An Approach to Mine SBVR Vocabularies and Rules from Business Documents

Pavan Kumar Chittimalli
TRDDC, TCS Research
Pune, India
pavan.chittimalli@tcs.com

Chandan Prakash
TRDDC, TCS Research
Pune, India
ch.pr@tcs.com

Ravindra Naik
TRDDC, TCS Research
Pune, India
rd.naik@tcs.com

Abhidip Bhattacharyya*
University of Colorado Boulder
Boulder, CO, USA
abhidip.bhattacharyya@colorado.edu

ABSTRACT

Enterprises model the behavior of their business to prepare a communication standard for business analysts and to specify requirements to Information Technology (IT) people. The communication gap between IT group and business analysts, who lie on the opposite end of the business spectrum exists due to the different terminologies used in their respective fields regarding the same context. This gap has led to major software failures which prompted the OMG group has come up with a new standard - Semantic of Business Vocabulary and Business Rules (SBVR). Declarative models are provided by SBVR to represent Business Vocabulary and Business Rules which can be understood by everyone working throughout the business spectrum. Each business is governed by business rules which are constrained by the regulation policy set up by the policy guidelines of the organization and government regulations set up on the organization. Business rules are specified in documents like user guides, requirement documents, terms and conditions, do's and don'ts. Typically a Business Analyst interprets the document and manually extracts rules based on his understanding which leads to potential discrepancies, ambiguities and quality issues in the software system. To minimize such errors, in this paper we present an unsupervised approach to automatically extract SBVR vocabularies and rules from domain-specific business documents. We also present our initial results and comparative study with our earlier approach.

KEYWORDS

Business Rules Extraction, Rule Document, Rule Components, SBVR, Natural Language Processing, Text Mining

*This author has previously worked at TRDDC, currently pursuing PhD.

ISEC 2020, February 27–29, 2020, Jabalpur, India

ACM ISBN 978-1-4503-7594-8/20/02…$15.00
https://doi.org/10.1145/3385032.3385046

ACM Reference Format:
Pavan Kumar Chittimalli, Chandan Prakash, Ravindra Naik, and Abhidip Bhattacharyya. 2020. An Approach to Mine SBVR Vocabularies and Rules from Business Documents. In *13th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference) (ISEC 2020), February 27–29, 2020, Jabalpur, India.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3385032.3385046

1 INTRODUCTION

Business services are provided with the help of rules that constraint the functioning of their operations and activities. Business rules are the implementation core of any software system that automates the process. The "if-then" rules are found in the source code of the software system (IT system). These rules are designed by Business Analyst & Business owners and found in requirements documents, manuals, user guides, terms and conditions and do's and don'ts of the business. Business rules are constrained by the guidelines and policies of government and business authorities.

Mostly Business rules are written in natural language by the Business Analysts and presently involves a lot of manual work to extract them from such documents. The rules which are mined from the documents if expressed in natural language, again becomes a cumbersome activity for the IT engineers while analyzing for inconsistencies. The major issue of software failures in the industry is because of the gap in communication between the IT people and the Business Analysts [24]. The losses incurred due to such software failures amount to nearly 250 billion dollars each year [3].

Semantics of Business Vocabulary and Rules (SBVR) [5] is a standard for business rule representation by Object Management Group (OMG) [2]. SBVR is a Controlled Natural Language (CNL) and describes rules considering only a set of predefined business vocabularies. SBVR provides a natural language interface with first order logic (FOL). These semantics of SBVR makes it ideal for representing business rules. SBVR can be used to formalize complex compliance rules, such as security policy, operational rules for an enterprise, regulatory compliance or standard compliance rules. SBVR has two popular notations for specifications;1) SBVR Structured English (SBVRSE) [23] 2) RuleSpeak [31]. These notations are based on Controlled Natural Language representation and yet provides underlying model for machine manipulation. With the SBVR representation, the problem can now be reduced to extracting SBVR rules from various documents. The closest work is by Bajwa

et al. [9], which generates SBVR format from a single line of English text. In this work, a user inputs one English rule sentence (having a subject, object, and a verb) at a time without any conjunctions is converted into a SBVR rule automatically. To the best of our knowledge, there has been no further work in mining SBVR rules or rules in any other form. A typical document consisting of rule statements with lot of noise words makes it hard to comprehend and identify meaningful business rules. Another challenge is rule sentences with several clauses that make rule comprehension even more difficult. Moreover adherence to a particular format of the document or format of the output may put question on the reusability of the approach. Several techniques are proposed for extracting rules from legacy source code [14, 18, 22, 33, 34] and knowledge extraction from documents [17, 25, 26]. In our previous works [10, 11], we proposed methods to extract atomic facts (rule intents) and relations among these rule intents. However, these the outcomes of our approaches are not machine processable representations.

To tackle this problem, in this paper we present an approach to mine business rules and vocabularies in SBVRSE notation and persist them in SBVR meta model to allow the IT engineers for further analysis like anomaly detection [7, 15]. Unlike our earlier approaches [10, 11], in this paper we present a new method to mine SBVR rules and vocabularies from domain-specific documents in an unsupervised (no-pre tagging required) manner. In this paper, based on the knowledge gathered by our discussions with Business Experts/Analysts, we tried to address multiple challenges like handling document with lots of inherent noise, a rule statement with several clauses and a rule having a rule nested inside, all of which makes the miner even harder to interpret and resolve. Moreover, in order to make our approach viable we must ensure that we do not become specific to a single type of document while developing our heuristics.

The paper is organized as follows: Section 2 describes the related work done in this area. Section 3 provides a motivating example to elaborate how the extracted SBVR vocabularies and rules look. In section 4, we describe the detailed approach of mining the SBVR rules and vocabularies. Section 5 illustrates the prototype tool and experimental results. Section 6 describes the conclusion.

2 RELATED WORK

The documents mining related research uses predictive classification techiques or populating a database or search index with information extraction techniques. The semi/automatic extraction of SBVR vocabularies and rules extraction from documents is not a very well explored research area. However, there appears little research specific to mining of business rules or creating their formal representations beside our research [10, 11] in this area. The related work can be broadly classified into following categories.

(1) Natural Language Processing (NLP) techniques using shallow parsing
(2) NLP Techniques creating finer levels of abstraction models
(3) NLP Techniques using dependency trees and machine learning (ML) for noise elimination.

