Recognizing and Splitting Conditional Sentences for Automation of Business Processes Management

Ngoc Phuoc An Vo, Irene Manotas, Octavian Popescu, Algimantas Černiauskas*, Vadim Sheinin
IBM Research, Yorktown Heights, US
*IBM Client Innovation Center Baltic, Vilnius, Lithuania
{ngoc.phuoc.an.vo, irene.manotas, algimantas.cerniauskas}@ibm.com
{o.popescu, vadims}@us.ibm.com

Abstract

Business Process Management (BPM) is the discipline which is responsible for management of discovering, analyzing, redesigning, monitoring, and controlling business processes. One of the most crucial tasks of BPM is discovering and modelling business processes from text documents. In this paper, we present our system that resolves an end-to-end problem consisting of 1) recognizing conditional sentences from technical documents, 2) finding boundaries to extract conditional and resultant clauses from each conditional sentence, and 3) categorizing resultant clause as Action or Consequence which later helps to generate new steps in our business process model automatically. We created a new dataset and three models to solve this problem. Our best model achieved very promising results of *83.82, 87.84*, and *85.75* for Precision, Recall, and F1, respectively, for extracting Condition, Action, and Consequence clauses using Exact Match metric.

1 Introduction

Business processes which serve as basis for many (if not all) companies are usually stored as unstructured data, especially as text documents (Blumberg and Atre, 2003). They can reflect all organization tasks and activities in order to provide services. Therefore, these documents could be converted into process models which allows to discover, analyze, redesign, monitor and control these business processes (M. Dumas, 2018). The discipline which is responsible for management of this life cycle is known as Business Process Management (BPM).

Today, this well known practice is very interdisciplinary and combines: organizational theory, management, and even computer science. One of the sub-areas of computer science is Natural Language Processing (NLP) and due to large number of freely available NLP tools such as taggers, parsers, and the lexical database WordNet, NLP is applied in a multitude of domains and BMP is no exception. The application of NLP techniques could be used to extract useful information from textual documents to infer a process model, and also to extract information from the process model to facilitate visual process analysis (Leopold, 2013).

Process modelling is a very complex and time-consuming task. In this work we mainly focused on the analysis of extracted information from text documents in order to have more accurate process models. Given a list of extracted sentences from technical documents, the contributions of our work consists of the following: 1) classifying if a sentence is a conditional sentence, 2) identifying and extracting the conditional and resultant clauses from a conditional sentence, and 3) categorizing the extracted resultants. Section 2 introduces business context and application for our tasks. Section 3 explains how we defined the tasks and created the new dataset TechCond towards solving the tasks. Section 4 presents our methods and models for the tasks. Section 5 shows experiments, evaluation results, and the error analysis that identifies the limitations of our models.

2 Business Context, Application, and Related Works

Process Discovery Traditional process models based on interviews often provide only a limited or biased picture of the actual process. Such highly abstract result can be interpreted in many ways. Therefore, during the past decade process mining and visualization tools (i.e., Celonis, UIPath) were developed to improve process visibility by generating highly adaptable, highly maintainable and validated business process models. However, these tools focus only on analysis of results from structured data - process sequences from actual user

Proceedings of Recent Advances in Natural Language Processing, pages 1490–1497
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_167

desktops, or systems logs. It is well known that Desktop Procedures (DTPs) and Standard Operation Procedures (SOP) are very common process documentation techniques (Phalp and Cox, 2007). Therefore, in order to see a broader picture of the process the information from textual source should be analyzed and included into process model. One of the tools which includes multiple sources is Process Discovery Accelerator (PDA).

