PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
FACULTY OF INFORMATICS
GRADUATE PROGRAM IN COMPUTER SCIENCE

IDENTIFYING POTENTIAL CONFLICTS BETWEEN NORMS IN CONTRACTS

JOÃO PAULO DE SOUZA AIRES

Thesis presented as partial requirement for obtaining the degree of Master in Computer Science at Pontifical Catholic University of Rio Grande do Sul.

Advisor: Prof. Felipe Rech Meneguzzi
Co-Advisor: Prof. Vera Lucia Strube de Lima

Porto Alegre
2016

Dados Internacionais de Catalogação na Publicação (CIP)

A298i Aires, João Paulo de Souza

Identifying potential conflicts between norms in contracts / João Paulo de Souza Aires. – 2016.
67 p.

Diss. (Mestrado) – Faculdade de Informática, PUCRS.
Orientador: Prof. Dr. Felipe Rech Meneguzzi.

1. Conflitos Normativos. 2. Lógica Deôntica. 3. Processamento da Linguagem Natural. 4. Automação de Contratos. 5. Informática. I. Meneguzzi, Felipe Rech. II. Título.

CDD 23 ed. 006.35

Ramon Ely CRB 10/2165
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "*Identifying Potential Conflicts Between Norms in Contracts*" apresentada por João Paulo de Souza Aires como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, aprovada em 07 de agosto de 2015 pela Comissão Examinadora:

Prof. Dr. Felipe Rech Meneguzzi – Orientador — PPGCC/PUCRS

Profa. Dra. Vera Lúcia Strube de Lima – Coorientadora — PPGCC/PUCRS

Profa. Dra. Renata Vieira – — PPGCC/PUCRS

Profa. Dra. Ingrid Oliveira de Nunes - — UFRGS

Homologada em 09/12/2015, conforme Ata No. 022 pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

IDENTIFYING POTENTIAL CONFLICTS BETWEEN NORMS IN CONTRACTS

RESUMO

Contratos são utilizados para formalizar acordos envolvendo troca de bens e serviços entre duas ou mais partes. Eles definem ações esperadas durante o período de vigência do contrato através de normas. Tais normas seguem conceitos baseados em lógica deôntica, definindo permissões, proibições e obrigações. No entanto, conflitos podem ser gerados quando duas normas são aplicadas a um mesmo contexto tendo sentidos deônticos diferentes, como a proibição e a obrigação da realização de uma mesma ação. Estes conflitos invalidam as normas e criam uma inconsistência para o contrato. Para evitá-los é necessário que um revisor leia as normas e encontre quais apresentam elementos conflitantes. Uma vez que contratos podem ser longos e complexos, esta tarefa consome tempo e é passível de erro humano. Para automatizar o processo de identificação de conflitos, neste trabalho criamos uma abordagem que busca identificar potenciais conflitos entre normas através da comparação de suas estruturas. Nossa abordagem é dividida em duas fases: na primeira, identificamos as normas e seus elementos dentro de um contrato; na segunda, utilizamos os elementos extraídos para comparar diferentes normas e identificar quais apresentam características de um conflito normativo. Nós avaliamos a abordagem aplicando-a em contratos contendo conflitos e obtivemos resultados com acurácia superior a 70%.

Palavras-Chave: conflitos normativos, lógica deôntica, processamento de linguagem natural, automação de contratos.

IDENTIFYING POTENTIAL CONFLICTS BETWEEN NORMS IN CONTRACTS

ABSTRACT

Contracts formally represent agreements between parties and often involve the exchange of goods and services. In contracts, norms define the expected behaviors of the parties using deontic statements, such as obligations, permissions, and prohibitions. However, norms may conflict invalidating themselves and producing a contract inconsistency. A conflict arises when two or more norms are applied to the same context but have different deontic statements, such as permissions x obligations and prohibitions x obligations. The identification of such conflicts is often made by humans, which makes the task time consuming and error-prone. In order to automate such identification, in this work we propose an approach to identify potential conflicts between norms in contracts written in natural language. We build a two-phase approach that extracts norms and norm elements from contracts, creating a norm representation that we use to compare norms and identify potential conflicts. We evaluated the approach using a corpus of contracts with norm conflicts inserted, and we measured the accuracy for different cases of conflict, which resulted on values higher than 70%.

Keywords: norm conflicts, deontic logic, natural language processing, contract automation.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CONTENTS

1. INTRODUCTION

In social groups, interactions between members often follow some kind of regulation to minimize conflicting behavior. This regulation is based on expected behaviors for each group member, which ensures that members follow a socially accepted behavior. In order to formalize expected behaviors, norms are created as group regulators enforcing a defined conduct for each specific case. Their definitions usually respect a social consensus, which is formed by group representatives. In general, norms follow deontic concepts that define three types of modalities: permissions, obligations and prohibitions [35]. A permission defines a behaviour that is allowed to be executed. An obligation defines a behaviour that must be executed. Finally, a prohibition defines a behaviour that must not be executed. These concepts are present in most social relationships involving agreements between members of a society. Agreements are usually formed by two or more parties that agree in the exchange of goods or services. Such agreements are often formalized by contracts, which describe the parties and a set of clauses. Within the clauses, contracts describe pre-defined norms to be agreed upon.

