should be included by the summaries, nevertheless, precision becomes equally crucial in cases where the generated summaries should be concise in nature. Therefore, in our scenario where the summaries, namely the high-level activity labels, should be both inclusive and precise in covering crucial content, it is best to compute both the precision and recall and then report the F1 measure.

Since the label summaries we expect to generate are relatively concise and potentially abstract and high-level compared to the usual text summaries, it is very likely for us to encounter situations where two or more labels are equally suitable but do not share the exact n-gram match. This is where BERTScore comes into play.

BERTScore computes similarity scores by aligning generated and reference summaries on a token-level. It leverages the pre-trained contextual embeddings from BERT to match tokens or words in generated and reference sentences by maximizing the cosine similarity through greedy computation. It has been shown to correlate with human judgment on sentence-level and system-level evaluation. Similar to ROUGE, BERTScore computes precision, recall, and F1 measure as well, which is useful for our evaluation.

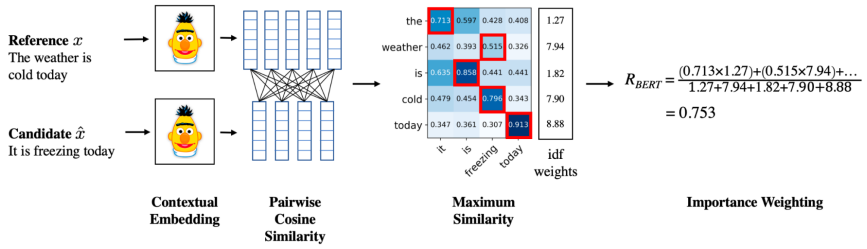


Figure 5.4: BERTScore

To conclude, we chose BERTScore as our automatic evaluation metric as it fits naturally in our condition. Due to the characteristics of process labels, aligning summaries through semantic similarity is more reasonable than matching the exact tokens. This is especially true in our single-reference setting.

5.1.3 Human Evaluation

Motivation

Apart from the limited information gained from the automatically applied evaluation, the main motive behind conducting a human evaluation is due to the subjectiveness of the labeling task. Process modelers have different preferences regarding labeling strategies and make labeling choices based on different reasons. Therefore, it is essential to evaluate the usefulness of the generated labels by humans. In addition, after examining the model results, we made an assumption that the models could provide different perspectives on processes. Some models might be better at providing a local perspective, capturing the most important task such as the decision point or main outcome within the process, while some others might be better at offering a global perspective or more general point of view, producing high-level abstraction to provide a good overview.

As a result, a user study in the form of an evaluation survey was conducted to evaluate the results in several aspects. We aimed to evaluate the suitability of the generated labels by assessing the quality of individual labels and comparing the proposed labels of different models to each other and to a gold standard created by humans. The survey results help differentiate the performance between the model generated labels and human created labels as well as the performance within different models.

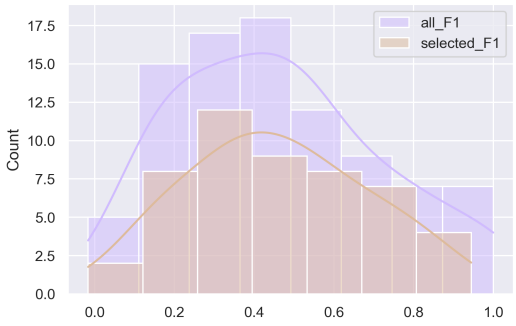


Figure 5.5: Test Set BERTScore F1 Distribution

Design Logic of Evaluation Survey

In this subsection, we briefly explain how we selected and distributed the study cases of the evaluation survey as well as the assessment process and criteria for each case evaluation.

We selected 50 cases out of 90 in total from the test set and distributed them into 5 batches. In order to be certain about choosing the cases representative of the whole test set, we made the selection based on the test set distribution of BERTScore F1 measure. From Figure 5.5, it shows that the selection set distribution of the metric score follows the test set distribution to a certain degree. One thing to note here is that the test examples with full scores, 1.0, are filtered out from the selection. As full score means that all model generated labels are identical to its human created label, there is no need for the evaluators to compare the results. After the study cases were chosen, we sorted them based on their process complexity determined by the number of low-level labels each process contains. The sorted 50 cases were then evenly assigned to 5 batches in terms of the complexity level, and the 10 cases within each batch were naturally sorted as well.

Each evaluator was assigned to one of the 5 batches and asked to make assessments for a total of 10 cases. For each case, we showed the evaluators either a sub-process or a whole process model with its process text extracted from the labels of its process steps. Following the process information, at most 4 unique generated labels were presented. We then asked the evaluators to (1) pick the best label from the options given, (2) assign one (or more) reason(s) for this selection, and (3) rate the selected label with respect to the criteria described below. Then, (4) compare against a label created by a human stating if this is better than the selected generated label and rating it according to the same criteria.

For the step-2 of the assessment process, we asked the evaluators to consider different reasons that make a label suitable for a model fragment. Ideally, a label should reflect one or more of the reasons we mentioned previously, main reasons for labeling . Simply put, the evaluators should consider a selected label is good because it:

- reflects the most important task of the process fragment
- reflects the main outcome of the process fragment
- provides a good overview of the process fragment

As for the step-3 of the assessment process, we asked the evaluators to rate a label based on the following criteria on the scale of 1 to 5:

- Relevance: How applicable is the label to the presented process fragment (Precision)?
- Informativeness: Is the label a sufficient representative of the presented process fragment (Recall)?
- Fluency: Is the label easy to read and understand?

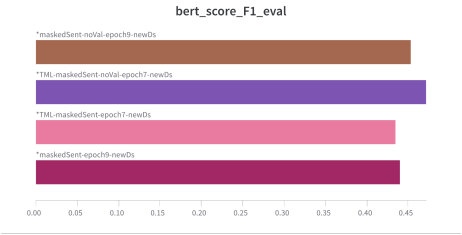
5.2 Evaluation Results

5.2.1 Automatic Evaluation

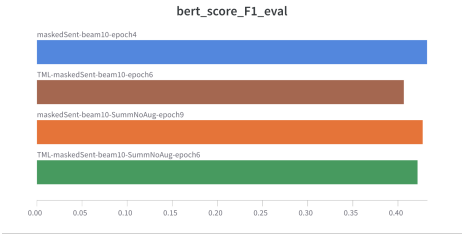
First stage of evaluation is to automatically apply BERTScore on the model generated labels compared against human created labels. As shown on Figure 5.6a and 5.6b, there is no obvious winner among the models as all F1 scores of Pegasus and Pegasus-TML with or without data augmentation lie between 0.4 to 0.45. Figure 5.6a shows the BERTScores of the two models in two different settings in terms of training iterations (epoch number) and with or without validation set included in testing. Figure 5.6b shows the BERTScores of two models in data augmentation settings with or without summary augmented. More results can be found in appendix B.1 and B.2.

