Natural language techniques supporting decision modelers

Peer-reviewed author version

Arco, Leticia; NAPOLES RUIZ, Gonzalo; VANHOENSHOVEN, Frank; Lara, Ana Laura; Casas, Gladys & VANHOOF, Koen (2021) Natural language techniques supporting decision modelers. In: Data mining and knowledge discovery, 35(1), p. 290-320.

DOI: 10.1007/s10618-020-00718-4
Handle: http://hdl.handle.net/1942/32589

Noname manuscript No.
(will be inserted by the editor)

Natural Language Techniques Supporting Decision Modelers

Leticia Arco · Gonzalo Nápoles · Frank Vanhoenshoven · Ana Laura Lara · Gladys Casas · Koen Vanhoof

Received: date / Accepted: date

Abstract Decision Model and Notation (DMN) has become a relevant topic for organizations since it allows users to control their processes and organizational decisions. The increasing use of DMN decision tables to capture critical business knowledge raises the need for supporting analysis tasks such as the extraction of inputs, outputs and their relations from natural language descriptions. In this paper, we create a stepping stone towards implementing a Natural Language Processing framework to model decisions based on the DMN standard. Our proposal contributes to the generation of decision rules and tables from a single sentence analysis. This framework comprises three phases: (1) discourse and semantic analysis, (2) syntactic analysis and (3) decision table construction. To the best of our knowledge, this is the first attempt devoted to automatically discovering decision rules according to the DMN terminology from natural language descriptions. Aiming at assessing the quality of the resultant decision tables, we have conducted a survey involving 16 DMN

L. Arco
AI Lab, Computer Science Department
Vrije Universiteit Brussel
Pleinlaan 9, 1050 Brussels, 3rd floor, Belgium
E-mail: larcogar@vub.be

L. Arco · G. Nápoles · F. Vanhoenshoven · K. Vanhoof
Business Informatics Group, Faculty of Business Economics
Hasselt University
Diepenbeek Kantoor A50, Hasselt, Belgium

A. L. Lara
AI Lab, Computer Science Department
Central University of Las Villas
Carretera a Camajuaní km 5 1/2, Santa Clara, Villa Clara, Cuba

G. Casas
Weast Coast University
Miami Campus, 9250 NW 36th St, Doral, FL 33178, USA

experts. The results have shown that our framework is able to generate semantically correct tables. It is convenient to mention that our proposal does not aim to replace analysts but support them in creating better models with less effort.

Keywords Decision Modeling and Notation · Decision Rules · Decision Tables · Natural Language Processing

1 Introduction

To facilitate communication on and analysis of process logic and decision logic, businesses often rely on representations such as process diagrams and decision tables. The graphical Business Process Model and Notation (BPMN)[1] that has been developed explicitly for process modeling has been recently complemented with the addition of Decision Modelling and Notation (DMN)[2], aimed explicitly at decision modeling. It is the job of a business analyst to create these processes and decision models based on the knowledge it has gathered from, for example, observations, interviews, policy documents, regulatory documents and legacy code (IIBA, 2009).

Gathering the business rules needed to model the decisions has been coined *rule harvesting* (Boyer and Mili, 2011) and can be a cumbersome and complicated task (Figl et al., 2018). Contrary to decision modeling, process modeling has seen the emergence of some techniques that support the analysts. Process mining has emerged as a technique to avoid subjective interpretations and discover process logic in historical data (Van Der Aalst, 2011). In parallel, Natural Language Processing (NLP) has been applied to discover process flows from written documentation (Riefer et al., 2016).

The task of modelers is challenging since they have to transform ideas, observations, and discussions into a correct representation. Thus, there are various requirements for a person to transition from a novice to a DMN modeler expert: learning the language and its concepts, developing patterns of capturing recurring problems, and deliberate training over a more extended period (Figl et al., 2018). Nevertheless, support for modeling decisions has, until now, received significantly less attention in the academic field. Data mining techniques are indeed applied to discover decision logic. However, their primary goal is mostly to produce decision outcomes themselves rather than providing an interpretable representation of the decision logic. To the best of our knowledge, academics had not yet devoted research to automatically discovering decision rules according to the DMN terminology from natural language descriptions. It would be convenient to develop a framework that supports modelers to create DMN decision tables and diagrams. Initializing the decision discovery phase from natural language descriptions would speed up the process.

[1] http://www.omg.org/spec/BPMN/2.0/
[2] http://www.omg.org/spec/DMN/

In this paper, we create a stepping stone towards implementing an NLP framework to model decisions based on the DMN standard. Our proposal contributes to the generation of decision rules and tables from a single sentence analysis. The framework comprises three phases: (1) discourse and semantic analysis, (2) syntactic analysis and (3) decision table construction. In phases 1 and 2, we directly apply existing NLP techniques, whereas phase 3 is conceptually novel. The framework is meant as a supporting tool and does not entirely replace a business analyst in the modeling task but helping them to create better models in less time. The main contribution of this paper is that we identify and extract the decision rule components from English sentences, without the need for the strict supervision of experts during the rule formalization process. To do this, we design novel algorithms that generate decision rules from the syntax trees and dependency relationships. It is worth mentioning that our proposal –beyond DMN as an underlying model– comprises a suitable starting point to cope with other kinds of problems.

The rest of this paper is organized as follows. Background about DMN and its purpose within BPM is explained in Sect. 2. The principal approaches for extracting rules from texts are presented in Sect. 3. Sect. 4 describes the NLP framework for supporting decision modeling from single sentence analysis. The third phase of our framework is the most complex one and it can be considered the cornerstone supporting our study. Sect. 5 provides its detailed description. Sect. 6 presents the evaluation of our proposal based on the survey results applied to expert modelers. Towards the end, we provide concluding remarks and future research directions.

2 Decision Modeling and Notation and its purpose within Business Process Modeling

Organizations are continuously trying to analyze and improve their business processes (Corradini et al., 2018; De Smedt et al., 2018; Kuss et al., 2018; Satyal et al., 2018). The attempts to improve processes can be supported by representing process knowledge in a structured and standardized format, such as the BPMN standard. Most processes contain decisions; however, BPMN is not very suited to model these decisions. Modeling decisions in BPMN can be made using gateways, which would force decisions to be procedural rather than declarative. Moreover, by adding decisions as gateways, the diagram grows in size and can become hard to understand. Lastly, the procedural logic applied and expressed in BPMN modeling does not trigger the user to think about decisions in a more declarative way. Therefore, rather than being unable to, the user is unlikely to decouple decisions and identify sub-decisions. To fill these gaps, DMN emerged as a recent Object Management Group (OMG) standard (Calvanese et al., 2018; De Smedt et al., 2017; Taylor et al., 2013). It complements BPMN, which does not model the decision logic in detail, with a notation for modeling decision logic and dependencies between decisions and data elements. DMN has incorporated many principles from the

Decision Model (Goldberg and Halle, 2009) for the construction of its decision requirement diagram as well as its decision tables. There are, however, many ways of creating decision tables that are not DMN-compliant. For example, Semantics of Business Vocabulary and Rules (SBVR)[3] and Predictive Model Markup Language (PMML)[4]. Both also cover parts of decision modeling, but neither is considered to be part of DMN (Silver, 2016). Other standards for expressing rules, such as RuleSpeak[5], which are heavily focused on structuring decision logic in text format, are not part of DMN either.

DMN advocates the decomposition of the subsequent decision logic into simple, atomic sub-decisions. DMN involves an unequivocal expression language oriented to business users rather than technical experts. Furthermore, it can express links between decisions, knowledge models such as data mining models and information structures. This standard aims to combine understandability with semantic richness (i.e., the ability to model complex situations). As mentioned, DMN is supported as a global standard and therefore, software vendors commonly accept it. This is evidenced by the inclusion of DMN in software tools such as Signavio[6] and Trisotech[7].

It is crucial to notice the added value of a decision model compared to a text-based requirement document. A decision model communicates through structured diagrams and tables, reducing the ambiguity and making the decision more straightforward to implement in an information system. Efforts have been made to harvest business rules from legacy code (Paknikar et al., 2014; Wang et al., 2004), including some patented techniques (Garza, 2014). Whereas legacy code typically adheres to a structural format, the same can not be said about decisions that are expressed in policy documents. Obtaining DMN-compliant decision models from text documents involves challenging, knowledge-dependent, time-consuming and labor-intensive tasks, especially when large volumes of texts need to be transformed into a structured representation. Such tasks create a significant bottleneck between the semantic content of the source material (expressed in natural language) and the computer-based, automatic use of that content. To address the bottleneck, NLP techniques can be applied.

The literature includes several approaches that use NLP and text mining techniques for supporting business process modeling (Caporale, 2016; Friedrich et al., 2011; Ghose et al., 2007; Gonçalves et al., 2009; Riefer et al., 2016; Sinha and Paradkar, 2010). Although some of these models cover the problem of obtaining BPMN models from natural language process descriptions, the results are still insufficient. All these approaches face the challenge related to the flexible and inherently ambiguous nature of natural languages. Humans have the ability to exploit and interpret such characteristics (van der Aa et al., 2017). Nevertheless, the automatic interpretation power mostly depends on

[3] https://www.omg.org/spec/SBVR/About-SBVR/
[4] http://dmg.org/pmml/v4-3/GeneralStructure.html
[5] http://www.rulespeak.com/en
[6] https://www.signavio.com/
[7] https://www.trisotech.com/

the level of linguistic analysis chosen (Liddy, 1998). Most studies apply techniques related to the lexical, syntactic, semantic and discourse levels. However, graphemic and lexical levels are not enough for supporting modeling from texts since these levels only use statistical facts about the text. On the other hand, they cannot entirely capture the meaning of documents because there is only a weak relationship between term occurrences and document content. In contrast, syntactic, semantic and pragmatic levels try to capture internal relations and more semantic content by exploiting an increasing amount of contextual information. Thus, this challenge demands go in deep in the syntactic, semantic and discourse analysis. On top of that, when it comes to producing DMN models from textual descriptions of decisions, the literature becomes significantly scanter. For this reason, in the next section, we will describe different approaches for extracting rules in diverse textual application domains. Our goal is to identify which tools, levels of linguistic analysis, resources and pipelines could be useful for discovering decision rules.