The existing shallow parsing techniques such as [27, 32, 35], uses a shallow parser [12] with predefined templates, making these techniques tightly coupled and dependent on the document structure. The API document mining by R. Pandita et al. [27] proposed a method to generate code contracts which consists of variables and the relation among them as API contracts. The authors feed the data to the shallow parser after pre-processing. The sentences are subsequently mapped to set of First Order Logic (FOL) formulae using predefined templates on shallow parser. Xaio et al. [35] used a method called Text2policy. This technique use a shallow parsing method to extract access rule and its corresponding action steps from access control policies (ACP) defined in the document. The authors also implemented a negative meaning implication (e.g. verb like *disallow*) and a negative inference (word like *never, not*). A. Sinha et al. [32] defined a Linguistic Analysis Engine to infer Use Case models from Use Case Description (UCD) based on shallow parser using Standford NLP. This engine is built on top of Unstructured Information Management Architecture (UIMA). The authors also incorporated context annotators to allocate roles to the nouns occurring in the predicates. These roles are either ACTORs or SYSTEMs. Ultimately, their technique builds process models by identifying the use case action sequence and by scanning the actions against known (pre-defined) use case patterns. The three works described above use a pre annotated domain dictionary to classify the verbs into some predefined classes depending on their semantic equivalence. Their techniques serve specific purposes and are centred on specific documents. These techniques largely benefit from the structure of the documents and business vocabularies.

In the second category of related work (techniques uses a much finer level models than shallow parsing), uses POS tag sequence. One such method is proposed by S.Ghaisas et al. [20], focuses on retrieving rule intents from requirement documents by matching them against patterns made up of sequence of Part Of Speech (POS) tags, key words and their repetitions denoted by wild characters like '*', '+' etc. They discovered 29 intents (such as threshold, chronology, activity, temporal check to denote the intention of *rule intent*) and 517 rule intent patterns. The authors used agglomerative clustering to place co-occurring rules together. The effectiveness of their technique has been shown on the internal/industrial case studies.

Apart from these techniques, there are few other works that address the knowledge extraction from document using Information Extraction technique. Colette Rolland et al. [30] proposed an approach to guide the construction of textual use case specifications from documents. The authors used the concept of case grammar introduced by Fillmore [19]. This approach classifies the semantic patterns into Clause and Sentence semantic patterns to capture different surface structures of the sentences having same deep meaning. This approach further used set of rules and guidelines for converting the input into a use case. Atkinson et al. [8] used genetic algorithms for inferring hypothesis from scientific document written in natural language. H.Zhong et al. [36] proposed a method of inferring resource specifications from API document using machine learning. This research uses Hidden Markov Model (HMM) [29] based named entity recognition for knowledge extraction. The research by Putrycz et al. [28] for understanding the business rules embedded in source code. This technique use patterns and keywords derived from source code to create mappings between business rules from source code to documents.

The above related work techniques basically focus on domain specific, project specific class of documents and take advantage of the structure and format of the document. As the patterns, style, structure of document is pre defined, the techniques uses a predefined set of templates. The second category of work uses predefined templates based out of POS tag sequence. The techniques also makes the technique vulnerable to noise since the POS tag sequence can produce incorrect rule intents due to noisy word sequences.

The third category is combining machine learning techniques with NLP. Our earlier works [10, 11] addresses the problems such as eliminating noise, identifying rule sentences in the structured or unstructured documents, extracting rule intents from sentences, and extracting relationship among the rule intents.

3 MOTIVATING EXAMPLE

In this section, we use a rule text from EURent-A-Car [1] example to illustrate SBVR rule and vocabulary identification.

> If a rental request does not specify a particular car group or model, the allocated car is A.

The first task in our approach is to automatically extract relevant atomic facts from the *rule sentence*. The facts extracted from the rule sentences are shown below.

f_1 : $specify$(rental request, car group)
f_2 : $specify$(rental request, car model)
f_3 : is(allocated car, A)

Table 1: Rule intents of the sentence from earlier example

The above example document fragment has total 3 atomic facts (rule intents). The next step in the workflow is identifying the relations among the extracted rule intents. The relation among rule intents extracted for the example are shown below:

$$Rule_1 : \neg f_1 \vee \neg f_2 \rightarrow f_3$$

Subsequently, the extracted rules (rule intents and relations between them) will be converted to SBVR models, which is a machine manipulatable format, to perform analysis for verification & validation [15]. SBVR is a Controlled Natural Language (CNL) and describes rules considering only a set of predefined SBVR business vocabularies. The SBVR vocabulary consists of *terms concepts, name concepts, fact concepts*. SBVR Structured English (SE) specified using coloring notation for easy readability of the rules. The green underlying text denotes the terms, blue italic text denotes verbs, bold face underlying text denotes the name (individual noun concept). The SBVR Vocabulary and Rule in Structured English (SE) for the example rule sentence is shown in Table 2 and Table 3

request

rental_request
General Concept: request
Definition: The request that is of rental car

allocated_car
General Concept: car
Definition: The car that is allocated

car_model
Definition: The model that is of car

car_group
Definition: The group that is of car

A
General Concept: car

rental_request *specify* car _group
rental_request *specify* car _model
allocated_car *is* **A**

Table 2: The vocabulary in SBVRSE for the example sentence

It is necessary that if the rental_request doesn't *specify* car _group or the rental_request doesn't *specify* car _model then the allocated_car *is* **A**

Table 3: The rule in SBVRSE for the example sentence using SBVR Vocabulary

4 DETAILED METHODOLOGY

In this section, we first introduce our approach, and describe each phase of the approach in later parts of the section.

The block diagram in Figure 1 illustrates our approach. Rule sentences are parsed by natural language parser (SpaCy) [21] to produce the dependency trees. Our approach for SBVR rule extraction takes this dependency tree as input and provides possible vocabulary and rules as output. The extraction of rule intents from dependency tree is done in four stages. In first step, the noise elimination from the document to identify only the rule text from document, followed by the identification of the entities and named entities in the second step. In third step, the atomic facts in each rule sentence has been identified. In fourth step, the relation between the facts have been identified and corresponding SBVR rule is constructed based the extracted SBVR vocabulary.

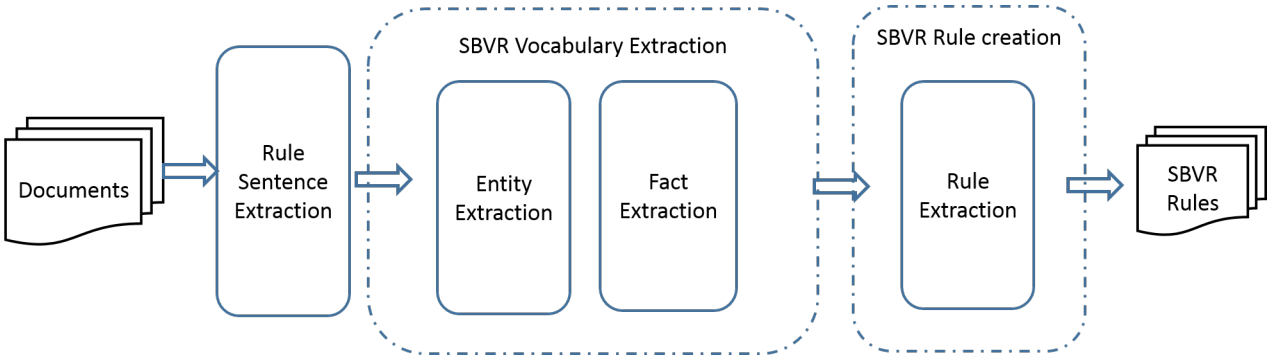

Figure 1: Block diagram of our approach.