5.2.2 Human Evaluation

We analyzed the results of the evaluation survey in several aspects. Firstly, the performance is compared between the selected model generated label and human created label, then we examined the inter-rater reliability to check the agreement



(a) BERTScore F1 without Data Augmentation



(b) BERTScore F1 with Data Augmentation

Figure 5.6: BERTScore F1 with and without Data Augmentation

level between the evaluators. Secondly, the performance is compared between 6 different models based on the evaluators' label selection, and next, we look into the evaluators' main reasons behind their label selection. Lastly, we conducted a qualitative study based on the labeling suggestions we received from the evaluators.

Performance Comparison between Models and Humans

We explored two different aspects when conducting the performance comparison between the selected model generated label and human created label. We first analyzed the direct comparison between the two, that is whether the automatically generated label is better, equally good, or worse than the manually created label, then we compared the two based on the ratings of the metric scores, namely relevance, informativeness, and fluency. The results shown below were aggregated from the 5 different batches distributed to the evaluators based on the assumption that the 10 cases in each batch share similar traits and process complexity.

For direct comparison, we computed the percentage distribution over the model performance against humans. If a model generated label (M) was selected to be representative of the underlying process by the evaluator, one of the following options should be chosen after comparing it with a human created label (H), M is Better, Equally good, or Worse than H. On the other hand, if none of the presented generated labels was considered to be reasonable, then the option No relevant result should be chosen and no further comparison was required. Due to the data quality issue or the lack of background knowledge for certain processes, No relevant result could also be chosen in cases where the underlying process was unclear for the evaluators to make labeling choices.

From the Figure 5.7, the green bars stand for option better, blue bars for equally good, yellow bars for worse, and red bars for no relevant results. The 4 bars representing the percentages accounted for each option are stacked in sequence and

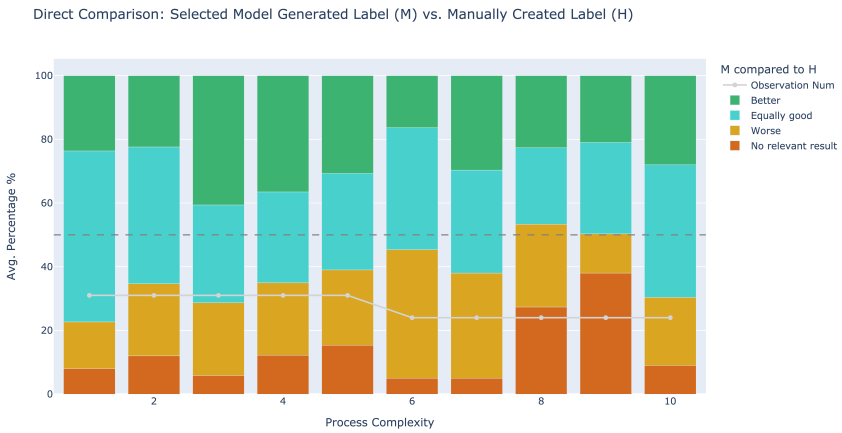


Figure 5.7: Direct Comparison

summed up to 1, from top to bottom, green, blue, yellow and red. The horizontal dash line stands for the average 50 percent. As mentioned, the 10 cases are sorted in ascending order based on the process complexity, and averaged across 5 batches. The lower-granularity of performance comparisons shown for each 50 cases, namely without aggregation, can be found in the appendix, Figure B.3.

The analysis shows that for most cases, more than half of the evaluators think the selected model generated label is either equally good or better than the human created label on average, except for case 8 and 9 where No relevant result accounts for a relatively large proportion. There is a slight trend shown in the figure that the models perform worse when process complexity increases. However, data quality issues could also play a part in the worse performance of the latter cases according to the feedback received from some evaluators. It is also more likely that the latter cases tend to be too complicated for the evaluators to grasp given the limited background knowledge and time provided to conduct the survey. In addition, the analysis accounted for the contradictory answers received from the evaluators. An answer is considered as contradictory where an evaluator labeled a worse performance for the selected generated label when the presented label collection contained one identical to the human created label. The contradictory answers account for around 3.6 percent, 10 answers out of a total of 275 to be exact. This number can be seen as a rough human error in the evaluation survey, which is tolerable.

For the metric score comparison, we analyzed each metric through creating a line plot where each point stands for the average rating of the respective case. From Figure 5.8, the results of selected model generated labels and human created

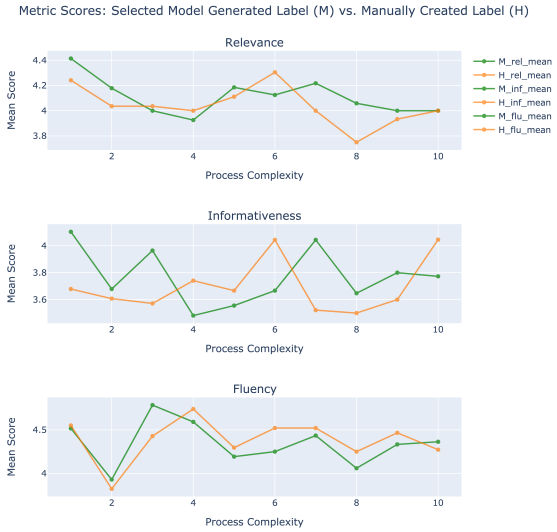


Figure 5.8: Metric Scores Comparison

Table 5.1: Metric Scores Comparison

	Relevance	Informativeness	Fluency
M_mean	3.99	3.66	4.29
H_mean	3.95	3.64	4.35
M_std	0.75	0.82	0.63
H_std	0.82	0.86	0.66

labels seem to be comparable to each other, especially in relevance and informativeness. There is no clear pattern shown in the figure such that humans outperforms the models when process complexity increases or vice versa, except when looking solely at fluency. As mentioned, the lower-granularity version of performance comparisons is provided in the appendix, Figure B.4.

To further aggregate the results, we averaged the ratings across all cases for each metric. We can see from the table 5.1 that the models slightly outperformed humans in the mean of the metric scores regarding relevance and informativeness and slightly under-performed regarding fluency. Furthermore, the models outperformed in the standard deviations of all metric scores implying that the models produce more stable results than humans.