3 Approaches for extracting rules from texts

Extracting patterns, relationships, and rules from textual information is an issue that has been researched intensely since the 1990s. The first approaches aimed at extracting relationships between elements from natural language texts were AutoSlog (Riloff, 1993), PALKA (Parallel Automatic Linguistic Knowledge Acquisition) (Moldovan, 1995), LIEP (Learning Information Extraction Patterns) (Huffman, 1996), CRYSTAL (Soderland, 1997), WHISK (Soderland, 1999), RAPIER (Robust Automated Production of Information Extraction Rules) (Califf and Mooney, 1999) and $(LP)^2$ (Ciravegna, 1999). Most of these systems perform syntactic analysis, recognizers for domain objects such as person and company names, and discourse processing that makes inferences across sentence boundaries. However, a syntactically-informed representation structure faces the problem of linguistic variations, the phenomenon in which similar semantic content may be expressed in different surface forms. For that reason, other analyses are necessary for some applications. Unfortunately, most of them are domain-specific, need annotated corpus or required human interaction. Most successful algorithms make scarce use of NLP, thus tending to avoid any generalization over the flat word sequence. Thus, they do not identify the relations among words or phrases, because the relations between different patterns was not investigated yet. Establishing such relations is desirable for both the efficiency of representation and the flexibility of interpretation. In addition, it was not usual to use other knowledge sources such as dictionaries or other lexical resources. These systems often rely on approaches based on basic NLP (e.g., using parsing) and require the manual development of resources (e.g., grammars and finite-state automates). Other systems use machine learning techniques, such as inductive and relational learning.

Despite the identified disadvantages, those initial experiences have been successfully used in more recent systems that attempt to discover patterns

and their relationships from textual documents. Although the extraction of patterns from texts arose from the need to enrich information retrieval systems, there are many real applications where extracting patterns and their relationships from texts is of great importance. For instance, a system for extracting protein-protein interactions from scientific texts was proposed in (Ono et al., 1999).

DIRT (Discovery of Inference Rules from Text) (Lin and Pantel, 2001) is an unsupervised method based on dependency trees for discovering inference rules from text. It essentially considers that, if two paths tend to link the same sets of words, then their meanings are similar. Since a path represents a binary relationship, it generates an inference rule for each pair of similar paths. DIRT considers few dependence relations and it uses them for discovering similar meanings.

A better understanding of the clinical narrative text might be gained by identifying and extracting meaningful relationships between the identified entities and events (Demner-Fushman et al., 2009). Another kind of rule (cause-effect relation) is extracted from natural language texts (Sorgente et al., 2013). This approach first identifies a set of plausible cause-effect pairs through a set of logical rules based on the Stanford dependencies relations between words (Marneffe and Manning, 2010). For each lexico-syntactic pattern, a regular expression is defined to recognize the sentences that contain such a pattern.

A framework for the automatic rule extraction from online texts is presented in (Hassanpour et al., 2011). This approach starts with existing domain knowledge in the form of Web Ontology Language (OWL) ontologies and Semantic Web Rule Language (SWRL) rules and applies NLP and text-matching techniques to deduce classes and properties. The authors apply the Stanford Parser to each sentence. Then, they use the obtained dependencies to help find the relationships between terms in the text.

Several works have been developed to extract patterns and relationships from legal documents and regulations (Dragoni et al., 2015, 2016; Papanikolaou, 2012; Wyner and Peters, 2011). In (Wyner and Peters, 2011) the authors proposed an approach for identifying and extracting conditional and deontic rules, specifying the antecedents, consequences, agents, themes, actions, and exceptions. They identify and extract high-level components of rules from regulations in English, thus applying and extending widely available NLP tools (General Architecture for Text Engineering (GATE)[8] and Stanford Analyser[9]). Rules make use of syntactic analyses and lexical-semantic information. To associate grammatical roles with thematic roles, the authors use grammatical information from the Stanford Parser (passive annotation and dependency information) along with information on thematic roles derived from VerbNet[10]. This approach identifies and extracts relevant elements; however, it still has limitations in extracting relationships between identified elements.

[8] https://gate.ac.uk/

[9] https://nlp.stanford.edu/

[10] https://verbs.colorado.edu/verbnet/

In (Papanikolaou, 2012), the author proposed an NLP tool to automatically analyze and extract information from legal and regulatory texts. This approach requires experts to highlight and mark up portions of texts that are to be translated to machine-readable rules. As evidenced in several approaches, automatically identifying in a textual corpus the fragments corresponding to the rules to be described requires a complex discourse analysis. Besides, a database of concepts and relationships was manually built. In (Dragoni et al., 2015), a framework for the automated rule extraction from legal texts is presented. It uses two ontologies, one of them indicates the lexicon and the other one expresses the structure. The framework combines the linguistic information provided by WordNet[11] with a syntax-based rule extraction by exploiting the Stanford Parser, and a logic-based extraction of dependencies between chunks of such texts by using the Combinatory Categorial Grammar (CCG) parser tool[12]. This proposal has the disadvantage that the terms used in the legal text and those used in the rules are not aligned.

A polyvalent framework for acquiring more complex relationships from texts while coding them as rules is presented in (Boufrida and Boufaida, 2014). This approach starts with existing domain knowledge represented as OWL ontology and SWRL rules by applying NLP tools and text-matching techniques to deduce different atoms as classes and properties. This rule extraction system requires two inputs: the existing knowledge and free texts. This approach does not exploit the main benefits of NLP since it only uses the part-of-speech (POS) tags of the terms and defines elementary patterns to identify the relationships between them by a triple [noun modal verb]. Semantic analysis is done in order to bridge the semantic gap between the terms and the vocabulary of the ontology.

In (Liu et al., 2015), the authors proposed a new approach to extract entities and aspects from opinion texts. This work uses dependency relations between opinion words and aspects. Rules are obtained based on syntactical dependency relations. The new task, Emotion-Cause Pair Extraction (ECPE), which aims to extract the potential pairs of emotions and corresponding causes in a document, is proposed in (Xia and Ding, 2019). A manually annotated corpus is required for training the model. Another supervised method for extracting relations from textual data is presented in (Lima et al., 2016). A set of rules is induced from an annotated learning corpus given as input. The set of the induced rules is applied to the candidate relation instances from an unseen document. This process identifies relation instances that are used for populating the domain ontology. The method currently relies on shallow syntactic parsing of sentences, which does not consider semantic aspects relating entities to verbs.

The closest proposal to our paper was published in (Bajwa et al., 2011). The proposal entitled NL2SBVR takes two inputs: the English specification of a business rule and the Unified Modeling Language (UML) class model,

[11] https://wordnet.princeton.edu/

[12] http://groups.inf.ed.ac.uk/ccg/software.html

which provides a business domain. A rule-based parser is used to process the POS tagged information to extract basic SBVR elements, e.g., noun concept, fact type, etc. The SBVR vocabulary is mapped to an SBVR rule using a rule-based approach. Thus, this approach does not need to consider the dependency relation because it only extracts the basic elements from the natural language text. Then, it extracts their relations from the UML class model. However, this approach is not able to obtain business rules only considering the natural language text description, while also being domain-dependent. There have been other efforts to formalize natural language regulations for SBVR (Fortineau et al., 2013; Lévy and Nazarenko, 2013).

The main disadvantages detected in these previous works are: i) most approaches do not exploit the main benefits of NLP since they use a limited number of types of relationships among terms, ii) most of them are domain-dependent or need additional information such as ontologies, diagrams or models, and iii) some of them require human interaction. The main conclusion we can obtain from these related works is that most approaches are based on dependency trees, and they are mostly based on the Stanford dependencies representation (Marneffe and Manning, 2010). As a final remark, it should be noticed that each referenced approach presents a proposal adjusted to the type of rules to be obtained, the application domain, and the main goals when extracting the relationships. Therefore, each approach reuses a few methods from others.

Considering the advantages and disadvantages of the approaches described above, we should determine which tasks are necessary to extract the desired patterns from textual decision descriptions. Sentence breaking, POS tagging and parsing are the most useful syntax subtasks for supporting rule extraction. Determining the syntax tree of each sentence helps identify patterns and their connections in decision descriptions. At the same time, Named Entity Recognition (NER), natural language understanding, and relation extraction are semantics subtasks also frequently used. The following section explains how these tasks, subtasks, and the corresponding techniques are amalgamated to bring our proposal to life.

4 A new framework for supporting decision modeling from texts

In this section, we introduce a new NLP framework that contributes to the generation of decision rules from a single sentence analysis. Our proposal supports the construction of decision tables through three phases, namely: (1) discourse and semantic analysis, (2) syntactic analysis and (3) decision table construction. In phases 1 and 2, we directly apply existing NLP techniques, whereas phase 3 is conceptually novel. Several algorithms were designed that allow generating the decision rules from the outputs of the first two phases.

DMN's two most prominent features are the Decision Requirements Diagram (DRD) and the decision tables (Silver, 2016). When processing single sentences, the creation of the DRD is relatively trivial and may seem redun-

dant. For that reason, to accomplish our goal, we will focus on decision tables. In short, a decision table is a tabular representation of a set of related input and output expressions, organized into rules indicating which output entry applies to a specific set of input entries. As an illustrative example, Fig. 1 shows the decision table corresponding to the description: "If the keys of the insured home were stolen, we would also pay the cost of replacing the locks.". This example illustrates some challenges of automatic decision modeling since a certain level of semantic understanding would be required (e.g., the expert modeler took the liberty of using the word "refund" instead of "pay".).

F	Home	Keys	Cost
	{INSURED,NOT INSURED}	*{STOLEN,NOT STOLEN}*	*{REFUND,DO NOT REFUND}*
1	= INSURED	= STOLEN	REFUND
2	-	-	DO NOT REFUND

Fig. 1 Example of a decision table.

The first challenge is determining the input and output values, and the decision logic from the natural language descriptions in an automated fashion. To cope with this issue, it is necessary to consider the main linguistic characteristics of the decision texts. We examine many documents where decisions were presented; for instance, documents where working procedures contain business rules, policy documents from insurance companies, rules to grant research funding, rules of some sports, and codes of conduct of diverse companies. When analyzing these diverse decisions, we were able to extract the main linguistic characteristics of these texts, which we summarized in Table 1. The second column in Table 1 indicates the features supported by our proposal.