Application PDA is a business process mining and discovery tool currently available only for automation consultants of our company. It runs as a web application in production on the Services Essentials for Automation platform of our company. PDA has three main components: 1) Standard operation procedure file analysis; 2) Click stream file analysis, sequential structured data which represents actual users activities; and 3) system logs file analysis which is also structured data from applications (system, e-mail, chat). Process Discovery Accelerator is able to discover processes using separate source components or all available at once, this procedure is called process harmonization. The final view of this procedure is single, correlated or uncorrelated processes from all available sources. The example of a single (only SOP) simple process graph result in PDA UI is shown in Figure 1. It illustrates how process flow looks like when conditional and resultant clauses are split into separate steps.

The analysis of Standard operation procedures is very challenging not only because it is based on unstructured data but also because it is very non-homogeneous, domain specific and biased since it is highly dependent on a process description creator (Baier and Mendling, 2013). These SOP files usually contain complex step descriptions, i.e., "if something, then do this" (Table 1) which sometimes could act as gateways or decision points. In order to have a more accurate process model we need to detect specific clause (conditions and actions) from already extracted process steps.

The main objective of PDA is to find automation opportunities by analyzing available data from Standard operation procedures, Click stream, and log files. Without accurate analysis of SOP files it is difficult to measure differences between different sources. Our proposed solution for conditional splitting is crucial for the SOP analysis, since it allows to see actual decision points and how complex a process is by detecting actions and/or consequences.

Related Works The study (Wenzina R., 2013) proposed heuristic, rule-based method to identify condition-action sentences. By using a set of linguistic patterns authors were able to split up sentences semantically into their clauses showing the condition and the consequence with recall of 75% and precision of 88%. Note: these numbers are for specific patterns - for all activities recall is 56%. Another rule-based work was presented by (Hematialam H., 2017). This work is based on combinations of part of speech tags used as features. Authors applied more statistical approach to automatically extract features, however, the recall value is very similar to (Wenzina R., 2013). Both works were tested using sentences from medical SOP's.

Another good example of NLP and Business Process Management combination is the work from (Qian et al., 2019). Authors used latest NLP techniques (transformers) to classify textual information from word-level to sentence-level. Although this work is most similar to our work it focuses more on automatic process graph extraction, which aims to extract multi-granularity information without manually defined procedural knowledge.

3 Task Definition and New Dataset

In this section, we first analyzed the sampled data from our client, then we defined the task Conditional Sentence Splitting, and created a new dataset TechCond to solve it.

3.1 Preliminary Data Analysis

We received a dataset of 4,098 sampled sentences provided by our client from SOP documents that is based on few industries: financial services (banking) and retail. This dataset was manually annotated by client in four categories (at sentence level) with unbalanced label distribution:

- No Condition (NC) (86.5%): sentence that has no conditional logic, no condition is found.
- Condition Action (CA) (9%): sentence that has condition, and resultant is an Action.
- Condition Consequence (CC) (3%): sentence that has condition, resultant is a Consequence.