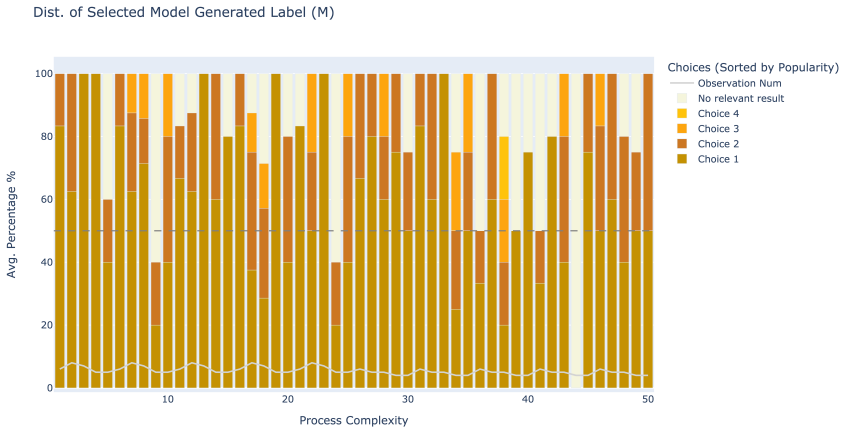


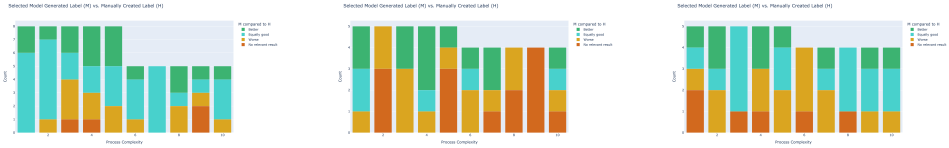
Figure 5.9: Inter-rater Reliability 1

Inter-rater Reliability

After analyzing the model performances against humans, we conducted several experiments to test the inter-rater reliability, which gives us insights into the extent the evaluators agreed with each other during the assessment.

The first thing we wanted to explore is whether the evaluators made similar selections towards the model generated labels. We sorted the provided labels based on their popularity in terms of the vote counts and computed the percentage distribution over all options presented. Choice 1 shown in the Figure 5.9 means that the respective label was the top selected option by the evaluators, Choice 2 simply means the second most selected label, and the logic applies to the rest of the options. As mentioned previously, No relevant result could be chosen in cases where the evaluators found no suitable label provided or the presented process was unclear to the evaluators. Judging from the figure with all 50 cases presented, the agreement among the evaluators seems to be high as Choice 1 and Choice 2 account for over 50 percent in most cases. Nevertheless, there are few cases where equal or more than three options were selected among the evaluators. For the cases showing less agreement, we should keep in mind that there is a possibility of false disagreement where the evaluators were indifferent towards all options presented. In such situations, the evaluators might find it hard to make labeling choices as most labels were considered equally good or bad concerning the process.

The next exploration section is regarding the evaluators' agreement within the performance comparisons between the model generated labels and human created labels, which include the direct comparison and metric score comparison. For



(a) Vote Counts Batch 2 (b) Vote Counts Batch 4 (c) Vote Counts Batch 5

Figure 5.10: Inter-rater Reliability 2

(a) Heapmap Batch 2 (b) Heapmap Batch 4 (c) Heapmap Batch 5

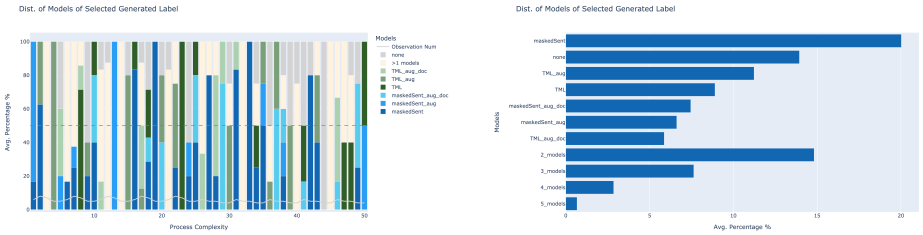
Figure 5.11: Inter-rater Reliability 3

the direct comparison, we simply counted the votes for each option in all cases as shown in Figure 5.10. The agreement level seems to drop compared to purely looking at the label selection. This can be expected since the evaluators could select the same model generated label and feel differently about the comparison made with the human created labels. Another finding worth mentioning is that the models performed worse in batch 4. More figures in appendix B.5.

As for the metric score comparison, we computed the Pearson Correlation between the ratings of selected model generated labels and human created labels given by the evaluators in each batch. One caveat in this particular study is that the ratings were partially given based on different labels as one could select different generated labels provided in the selection. We visualized the correlations using the heatmaps as shown in Figures 5.11. The heatmaps show general consensus among the evaluators, with most ratings positively correlated. However, the agreement level is not considered high as most correlations are within the region of 0 to 0.5, except for batch 4. Interestingly, we also noticed that batch 4 is the worst performing batch for the models. More figures in appendix B.6.

Performance Comparison among Models

Based on the labeling selection evaluators made, the performances between 6 different models can be compared, namely the two base models, Pegasus (maskedSent) and Pegasus-TML (TML), and two augmented versions of them, Pegasus-Aug (maskedSent_aug), Pegasus-Aug-Doc (maskedSent_aug_doc), Pegasus-TML-Aug (TML_aug), Pegasus-TML-Aug-Doc (TML_aug_doc). For each case, at most



(a) Selected Model Distribution (b) Distinct Selected Model Dist

Figure 5.12: Selected Model Distribution

4 generated labels produced by the models were presented to the evaluators. Each label could represent the results produced by more than one model, that is to say, the models could produce the identical labels and it often happens in scenarios where the underlying processes are clear or relatively easy to them based on the training.

As we can see in the Figure 5.12a, there is no clear winner among the different models across cases. There are cases where Pegasus maskedSent-based models clearly dominate, colored in blue, and similarly, there are cases where Pegasus-TML TML-based models outperform, colored in green. Surprisingly, an equally large portion of the selected labels goes to the ones produced by more than one model, colored in light yellow. When the evaluators selected none of the presented labels, it is counted as none, colored in gray.

To be able to see the individual model performance clearer, we created another bar chart that shows the percentage distribution over the selected output of the individual models or more than one model 5.12b. All percentages of the horizontal bars sum up to 1. 20 percent of the selected labels were uniquely generated by Pegasus (maskedSent), which accounts for the highest percentage among all models but not yet leaving the rest of them replaceable. The model following Pegasus in the ranking is Pegasus-TML-Aug (TML_aug), accounting for around 12 percent. In addition, around 26 percent of the time, more than one model produced identical labels that got selected.