Table 1 Main linguistic characteristics of decision texts.

Main characteristics	Supported
Have long sentences of several coordinated/subordinate clauses	✓
Include sentences that do not contribute to the decision modeling	✗
Use list punctuation, including enumerations, colons, etc.	✗
Contain a mix of punctuation and alpha-numeric characters	✓
Use more than one sentence for expressing one decision	✗
Express more than one decision in one sentence	✓
Describe the same decision in different parts of the text	✗
Use parentheses for describing examples, condition values, etc.	✓
Use co-reference (e.g., different terms for the same entity)	✓
Express some conditions in a negative way	✓
Contain embedded exception clauses	✓
Contain active and passive sentences	✓

These characteristics impose challenges when identifying the elements needed to obtain the decision tables such as the glossary, variable names, output labels, input expressions, input and output values, decision rules and compound outputs. The proposed solution (see Fig. 2) combines NLP techniques to analyze decision descriptions in the discourse, semantic and syntactical levels and procedures which mostly adopt syntactic dependencies between terms for constructing the decision tables. This amalgamation results in an unsupervised proposal that exploits the NLP advantages to obtain decision tables from decision descriptions in natural language automatically.

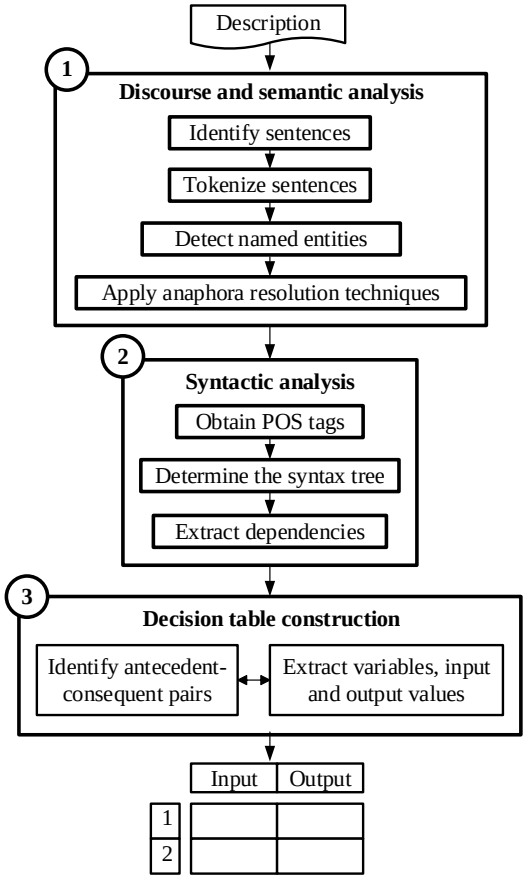

Fig. 2 A new proposal for supporting decision modeling from texts.

Our proposal requires the use of tools that can provide a grammatical structure of the text, which can be exploited for inferring the different components of descriptions. By analyzing the state-of-the-art, as shown in Sect. 3, we noticed that one of the most prominent libraries is the one provided

by Stanford NLP. The Stanford Parser can determine a syntax tree, thus displaying the dependencies between the words of the sentence through the tree structure (Melkuc, 1988). Additionally, each word and phrase is labeled with an appropriate POS tag or phrase-tag. The tags that the Stanford Parser uses are the same that can be found in the Penn Treebank[13] (Marcus et al., 1993). On top of that, the Stanford NLP also produces Stanford Dependencies (De Marneffe et al., 2006; Marneffe and Manning, 2010), which is the main reason for using it in our proposal. These dependencies reflect the grammatical relationships between words, thus producing a grammatical representation of sentences.

4.1 Phase 1. Discourse and semantic analysis

Given a textual description, the first step in Phase 1 refers to the discourse and semantic analysis. By doing so, it is necessary to apply the sentence breaking task to detect the sentence boundaries. The second step is devoted to the tokenization (i.e., the task of cutting it up into pieces, called tokens). The third step performs the detection of named entities. NER seeks to locate named entities in texts and classifies them into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. The Stanford NLP library is used for recognizing general named entities. However, it should be mentioned that it does not have a tool that fits the linguistic style of process descriptions. To cope with this limitation, we analyze dependencies (e.g., dependency *compound*[14]) to recognize more specific entities as illustrated in Sect. 5. As a final step, it is necessary to apply anaphora resolution techniques in this phase to deal with co-reference situations.

As a summary, this phase receives a corpus with the decision description and returns a list of sentences, where the tokens and named entities have already been identified, and the co-reference has been resolved.

4.2 Phase 2. Syntactic analysis

The first step in this phase is to determine the part of speech for each word in a given sentence. Such POS tags were obtained using the Stanford NLP library, which employs the POS name abbreviations agreeing to the Penn Treebank tag set[15]. In our proposal, the syntactic information is derived from the syntax tree and the grammatical relations between the sentence parts. The foundation on which the present paper is grounded is that the syntactic information is valuable when processing lengthy and complex sentences on which particular

[13] The Penn Treebank is a corpus of manually parsed newspaper articles

[14] The description of each dependency tag appears in the Stanford typed dependencies manual (Marneffe and Manning, 2010)

[15] https://catalog.ldc.upenn.edu/docs/LDC99T42/tagguid1.pdf

relationships among terms must be discovered. This contributes to the decision rule generation and supports the construction of decision tables, which is the underlying goal behind our research.

The second step is devoted to determining the syntax tree of each sentence. We utilize the factored model instead of the pure probabilistic context-free grammar in the Stanford Parser. This is motivated by the fact that it provides better results in determining antecedent and consequent clauses, which are essential for the decision rule generation (Friedrich et al., 2011). Considering the Guidelines for Treebank II Style Penn Treebank Project[16], it is possible to recognize the notation used in the syntax tree at three levels: word, phrase and clause. The notation used at the word level exactly matches the POS labels, which will help us identify some sentence parts and their links with antecedents and consequents of the decision rules to be obtained. However, the syntax tree is not enough to find all the desired rule components and their relations. Each phrase and clause identified in the syntax tree allows identifying where to look for dependencies. In a nutshell, the coherent combination between the syntax tree and the dependency relations allows detecting the decision's value expressions, the input variables, and the input and output values.

The third step refers to the extraction of dependencies. A dependency relationship (Hays, 1964) is an asymmetric binary relationship between a word called *head*, and another word called *modifier*. The structure of a sentence can be represented by a set of dependency relationships that form a tree. A word in the sentence may have several modifiers, but each word may modify at most one word. The root of the dependency tree does not modify any word. It is also called the head of the sentence (Lin and Pantel, 2001). Stanford Dependencies[17] allow extracting the dependency tree. The Stanford typed dependency representation was designed to provide a simple description of the grammatical relationships in a sentence that can easily be understood and effectively used by people without linguistic expertise who want to extract textual relations. In the plain text format, a dependency is written as *abbreviated_relation_name*(*governor*, *dependent*) where the *governor* and the *dependent* are words in the sentence to which a number indicating the position of the word in the sentence is appended[18].

Five variants of the typed dependency representation are available in the dependency extraction system provided by Stanford NLP (Marneffe and Manning, 2010). The differences between the five formats are that they range from a more surface-oriented representation, where each token appears as a dependent in a tree, to a more semantically interpreted representation where certain word relationships (such as prepositions) are represented as dependencies, and the set of dependencies becomes a possibly cyclic graph (Marneffe and Manning, 2010). In our proposal, we consider the *collapsed representation*

[16] http://languagelog.ldc.upenn.edu/myl/PennTreebank1995.pdf
[17] https://nlp.stanford.edu/software/stanford-dependencies.shtml
[18] https://nlp.stanford.edu/software/dependencies_manual.pdf

since dependencies involving prepositions, conjuncts, and relative clauses are collapsed to get direct dependencies between content words.

Handling multi-words is useful for detecting the exact terms or expressions corresponding to input and output values and entries from decision descriptions. Stanford NLP identifies some multi-word prepositions, collapsed as prepositional relations, such as two-word-prepositions (e.g., "depending on") and three-word-prepositions (e.g., "in addition to"). Therefore, some dependencies express quantification (e.g., "at least one", "exactly one"), others allow quantifying concepts (e.g., "greater than"), while others express a logical formulation (e.g., "it is obligatory"). Besides, dependencies involving conjunctions will be collapsed into a single relation. For instance, the basic dependencies *cc*(valid-5, and-6) and *conj*(valid-5, signed-7) obtained from the condition "if official documents are valid and signed" will be collapsed into the single relation *conj:and*(valid-5, signed-7). This suggests that the collapsed dependency representation leads to better logic transparency and a smaller gap between modeling and implementation by considering DMN style (Silver, 2016).

As a summary, this phase receives a list of sentences and returns for each sentence its POS tags, syntax tree and list of dependencies.

4.3 Phase 3. Decision table construction

The third phase deals with the extraction of the elements that make up the decision rules. Dependencies are a syntactic resource that allows identifying conditions and actions involved in such rules. Depending on how the descriptions are expressed, we must make use of some or other dependencies. Therefore, the algorithms proposed in this paper to obtain the decision rule elements have been designed to analyze certain forms of expressing decisions. Fortunately, the most common sentences in decision descriptions are expressed in a *condition -> action* and *condition -> actionable-value* forms, and this is a natural way for representing decision rules in decision tables. We show that it is possible to find links between this kind of description and the IF-THEN rules to be included in decision tables. Because the third phase of the framework is the most complex, we will dedicate a whole section to its description.

As a summary, this phase receives a sentence and its corresponding POS tags, syntax tree and list of dependencies. Then, it returns the corresponding set of decision rules formalized in a decision table.

5 Detailed description of the decision table construction

The third phase of our framework is novel and the most complex one. Firstly, we identify a set of plausible antecedent-consequent pairs based on the information provided by the syntax trees and dependencies between words. Then, two general rules are applied: if the sentence contains the keyword "if", dependency *mark* indicates the first antecedent term; if not, dependency *root*

indicates the first consequent term. Secondly, we identify each antecedent and consequent component by extracting the variables, the input and output values from each sentence. The original terms are not replaced by synonyms, symbols, or semantically equivalent expressions. Future implementations of the proposed framework should incorporate a level of semantic understanding that allows convergence in terms that are shared among tables and decisions, as described in Sect. 8.