4.1 Rule Sentence Extraction

The primary objective of the Rule Sentence Extraction is to classify the sentences in a given document into rule sentences or noise. For this purpose, we use the concept of n-gram language model. The trigram language model, a specialization of *n-gram* language model [13], is well-known in NLP, and has been widely used in many NLP tasks, like machine translations. A *Trigram* language model consists of:

(1) A finite set of words v.
(2) A parameter $q(w|u,v)$ for each trigram u, v, w such that $w \in v \cup \{STOP\}$ and $u, v \in v \cup \{*\}$, where $STOP$ is a delimiter word (e.g. '.') and u, v, w are any three consecutive words in a sentence. The symbol '*' denotes prefix of a sentence when handling the first word of the sentence.

For any sentence $x_1x_2x_3 \ldots x_n$ where $x_i \in v$ for $i = 1, 2, \ldots, n-1$ and $x_n = STOP$, the probability of the sentence under trigram language model is given as:

$$p(x_1x_2x_3 \ldots x_n) = \prod_{i=1 \ldots n} q(x_i|x_{i-2}x_{i-1}) \tag{1}$$

Where $x_{-1} = x_0 = *$

$$q(x_i|x_{i-2}x_{i-1}) = \lambda_1 q_{\text{ML}}(x_i|x_{i-2}x_{i-1}) + \lambda_2 q_{\text{ML}}(x_i|x_{i-1}) + \lambda_3 q_{\text{ML}}(x_i) \tag{2}$$

$$q_{\text{ML}}(x_i|x_{i-2}x_{i-1}) = \frac{count(x_{i-2}x_{i-1}x_i)}{count(x_{i-2}x_{i-1})} \tag{3}$$

$$q_{\text{ML}}(x_i|x_{i-1}) = \frac{count(x_{i-1}x_i)}{count(x_{i-1})} \tag{4}$$

$$q_{\text{ML}}(x_i) = \frac{count(x_i)}{|v|} \tag{5}$$

where λ_is are weights such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$ and $\lambda_i \geq 1$. q_{ML} s are the maximum likelihood estimates of trigram, bigram and unigram word porbability. The q_{ML} parameters are shown in equations (3), (4), (5) for trigram, bigram and unigram, respectively.

In our approach, we propose two such language models:

(1) A trigram language model for relevant sentences and,
(2) A trigram language model for non-relevant or noise.

The two language models are trained; one using relevant sentences and the other using non-relevant sentences. During testing, a test sentence is given as input to the two language models to predict the probabilities. A sentence will be classified as per the language model which has a greater value than the other i.e. the sentence will be classified as a rule sentence or a noise.

> No account is opened in anonymous or fictitious name

Consider the above sentence as a test sentence. Let us assume that the training data does not have the word '*opened*', but has the word '*create*' in the same context. The q factor of q(opened| account is) will be calculated as '0'. This makes the probability of the whole sentence under the particular language model as '0', as described by equation (1). Thus during testing if a word occurs in the test sentence that is not seen during training in relevant sentences or in noise, or in either of them, then such particular word sequence results in the q factor as '0' in that particular language model or in both. Therefore classification of a sentence is controlled by a single word in the sentence, and its absence or presence in the training data. Such a solution makes the result biased, and is unacceptable.

To overcome this problem, a POS tag trigram is used for that particular word sequence. Note that the particular word can occur in at most three combinations of word-trigrams. However, the q factor can only be '0' for the particular word (not seen in the training data) that is conditioned on previous two words. In this particular scenario of a trigram $w_i|w_{i-2}w_{i-1}$ (w_i is word unseen in training data), we use $t_i|t_{i-2}t_{i-1}$ where t_i's are the POS tag of w_i's, leaving all other q's unchanged. We use the modified language model with POS tag for both, the relevant and noise language models, if the word is unseen in the training data for any one of the models. This makes the classification unbiased.

$$\underbrace{No}_{DT}\ \underbrace{account}_{NN}\ \underbrace{is}_{VBZ}\ \underbrace{opened}_{VBN}\ \underbrace{in}_{IN}\ \underbrace{anonymous}_{JJ}\ \underbrace{or}_{CC}\ \underbrace{fictitious}_{JJ}\ \underbrace{name}_{NN}$$

In the above test sentence, we use POS tag sequence of q(VBN| NN VBZ) instead of using q(*opened*| *account is*). This makes the values of q same for 'created' and 'opened'. The intuition behind incorporating POS tag in place of word is that a same POS tag sequence can be treated as a representative set (template) of a group of word sequences.

4.2 Entity Extraction

The initial step involves extracting the *entities* and *named entities* from the document. *Entities* are the common nouns, such as `car`, `person`, `sport` etc, while *Named Entities* are proper nouns such as `EU-Rent`, `M S Dhoni`, `Ford`, `Orange` etc. The `SBVR facts` that are developed will have these *entities* as their subjects and the objects, which are then used to form `SBVR rules`. We perform this step in a completely unsupervised manner (no user tagging data is required) which is achieved by making use of the different POS tags that are assigned to the individual words in the document by the SpaCy parser [21].

The input to this step is the business document cleaned by removing text noise such as punctuations, words within brackets, words within hyphens etc. We assume that the document is cleaned beforehand by standard techniques [6]. By observing several sentences in the document and the corresponding sequence of POS tags, we propose a set of heuristics which help us to identify and extract the entities. This however is not an intuitive process but requires going through several statements in the document, identifying the different POS tags that occur together and deciding which pairs of such POS tags should be treated as a single entity. It may also happen that some POS tag combinations only appear in a few statements in the document. In this case it is necessary to ignore such POS tag combinations otherwise it may cause our model to be an over fitted one causing the precision of our entity extraction model to fall.

We present two heuristics which we use for *Entity* extraction.

H_1- The sequence of `NOUN/NUM` appearing in dependency tree:

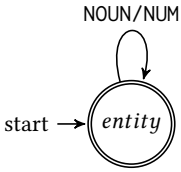

Figure 2: Heuristic 1 for entity extraction.

Any number of nouns or numbers occurring together in the document result in the entire group being classified as a *single entity* as shown in Figure 2. The POS tags for the words thus maybe `NN`, `NNS`, `NNP`, `CD` or `NNPS`. All of these are just different type of nouns (like singular noun, plural noun, singular proper noun, plural proper noun etc) or numbers (like 123). This heuristic follows because if several nouns or numbers occur all together, then they will always belong to the same entity. Had they belonged to different entities, there would have been some sort of a separator word between the nouns. The separator word could be a punctuation mark, a conjunction, a verb or any other non-noun word. As an example consider Figure 3. The sentence is: '*A rental request specifies car group*'. The words *car* and *group* occur together and have the tags `NN` and `NN` respectively. Thus, this heuristic gets applied and thus '*car group*' becomes a single entity.