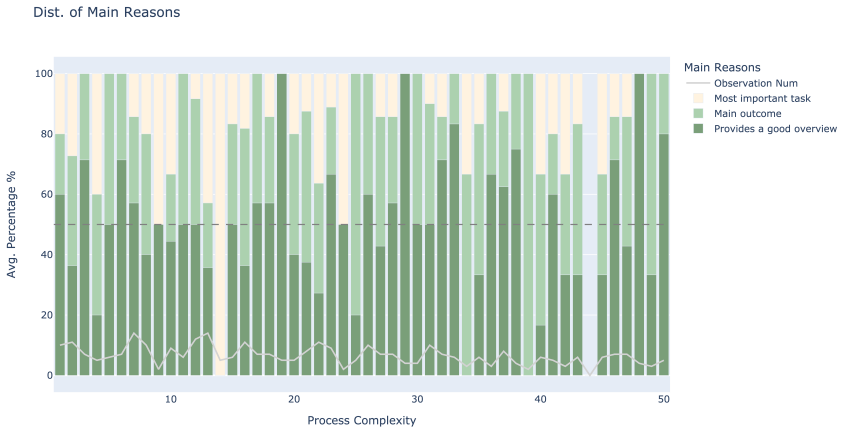


Figure 5.13: Main Reasons behind Label Selection

Main Reasons behind Label Selection

Next, we analyzed the evaluators' main reasons behind their label selection. We provided three main reasons that the labels could reflect on, namely the most important task, the main outcome, and a good overview provided. The evaluators can select more than one reason for their labeling choice. Through this analysis, we gained insights into the main qualities provided by the selected generated labels.

The percentage distribution of the main reasons across all cases is shown in Figure 5.13. As it is a multi-choice task, the percentages stand for the relative prevalence based on the vote counts each reason received, given that every evaluator had at most 3 votes. The figure shows that the reasons Provides a good overview and Main outcome prevail across cases especially with the reason Provides a good overview dominating in general. We averaged the percentages across the cases to obtain the final aggregated results, which show that Provides a good overview accounts for 49.70 percent, Main outcome accounts for 33.65 percent, and Most important task accounts for 16.65 percent. This can be expected as the evaluators could find the labels reflect the Main outcome or Most important task and at the same time Provides a good overview of the process since all other information subsided compared to the core tasks. Overall, the results indicate that the generated labels are considered as high-level, which serves the main purpose of our work.

We also examined the distribution of the main reasons with respect to each model. It is again the relative prevalence based on the vote counts of the selected model output. From the Figure 5.14, we can see that the top performing models Pegasus (maskedSent) and Pegasus-TML-Aug (TML_aug) seem to provide a more

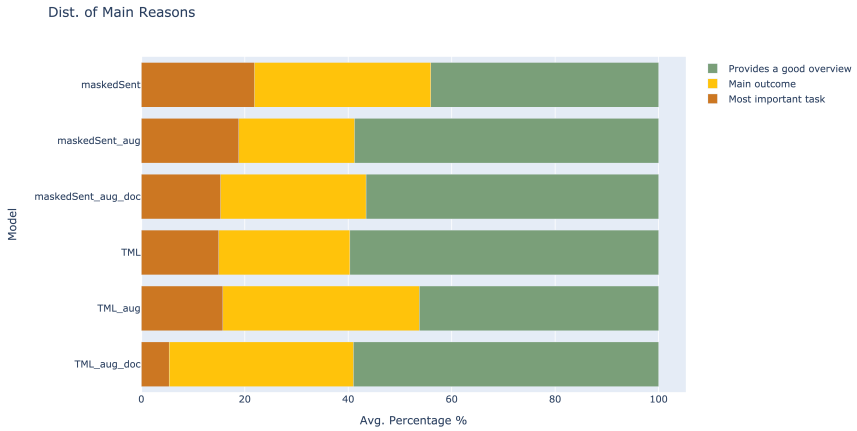


Figure 5.14: Main Reasons for Models

balanced view with comparably larger portions of percentages distributed to the reasons Main outcome and Most important task. To be more specific, top performing here means that the selected labels produced by Pegasus and Pegasus-TML-Aug are less replaceable by other models as shown in the figure. Once again, this observation can only be considered as a conjecture since the sample size is limited.

Qualitative Study

In the final analysis of the assessment, we conducted a qualitative study based on the labeling suggestions we received from the evaluators. At the end of each case evaluation, we asked the evaluators to optionally suggest alternative labels that they think best suited the underlying process. Although it was not the main focus of the evaluation survey, it allows us to better comprehend the qualities that an ideal label should have in certain scenarios. We provide three examples below to illustrate our main findings.

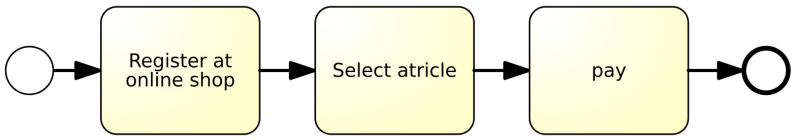


Figure 5.15: Example 1 of Qualitative Study

Example 1

- **Process Text:**
  "register at online shop, select atricle, pay"
- **Model Generated Labels:**
  - "shop at atricle"
  - "register online shop"
  - "shop online"
  - "register account"
- **Human Created Label:** "shop online"
- **Label Suggestion:**
  - "E-commerce purchase process"
  - "register and shop online"

This simple online purchase process, Figure 5.15, contains two main activities, registration and purchase, and the model generated output contains labels that reflect them. The labels "register online shop" and "register account" focus only on the activity registration, while the label "shop online", identical to the human created label, presents a part-of relation comprising both activities. The label suggestions also deliver information that embodies the two main activities. Compared to "shop online", "E-commerce purchase process" is a more abstract and high-level term, and "register and shop online" adds on the activity registration based on the context online instead of having it implied in the action shop online.

Example 2

- **Process Text:**
  "check application for basic completeness, sort applications, disregard application, add to rejection notification que"
- **Model Generated Labels:**
  - "check application for basic completeness"
  - "process application"
  - "registration for incoming applications"
  - "check application"
- **Human Created Label:** "manage application received"
- **Label Suggestion:**
  - "pre-filter applications"
  - "Basic Application Requirements Check"

This sub-process in Figure 4.14 revolves around a clear decision point "check application for basic completeness", which can be considered representative of the underlying process. As expected, it is generated by one of the models as a high-level activity label. Other generated labels "process application" and "check application" suggest more generalized terms for handling application. Nevertheless, one can see that this sub-process refers only to an initial procedure of the whole application handling process, which is implied by the human created label "manage application received". On the other hand, label suggestions "pre-filter applications" and "Basic Application Requirements Check" are semantically more specific into the actual tasks of the sub-process and at the same time show a part-of relations for the low-level activities.