[1] https://www.foodandbeveragetrainer.com/sop/

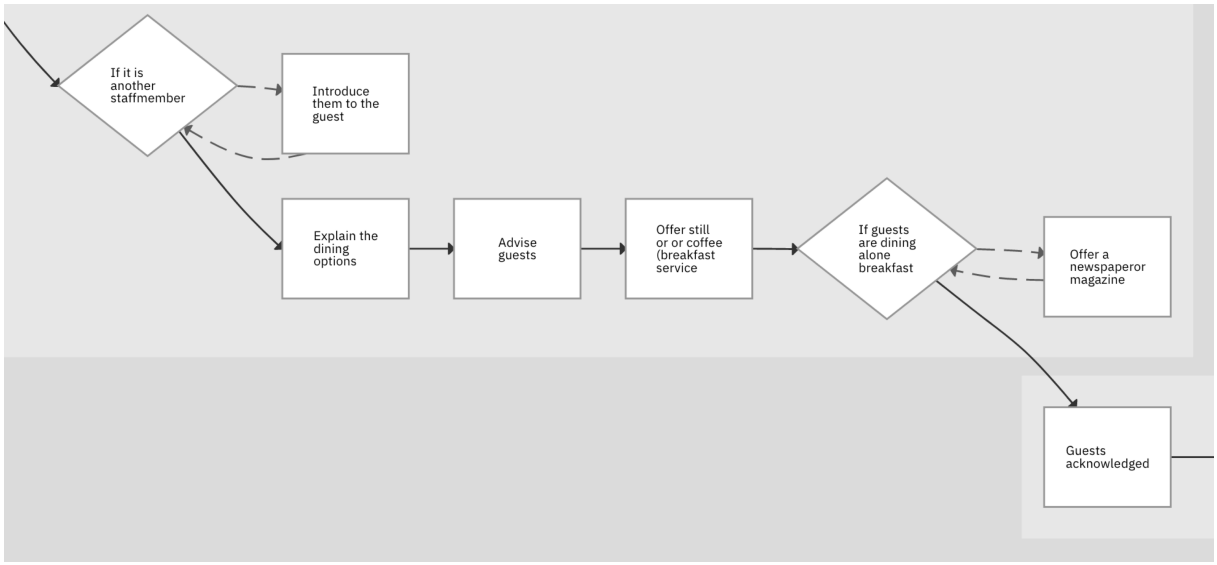

Figure 1: Standard operation procedure example of welcoming guest in restaurant from food and beverage training [1] in Process Discovery Accelerator tool UI. The process step in diamond shape indicates condition clause and dashed arrow shows resultant. Solid lines connects different process steps.

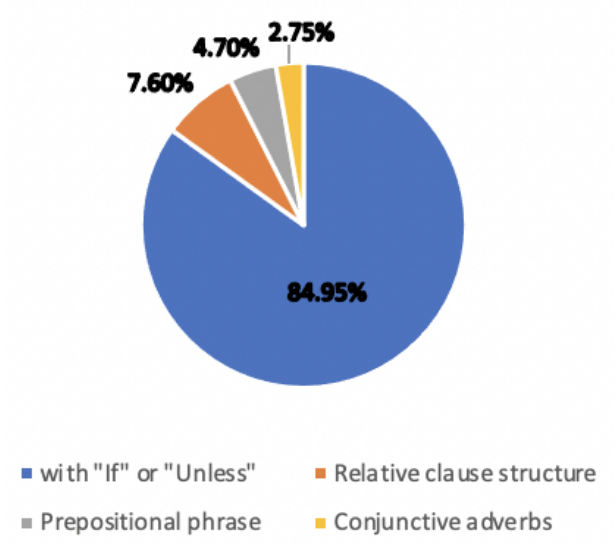

Figure 2: Syntactic structures of CA sentences.

- Only Condition (OC) (1.5%): sentence that has only condition, no resultant found.

We extracted 352 CA sentences from this client's dataset then analyzed syntactic structures of CA sentences and different types of Action resultants (Figures 2 and 3). Based on this preliminary analysis, we then created the new dataset TechCond (Section 3.3) which has better label distribution and more fine-grained annotation. We also constructed our rule-based model (Section 4) based on the analysis of syntactic structures and Action resultant types of CA sentences.

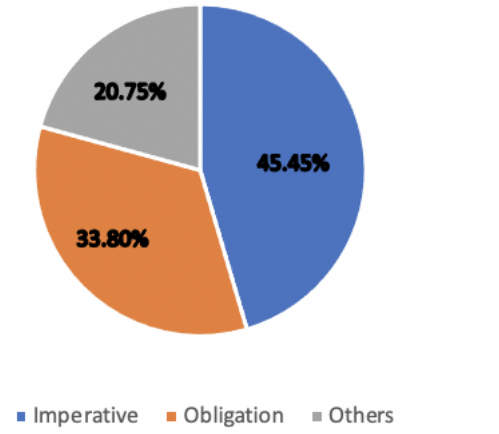

Figure 3: Action resultant types of CA sentences.