H_2- The sequence of `Adjectives` followed by sequence of `Noun/Nums`:

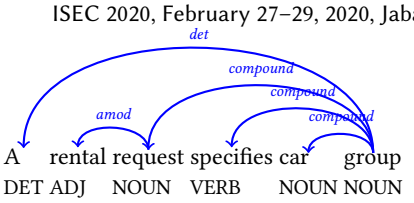

Figure 3: Dependency tree for a sample sentence for Heuristic1.

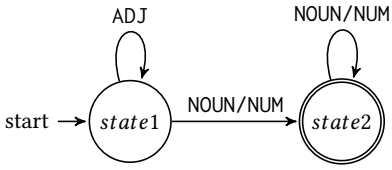

Figure 4: Heuristic 2 for entity extraction.

Any number of adjectives(describer of a noun) coming together and then followed by any number of nouns or numbers will be considered as a single entity. Also, there may be multiple qualifiers for a noun and hence many adjectives may precede a noun or a number, which in turn may be followed by any number of nouns or numbers. The DFA for this heuristic is shown in Figure 4. The tags for the adjective will be `JJ` while that for a noun can be any of `NN`, `NNP`, `NNS`, `CD` or `NNPS`.

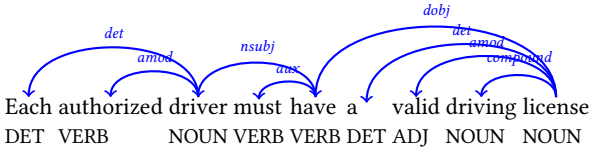

Figure 5: Dependency tree for a sample sentence for Heuristic2.

The heuristic follows because many adjectives one after the other will always belong to the same entity. Else, there would have been some non-adjective and non-noun word in between to separate them out. The separators could be punctuation, conjunctions, nouns etc. In Figure 5, consider the sentence '*Each authorized driver must have a valid driver license.*' The word *valid* is an adjective, *driver* is a noun and *license* is also a noun. Hence since these 3 words appear together, this heuristic fires and the three words get combined into a single entity. Hence the new entity in this case is '*valid driver license*'.

4.3 Fact Extraction

After the completion of `Entity Extraction`, all the multi word entities are clubbed into a single word by inserting an underscore between the different words. e.g., a multi word entity `Government of India` becomes *Government_Of_India* after the completion of

first stage. This underscore insertion step is carried out on the entire document and the new document is fed as input to the second stage.

Consider the following sentence as an example: *'If the customer requesting the rental has been blacklisted, the rental must be refused'*. In the given business sentence, the atomic facts are:

(1) customer requesting rental
(2) customer has been blacklisted
(3) rental must be refused

Every business domain comprises of various atomic facts. Every atomic fact can have one of the three following structures:

- **Verb Phrase has both Subject and Object**: In this structure, the verb phrase has both the subject as well as the object connected directly to the words on the verb phrase. The linkage maybe through subject tags such as `nsubj, nsubjpass`, and object tags such as `dobj, pobj`. This is one of the simplest structures possible for an atomic fact.

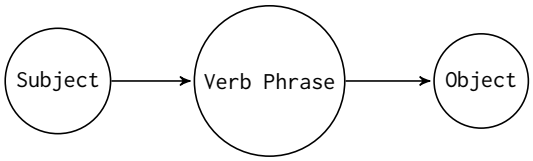

Figure 6: Heuristic 1 for fact extraction.

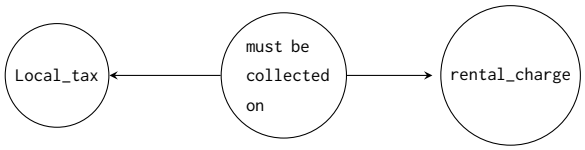

Figure 7: Structure for the fact extraction heuristic 1.

Example: Consider the sentence '*Local tax must be collected on the rental charge*'. In this sentence, *must be collected* is the verb phrase. The subject is *Local tax* and the object is *rental charge*, which are directly connected to one of the verb phrase words as shown in Figure 7.

- **Verb phrase has a subject, but no object**: In this structure the verb phrase has a set of words, one of which is directly connected to a subject via one of the subject links. However, there is no object tag connection between any of the verb phrase words and some other noun in the document. Hence the right pointer of the verb phrase is `null`, as shown in Figure 8.

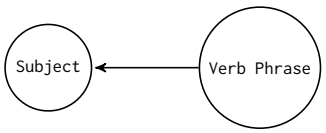

Figure 8: Heuristic 2 for fact extraction.

Example: For the sentence '*If the customer requesting the rental has been blacklisted, the rental must be refused*', the verb phrase is *has been blacklisted* and subject is *customer*

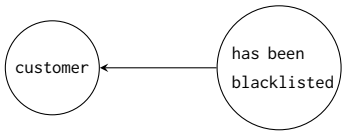

Figure 9: Structure for the fact extraction heuristic 2.

which is directly connected to one of the verb phrase words. However there is no object tag connection from any of the verb phrase words. Hence the right pointer of the verb phrase is null. Figure 9 shows the structure of the example.

- **Verb phrase has an object, but no subject**: In this structure the verb phrase has a set of words, one of which has a direct object tag connection to some object. There is however no subject tag connection from any of the verb phrase words to any noun in the document. Hence the left pointer of the verb phrase is null.
  Example: In the sentence *'If there are not sufficient cars in a group to meet demand, a one-group free upgrade may be given if there is capacity'*, the verb phrase is *to meet*. The word *demand* is connected to one of the verb phrase words via an object tag. However there is no subject tag connection from any of the verb phrase words to any noun in the document. Thus the left pointer of the verb phrase is null.

We now present the sequential steps that is followed in our approach for Identification of Verb Phrase, Subject & Object.

4.3.1 *Identification of the Verb Phrase.*

(1) Get input tokens from a rule sentence.
(2) Identify the POS tag.
(3) If the POS tag is a VERB, then add the token in the VERB PHRASE.
(4) Look for the child of the VERB added.
(5) If the child of the verb is not a noun and the dependency link between the child and the verb is any of the following: `aux, auxpass, prep, advmod, acomp, xcomp, agent, expl`, then add the child in the Verb Phrase and if the child is a verb then recursively check for its children.
(6) Take the next token in the rule sentence, if it is a verb repeat the same activity and if the token is not a verb break the Verb Phrase formation.

Figure 10 shows the flowchart of the process presented above.