Example 3

- **Process Text:**
  "analyze order, check stock, make products, ship products, send bill, receive payment"
- **Model Generated Labels:**
  - "handle order"
  - "perform order"
  - "verify stock and prepare shipment"
  - "handle purchase order"
- **Human Created Label:** "fulfill order"
- **Label Suggestion:**
  - "Order-to-Cash"
  - "Fulfill order and handle payment"

Judging from the label suggestions received for this case in Figure 4.2b, we can clearly see that for some evaluators, there are two high-level activities, order and payment, involved in this process. The suggested label "Order-to-Cash" concludes the process with a domain specific term, while the other suggested label "Fulfill order and handle payment" directly combines two activities together as one. On the contrary, our models output labels "handle order" and "handle purchase order" that provide good overviews but in a rather general tone. While the suggested labels are more informative, it is not the main purpose of our models to produce labels that include two distinct high-level activities.

The labels suggested by the evaluators are generally more explicit in expressing the core content of the underlying process, and at the same time, provide a decent level of abstractions. Overall, the suggestions we received from the evaluators are useful and can be utilized to retrain the models. In this manner, the models can be adjusted according to the preferences of the process modelers and improve on generating the ideal labels.

5.3 In-depth Analysis and Discussion

This section provides in-depth analysis of individual cases and overall discussion of the evaluation. In-depth analysis considers two types of case study, namely qualitative analysis and error analysis. Qualitative analysis provides 2 examples to reflect the initial observation that the models often deliver different perspectives, and error analysis examines the 4 cases that were rated with worse performance. As for the discussion, we conclude the findings of human evaluation results with a summary and discuss the challenges and limitations faced in the evaluation process.

5.3.1 Qualitative Analysis

From example 1, Figure 5.16, we are able to see different perspectives provided by the models. “request crowd-based solutions” and “upload crowd-request” offer more localized point of view focusing on the actions request and upload respectively, while “handle crowd request” and “operate problem solving process” provide more global and generalized perspectives. Compared to the human created label “execute crowdsourcing”, model generated labels “request crowd-based solutions” and “handle crowd request” lack the specific term crowdsourcing but are fairly representative and accurate in the semantics.

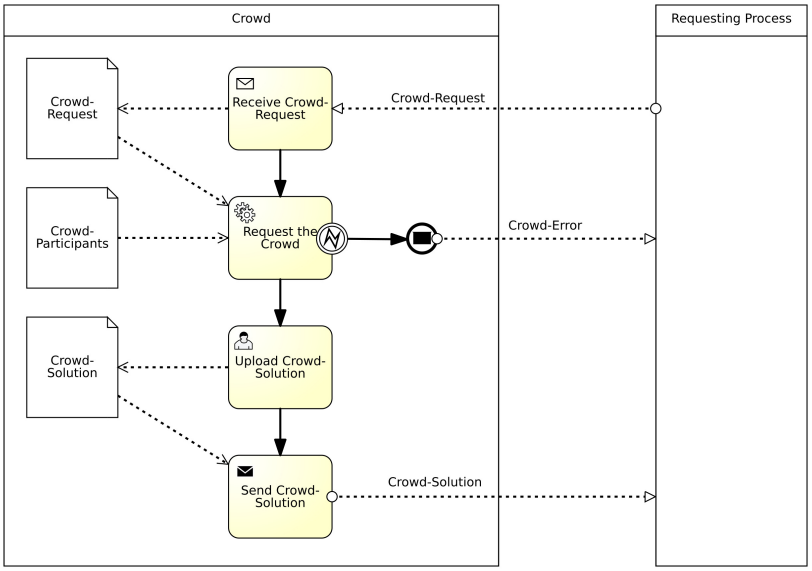


Figure 5.16: Example 1 of Qualitative Analysis

Example 1

- **Process Text:**
  "receive crowd-request, request the crowd, upload crowd-solution, send crowd-solution"
- **Model Generated Labels:**
  - "request crowd-based solutions"
  - "upload crowd-request"
  - "handle crowd request"
  - "operate problem solving process"
- **Human Created Label:** "execute crowdsourcing"

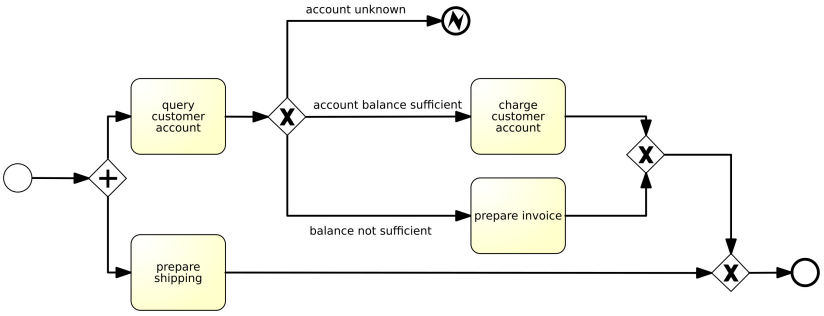


Figure 5.17: Example 2 of Qualitative Analysis

Example 2

- **Process Text:**
  "query customer account, charge customer account, prepare invoice, prepare shipping"
- **Model Generated Labels:**
  - "prepare shipping invoice"
  - "prepare customer order"
  - "manage customer account configuration"
  - "process shipping order request"
- **Human Created Label:** "execute automated order processing"

Similarly, in example 2, Figure 5.17, we can also see different perspectives provided by the models. "prepare shipping invoice" and "manage customer account configuration" offer more localized point of view focusing on the objects invoice and customer account respectively, while "process shipping order request" and "prepare customer order" provide more global and generalized perspectives. Compared to the human created label "execute automated order processing", model generated labels "process shipping order request" and "prepare customer order" fail to capture the context information automated from action query but in general, provide good overviews of the underlying process.