3.2 Task Definition

We defined the end-to-end task Conditional Sentence Splitting in which the input is a given sentence, and the output is the extracted clauses classified into Condition and other resultant categories such as Action or Consequence. Since this is a large and complex task, it can be split into the following three subtasks.

Subtask 1: Conditional Sentence Classification This task recognizes whether a given sentence is in conditional form. Conditional sentence is a sentence that expresses one or more thing(s) contingent on something else. The most basic form of a conditional sentence is

$$IfP,Q \quad (1)$$

where P is the conditional clause, and Q is the resultant. Recognizing conditional sentences is not a trivial task because there are various ways of expressing the meaning in conditional sentences. A basic approach to identify conditional sentences is to search for a conditional clause within a sentence that has keywords like *If* or *Unless* as condition indicators. However, conditional sentences do not always appear in standard conditional form in equation (1) with these indicators. Table 1 shows examples of conditional sentences that appear in different ways.

Subtask 2: Sentence Boundary Finding and Splitting This task finds the boundary of conditional and resultant clauses in a conditional sentence. A basic approach is to locate the comma that separates between conditional and resultant clauses. For example: *If it rains, children stay inside.* The comma in this example is a strong indicator showing that the first part of the sentence that has condition indicator *If* is a conditional clause, and the second part is a resultant. However, as conditional sentences can appear in numerous ways (Table 1), this makes conditional sentence splitting a complex task.

Subtask 3: Resultant Categorization After splitting a conditional sentence into conditional and resultant clauses, the next step is to categorize if the resultant is an Action or a Consequence clause. In our application, we are interested in Action clause as it helps us to create a new action branch for our process model. For example:

- *Refer to the author if you are in any doubt about the currency of this document.* The resultant "*Refer to the author*" is an Action clause which means the action will be taken should the condition be satisfied.

- *If the entered password is matched with the one stored in system, the user is authenticated.* The resultant "*the user is authenticated*" is a Consequence clause which shows the result should the condition be satisfied. Thus, no action is captured and no action branch is created for our process model.

Some common types of Action resultant are:

- Imperative type is phrase that has an imperative verb to give a command or an order.

- Obligation type is phrase that expresses something necessary to follow or execute. It usually has modals of obligation such as *must, have to, have got to, need to, etc.*

- Others are phrases that are neither in imperative nor in obligation form but actual meaning expresses the same.

3.3 TechCond Dataset

We created the new TechCond dataset with conditional sentences to predict sentences with conditions and different types of resultants. We extracted sentences from real text documents collected from public technical manuals and regulation documents from different websites. Technical documents included manuals and troubleshooting guides for different software products and applications available online as documents that could be downloaded. These documents were processed by the English Slot Grammar (ESG) parser (McCord et al., 2012) to identify sentences and their linguistic features. We filter out sentences having conditional words and a subordinating conjunction in their constituency parse. We tried to reduce the bias towards a specific type of conditional sentences by considering a diverse set of conditional words: *'after','as a consequence of','as a result of','assuming','if','if only','if not','only if','unless','until','when', 'because','as soon as','as long as','but for','even if','once','on the condition','provided','providing'*. After we filtered out sentences potentially describing a condition, a total of $1,936$ sentences with conditional statements were extracted for annotation to be part of the TechCond dataset.

Annotation Methodology We selected the Doccano [2] annotation tool to annotate sequences in the sentences according to our task. Guidelines for the annotation job included possible types of sentences and clauses to annotate. The labels included in the annotation were the following:

- Condition (CD): Clause describing a condition. A conditional clause can come before or after the main clause.

- Action (AC): Clause describing an actionable result that is related to a condition. The condition can be described before or after the Action clause in the same sentence.