4.3.2 *Identification of the Subject.*

(1) Iterate over every child of every element present in the Verb Phrase, If the dependency link between the child and the element of the Verb Phrase is one of the following: `amod, comp, compound`, then assign the child as the subject of the Verb Phrase and make the right child of the verb phrase null, structure of such a verb phrase is shown in Figure 11. For such cases, the verb phrase is connected via an amod, comp or compound link to the subject and there will be just a subject for the verb phrase, and no object.
(2) Iterate over every child of every element present in the Verb Phrase. If the dependency link between the child and the element of the Verb Phrase is one of the following: `acl`,

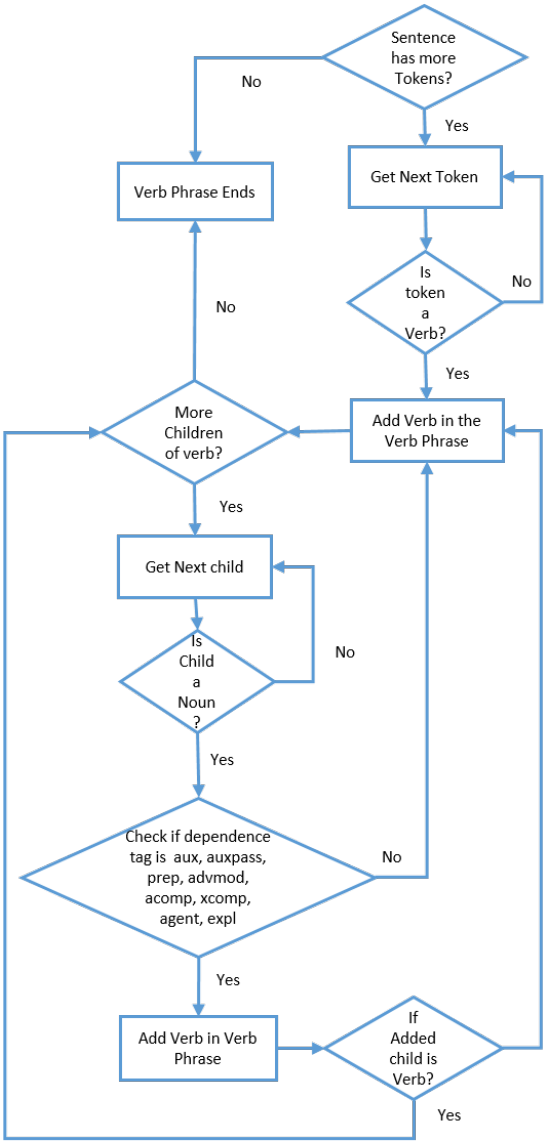

Figure 10: Flowchart of the algorithm for verb phrase extraction.

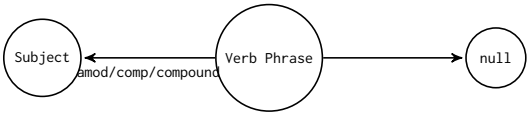

Figure 11: Heuristic for subject identification

`relcl` and the child is Noun, then assign the child as the subject of Verb Phrase. The structure of such a verb phrase is shown in Figure 13, In this case, the word in the verb phrase is connected directly to the subject in the document via a `relcl` or an `acl` tag. This condition is given a priority over the normal subject tags heuristic.

e.g., In Figure 12 for the verb phrase *requesting*, the subject will be *customer* because it is directly connected to *requesting* via an `acl` tag.

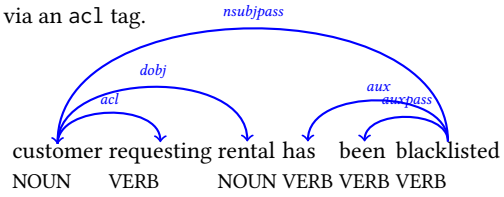

Figure 12: Dependency tree for a sample sentence for subject identification.

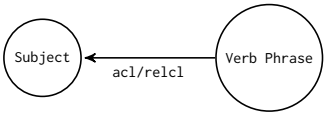

Figure 13: Structure of fact with acl link.

(3) Iterate over every child of every element present in the Verb Phrase. If the dependency link between the child and the element of the Verb Phrase is one of the following: `nsubj`, `nsubjpass`, `csubj`, `csubjpass` (structure of which is shown in Figure 14) and the child is Noun, then assign the child as the subject of Verb Phrase. For this rule, we try to find a direct subject link between one of the verb. phrase words and any other noun in the document, to which the word is connected to. This noun is chosen as the subject for the verb phrase. However, this step happens if and only if there is no acl or relcl link from any of the verb phrase words.
e.g.,In Figure 15, *Reservations* is directly connected to *may* which is a word on the verb phrase. Thus, for the verb phrase *may be accepted*, the subject will be *Reservations*(the dependency tag is `nsubjpass`).

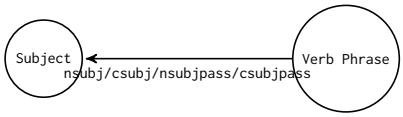

Figure 14: Structure of fact with direct subject link.

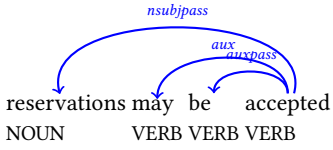

Figure 15: Dependency tree for direct nsubjpass link.

(4) Iterate over every child of every element present in the Verb Phrase. If child is a Verb and the child belongs to a different verb phrase then the subject of the verb phrase is the subject of the verb phrase that the child belongs to.
e.g.,In Figure 16, the verb phrase *may have* has the subject as *customer*. For the verb phrase *may have* only, there is no direct subject link, neither through a direct object link,

nor through `acl` or `relcl`. Thus, we check if any word in this verb phrase is connected to some other word in some other verb phrase. Here the *have* of the second verb phrase is connected to the *have* of the first verb phrase. We infer that since the first verb phrase has a subject i.e. *customer*, the second verb phrase will also have the same subject i.e., *customer*.

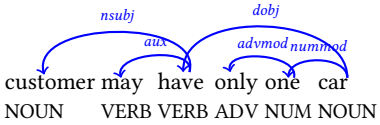

Figure 16: Example for verb phrase connected to a different verb phrase.

(5) If the subject of the verb phrase assigned in the previous four stages has a "conj" link then identify the child with which the subject has a conj link and replicate the same structure with the child as the new subject.
e.g., In Figure 17, the verb phrase *requested* has *model* as the subject. However, since *model* itself is connected to *group* via a `conj` tag, we replicate the entire fact structure and replace *model* with the word that is connected to *model* via the `conj` link i.e. *group*. Hence, we get two facts out of one rule sentence.

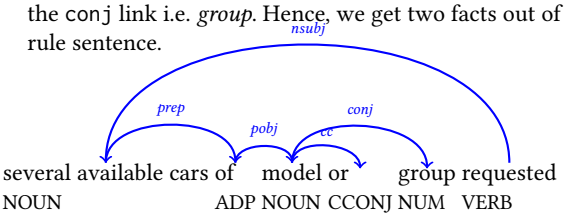

Figure 17: Example for handling conjunction for subjects.

(6) If there is no subject for a verb phrase, and after finding the object, there is a noun connected to the verb phrase that has not been used yet, assign this noun as the subject of the verb phrase.

4.3.3 Identification of the Object.

(1) Iterate over every child of every element present in the Verb Phrase. If the dependency link between the child and the element of the Verb Phrase is one of the following: `pobj, dobj, iobj, xcomp, npadvmod` and the child is Noun and the element is not a preposition then assign it as an object. The structure of this rule is shown in Figure 18. The word in the verb phrase is attached to the object via any one of the object dependency tags.
e.g., As illustrated in Figure 19, the verb phrase *authorized to drive* has the object *car* directly attached to a word in the verb phrase via the `dobj` link.