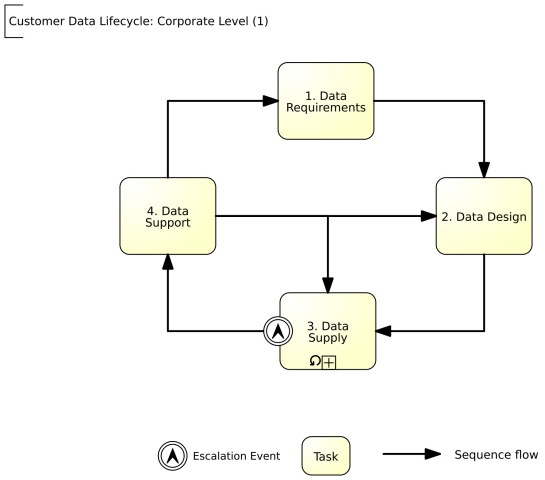


Figure 5.18: Example 1 of Error Analysis

5.3.2 Error Analysis

Example 1

- **Process Text:**
  "data requirements, data design, data supply, data support"
- **Model Generated Labels:**
  - "manage data requirements"
  - "perform data analysis to conceptualise data"
  - "perform data analysis"
  - "manage data supply"
- **Human Created Label:** "manage customer data lifecycle"

For example 1, Figure 5.18, 60% of the evaluators found that the models produced no relevant result and 40% found the results produced were worse than the human created label "manage customer data lifecycle". The process name can be revealed rather easily for the evaluators by spotting the additional information Customer Data Lifecycle: Corporate Level (1) provided at the top left corner. On the contrary, the models had no access to the information other than the event and activity labels. In general, the models managed to generate process-related labels

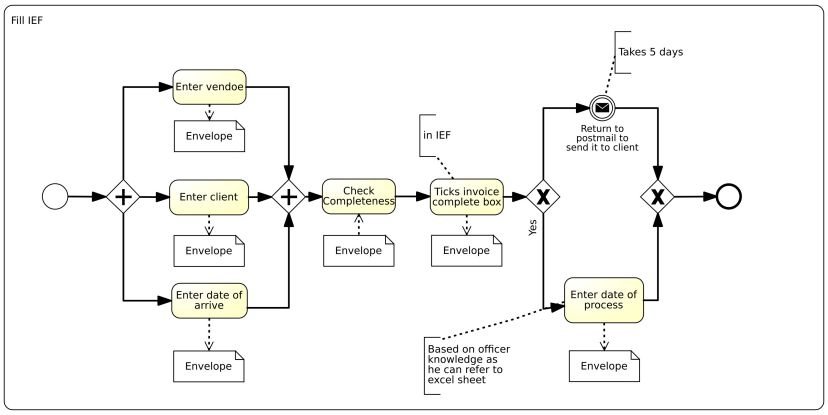


Figure 5.19: Example 2 of Error Analysis

centered around data related concepts. Judging from the feedback we received from evaluators, the underlying process might have been unclear to the evaluators as well without the context information provided or having the background knowledge of the process.

Example 2

- **Process Text:**
  "enter vendoe, enter client, enter date of arrive, check completeness, ticks invoice complete box, enter date of process, return to postmail to send it to client"
- **Model Generated Labels:**
  - "complete invoice entry form"
  - "tick the invoice box"
  - "complete invoice configuration"
- **Human Created Label:** "fill information and check completeness"

For example 2, Figure 5.19, 25% of the evaluators found that the models produced no relevant result and 75% found results produced were worse than the human created label "fill information and check completeness". Instead of combining two high-level activities as one, the models generated labels focusing on one main high-level activity, invoice completion. In fact, the generated label "complete invoice entry form" matches the original sub-process name "Fill IEF" (IEF shorts for

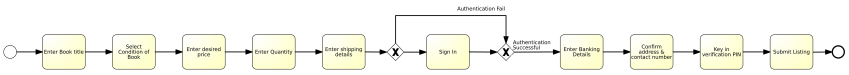


Figure 5.20: Example 3 of Error Analysis

Invoice Entry Form) as shown in the figure. In addition, the model results mainly reflect on the important tasks "Check Completeness" and "Ticks invoice complete box" from the perspective of the control flow, the task that either follows or is followed by a gateway. Naturally, the output labels match the model pre-training scheme, important sentence masking. Through this example, we could see that the evaluators prefer to explicitly include core activities of the process in the high-level activity label despite the main object invoice not stated in the human created label.

Example 3

- **Process Text:**
  "enter book title, select condition of book, enter desired price, enter quantity, enter shipping details, sign in, enter banking details, confirm address & contact number, key in verification pin, submit listing"
- **Model Generated Labels:**
  - "enter book title"
  - "apply for the first time"
  - "registration for the book"
  - "book seller ebay account"
- **Human Created Label:** "list book for sale"

For example 3, Figure 5.20, 50% of the evaluators found that the models produced no relevant result and 50% found results produced were worse than the human created label "list book for sale". Through observing the model results, we noticed that some models recognized certain attributes of this process. Generated labels "registration for the book" and "apply for the first time" show that it is an application process, while the label "book seller ebay account" shows that it is a bookseller activity on the e-commerce platform. The observation confirms that the models are not yet able to produce labels with high abstraction level. With more training examples as such, the models can be trained to generate labels that exhibit strong part-of relations.

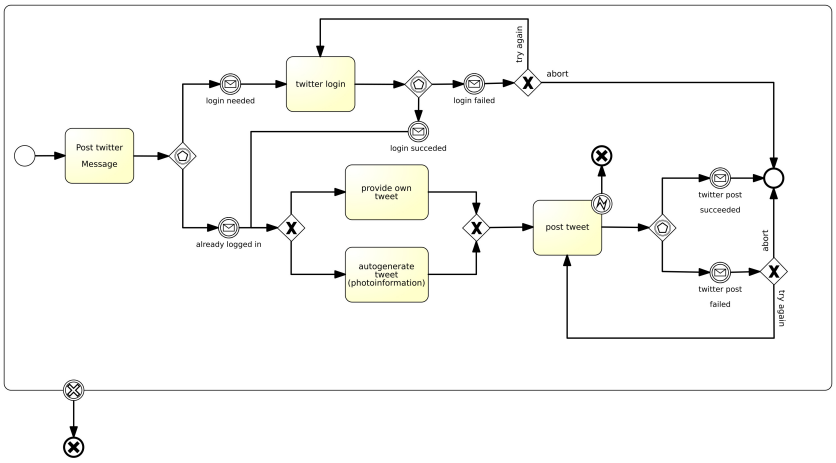


Figure 5.21: Example 4 of Error Analysis

Example 4

- **Process Text:**
  "post twitter message, login needed, twitter login, login succeded, login failed, already logged in, provide own tweet, autogenerate tweet (photo information), post tweet, twitter post succeeded, twitter post failed"
- **Model Generated Labels:**
  - "log in"
  - "handle twitter login"
  - "login twitter"
- **Human Created Label:** "post twitter message"

For example 4, Figure 5.21, all evaluators voted for no relevant result. It is fairly clear that the models failed to capture the most important task in the process post tweet, which is mentioned at both the start and end activities. The models generated labels focusing on the other task in the process login, which accounts for half of the process. It could be due to the selection bias introduced by the current training data, and therefore, this issue could be alleviated by adding more data that offers diversity in the processes involving login. Another possible reason is that the models see no difference between an event label and an activity label, and it could lead to the models giving equal weights to all labels, including the intermediate

events. In most cases, intermediate events are less important, which is especially true in this example as most of them are only about login status. This issue could be addressed in several ways. One potential solution in the context of the transformer language model is to utilize the attention layer for weight assignment, and another is to simply optimize the data preprocessing process such as adding in a step to randomly filter out labels of intermediate events.