A contract defines the parties, their relations, and a set of clauses containing norms that explain what each party must comply with [10]. Since the use of contracts to regulate the exchange of goods and services through the Internet has increased, contracts have been formalized for use in the online context, Online contracts are increasingly used for different kinds of agreements involving the trading of products and services. This increase demands much more effort in the process of contract creation and analysis, since contracts tend to be long and complex documents.

Contractual norms include some kind of logical description of behaviors, defining what is obliged, permitted or forbidden. For example, in a restaurant, customers are prohibited from smoking and obliged to pay for the food they eat. However, conflicts between norms may occur depending on the objects they refer to and the modalities used in them. A conflict arises when norms that represent different deontic concepts are applied to the same target. One example is when a norm obliges a student to hand in an assignment and another one prohibits him doing so. Conflicts of such type may invalidate norms involved, confusing the party according to what is expected to do in the situation described. Thus, a specific requirement in this scenario is to ensure that norms described in contracts do not introduce inconsistencies into them. Such effort can prevent the case when long and complex contracts have conflicts introduced unwittingly. Automating the identification of these conflicts avoids contract inconsistency and assists the human analysis, facilitating the task of writing and correcting contracts.

Identifying potential conflicts between norms within contracts is usually done by humans that manually check each norm and compare their meaning. This is a laborious and time-consuming process, which in turn makes contract verification slow, difficult, and error-prone. There is a great potential to improve the speed and quality of verification by automating such process. Recent efforts in identifying conflicts between norms include two important works. First, Vasconcelos *et al.* [33] present an approach for identifying and resolving norm conflicts in multi-agent systems. They use

deontic logic concepts and a norm formalization to identify when two norms present a conflict and then resolve them. Second, Figueiredo and Silva [7] present an algorithm that identifies conflicting norms. The algorithm is based on a logical language that facilitates the comparison between norms and as a consequence identify conflicting norms by logical conflicts.

To make conflict analysis possible, we need to, first, extract the elements that compose a norm. For this task, two works develop approaches that deal with the structure of contracts written in natural language and their main elements. In the first one, Gao and Singh [9] extract normative relationships in business contracts. In the second one, the same authors [8] present an approach for the extraction of other norm properties and information, such as contract events and their temporal constraints. These works focus on information extraction from contracts and resolution of conflicting norms. However, to the best of our knowledge, no existing approach deals with normative conflicts in contracts written in natural language.

In order to develop an approach to identify potential conflicts, we divide our efforts into two main phases. First, we need to deal with the contract structure and its norms and elements. For this end, we use natural language processing techniques, which allow us to extract norm elements needed to create a norm representation. In the second phase, we use the extracted norm representations to compare the norms according to the parties to which they are applied. In this phase, we focus on the comparison of norm elements, using concepts of deontic logic and language similarity to identify corresponding information in norm pairs that may produce a conflict. To evaluate our approach, we developed a system to assist a user to insert conflicts within real contracts. We selected two volunteers that inserted norms to conflict with existing norms in the contracts. Thus, using the set of contracts with conflicts we test our identifier that obtains an accuracy of 78% overall inserted conflicts.

In this work, we have two main contributions. The first one is related to the development of the first step to the automation of contract analysis. Since contract analysis is poorly developed ([5, 9, 8]), this work improves the area by presenting an approach that provides a basis to more complex approaches. Our second contribution is a tool that can be used to assist in preventing conflicts from being inserted into contracts. Such contribution is related to the practical use of the tool, improving the contract creation by warning about potential conflicts.

This document is divided into six chapters. Chapter 2 provides an overview about the main concepts we use in this work, such as norms, contracts, deontic logic, and natural language processing. In Chapter 3, we describe extract information phase designed to create a norm representation that allows us to make comparisons between norms. Chapter 4 presents the second phase of the work in which we use deontic logic concepts and natural language processing to determine which norm pairs are potentially conflicting in a contract. In Chapter 5, we present the most relevant works in the area, highlighting three contexts and presenting different approaches in which we base and compare our work. Finally, in Chapter 6 we conclude our work making a review and discussing about future work.

2. BACKGROUND

The identification of potential conflicts between norms involves two main concepts: natural language processing (NLP) and deontic logic. In this chapter, we describe the structure of norms and contracts, the concept of conflict between norms, and the main tasks in NLP that allow us to detect and identify potential conflicts in this context.

2.1 Norms and Deontic Logic

According to Axelrod [2], "a norm exists in a given social setting to the extent that individuals usually act in a certain way and are often punished when perceived not to be acting in this way". Norms arise when a coordinated behavior serves to regulate conflict under a large number of individuals, and provide a powerful mechanism for regulating conflict in groups, governing much of our political and social lives. In contracts, clauses define a set of norms. In such case, these norms indicate what is expected from each party according to the contract definitions. Most norm representations are based on deontic logic, using concepts such as permissions, obligations and prohibitions. These concepts describe the type of restriction intended of a norm.