(2) If the Verb Phrase has a preposition which has an object and the Verb Phrase also has a word which has an object then the fact will have a structure like the one below without the object. To handle the object of this fact, the verb phrase is broken down into two parts, the verb phrase part without the preposition and with the preposition. The preposition

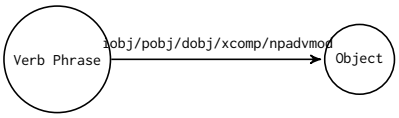

Figure 18: Structure of direct object link.

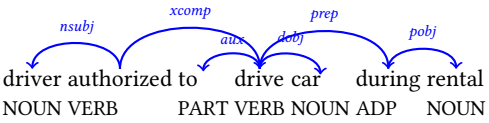

Figure 19: Dependency tree for direct object link.

is attached to the right of the verb phrase. The object of the preposition is attached to the right of the new preposition node created, while the object of the non-preposition word is attached to the left of the preposition node, as shown in Figure 20 and 21.
e.g., In sentence 'customer may have only one car at a time' the verb phrase *may have only at* has the word *only* has the noun and *one car* attached to it. The preposition *at* has the noun *time* attached to it. Thus, this is a verb phrase in which there are two words in the phrase having their own different objects.

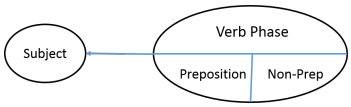

Figure 20: Structure of fact without object.

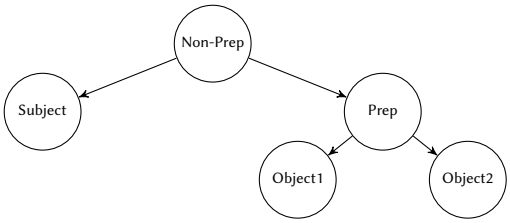

Figure 21: Structure of a verb phrase having 2 objects connected to it.

(3) If the Verb Phrase has two prepositions, both the prepositions having an object associated with them, then we remove the two prepositions from the verb phrase and separate them out into two different nodes. Thus the verb phrase has the prepositions taken away from it. The first preposition has the left child of it as null and the right child as the second preposition. The second preposition has the child of the first preposition as its left child and its own child as the right child. The structure is shown in Figure 22.

(4) If there is any preposition in the verb phrase connected to an object, then this object becomes the object of the preposition.

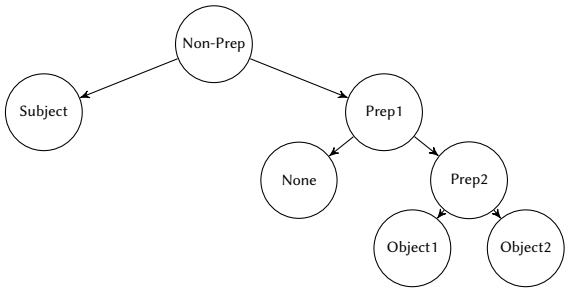

Figure 22: Structure of a verb phrase having 2 objects connected to it.

e.g., Consider the sentence 'end date of rental must be before any schedule maintenance of the car'. The verb phrase *must be before*, the object of this verb phrase *booking* is connected to the preposition *before* in the verb phrase and so this rule gets applied.

(5) If the object of the verb phrase assigned in the previous four stages has a `conj` link then identify the child with which the object has a `conj` link and replicate the same structure with the child as the new object.
e.g., In the example sentence 'if a rental request doesn't specify particular car group or model the default group is allocated'. The object of the verb phrase *does not specify*, i.e. *particular car group* has a `conj` link attached to it. Thus, the same structure is replicated and the node *particular car group* in the tree is replaced by *model*, which is the `conj` link word connected to *particular car group*.

(6) If there is no object for a verb phrase, and after finding the subject, there is a noun connected to the verb phrase that has not been used yet, assign this noun as the object of the verb phrase.

4.4 Rule Extraction

The input to the final stage is the set of facts extracted from the document. *Rules* are defined as the constraints on the *facts*. Any statement present without any constraint such as less than, more than etc. is just a *fact*. When we impose some sort of a constraint on the fact, it becomes a *rule*. It is these rules that we need to extract from the document.
As mentioned earlier, we represent the rules in SBVR. The types of rules that we consider in our approach are:

- **If-then rule**
  (a) IF fact1 and/or fact2 and/or $\cdots$ n times THEN fact1 and/or fact2 and/or $\cdots$ n times e.g., If the age of driver is greater than eighteen and he has a valid driving license then he is eligible to drive the car
  (b) fact1 and/or fact2 and/or $\cdots$ n times IF fact1 and/or fact2 and/or $\cdots$ n times. e.g., A car from another branch may be allocated, if there is a suitable car available and there is time to transfer it to the pick-up branch.
  (c) IF fact1 and/or fact2 and/or $\cdots$ n times THEN fact1 and/or fact2 and/or n times IF fact1 and/or fact2 and/or $\cdots$ n times.
  e.g., If there are not suffcient cars in a group to meet demand, a one group free upgrade may be given if there is capacity.
- m $\times$ n subject-object relations Subject1 and/or Subject2 and/or Subject3 $\cdots$ n times Verb Phrase Object1 and/or Object2 and/or Object3 $\cdots$ n times.
  e.g., Extras such as insurance, fuel and taxes must be paid by cash or credit card.
- atomic facts as obligatory rules.
  e.g., Local tax must be collected on rental charge.

Algorithm for Rule Extraction

The following algorithmic steps are carried out converting to SBVR rules using the entities and facts extracted.

(1) All the facts that were extracted are put into a single list. This list actually contains the roots of all the *fact-trees*.
(2) If the size of the list is 1 ie; there is only one fact for the entire sentence, then this rule sentence is an atomic fact. We add the rule header *It is obligatory that* to the atomic fact to get the rule.
(3) If the sentence contains an *if*, we find out which is the fact that has a word of it connected to this *if*. This fact is put in the if-list.
(4) If more than one *Ifs* are present, all facts connected to them via a `mark` tag are put into the if-list.
(5) The fact that comes just after the fact in the if-list is then put in the then list.
(6) If the fact that was put in the if-list was the last fact of the rule sentence, the first fact of the rule sentence is put in the then-list.
(7) If there are any conjunctions connected to the fact in the if-list, the other fact connected via the conjunction link is also put in the if-list. If this fact is in the then-list, remove it from the then-list.
(8) If there are any conjunctions connected to the fact in the then-list, the other fact connected via the conjunction link is also put in the then-list.
(9) Find out whether the phrase *other than* appears in the rule statement. If it does, create an other-than list and put the fact that comes just after the other-than in this list. Also, if the fact was present in the then-list, remove it from the then-list.
(10) Put all the remaining facts in the then-list.
(11) Connect all the facts in the if-list with the appropriate conjunctions. Also, connect all the facts in the then-list with the appropriate conjunctions. The conjunctions can be like `and` or `or`. The conjunction depends on that present in the sentence.
(12) Thus we classify all the facts extracted into one of the three lists if-list, then-list, or other-than-list.