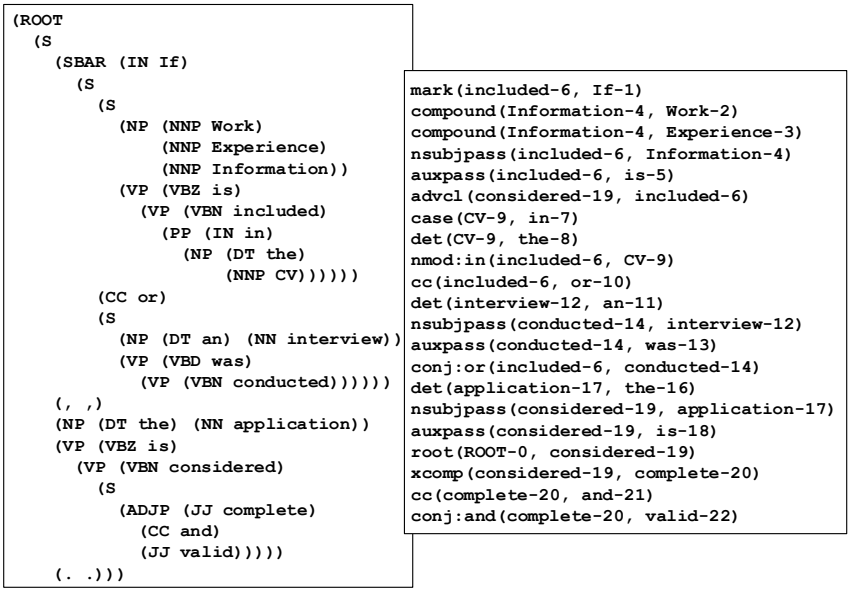

Fig. 3 Syntax tree (left) and collapsed dependency representation (right) corresponding to the sentence: "If Work Experience Information is included in the CV or an interview was conducted, the application is considered complete and valid." .

Identifying the variables, the input and output values and their relations from the syntax tree and the collapsed dependency representation of each sentence is a non-deterministic problem. We identify the regularities in this type of sentences, so we can define several rule-based functions for identifying the rule components and their relationships. The solution we propose allows analyzing a large number of sentences, although some including very complex grammar, could not be covered. Let us consider the following example: "If Work Experience Information is included in the CV or an interview was conducted, the application is considered complete and valid". Fig. 3 shows the syntax tree and dependencies detected by the Stanford NLP for this sentence.

5.1 General procedure for extracting decision rules from isolated sentences

Algorithm 1 displays the general procedure for extracting decision rules from isolated sentences. It results from a generalization process that allows discovering the connection between syntactic structures and decision rules. This procedure receives a decision sentence as input, and the collapsed dependencies and syntax tree corresponding to this input sentence. In short, the algorithm divides the sentence in antecedent and consequent. Subsequently, it detects all antecedent and consequent components useful to create the decision rules corresponding to the input description. Each antecedent component consists of an input expression and an input entry, while each consequent component consists of an output name and an output entry. Sentences with the keyword "if", like the previous example, are much easier to process. In these cases, Algorithm 1 identifies the term "if" using the *mark* dependency; otherwise, it is helped by the *root* dependency to process the rest of the sentences. In a decision table, the input conditions of any decision rule must be logically ANDed together (i. e., joined by conjunctions expressed, for example, by the terms "and" or "but"). Combining them with logical OR is not allowed by the DMN standard because the order of operations is ambiguous without parentheses (Silver, 2016). For that reason, in the presence of OR disjunction, more than one rule will usually be generated from an input sentence.

Algorithm 1 ExtractDecisionRules

Input: A decision sentence, the collapsed dependencies and syntax tree corresponding to this input sentence
Output: Decision rules corresponding to the input decision sentence

```
1: if exists dependency mark then
2:     FirstAntecTerm ← Term related to "if" by dependency mark
3:     FirstConseqTerm ← Term related to FirstAntecTerm by dependency advcl
4: else
5:     FirstConseqTerm ← Root term by dependency root
6:     FirstAntecTerm ← Term related to FirstConseqTerm by dependency nsubj
7: end if
8: ACompList, ARel ← CompleteElements(FirstAntecTerm)
9: CCompList, CRel ← CompleteElements(FirstConseqTerm)
10: RuleList ← CombineAntecConseq(ACompList, ARel, CCompList, CRel)
```

Algorithm 1 looks for dependency *mark*, which is considered the word introducing a finite clause subordinate to another clause (line 1). This mark is dependent on the subordinate clause head. As shown in Fig. 3 (right), *mark*(included-6, If-1) allows identifying the term "included" which belongs to the first antecedent rule component (line 2). Sometimes descriptions appear in the consequent-antecedent order, for example: "An order may be accepted if the credit is good.". Fortunately, *mark* dependency always indicates the condition when the keyword "if" is explicitly in the sentence. In those sentences where the keyword "if" is not present, it is necessary to look for the root word that is always an action in the consequent (line 4). Let us consider the fol-

lowing example: “Every employee receives at least 22 vacation days.”. Fig. 4 shows the syntax tree and dependencies detected by the Stanford NLP for this sentence. Note in Fig. 4 (right) that *root*(ROOT-0, receives-3) allows identifying the term “receives” which belongs to the first consequent rule component (line 5).

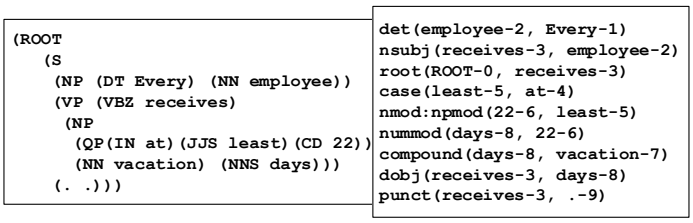

Fig. 4 Syntax tree (left) and collapsed dependency representation (right) corresponding to the sentence: “Every employee receives at least 22 vacation days.” .

If an antecedent term is accessed, it is necessary to look for its connection with a consequent term. Then, we must identify the first consequent term (*FirstConseqTerm*). A relevant dependency for detecting the relation between the antecedent and the consequent of a rule is *advcl*. An adverbial clause modifier (*advcl*) of a VP or S is a clause modifying the verb (temporal clause, consequence, conditional clause, purpose clause, etc.). As Fig. 3 (right) shows, *advcl*(considered-19, included-6) connects the principal word in the antecedent with the principal word in the consequent. In a nutshell, this dependency allows moving from the antecedent to the consequent. For that reason, algorithm *ExtractDecisionRules* searches for dependency *advcl* in order to extract the first consequent term (*FirstConseqTerm*) (line 3). Otherwise, if we have accessed a consequent term, then the *nsubj* dependency will help us connect with an antecedent term (line 6). A nominal subject, indicated by *nsubj*, is a noun phrase which is the syntactic subject of a clause. The governor of this relationship might not always be a verb: when the verb is a copular verb, the root of the clause is the complement of the copular verb, which can be an adjective or noun. As Fig. 4 (right) shows, *nsubj*(receives-3, employee-2) connects the a word in the consequent with a word in the antecedent. In a nutshell, this dependency allows moving from the consequent to the antecedent.

5.2 Extracting all antecedent and consequent components

After extracting one antecedent and one consequent term, we need to extract all antecedent and consequent components. By doing so, we must call the function *CompleteElements* (lines 8-9). This function, as shown in Algorithm 2, receives an antecedent/consequent term and generate all antecedent/consequent components and their relations. The first step is devoted to identifying the first antecedent/consequent rule component (*FirstRuleComp*), corresponding

to the first term (*FirstTerm*) (line 1). The *ExtractRuleComp* function is called by the *CompleteElements* function every time it is necessary to complete an antecedent or consequent component, as shown in lines 1 and 7 of Algorithm 2.

Algorithm 2 CompleteElements

Input: A decision sentence, an antecedent or a consequent term, the collapsed dependencies and syntax tree corresponding to this input sentence
Output: A list of all antecedent or consequent components depending on which kind of term was introduced and the relation among these components

```
1: FirstRuleComp ← ExtractRuleComp(FirstTerm)
2: Rel.update(FirstRuleComp)
3: CompList.add(FirstRuleComp as UNMARKED)
4: while exist a component Comp UNMARKED in the CompList do
5:     Mark Comp
6:     if exists any term T in Comp related to other by dependencies conj : and or
   conj : or then
7:         RuleComp ← ExtractRuleComp(T)
8:         if RuleComp ∉ CompList then
9:             CompList.add(RuleComp as UNMARKED)
10:        end if
11:        Rel.update(RuleComp)
12:    end if
13: end while
```

The function *ExtractRuleComp* (shown in Algorithm 3) helps compute a whole rule component, which consists of the component name, the component value and the negation if present. *ExtractRuleComp* receives a term (*Term*), so it uses the decision sentence, the collapsed dependencies and the syntax tree corresponding to this input sentence for constructing the antecedent/consequent component. When completing an antecedent component, the component name corresponds to the input entry, and the component value corresponds to the input expression. On the other hand, when completing a consequent component, the component name corresponds to the output name, and the component value corresponds to the output entry.

Dependencies *nsubpass* and *nsubj* are used for detecting the link between component name and value. Algorithm 3 needs to identify whether the term received as input is part of the input entry/output name or input expression/output entry, since this depends on the way in which the decision is presented. Considering the example described in Fig. 3, when the function *ExtractRuleComp* receives the first antecedent term "included", it is able to use dependency *nsubjpass*(included-6, Information-4) for constructing the corresponding antecedent rule component. Also, dependencies *compound*, *auxpass*, *case*, *det* and *nmod* are necessary for this analysis. Finally, lines 11-12 are very important in algorithm *ExtractRuleComp*. Dependency *neg*, if it appears, changes completely the meaning of the condition.