5.3.3 Discussion

To conclude, the overall model performance is comparable to humans. Although the general satisfaction seems to decrease when process complexity increases , data quality issues might be involved. The agreement between evaluators is high when selecting model generated labels but the agreement level drops in finer levels of performance assessment. Regarding the performances compared between the models, Pegasus (maskedSent) outperforms the rest of the models but is not yet able to replace them. In addition, all models are capable of providing good overviews of the underlying processes. As for the error analysis, most errors we've seen can be alleviated with adding more labeled training data or through optimizing data preprocessing and model learning steps.

There are a few challenges and limitations of the evaluation. Firstly, our data collection presents significant data quality issues which affect the model performance as well as model assessment conducted by humans. Secondly, only limited test data is available for evaluation even after manual labeling was in place. Lastly, the user study was carried out in a limited scale consisting of a total of 31 surveys with each case evaluated around 5 to 8 people.

Chapter 6

Conclusion

6.0.1 Main Purpose and Contribution of the Thesis

Event and activity abstractions are important tasks to reduce the complexity of either the discovered or existing process models, which enables process stakeholders to enhance process understanding and gain insights from it. We propose an approach to automatically generate semantically meaningful labels for process data. This labeling technique is built upon insights gathered from the previous work and it can be utilized in both event log and process model abstraction scenarios. Our main contribution is to achieve the goal with a learning approach optimized for the process field. We made use of Machine Learning techniques and adapted the existing resource to fit our process domain and solve process specific tasks. Specifically, we translated processes into process texts as input for the fine-tuned models to return educated summaries used as labels. Furthermore, the approach can be developed in a sustainable fashion through optimizing the results with user feedback.

The automatic generation approach provides a foundation not only for proposing labels of process fragments, but also for whole processes. In this regard, our approach can be used to produce abstractions of different granularity levels. Users can better comprehend the processes at hand by dynamically adjusting the abstraction level through interacting with process mining tools. In general, different aspects of process model management can be improved by having labels available at different scales, ranging from high-level activities or sub-processes, lanes and pools to process models.

To validate our approach, we evaluated fitness of the generated labels with automatic evaluation metrics and assessed applicability through human evaluation. Since automatic evaluation only allowed us to gain limited insights into the performance comparisons between the models, we incorporated humans in the assessment process. Specifically, a user study was conducted in the form of an evaluation

survey designed for this work. In spite of limited labeled training data, the overall performance achieved by the models is comparable with humans.

6.0.2 Implications for Practice

Besides performance validation, it is crucial to give thoughts on the implication for practice and potential use cases of this labeling technique. Firstly, providing label suggestions can substantially reduce manual labeling time for the users. Secondly, labels can be produced at different granularity levels of processes with ease. Generally speaking, apart from enhancing process understanding, labeling can improve data management to a great extent. For instance, it ameliorates the search efficiency in process knowledge base and naturally increases process reusability. In addition, a boost in search efficiency can indirectly enhance resource management. With better search functionality in existing process databases, process stakeholders can integrate similar processes within or across departments for resource optimization.

Once the implementation of the labeling technique is in place or used in production, it is imperative to quantify the improvements in order to monitor the performance constantly. Keeping process modelers or experts in the feedback-driven training loop allows us to measure the revision time or frequency throughout the testing or production phase. Monitoring the revision measurements is one way to observe the model performance regularly, and there are certainly other approaches to be explored from different aspects.

6.0.3 Future Work

For future work, it is important to address limited data issues in order to produce better results. Furthermore, data quality should be measured and taken into account in the evaluation as it affects the model performance, especially for this data collection. Currently, only labels of events and activities are fed into the models. One can try to include more information such as incorporating contextual knowledge of a process. Additionally, more work can be done to optimize the models, including applying data filtering and guided training. We also realized that the simplest version of model Pegasus could achieve decent performance by fine-tuning with more quality data. In order to implement it in practice, one can train the model in a continuous fashion by collecting and integrating user or expert feedback in the training pipeline. Lastly, a ranking mechanism for the generated results can be designed to provide users a list of label suggestions ordered by a ranking algorithm.

Bibliography

[1] D. R. Beddiar, M. S. Jahan, and M. Oussalah, "Data expansion using back translation and paraphrasing for hate speech detection," *Online Social Networks and Media*, vol. 24, p. 100153, 2021.

[2] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. van der Aalst, and P. J. Toussaint, "Guided process discovery–a pattern-based approach," *Information systems*, vol. 76, pp. 1–18, 2018.

[3] M. de Leoni and S. Dündar, "Event-log abstraction using batch session identification and clustering," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 36–44.

[4] J.-R. Rehse and P. Fettke, "Clustering business process activities for identifying reference model components," in *International Conference on Business Process Management*. Springer, 2018, pp. 5–17.

[5] N. Tax, N. Sidorova, R. Haakma, and W. M. van der Aalst, "Event abstraction for process mining using supervised learning techniques," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2016, pp. 251–269.

[6] J. Mendling, H. A. Reijers, and J. Recker, "Activity labeling in process modeling: Empirical insights and recommendations," *Information Systems*, vol. 35, no. 4, pp. 467–482, 2010.

[7] H. Leopold, J. Mendling, and H. A. Reijers, "On the automatic labeling of process models," in *International Conference on Advanced Information Systems Engineering*. Springer, 2011, pp. 512–520.

[8] H. Leopold, J. Mendling, H. A. Reijers, and M. La Rosa, "Simplifying process model abstraction: Techniques for generating model names," *Information Systems*, vol. 39, pp. 134–151, 2014.

[9] R. Mitkov, *The Oxford handbook of computational linguistics*. Oxford University Press, 2022.

[10] Y. Goldberg and G. Hirst, "Neural network methods in natural language processing. morgan & claypool publishers (2017)," *zitiert auf*, p. 69, 2017.

[11] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[12] R. E. Wright, "Logistic regression." 1995.