Heuristics for Rule Mining

We list the heuristics that we follow for rule mining, based on our experiments and understanding of the problem.

- If there is a single *If* in a statement, the verb phrase which is connected by a `mark` tag to the *If* is put into the If-list.

The fact placed immediately next to that fact is put in the then-list. If the fact used was the last one, the first fact is put in the then list.
- If there are more than 1 Ifs in the statement, then both the facts which are connected to the 2 Ifs via the mark tag are put in the If list whereas the fact coming immediately after them is put in the then list.
- If there is a conjunction attached to one of the words in the fact attached to the If-list, then the other fact attached through the conjunction tag is also put in the same list as the one before (If-list or then-list).
- If there is an other than present in the sentence, an other-than-list is created and the fact coming right next to it is put in this list.
- If there is an 'unless' present in the sentence, an unless-list is created and the fact coming right next to it is put in this list.

5 EXPERIMENTAL STUDY AND RESULTS

In this section, we first present details of our experimental subjects and then discuss the various experiments that we conducted on our prototype tool which is embedded in the tool BuRRiTo [16].

5.1 Experimental setup

We have used SpaCY parser [21] for parsing the rule sentences, POS tagging, and for creating the dependency tree. We have built a prototype tool to implement our detailed approach. The experiments have been performed with the prototype on Windows 7 machine with COREi5 processor and 2 GB RAM.

We have evaluated our prototype tool with two sets of subjects: 1) Know Your Customer (KYC) [4] document consists of guidelines for banks about collecting various details of their customer in conducting their business. The set comprised of 630 sentences, out of them 185 sentences were marked as non-rule or non-relevant sentences. 2) The requirements for a fictitious car rental company EU-RentACar [1], the document has 64 rules.

5.2 Experiments conducted

We tried to address following research questions:
RQ1): The efficacy of our approach.
For the evaluation purpose, we have used precision and recall as the parameters for measuring the efficacy of our approach. Recall is defined as the ratio of number of True Positive Instances (TPI) to the number of actual positive instances (API). Precision is defined as the ratio of number of True Positive Instances (TPI) to the number of True Positive Instances and False Positive Instances (FPI). The below tables 4, 5 shows the precision and recall values observed for the EU-Rent, KYC documents, each for the entities, facts and rules.
RQ2: How well this technique performs comparing with earlier approach? We have compared results of our new approach with earlier approach [10, 11]. The following Table 6 shows the results of our study.

The results shows that our new approach has improved precision by 7-20%, recall by 7-25%, and speed up the process by 25%.

EU-Rent					
Test on	tool extracted	correctly extracted	incorrect extraction	precision	recall
Entities	277	271	6	97.83%	97.83%
Facts	177	159	18	89.83%	88.82%
Rules	64	54	10	84.38%	84.38%

Table 4: Results on EURent.

KYC Document					
Test on	tool extracted	correctly extracted	incorrect extraction	precision	recall
Entities	548	530	18	96.71%	96.71%
Facts	350	299	51	85.42%	85.42%
Rules	638	534	98	83.69%	84.49%

Table 5: Results on RBI KYC document.

Subject	Technique	time(sec)	precision	recall
EURent	earlier	245	89.82%	89.82%
EURent	current	181	96.71%	96.71%
KYC	earlier	683	63.81%	59.65%
KYC	current	597	83.69%	84.49%

Table 6: The comparison of this approach to our earlier approach.

6 CONCLUSION AND FUTURE WORK

In this paper, we have presented an unsupervised approach to mine SBVR vocabularies and rules from business documents. To achieve this, we have broken down the problem into four parts i.e. rule sentence extraction, entity extraction, fact extraction and rule mining. The first step was part of our earlier approach [], the second and third steps constitute the extraction of vocabulary from the document. The novelty of this approach lies in the fact that this is totally unsupervised and hence doesn't require the labelling of large amounts of training data. The heuristics that we have developed are completely domain independent. We need to experiment with many other documents to ascertain the acceptability of our system. For now, we have designed the system for the EU-rental document and have tested it on the EU-rental and KYC documents. The results of the experiments that we have conducted so far have indeed been promising in terms of the measures of accuracy, recall and precision. This all the more increases our motivation towards using a formal standard notation such as SBVR and making improvements to our techniques to make them useful for the industry application. In the future the system can be made better by refining the heuristics that we have developed so as to increase its efficiency, precision and recall measures. Also, the rules that we have handled for now are just the explicitly written if-then rules and the single fact rules. This can be extended to handle all the implicitly written if-then rules as well. One more extension could possibly be handling business documents written in different languages. It is highly probable that entities, facts and rules in the business documents written in a non-English language could also be extracted and converted into a

suitable standard format for that language. Another aspect of the task that is not handled currently is the feature of co-referencing. SpaCy does not support co-referencing and thus our system does not resolve the co-references used in the document. In the future however, this can be dealt with by using libraries provided for spaCy to resolve the problem of co-referencing.

REFERENCES

[1] [n. d.]. EU-RentACar case study. http://www.businessrulesgroup.org/first_paper/br01ad.htm. [Online; accessed 18-April-2016].

[2] [n. d.]. Object Management Group(OMG). http://www.omg.org. [Online; accessed 29-September-2015].

[3] [n. d.]. Poor Communication Leads to Project Failure One Third of the Time. Retrieved Oct 20, 2018 from https://www.coreworx.com/pmi-study-reveals-poor-communication-leads-to-project-failure-one-third-of-the-time/

[4] [n. d.]. Reserve Bank of India (RBI), Master Circulars. https://rbi.org.in/scripts/BS_ViewMasCirculardetails.aspx/?id=9031. [Online; accessed 18-April-2016].

[5] [n. d.]. Semantics Of Business Vocabulary And Rules (SBVR). http://www.omg.org/spec/SBVR/. [Online; accessed 29-September-2015].

[6] Mehdi Allahyari, Seyed Amin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. *CoRR* abs/1707.02919 (2017). arXiv:1707.02919 http://arxiv.org/abs/1707.02919

[7] Kritika Anand, Pavan Kumar Chittimalli, and Ravindra Naik. 2018. An Automated Detection of Inconsistencies in SBVR-based Business Rules Using Many-sorted Logic. In *Practical Aspects of Declarative Languages*, Francesco Calimeri, Kevin Hamlen, and Nicola Leone (Eds.). Springer International Publishing, Cham, 80–96.