For following up the component name and component value expressions, the *CompleteName* and *CompleteValue* functions are useful. These functions

Algorithm 3 ExtractRuleComp

Input: A term (the collapsed dependencies and syntax tree corresponding to the input sentence are also used)
Output: The rule component corresponding to the input term

1: $RelatedTerm \leftarrow$ term related to $Term$ by dependencies *nsubpass* or *nsubj*
2: **if** $Term$ is a name **then**
3: $CompName \leftarrow Term$
4: $CompValue \leftarrow RelatedTerm$
5: **else**
6: $CompValue \leftarrow Term$
7: $CompName \leftarrow RelatedTerm$
8: **end if**
9: $RuleComp.FullName \leftarrow CompleteName(CompName)$
10: $RuleComp.FullValue \leftarrow CompleteValue(CompValue)$
11: **if** exists any term T related to *not* by the depedency *neg* **then**
12: $RuleComp.update$
13: **end if**

search for a sequence of *compound* dependencies (and other auxiliary dependencies [19]) between terms that belong to the same noun phrase in the syntax tree. It should be highlighted that we obtained the multi-word input expression "Work Experience Information" by using the *compound* dependency (see Fig. 3 (right)). Here the following question arises: how to concatenate the terms expressed in such dependencies to properly name the entries and their values? The OMG standards normally try to avoid anything that smacks of "methodology" or "stylistic recommendations". However, the DMN style (Silver, 2016) adds constraints to provide a more uniform look and feel, greater logic transparency and ease to use, and a smaller gap between modeling and implementation. Hence, we try to follow these constraints in the antecedent and consequent component construction, such as naming. Naming the entries and their values in an automated way ensures homogeneity and complies with the defined styles in an easier way when compared with the inconsistencies that might be introduced by modelers.

We already know how to complete a specific antecedent or consequent component, but we have not yet described how to generate all antecedent or consequent components. To achieve this, we need to return to the Algorithm 2 and describe which dependencies help us find all components. More components are identified by searching each *conj*:*and* or *conj*:*or* dependency and looking for each simple declarative clause which is connected with the (*cc and*) or (*cc or*) node in the syntax tree. The POS tag related to the words in conjunctions or disjunctions is quite important since such logical operators can connect, for instance, two input entries of the same input expression, or two different input expressions. Although in the line 6 we only refer to *conj*:*and* and *conj*:*or* dependencies, these conditional sentences should consider some exceptions. For example, "if" can be used as a conjunction in the textual descriptions. When looking for more rule components, it is not only important to

[19] *amod*, *aux*, *auxpass*, *case*, *compound*, *cop*, *dep*, *det*, *dobj*, *mark*, *mwe*, *nmod*, *nummod*, *xcomp* and *or*

consider the *conj*:*and* and *conj*:*or* dependencies, but to search other *mark* dependencies that might connect the antecedent components with term "if" as a conjunction. In some descriptions, the term "but" also expresses conjunction, so it is also necessary to search for *conj*:*but*. Finally, the syntax tree allows identifying multiple antecedents or consequent components split by commas (,) through its clause structure.

Rel is a list of antecedent/consequent components, where their AND/OR relations are stored. Each time a new antecedent/consequent is discovered, this list must be updated to derive the sets of antecedents/consequents to be concatenated with AND conjunctions in each rule. Antecedent components concatenated with logical OR will imply creating new rules since the input conditions of any decision rule must be ANDed together. The same happens when concatenating consequent elements with OR disjunctions. Aside from that, the component list (*CompList*) also must be updated to store all the elements that have been found. Each new antecedent/consequent component is stored as UNMARKED, since it will be analyzed by the *ExtractRuleComp* algorithm to find new components. The example described in Fig. 3 (right) shows when the "or" disjunction suggests the new rule generation. A second antecedent component is identified starting from dependency *conj:or*(included-6, conducted-14). This dependency links two input entries related to two different input expressions ("Work Experience Information" and "CV"). Besides, the syntax tree offers good information: the (*CC or*) node connects two different simple declarative clauses, as portrayed in Fig. 3 (left).

Starting with the output entry and analyzing the dependencies and the simple declarative clause in the syntax tree, it is possible to obtain all consequent components, as we illustrated for the antecedent. In our example, see Fig. 3 (right), dependency *conj:and*(complete-20, valid-22) allows identifying two consequent components. The syntax tree in Fig. 3 (left) shows that this dependency connects two different output entries ("complete" and "valid") that are associated with the same output name ("application"). If we change the decision by: "the application is considered complete and the registration is valid", then the analysis must be different. The dependencies *conj:and*(complete-20, valid-22) (from the original example) and *conj:and*(considered-19, valid-25) (from the modified example) seem to represent the same kind of conjunctions, as shown in Fig. 3 and Fig. 5. However, the syntax tree (see Fig. 5 (left)) illustrates that dependency *conj:and*(considered-19, valid-25) links two different output entry fragments ("considered" and "valid") that are associated with two different output names ("application") and ("registration").

It is important to highlight that the *ExtractRuleComp* function, as well as, line 6 in Algorithm 2, have a relevant role in our proposal. *ExtractRuleComp* is able to identify when a conjunction/disjunction involves the same or different output entries or output names, or input entries or input expressions. Line 6 in Algorithm 2 search for *conj:or* or *conj:and* dependencies, whereas Algorithm 3 search for each simple declarative clause which is connected by (*CC or* or *CC and*) in the syntax tree.

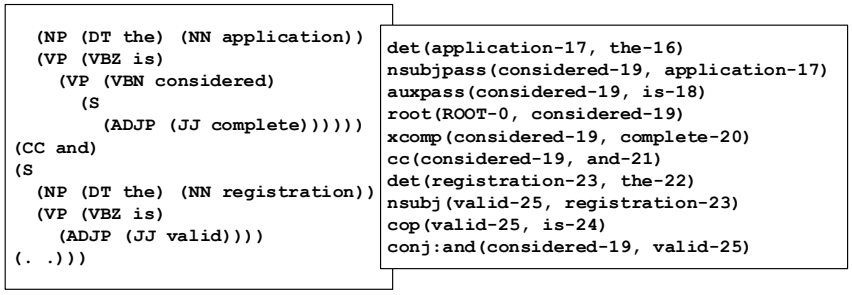

Fig. 5 Excerpt of the syntax tree (left) and collapsed dependency representation (right) corresponding to the sentence: "If Work Experience Information is included in the CV or an interview was conducted, the application is considered complete and the registration is valid." .

Likewise, it is necessary to take into account the POS tag assigned to the words related by *conj:or* or *conj:and*, since disjunctions/conjunctions could connect two entries of the same input expression or the same output name, or connect two different input expressions or two different output names. Hence, the POS tags, the syntax tree and the *conj* dependency are quite useful for detecting other antecedent and consequent components.

Finally, decision rules are generated in line 10 of Algorithm 1. The *CombineAntecConseq* function harmonically combines the information contained in variables $AcompList$, $ARel$, $CCompList$ and $CRel$, and formalizes each decision rule discovered from the input description. Variables $AcompList$ and $ARel$ involve the antecedent components and their relations, whereas variables $CCompList$ and $CRel$ are related to the consequent components. Fig. 6 shows the decision table obtained by DecisionRuleMiner[20] for our toy example, where two rules were generated by Algorithm 1. This is an *any* decision table, but *first* would work as well. In fact, DecisionRuleMiner usually generates first hit decision tables everywhere.

	Input	**Input**	**Output**	**Output**
	Work Experience Information	interview	application	application
1	is included in CV		is complete	is valid
2		was conducted	is complete	is valid

Fig. 6 Decision table for the description: "If Work Experience Information is included in the CV or an interview was conducted, the application is considered complete and valid." .

Our proposal is desirable when modeling real-world situations since it is domain-independent and unsupervised (i.e., there is no need to learn how to

[20] DecisionRuleMiner, the developed prototype tool that implements the main stages of our framework, will be described in Sect. 6.

extract the rules, which builds a massive annotated data set of descriptions as for supervised machine learning approaches). It extracts decision rules through algorithms that use dependency relationships and syntax trees for identifying input entries and expressions, and output names and entries. All this allows support modelers to create DMN decision tables and diagrams and to initialize the decision discovery phase from natural language descriptions in less time.

6 Evaluation

Aiming to evaluate our proposal, we applied it over various decision descriptions collected from textbooks, academia, industry and public sectors, with different characteristics. The overall goal is to assess whether the obtained decision rules truly represent the decision logic expressed in each description. To fulfill this goal, we developed the prototype tool DecisionRuleMiner that implements the main stages of our framework. DecisionRuleMiner returns as output the decision tables corresponding to the decision rules generated by Algorithm 1. Our tool creates a decision table with as many rows as rules were obtained by Algorithm 1, and as many columns as inputs and outputs. Each cell will be filled with the values corresponding to the inputs and outputs defined for each rule.

DecisionRuleMiner was written in Java for easy integration with Stanford NLP. Our tool processes sentence-by-sentence while assuming that the anaphora resolution step was done. The tool also offers tokens, POS tags, syntax trees, dependencies, and recognized named entities of each processed sentence.

To the best of our knowledge, there is no metric able to quantify the fitting between a textual description and its rule-based representation in a decision table. The lack of a fitting metric implies that we cannot use an error function to automatically and objectively assess the performance of the tool. Moreover, it seems unfair to use a binary approach, where the generated decision table and the actual decision table are either an exact match or no match at all. Thus, a binary approach would be too harsh in a field that is subject to contextualization and interpretation. After all, there are different ways of expressing the same logic that all may be equally valid. For these reasons, we decided to rely on human judgment and presented the results of the DecisionRuleMiner to a panel of experts.

The rules were evaluated by 16 expert modelers through a survey. For each combination of rule statements and tables, experts were requested to rate the decision table that had been generated by the tool. Apart from the generated table, the panel received the same input provided to DecisionRuleMiner, being the decision sentence. Afterward, they had to evaluate the *semantic* as well as *syntactic* correctness of the generated decision table. Sect. 6.1 details the conformation of our collection, while Sect. 6.2 provides more insights into the survey itself. Finally, Sect. 6.3 presents the main findings of the experiments, whereas Sect. 6.4 discusses some limitations of our evaluation.