[2] https://doccano.herokuapp.com/

Index	Conditional Sentences	Actual Meanings
1	If it rains, children should stay home.	If it rains, children should stay home.
2	Unless it rains, children can go out.	If it does not rain, children can go out.
3	1. Otherwise, they can go out.	If it does not rain, children can go out.
4	Come now and I'll give you the book.	If you come now, I'll give you the book.
5	Do you like it? You can have it now.	If you like it, you can have it now.
6	For rainy days, children stay home.	If it rains, children stay home.
7	Anyone who skips class will be disciplined.	If anyone skips class, they will be disciplined.

Table 1: Examples of Conditional Sentences appear in various ways.

Data	CD	CS	AC	OC	NC	UA
Train	1,446	568	886	18	26	83
Test	184	71	114	3	4	13
Dev	184	67	114	3	5	23
Total	1,814	706	1,114	24	35	122

Table 2: Frequency of labels by data split.

- Consequence (CS): Clause describing a non-actionable activity i.e., it does not describe an action that a system or user needs/should take.
- Only-Condition (OC): Sentence describing only a condition without additional clauses for actions or activities.
- No Condition (NC): Sentence with no conditional logic.
- Unconditional-Action: Clause describing only an action without an associated condition.

Annotators were guided to use labels at the sentence or clause level. *Sentence-level* labels included No-Condition (NC) and Only-Condition (OC) labels, and can only be assigned to a whole sentence. *Clause-level* labels included Condition (CD), Action (AC), and Consequence (CS) labels can be assigned only to a clause in a sentence having at least two clauses.

Before the annotation task, annotators were guided to analyze patterns of conditional sentences that could be found during the annotation task. For instance, sentences with short phrases such as *if possible* or *if any*, were not considered conditional phrases. Three annotators completed the annotation job for $1,936$ sentences. Since a sentence having a condition can have a resultant of type consequence or action, there could be sentences with two or more types of labeled sequences. Table 2 shows the different data splits of the TechCond dataset with the final number of annotated sequences per label.

4 Model Descriptions

We present three models using both rule-based and deep learning approaches to recognize and split conditional sentences into conditional clause and other resultant categories. The rule-based model implements a pipeline that solves the end-to-end task by solving each subtask. In contrast, we considered this large task as a sequential labeling task for deep learning approach.

Rule-Based Model Syntactic and semantic manually written rules are the basis for many systems that analyze technical documents, for example (Papanikolaou, 2012; Zhang and El-Gohary, 2012). In a nutshell, these approaches adopt a general architecture, which extracts the constituents of the sentence, their type and their dependency relationships using constituency or dependency parsers. Among the extracted entities, there could be conditional constituents which are further analyzed. However, we have found out that for our corpus of technical documents the systems trained on standard corpora, (Marcus et al., 1993; Taylor et al., 2003) do not perform well. A source of error might be the text itself, as the technical corpora is not necessarily written by native English speakers, but most importantly, conditional clauses have many peculiarities, both at syntactic and semantic level, which could be exploited more efficiently than a general approach does.

To achieve a high quality of parsing of the conditional sentences, we developed a system of rules that proceeds sequentially to determine (i) the conditional triggers - using lexical and semantic rules (ii) boundaries of conditional clause - using syntactic and semantic rules, (iii) the border of adjacent

	Rule-based			BERT-based			XLM-R-based			
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	support
Condition	80.52	76.86	78.65	84.21	92.56	**88.19**	78.12	81.52	79.79	242
Action	69.74	70.2	69.97	79.63	85.43	82.43	80.24	88.74	**84.28**	151
Consequence	51.52	55.43	53.4	67.92	78.26	72.73	88.21	89.67	**88.93**	92
Average	71.66	70.72	71.16	79.7	87.63	83.46	**83.82**	**87.84**	**85.75**	485

Table 3: Conditional Sentence Splitting performance of three models on Test set.

constituents - using syntactic rules and (iv) the relationship between constituents and the corresponding conditional - using semantic rules.