[8] J. Atkinson-Abutridy, C. Mellish, and S. Aitken. 2004. Combining Information Extraction with Genetic Algorithms for Text Mining. *IEEE Intelligent Systems* 19, 3 (May 2004), 22–30. https://doi.org/10.1109/MIS.2004.4

[9] Imran Sarwar Bajwa, Mark G. Lee, and Behzad Bordbar. 2011. SBVR Business Rules Generation from Natural Language Specification. In *AI for Business Agility, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-03, Stanford, California, USA, March 21-23, 2011.*

[10] Abhidip Bhattacharyya, Pavan Kumar Chittimalli, and Ravindra Naik. 2017. An Approach to Mine Business Rule Intents from Domain-specific Documents. In *Proceedings of the 10th Innovations in Software Engineering Conference (ISEC '17)*. ACM, New York, NY, USA, 96–106. https://doi.org/10.1145/3021460.3021470

[11] Abhidip Bhattacharyya, Pavan Kumar Chittimalli, and Ravindra Naik. 2018. Relation Identification in Business Rules for Domain-specific Documents. In *Proceedings of the 11th Innovations in Software Engineering Conference (ISEC '18)*. ACM, New York, NY, USA, Article 14, 5 pages. https://doi.org/10.1145/3172871.3172884

[12] Branimir K Boguraev. 2000. Towards finite-state analysis of lexical cohesion. In *Proceedings of the 3rd international conference on finite-state methods for NLP.*

[13] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based N-gram Models of Natural Language. *Comput. Linguist.* 18, 4 (Dec. 1992), 467–479. http://dl.acm.org/citation.cfm?id=176313.176316

[14] C. C. Chiang. 2006. Extracting business rules from legacy systems into reusable components. In *2006 IEEE/SMC International Conference on System of Systems Engineering*. 6 pp.–. https://doi.org/10.1109/SYSOSE.2006.1652320

[15] Pavan Kumar Chittimalli and Kritika Anand. 2016. Domain-independent method of detecting inconsistencies in SBVR-based business rules. In *Proceedings of the International Workshop on Formal Methods for Analysis of Business Systems@ASE 2016*. ACM, 9–16.

[16] Pavan Kumar Chittimalli, Kritika Anand, Shrishti Pradhan, Sayandeep Mitra, Chandan Prakash, Rohit Shere, and Ravindra Naik. [n. d.]. BuRRiTo: A Framework to Extract, Specify, Verify and Analyze Business Rules. Automated Software Engineering (ASE 2020).

[17] Fabio Ciravegna. 2001. Adaptive Information Extraction from Text by Rule Induction and Generalisation. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2 (IJCAI'01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1251–1256.

[18] A. B. Earls, S. M. Embury, and N. H. Turner. 2002. A Method for the Manual Extraction of Business Rules from Legacy Source Code. *BT Technology Journal* 20, 4 (Oct. 2002), 127–145.

[19] Charles J. Fillmore. 1968. The Case for Case, Dins. In *Universals in Linguistic Theory*, Emmon Bach and R. Harms (Eds.). Holt, Rinehart, and Winston.

[20] S. Ghaisas, M. Motwani, and P.R. Anish. 2013. Detecting system use cases and validations from documents. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*. 568–573. https://doi.org/10.1109/ASE.2013.6693114

[21] Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 1373–1378. https://aclweb.org/anthology/D/D15/D15-1162

[22] Hai Huang. 1996. Business Rule Extraction from Legacy Code. In *Proceedings of the 20th Conference on Computer Software and Applications (COMPSAC '96)*. IEEE Computer Society, Washington, DC, USA, 162–. http://dl.acm.org/citation.cfm?id=872750.873408

[23] François Lévy and Adeline Nazarenko. 2013. Formalization of Natural Language Regulations through SBVR Structured English. In *Theory, Practice, and Applications of Rules on the Web*, Leora Morgenstern, Petros Stefaneas, François Lévy, Adam Wyner, and Adrian Paschke (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 19–33.

[24] Doug McDavid. [n. d.]. The Business-IT Gap: A Key Challenge. Retrieved Oct 20, 2018 from http://www.almaden.ibm.com/coevolution/pdf/mcdavid.pdf

[25] Ion Muslea et al. 1999. Extraction patterns for information extraction tasks: A survey. In *The AAAI-99 Workshop on Machine Learning for Information Extraction*, Vol. 2.

[26] T. Nasukawa and T. Nagano. 2001. Text analysis and knowledge mining system. *IBM Systems Journal* 40, 4 (2001), 967–984. https://doi.org/10.1147/sj.404.0967

[27] Rahul Pandita, Xusheng Xiao, Hao Zhong, Tao Xie, Stephen Oney, and Amit Paradkar. 2012. Inferring Method Specifications from Natural Language API Descriptions. In *Proceedings of the 34th International Conference on Software Engineering (ICSE '12)*. IEEE Press, Piscataway, NJ, USA, 815–825.

[28] Erik Putrycz and Anatol W. Kark. 2008. *Rule Representation, Interchange and Reasoning on the Web: International Symposium, RuleML 2008, Orlando, FL, USA, October 30-31, 2008. Proceedings.* Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter Connecting Legacy Code, Business Rules and Documentation, 17–30. https://doi.org/10.1007/978-3-540-88808-6_5

[29] Lawrence R Rabiner and Biing-Hwang Juang. 1986. An introduction to hidden Markov models. *ASSP Magazine, IEEE* 3, 1 (1986), 4–16.

[30] Colette Rolland and Camille Ben Achour. 1998. Guiding the Construction of Textual Use Case Specifications. *Data Knowl. Eng.* 25, 1-2 (March 1998), 125–160. https://doi.org/10.1016/S0169-023X(97)86223-4

[31] Ronald G. Ross. [n. d.]. The RuleSpeakÂ® Business Rule Notation. Retrieved Oct 20, 2018 from http://www.brcommunity.com/articles.php?id=b282

[32] A. Sinha, A. Paradkar, P. Kumanan, and B. Boguraev. 2009. A linguistic analysis engine for natural language use case description and its application to dependability analysis in industrial use cases. In *Dependable Systems Networks, 2009. DSN '09. IEEE/IFIP International Conference on*. 327–336. https://doi.org/10.1109/DSN.2009.5270320

[33] H. M. Sneed. 2001. Extracting business logic from existing COBOL programs as a basis for redevelopment. In *Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on*. 167–175. https://doi.org/10.1109/WPC.2001.921728

[34] Xinyu Wang, Jianling Sun, Xiaohu Yang, Zhijun He, and S. Maddineni. 2004. Business rules extraction from large legacy systems. In *Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings. Eighth European Conference on*. 249–258. https://doi.org/10.1109/CSMR.2004.1281426

[35] Xusheng Xiao, Amit Paradkar, Suresh Thummalapenta, and Tao Xie. 2012. Automated Extraction of Security Policies from Natural-language Software Documents. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE '12)*. ACM, New York, NY, USA, Article 12, 11 pages. https://doi.org/10.1145/2393596.2393608

[36] Hao Zhong, Lu Zhang, Tao Xie, and Hong Mei. 2009. Inferring Resource Specifications from Natural Language API Documentation. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering (ASE '09)*. IEEE Computer Society, Washington, DC, USA, 307–318. https://doi.org/10.1109/ASE.2009.94