6.1 Collection of decision descriptions

Aiming at creating the collection, we examine many documents where decisions were presented; for instance, documents from the Atlanta police department where working procedures contain business rules [21], policy documents from insurance companies [22] [23] [24], rules to grant research funding [25] [26] and football rules [27]. Moreover, we included sentences from codes of conduct of diverse companies, such as: AIRBUS [28], Apple [29], Citi [30], Beiersdorf [31], Walt Disney [32], IBM [33], Lidl [34], PMI [35], SANDVIK [36], PEPSICO [37] and Google [38]. These documents are publicly available, which facilitates the reproducibility of results.

In our study, we identified and analyzed nine different types of sentences. Table 2 portrays the number of sentences included in our collection for each identified type of sentence. We have inspected the structure of sentences that use the keyword "if" to indicate the beginning of the conditions. Additionally, we also inspected various sentences that do not present the keyword "if". The first eight types of sentences describe in detail the variants in which simple and complex antecedents and consequents can be presented. This study includes both IF-THEN and THEN-IF sentences with varying complexity in the antecedents and consequents.

The 95 sentences included in the collection express decision logic, however, some could be classified as decision/process logic while others could be labeled as decision/data logic. The selected sentences also cover different ways of expressing the decision logic (Yarahmadi, 2018): obligation, condition, prohibition and permission. *Obligation* is the most common category, while *Condition* is the less frequent one, although this category is regularly embedded

[21] http://www.atlantapd.org/Home/ShowDocument?id=810

[22] http://docplayer.fr/82860665-Ing-home-family-insurance-general-conditions-tenant.html

[23] https://www.chubb.com/us-en/terms-of-use.aspx

[24] https://www.esecutive.com/pdfs/Liability_Insurance_Conditions.pdf

[25] http://www.fwo.be/en/general-regulations/

[26] http://eureka-sd-project.eu/general_information?lang=en

[27] https://www.fifa.com/mm/document/footballdevelopment/refereeing/81/42/36/lawsofthegame_2012_e.pdf

[28] https://www.airbus.com/content/dam/corporate-topics/corporate-social-responsibility/ethics-and-compliance/Airbus-Ethics-Compliance-Code-Conduct-EN.pdf

[29] https://www.apple.com/supplier-responsibility/pdf/Apple_SR_2018_Progress_Report.pdf

[30] https://www.citigroup.com/citi/investor/data/codeconduct_en.pdf

[31] https://www.beiersdorf.com/investors/corporate-governance/code-of-conduct

[32] https://ditm-twdc-us.storage.googleapis.com/Manufacturer-Code-of-Conduct-Translations.pdf

[33] https://www.ibm.com/investor/pdf/BCG_Feb_2011_English_CE.pdf

[34] https://www.rspo.org/file/acop/lidl-stiftung-cokg/R-Policies-to-PNC-laborrights.pdf

[35] https://www.pmi.org/-/media/pmi/documents/public/pdf/ethics/pmi-code-of-ethics.pdf

[36] https://www.home.sandvik/en/about-us/sustainable-business/code-of-conduct/

[37] https://www.pepsico.com/Assets/Download/CodeOfConduct/English_GCOC_2014.pdf

[38] https://abc.xyz/investor/other/google-code-of-conduct.html

Table 2 Number of sentences for each identified type included in the dataset.

Type of sentences	Quantity
A) IF-THEN simple antecedent and consequent	14
B) IF-THEN simple antecedent and complex consequent	10
C) IF-THEN simple complex and simple consequent	8
D) IF-THEN complex antecedent and consequent	6
E) THEN-IF simple antecedent and consequent	22
F) THEN-IF simple antecedent and complex consequent	12
G) THEN-IF complex antecedent and simple consequent	6
H) THEN-IF complex antecedent and consequent	6
I) Diverse sentences where keyword "if" is not present	11

in most sentences. On the other hand, we can also categorize rules according to various OMG categories[39], such as eligibility or approval, calculation, validation, risk, fraud, and targeting, among others. Rules classified as *Eligibility or Approval* and *Validation* predominate in our collection. Our approach does not currently use these distinctions, nor does it automatically detect one of these labels. Since we lack a standard, accepted set of benchmark rules, these labels indicate to the reader that we are using a diverse sample of rule statements. Besides, sentences that form our collection are heterogeneous in dimensions, such as size, purpose, complexity, and domain. For instance, they describe the decision from different perspectives (e.g., "If the customer", "senior management approval is required", "If you are", "A single line indicates", "If no funding" and "If there are") so that they describe the context in different ways. Although, in our approach, a variety of patterns is more considered than a variety in context. All these issues make the collection diverse enough, which is needed to assess the solutions provided by our framework properly.

6.2 Survey

Once the collection was formed, the 95 decision sentences were processed by the DecisionRuleMiner prototype tool. Later on, such textual descriptions and the generated rules, exposed in the obtained decision tables, were shown to the expert modelers to evaluate the decision tables and their corresponding rules based on the following scale:

- *Good* - I can use the decision table as it is (semantically and syntactically correct)
- *Somewhat good* - I can use the decision table but it requires minor modifications (semantically correct with syntactic issues)
- *Average* - I can only use the decision table as a starting point (minor semantic issues with no/minor syntactic issues)

[39] https://www.omg.org/news/whitepapers/An_Introduction_to_Decision_Modeling_with_DMNv51-15-15

- *Somewhat bad* - I can hardly use the decision table (minor semantic issues with major syntactic issues)
- *Bad* - I cannot use the decision table (major semantic and syntactic issues)

The scale was defined by taking into account the semantic and syntactic correctness of decision tables. More specifically, a *semantically correct* decision table represents the actual decision logic of the business rule accurately but may use terms that are slightly odd or uncommon. A *syntactically correct* decision table correctly uses the terms but may misrepresent the actual logic.

The panel of 16 experts was formed by business professionals who work with decision tables when they perform business analysis tasks and academics who perform research in decision management or teach courses related to decision tables and similar techniques at the university level.

The survey was conducted in the Qualtrics tool[40]. Three randomly selected descriptions and their corresponding automatically generated decision tables for each type of decision sentence were shown to experts, except for types A, E, and I, where four descriptions and their corresponding decision tables were presented. Types A and E are the most common types in practice, whereas type I sentences impose additional processing challenges. The survey also offers experts the opportunity to express their opinions, suggestions, and comments on DecisionRuleMiner's performance. This qualitative evaluation was also useful to reach certain conclusions and define future work.

To be more precise, we illustrate some examples presented to the experts. Fig. 8 - 16 show selected sentences per defined type and their corresponding decision tables obtained by DecisionRuleMiner.

	Input	**Output**
	claimant_history	claim
1	fraudulent	considered fraudulent

Fig. 7 Generated decision table for the Type A description: "If the claimant has a fraudulent history, a claim must be considered potentially fraudulent.".

	Input		**Output**
	customer_funds	requested amount	withdrawal
1	sufficient	below 15000e	approved

Fig. 8 Generated decision table for the Type B description: "If the customer has sufficient funds and the requested amount is below 15000e, the withdrawal is approved.".

[40] https://www.uhasselt.be/Qualtrics

	Input	Output	
	published method	quotation marks	the source
1	directly quoted	use	cite

Fig. 9 Generated decision table for the Type C description: "If a previously published method is directly quoted, you must use quotation marks and cite the source.".

	Input		Output	
	official documents	official documents	application	student profile
1	are valid	signed	successful	opened

Fig. 10 Generated decision table for the Type D description: "If official documents are valid and signed, the application is successful and student profile is opened in the Core University System.".

	Input	Output
	customer	order
1	out-of-state	credit-checked

Fig. 11 Generated decision table for the Type E description: "An order must be credit-checked if the customer is out-of-state.".

	Input		Output
	payment	credit	order
1	received		shipped
2		good	shipped

Fig. 12 Generated decision table for the Type F description: "An order may be shipped if payment for the order has been received or the customer's credit is good.".

	Input	Output	
	computers_performance	computers_to sellers	computers
1	not good	sent back	rejected

Fig. 13 Generated decision table for the Type G description: "Computers are sent back to sellers and rejected, if they do not have good performance.".

6.3 Results

The evaluations carried out by experts were transformed into numerical values (good - 5, somewhat good - 4, average - 3, somewhat bad - 2 and bad - 1) to facilitate the application of descriptive statistics. First, we calculate the variance and the range of evaluations assigned to each decision table to explore

	Input	Output	
	solutions	Solutions_to students	solutions
1	are incorrect	sent back	deleted
2	incomplete	sent back	deleted

Fig. 14 Generated decision table for the Type H description: "Solutions are sent back to students and deleted, if they are incorrect or incomplete.".

	Input	Output
	employee	extra days
1	is a veteran	2

Fig. 15 Generated decision table for the Type I description: "An employee is a veteran, 2 extra days can be given.".

	Input	Output
	Employees age	days
1	45 or more	2

Fig. 16 Generated decision table for the Type I description: "These 2 days can also be provided for employees of age 45 or more.".

the coincidence of evaluations among experts. The maximum variance was 3.50, while 23.17% of the sentences received evaluations with a variance equal to or higher than 1.5. Likewise, the range of evaluations per sentence was 3 or 4 for the 36.84% of the sentences. To cope with this high dispersion, outlier evaluations per sentence and sentences with less than two evaluations were excluded. This resulted in a dataset where only 5.26% of sentences have a variance higher or equal to 1.5. The 16.84% of the sentences have the range of evaluations equal to 3 or 4.

Most generated decision tables (68.27%) were evaluated as *Good* or *Somewhat good*, which confirms that most of the obtained decision tables are semantically correct, as shown in Fig. 17. Fig. 18 depicts the mean of expert evaluations for each type of sentence, which indicates that, in most cases, the obtained decision tables are semantically correct.

The significance value attached to the Kruskal-Wallis test was 0.015. Thus, we can conclude that there are significant differences among the quality of decision tables obtained for each sentence type. The best-evaluated decision tables were obtained from type I descriptions, while the worst ones were those from type C descriptions, as depicted in Fig. 19.