The conditional triggers are lexical items, like *if*, *when*, but also the comma itself, *" , "*, or *that*, when it introduces a relative clause. The rules in (i) analyze the context and decide whether a trigger does indeed introduce a conditional. Then, the one predicate condition is used by the rules in (ii), that is, we determine exactly one verbal phrase whose boundaries are actually the boundaries of the conditional clause itself. We used semantic features, like *imperative*, or *nominalization* to decide whether a verbal group carries conditional relevance. Some of these features are extracted directly from ESG output (McCord et al., 2012), and some other are extracted by our rules via a deep analysis of copula, semantic transparent verbs and other verbal constructions that do not necessarily contain exactly one verb. The boundaries ambiguities are resolved by rules in (iii) , which decide whether a noun phrase is a part of a conditional or it belongs to the following clause. The rules in (iv) check for imperative, or imperative like constructions, because a clause expressing an action that logically follows a conditional must be understood as "to do" obligation. That is, these rules rank the candidates according to their imperative valency and their position with respect to the determined conditional clause.

This model achieved an accuracy of about 70% (Table 3).

BERT-based Model One sequence model for conditional splitting was based on the $BERT_{BASE}$ language model (Devlin et al., 2019). This model takes a maximum 512 input word piece token sequence $X = [x_1; x_2; ::: ; x_T]$ and uses a $L = 12$ layer transformer network, with 12 attention heads and 768 embedding dimensions, to output a sequence of contextualized token representations. We used the representation of the first sub-token as the input to the token-level classifier over the conditional label set. We fine-tuned the model using 5 epochs, learning-rate 0.1, and 256 maximum sequence length.

XLM-R-based Model XLM-RoBERTa (XLM-R) is a SOTA multilingual masked language model trained on 2.5 TB of newly created clean CommonCrawl data in 100 languages (Conneau et al., 2019). It obtains strong gains over previous multilingual models like mBERT and XLM on classification, sequence labeling and question answering. We used the $XLM - R$ model with standard settings: max sequence 256, learning rate 7e-5, warmup_proportion 0.1, epoch 5, batch size 8.

5 Experiments

To evaluate the performance of the models, we used the test set from the TechCond dataset (Section 3.3) following the data split shown in Table 2. Since our dataset was annotated at phrase/chunk level, we transformed our annotated data into Inside-Outside-Beggining (IOB) tagging format for the sequential labeling task. For example:

- Phrase level annotation: {*"id": 908, "text": "Include the date if the opt-out period expires.", "meta": , "annotation_approver": "admin", "labels": [[0, 16, "Action"], [17, 47, "Condition"]]*}
- IOB annotation scheme: {*Include B-Action, the I-Action, date I-Action, if B-Condition, the I-Condition, opt-out I-Condition, period I-Condition, expires I-Condition, . O*}

Since we are only interested in having sentences split into three main categories Condition, Action, and Consequence, we focused the evaluation of the models for these labels. Table 3 shows the performance of our models for the end-to-end conditional sentence splitting on the test set of 200 sentences. The XLM-R-based model outperforms other two counterparts by large margins.

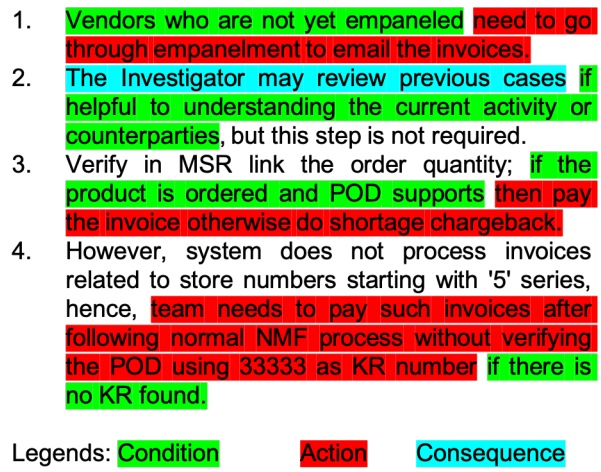

Figure 4: Sentences that our models failed to process. Annotation is shown for expected splitting and categorization.