[13] S. Suthaharan, "Support vector machine," in *Machine learning models and algorithms for big data classification*. Springer, 2016, pp. 207–235.

[14] L. Deng and Y. Liu, *Deep learning in natural language processing*. Springer, 2018.

[15] K. Jing and J. Xu, "A survey on neural network language models," *arXiv preprint arXiv:1906.03591*, 2019.

[16] P. F. Brown, V. J. Della Pietra, P. V. Desouza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–480, 1992.

[17] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738*, 2014.

[18] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, no. 3. Makuhari, 2010, pp. 1045–1048.

[19] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[22] N. Moratanch and S. Chitrakala, "A survey on extractive text summarization," in *2017 international conference on computer, communication and signal processing (ICCCSP)*. IEEE, 2017, pp. 1–6.

[23] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.

[24] A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?" *arXiv preprint arXiv:2002.08910*, 2020.

[25] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 328–11 339.

[26] M. Weske, G. Decker, M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers, "Model Collection of the Business Process Management Academic Initiative," Apr. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3758705

[27] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[28] C. Winship and R. D. Mare, "Models for sample selection bias," *Annual review of sociology*, pp. 327–350, 1992.

[29] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks," *arXiv preprint arXiv:1901.11196*, 2019.

[30] M. Bayer, M.-A. Kaufhold, and C. Reuter, "A survey on data augmentation for text classification," *ACM Computing Surveys*, 2021.

[31] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, "Qanet: Combining local convolution with global self-attention for reading comprehension," *arXiv preprint arXiv:1804.09541*, 2018.

[32] B. MacCartney, *Natural language inference*. Stanford University, 2009.

[33] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, and J. Tang, "Kepler: A unified model for knowledge embedding and pre-trained language representation," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 176–194, 2021.

[34] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[35] Z. Yang, M. Ding, C. Zhou, H. Yang, J. Zhou, and J. Tang, "Understanding negative sampling in graph representation learning," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 1666–1676.

[36] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," *arXiv preprint arXiv:1607.05368*, 2016.

[37] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[38] H. A. Reijers and J. Mendling, "A study into the factors that influence the understandability of business process models," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 3, pp. 449–462, 2010.

[39] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[40] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," *arXiv preprint arXiv:1904.09675*, 2019.

[41] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

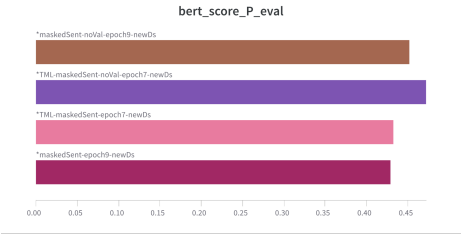
Appendix A

GitHub Repository

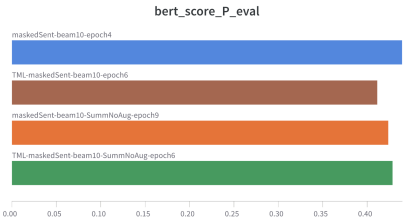
GitHub repository of this thesis containing the source code and a documentation:
https://github.com/YenTingWangTW/Thesis

Appendix B

Further Experimental Results

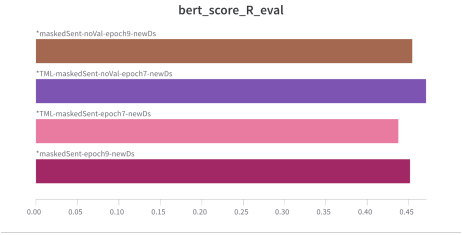


(a) BERTScore P without Data Augmentation

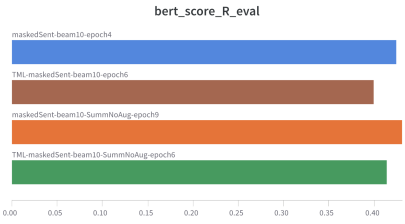


(b) BERTScore P with Data Augmentation

Figure B.1: BERTScore P with and without Data Augmentation



(a) BERTScore R without Data Augmentation



(b) BERTScore R with Data Augmentation

Figure B.2: BERTScore R with and without Data Augmentation

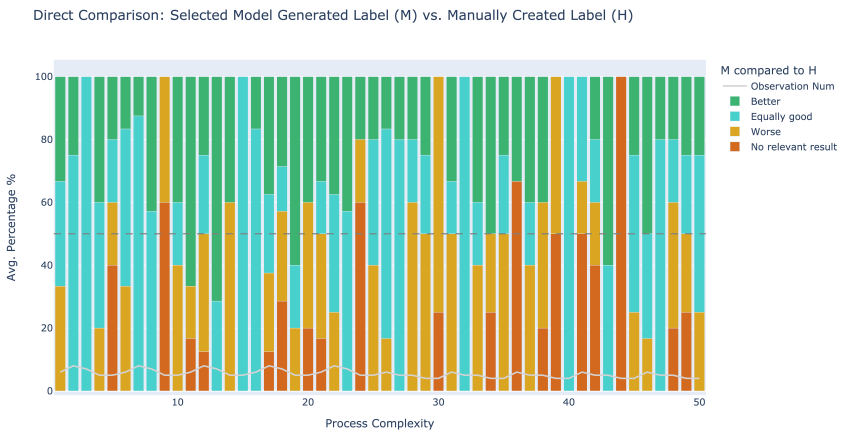
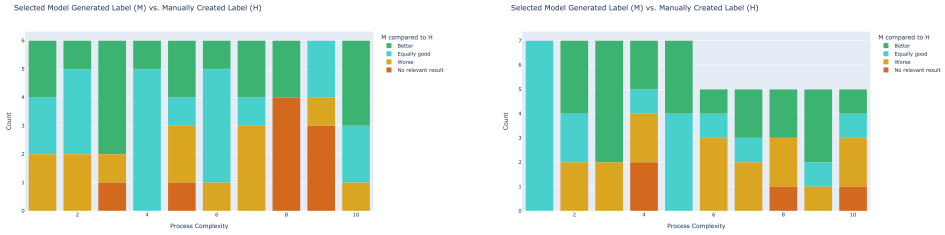


Figure B.3: Direct Comparison 50 Cases

Figure B.4: Metric Scores Comparison 50 Cases



(a) Vote Counts Batch 1

(b) Vote Counts Batch 3

Figure B.5: Inter-rater Reliability 2 - Appendix

(a) Heatmap Batch 1

(b) Heatmap Batch 3

Figure B.6: Inter-rater Reliability 3 - Appendix

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Master-/Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Er- klärung rechtliche Folgen haben wird.

Mannheim, den 09.08.2022 Unterschrift