It seems interesting to analyze possible causes behind the results obtained for decision tables generated from type C descriptions. Firstly, it is noticeable that the mean range is 3 for this type of sentence. Besides, some sentences received all possible scores of the defined scale. Both results allow us to con-

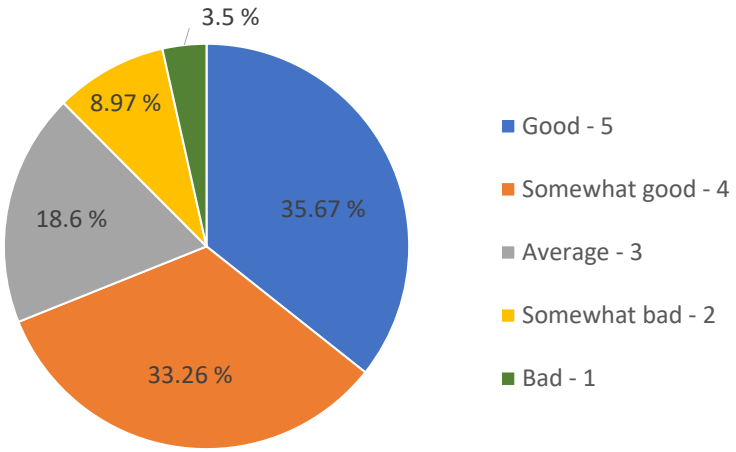

Fig. 17 Distribution of evaluation scores.

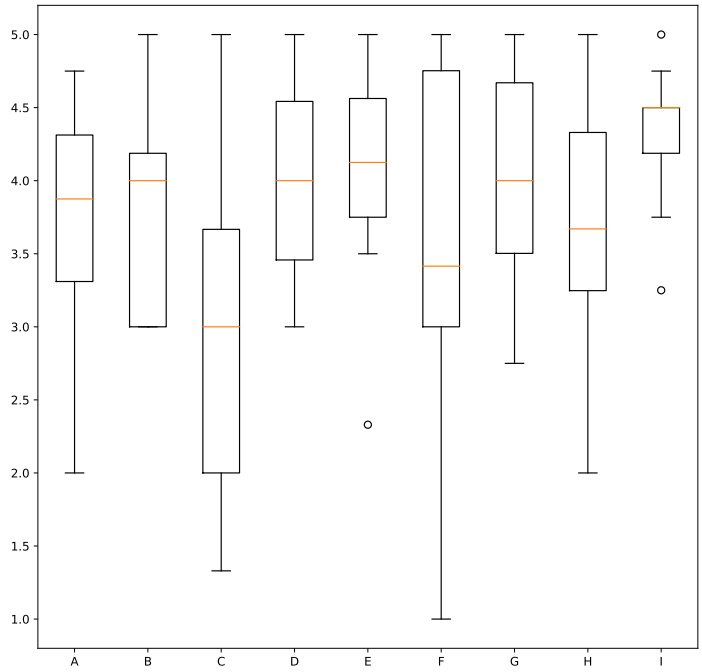

Fig. 18 Mean of evaluations for each type of sentence.

clude that the experts' consensus is shallow when evaluating the corresponding decision tables. Since these types of sentences are not common in real-world scenarios, few sentences of this type were included in our study. On the other hand, the complexity of the consequent part leads to different viewpoints when modeling the process. This means that some experts simplify the rules while

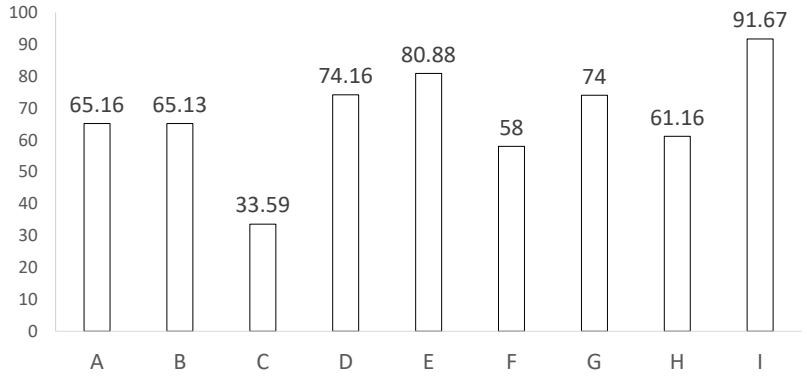

Fig. 19 Mean ranks of evaluations assigned to each type of sentences.

others expect more explicit decision rules, as they commented and exemplified at the end of the survey.

6.4 Discussion

The evaluation shows that our framework is able to generate correct decision tables from decision-type descriptions. The semantic correctness of the obtained rules is highlighted, evidencing that the antecedent-consequent pairs and their components were correctly obtained in most cases. However, some syntactic issues present in the input entries and expressions, and the output names and entries extraction, were detected by experts.

Despite the positive results, some limitations related to our proposal and the evaluation method should be discussed. Concerning the former, it is fair to mention that the Stanford NLP is not entirely accurate. Secondly, the syntax used by DecisionRuleMiner to express the decision rules does not fully match the syntax used by expert modelers, in some cases. Being more explicit, sometimes expert modelers reduce the decision rule to a binary statement (e.g., "when something is smaller than 500"). In contrast, DecisionRuleMiner keeps the comparison expression where the input name is the name of the variable, and the cell value is the expression "smaller than 500". Sometimes the experts do not match how DecisionRuleMiner expresses the name and value entries. For example, given the input expression "Student_exam" and the input entry "passed", some modelers prefer to transform the expressions as follows: input expression "Student passed exam" and input entry "True". The above examples, expressed by experts at the end of the survey, illustrate subjective and objective issues that make automatic modeling and its evaluation difficult.

As a result of these limitations, in some cases, our tool extracts the decision tables following a convention that does not always fit the expectations of experts. An intermediate language could be added in order to translate the obtained results into the language desired by the experts or by tools to be

used in later stages. Nevertheless, it can be concluded that the essential elements of the rules were correctly identified, which was the critical challenge in our study. Besides, there is the agreement that the results are semantically correct, and therefore the generated tables can support modelers to detect the antecedent and consequent components quickly. This implies that our proposal reduces the effort required to model decision rules and tables from texts.

Concerning the limitations of the evaluation, we would like to point out that the quantitative results are bound to the types of sentences used in our study. In fact, creating a statistically representative dataset is difficult since natural language offers a high degree of freedom. However, we tried to create a dataset as heterogeneous as possible by collecting descriptions from various external sources. However, more variety would still be desirable. On the other hand, the experts were not provided with the convention used by DecisionRuleMiner to specify noun_attributes expressions. This could have negatively affected the evaluations to some extent. Moreover, for the sake of simplicity, we asked experts to evaluate the quality on a single-dimensional scale, which was a combination of two dimensions (syntactical correctness and semantic correctness). Finally, how the survey was presented to the experts produces a bias in the evaluation, to some extent. Experts are conditioned by what they see; therefore, the evaluation is not performed independently. Ideally, we should ask the experts to model the decisions and then automatically compare the tables generated by the system and those obtained by the experts independently. This requires the definition and implementation of criteria that allow for the comparison of tables and their components.

7 Conclusions

This paper has presented an unsupervised NLP-based framework to support decision modelers according to the DMN standard. Our solution can identify and extract the components of decision rules from decision descriptions in English by defining some phases and functions that harmonically employ several NLP techniques. To the best of our knowledge, this is the first attempt to automatically discover decision rules and tables according to the DMN terminology from natural language descriptions. It is worth mentioning that our framework does not aim to replace the analysts but to support them to create better models with less effort.

More specifically, we contribute to the decision rule generation and the automatic decision table construction through three well-defined phases: (1) discourse and semantic analysis, (2) syntactic analysis and (3) decision table construction. The proposed algorithms based on dependencies between words, the syntax tree and the POS tags of each word obtained by the Stanford NLP allow extracting the variables, the input and output values from single sentences.

We have evaluated the feasibility of our proposal through a survey applied to expert modelers. The results have shown a high satisfaction degree of experts for the semantic correctness of generated decision tables.

8 Future work

In spite of the promising results, many questions remain open. Here are some lines of future work:

- The single sentence limits the applicability of the current research. Next-step research would be the aggregation of these decision tables into larger ones. Preferably, these decision tables would be linked to a DRD diagram. On the other hand, using an entire text as input and working its way down towards decision tables can be a sensible option as well, but would be an alternative research route focused on discourse analysis.
- We consider the hit policy of a decision table to be primarily a design choice of the decision modeler or business analyst, rather than a requirement imposed by the decision logic. Thus, the current implementation of DecisionRuleMiner produces first hit decision tables. Other hit policies can become part of future extensions to make the tool more flexible.
- We work within the current limitations of NLP. The existing discourse analysis (e.g., anaphora and co-reference resolution) and semantic approaches still have low effectiveness. The analysis and development of specific NLP methods to process textual descriptions, as well as the use of the Structured Business Vocabulary and domain ontologies, should boost DecisionRuleMiner as well.
- The identification of the sentences from a given business text was made manually. There are several solutions available that allow us to delimit a text in sentences with high effectiveness automatically. However, the main challenge here is not to delimit the text in sentences but to identify which sentences contribute to the construction of decision rules, i.e., which sentences express decisions. Determine whether a sentence is a decision is an interesting research topic that involves semantic analysis, lexical resources, and the development of specific NLP techniques.
- Since there is no adequate quantitative measure, human evaluation seems to be a valid alternative. However, a research line could be aimed at automatically comparing the rule-set generated manually by an analyst, and the set of rules automatically extracted applying DecisionRuleMiner. For this, certain criteria could be defined that quantify the number of rules, number of elements in the antecedent and consequent, and terms that match or not, among others.
- There have been some efforts to formalize natural language regulations for SBVR (Bajwa et al., 2011; Fortineau et al., 2013; Lévy and Nazarenko, 2013). Rules stipuled in SBVR and to some measure maybe PMML as well could be used as a basis for the logic upon which the miner discovers rule patterns.

Acknowledgments

This research was supported by the special research fund for incoming mobility of Hasselt University, Belgium. The authors gratefully acknowledge Veronika Boyanova and Aziz Yarahmadi for providing useful descriptions in our experiments, as well as the experts who kindly answered the survey.