Error Analysis Figure 4 shows sentences having complex structures that our models failed to recognize and extract correct Condition, Action, and Consequence clauses. Although the rule-based model has a comprehensive set of rules for high accuracy, the coverage is limited for sentences having complex syntactic structures. It struggles to capture non-standard conditional structures, such as relative clause where no Condition clause was found (Example 1); or it captured wrong resultant (Example 2) due to structure complexity. In contrast, deep learning models performed well on (Examples 1 & 2) but failed to capture correct resultant (Example 3) due to the coordination of verbs in the Condition clause; or failed to capture full resultant (Example 4) due to a very long sentence sequence. Our models also have difficulty to process sentences with multiple conditions or multiple resultants. For example:

- ***If you had dynamic SQL*** *(or* ***if you rebound static SQL****), your applications might be breached.*
- *If using PayPal for payment* ***then click on the PayPal tab*** *and* ***then click Pay Now.***

Example 3 shows another limitation that is to differentiate between Unconditional-Action clause *"Verify in MSR link the order quantity"* and the actual Action clause.

6 Conclusions and Future Work

Recognizing and splitting conditional sentences for automation of business process management is a complex and important task for many industries. We presented a non-trivial real-world system that can recognize and split technical instructions (at sentence level) into Condition, Action, Consequence clauses for business process modeling and automatic update. We defined an end-to-end task consisting of three subtasks that fits our business context and needs. We also created a new dataset and three models, using both rule-based and deep learning approaches, to solve this task. For future work, we plan to improve the task performance by 1) expanding the dataset and improving annotation to capture more examples of complex structures, and 2) trying better learning approaches or more linguistics attributes to help our models learn better.

References

Thomas Baier and Jan Mendling. 2013. Bridging abstraction layers in process mining by automated matching of events and activities. In *Business Process Management*, pages 17–32, Berlin, Heidelberg. Springer Berlin Heidelberg.

R. Blumberg and S. Atre. 2003. The problem with unstructured data. *DM Review*, 13:42–49.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. Association for Computational Linguistics.

Zadrozny W. Hematialam H. 2017. Identifying condition-action statements in medical guidelines using domain-independent features. *ArXiv Prepr.* Version 2.

H. Leopold. 2013. *Natural Language in Business Process Models: Theoretical foundations, techniques, and applications*, volume 1. Springer International Publishing.

J. Mendling H. Reijers M. Dumas, M. La Rosa. 2018. *Fundamentals of Business Process Management*, volume 1. Springer-Verlag Berlin Heidelberg.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

M. C. McCord, J. W. Murdock, and B. K. Boguraev. 2012. Deep parsing in watson. *IBM Journal of Research and Development*, 56(3.4):3:1–3:15.

Nick Papanikolaou. 2012. Natural language processing of rules and regulations for compliance in the cloud. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 620–627. Springer.

Vincent J. Phalp, K.T. and K. Cox. 2007. Improving the quality of use case descriptions: empirical assessment of writing guidelines. *Software Quality Journal*, 15(4):383–399.

Chen Qian, Lijie Wen, Mingsheng Long, Y. Li, Akhil Kumar, and J. Wang. 2019. Extracting process graphs from texts via multi-granularity text classification. *arXiv: Computation and Language*.

Ann Taylor, Mitchell Marcus, and Beatrice Santorini. 2003. The penn treebank: an overview. *Treebanks*, pages 5–22.

Kaiser K. Wenzina R. 2013. Identifying condition-action sentences using a heuristic-based information extraction method. *Process Support and Knowledge Representation in Health Care*, page 26–38.

J Zhang and Nora El-Gohary. 2012. Extraction of construction regulatory requirements from textual documents using natural language processing techniques. In *Computing in Civil Engineering (2012)*, pages 453–460.