References

Bajwa, I. S., Lee, M. G., and Bordbar, B. (2011). SBVR business rules generation from natural language specification. In *Proceedings of the AAAI Spring Symposium - AI for Business Agility, Palo Alto, California, United States*, volume SS-11-03, pages 2–8.

Boufrida, A. and Boufaida, Z. (2014). Automatic rules extraction from medical texts. In *Proceedings of the International Workshop on Advanced Information Systems for Enterprises (IWAISE)*, pages 29–33.

Boyer, J. and Mili, H. (2011). *Agile Business Rule Development: Process, Architecture, and JRules Examples*. Springer-Verlag, Berlin Heidelberg.

Califf, E. and Mooney, J. (1999). Relational Learning of Pattern - Match Rules for Information Extraction. *Computational Linguistics*, pages 9–15.

Calvanese, D., Dumas, M., Laurson, U., Maggi, F. M., Montali, M., and Teinemaa, I. (2018). Semantics, analysis and simplification of DMN decision tables. *Information Systems*, pages 1–14.

Caporale, T. (2016). A tool for natural language oriented business process modeling. In Hochreiner, C. and Schulte, S., editors, *CEUR 8th ZEUS Workshop Proceedings, Vienna, Austria, January 27-28, 2016*, volume 1562, pages 49–52.

Ciravegna, F. (1999). Adaptive information extraction from text by rule induction and generalisation. *Natural Language Engineering*, 10:145–165.

Corradini, F., Ferrari, A., Fornari, F., Gnesi, S., Polini, A., Re, B., and Spagnolo, G. O. (2018). A guidelines framework for understandable bpmn models. *Data & Knowledge Engineering*, 113:129 – 154.

De Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006), Genoa, Italy, May 22-28, 2016*, pages 449–454. European Language Resources Association (ELRA).

De Smedt, J., De Weerdt, J., Serral, E., and Vanthienen, J. (2018). Discovering hidden dependencies in constraint-based declarative process models for improving understandability. *Information Systems*, 74:40–52.

De Smedt, J., Hasic, F., vanden Broucke, S. K. L. M., and Vanthienen, J. (2017). *Business Process Management - 15th International Conference (BPM 2017), Barcelona, Spain, September 10-15, 2017, Proceedings*, volume 10445 of *Lecture Notes in Computer Science*, chapter Towards a Holistic Dis-

covery of Decisions in Process-Aware Information Systems, pages 183–199. Springer.

Demner-Fushman, D., Chapman, W. W., and McDonald, C. J. (2009). What can natural language processing do for clinical decision support? *Journal of Biomedical Informatics*, 42(5):760–772.

Dragoni, M., Governatori, G., and Villata, S. (2015). Automated rules generation from natural language legal texts. In *Proceedings of the Workshop on Automated Detection, Extraction and Analysis of Semantic Information in Legal Texts (ICAIL 2015), San Diego, USA*, pages 1–6.

Dragoni, M., Villata, S., Rizzi, W., and Governatori, G. (2016). Combining NLP approaches for rule extraction from legal documents. *Proceedings of the 29th International Conference on Legal Knowledge and Information Systems*, pages 1–13.

Figl, K., Mendling, J., Tokdemir, G., and Vanthienen, J. (2018). What we know and what we do not know about DMN. *Enterprise Modelling and Information Systems Architectures*, 13(2):1–16.

Fortineau, V., Paviot, T., Guissé, A., and Lamouri, S. (2013). A transformation model to express business rules from natural language to formal execution: an application to nuclear power plant. *IFAC Proceedings Volumes*, 46(9):1096–1101.

Friedrich, F., Mendling, J., and Puhlmann, F. (2011). *Advanced Information Systems Engineering, 23rd International Conference on Advanced Information Systems Engineering (CAiSE 2011), London, UK, June 20-24, 2011, Proceedings*, volume 6741 of *Lecture Notes in Computer Science*, chapter Process model generation from natural language text, pages 482–496. Springer.

Garza, D. (2014). Automated business rule harvesting with abstract syntax tree transformation.

Ghose, A., Koliadis, G., and Chueng, A. (2007). *Conceptual Modeling - ER 2007, 26th International Conference on Conceptual Modeling, Auckland, New Zealand, November 5-9, 2007, Proceedings*, volume 4801 of *Lecture Notes in Computer Science*, chapter Rapid Business Process Discovery (R-BPD), pages 391–406. Springer.

Goldberg, L. and Halle, B. v. (2009). *The Decision Model*. Taylor & Francis Group.

Gonçalves, J. C., Santoro, F. M., and Baião, F. A. (2009). Business process mining from group stories. In *Proceedings of the 13th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2009)*, number September, pages 161–166. IEEE.

Hassanpour, S., O'Connor, M. J., and Das, A. K. (2011). *Rule-Based Reasoning, Programming, and Applications*, volume 6826 of *Lecture Notes in Computer Science*, chapter A framework for the automatic extraction of rules from online text, pages 266–280. Springer-Verlag.

Hays, D. G. (1964). Dependency theory: a formalism and some observations. Technical report, Santa Monica, California.

Huffman, S. B. (1996). *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, volume 1040 of *Lecture Notes in Computer Science*, chapter Learning information extraction patterns from examples, pages 246–260. Springer.

IIBA (2009). *A Guide to the Business Analysis Body of Knowledge (BABOK Guide), Version 2.0*. International Institute of Business Analysis.

Kuss, E., Leopold, H., van der Aa, H., Stuckenschmidt, H., and Reijers, H. A. (2018). A probabilistic evaluation procedure for process model matching techniques. *Data & Knowledge Engineering*.

Lévy, F. and Nazarenko, A. (2013). *Theory, Practice, and Applications of Rules on the Web. RuleML 2013*, volume 8035 of *Lecture Notes in Computer Science*, chapter Formalization of Natural Language Regulations through SBVR Structured English, pages 19–33. Springer.

Liddy, E. (1998). Enhanced text retrieval using natural language processing. *Bulletin of the Association for Information Science and Technology*, 24(4):14–16.

Lima, R., Freitas, F., and Espinasse, B. (2016). Relation extraction from texts with symbolic rules induced by inductive logic programming. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Vietri sul Mare, Italy*, pages 194–201. IEEE.

Lin, D. and Pantel, P. (2001). DIRT - Discovery of Inference Rules from Text. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining (KDD-01), San Francisco, C.A.*, pages 323–328. ACM.

Liu, Q., Gao, Z., Liu, B., and Zhang, Y. (2015). Automated rule selection for aspect extraction in opinion mining. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume January, pages 1291–1297.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marneffe, M.-C. D. and Manning, C. D. (2010). Stanford typed dependencies manual. *20090110 Httpnlp Stanford*, 40(September):1–22.

Melkuc, I. A. (1988). *Dependency syntax: theory and practice*. State University of New York Press.

Moldovan, D. I. (1995). Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):713 – 724.

Ono, T., Hishigaki, H., Tanigami, A., and Takagi, T. (1999). Automatic extraction of information on protein-protein interaction from scientific literature. *Genome Informatics 1999*, pages 296–297.

Paknikar, S., Minds, H., Anand, A., Pandey, K. K., Kotkar, K., and Bhol, D. (2014). Rules Harvesting from Source Code.

Papanikolaou, N. (2012). *On the Move to Meaningful Internet Systems: OTM 2012*, volume 7566 of *Lecture Notes in Computer Science*, chapter Natural language processing of rules and regulations for compliance in the cloud, pages 620–627. Springer.

Riefer, M., Ternis, S. F., and Thaler, T. (2016). Mining process models from natural language text: a state-of-the-art analysis. In Nissen, V., Stelzer, D., Straßburger, S., and Fischer, D., editors, *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2016)*, pages 1–12. Springer.

Riloff, E. (1993). Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the 11th National Conference on Artificial Intelligence, Washington, D.C.*, pages 811–816.

Satyal, S., Weber, I., Paik, H.-y., Di Ciccio, C., and Mendling, J. (2018). Business process improvement with the ab-bpm methodology. *Information Systems*, pages 1–15.

Silver, B. (2016). *DMN Method & Style The practitioner's guide to decision modeling with business rules.* Cody-Cassidy Press, Altadena, CA.

Sinha, A. and Paradkar, A. (2010). Use cases to process specifications in business process modeling notation. In *Proceedings of the 8th International Conference on Web Services (ICWS 2010)*, pages 473–480. IEEE.

Soderland, S. (1999). Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1):233–272.

Soderland, S. G. (1997). *Learning text analysis rules for domain-specific Natural Language Processing.* PhD thesis, University of Massachusetts.

Sorgente, A., Vettigli, G., and Mele, F. (2013). Automatic extraction of cause-effect relations in natural language text. In *Proceedings of the 7th International Workshop on Information Filtering and Retrieval, Turin, Italy*, volume 1109, pages 37–48. CEUR Workshop Proceedings.

Taylor, J., Fish, A., Vanthienen, J., and Vincent, P. (2013). *Intelligent BPM Systems: Impact and Opportunity*, chapter Emerging standards in decision modeling - an introduction to Decision Model & Notation, pages 133–146. BPM and Workflow Handbook Series. Future Strategies, Incorporated.

van der Aa, H., Leopold, H., del Río-Ortega, A., Resinas, M., and Reijers, H. A. (2017). Transforming unstructured natural language descriptions into measurable process performance indicators using hidden markov models. *Information Systems*, 71:27–39.

Van Der Aalst, W. (2011). *Process mining: discovery, conformance and enhancement of business processes*, volume 2. Springer.

Wang, X., Sun, J., Yang, X., He, Z., and Maddineni, S. (2004). Business rules extraction from large legacy systems. In *Eighth European Conference on Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings.*, pages 249–258.

Wyner, A. and Peters, W. (2011). *Legal Knowledge and Information Systems*, volume 235 of *Frontiers in Artificial Intelligence and Applications*, chapter On rule extraction from regulations, pages 113–122. IOS Press.

Xia, R. and Ding, Z. (2019). Emotion-Cause Pair Extraction: a new task to emotion analysis in texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy*, pages 1003–1012. ACL.

Yarahmadi, A. (2018). *Enhanced machine learning approaches in text analysis for business intelligence: The appealing story of documents.* PhD thesis,

University of Hasselt.