Deontic Logic has its origins in philosophical logic, applied modal logic, and ethical and legal theory. The aim of deontic logic is to describe ideal worlds, allowing the representation of deviations from the ideal (i.e. violations). Thus, deontic logic and the theory of normative positions have strong relevance to legal knowledge representation, and consequently it is applied to the analysis and representation of normative systems [14]. Norms often present deontic concepts to describe permissions, obligations, or prohibitions. A prohibition norm indicates an action that must not be done, and, if such action is carried out, a violation occurs. Conversely, a permission norm indicates an action that can either be performed or not, and no violation occurs in either case. In most deontic systems, a prohibition is considered to be equivalent to the negation of a permission, thus, an action that is not permitted comprises a prohibition. Although these two modalities are sufficient to represent most norms, obligations are also commonly employed in norm representation. An obligation represents an action that must be done, and it is equivalent either to the negation of a permission not to act or a prohibition not to act. Formally, we represent norms using the modal operators of P for permission, F for prohibition, and O for obligation. Now, if ϕ is an action we can represent, based on Wright [35] definitions, the equivalences between these deontic concepts are as follows:

- $P\phi \equiv P\phi \vee P\neg\phi$
- $F\phi \equiv O\neg\phi \vee \neg P\phi$
- $O\phi \equiv F\neg\phi \vee \neg P\neg\phi$

Example 2.1.1 exemplifies sentences presenting the equivalence between obligation and prohibition. Equivalences allow us to compare and identify conflicts between norms. Conflicting norms often have different modal operators, thus, knowing theirs equivalences make it easy to identify differences.

Example 2.1.1. :

- Company X must follow rule Y.
- Company X must not disconsider rule Y.

2.2 Norm Representation

A norm can be represented in different ways in order to be understood and processed by a computer. In this section, we present four approaches to represent norms. First, we present Sergot's approach, which uses Horn clauses to represent norms. Second, we describe Benthan's approach that uses the logic of rights to represent a norm. Third, we present Singh's approach, which proposes defined structures for each norm element. Fourth, we present two approaches based on the RuleML markup language.

2.2.1 Sergot's approach

Sergot [29] presents an example using the acquisition of British Citizenship at birth to represent a norm into a Horn clause. Considering the clause:

1- A person born in the United Kingdom after commencement shall be a British Citizen if at the time of birth his father or mother is

(a) a British citizen; or

(b) settled in the United Kingdom.

Analyzing the clause, one can establish that 'after commencement' is related to a moment in time that occurs after or the on the data on which the Act comes into force. Such clause of Act can be formalized as a Horn clause, once we take into consideration the date on which an individual acquires citizenship and the section of the Act by which he does so. Sergot [29] represent this clause as Example 2.2.1 shows:

Example 2.2.1. :
x acquires British citizenship on date y by section q.w
if x was born in the UK
and x was born on date y

and y is after or on commencement
and z is a parent of x
and (z is a British citizen on date y by section y1
or
z is settled in the UK on date y).

Although Sergot's approach is a considerable way to represent norms logically, human intervention is necessary to select which predicates to use and what is important to select in a clause. Once represented, norms using Sergot's approach can be processed by computers that can check whether a pair of norms is conflicting. However, in this work we want to automate this process, whereas using Sergot's approach would not satisfy our goal.

2.2.2 Bentham's approach

The Horn clause representation is one of the possible approaches to represent norms. Bentham [3] also presents an approach to represent legal rules using logic. To represent legal rules, he presents the following conventions:

(1) p, r, q are variables for persons;

(2) A_p, B_p are variables for action-propositions, where p is the agent;

(3) A_q, B_q are variables for action-propositions, where q is the agent;

Where A_p is read as "p performs action A", and B_q is read as "q performs action B". Bentham defines the logic of imperation, which deals with the expressions of the speaker's will and provides the basis of a logic of rights and obligations. An example of the logic of rights and obligations as follows:

> Obligation(p, A_p) for:
> p has an obligation to the effect that A_p;
> Obligation(p, q, A_p) for:
> p has to q an obligation to the effect that A_p;
>
> the logic of rights and obligations can be represented by the following axioms and definitions:
>
> $Axioms$:
> B1. ¬(Obligation(p, A_p) & Obligation(p, $\neg A_p$))
> B2. Obligation(p, q, A_p) ⇒ Obligation(p, A_p).

$Definitions$:
Def. 1: Right(q, p, A_p) $=_{def.}$ Obligation(p, q, A_p) & ($p \neq q$).
Def. 2: Liberty(p, A_p) $=_{def.}$ Obligation(p, A_p).

In the axioms, B1 refers impossibility to subject p be obliged to perform A at the same time he is obliged to perform $\neg A$; B2 says that the obligation(p, A_p) is a commanded from a logical consequence of commanded that A_p. Def. 1 says that, in cases where the parties are not identical, when q has the right to a service A made by p, it means the obligation of p to perform A. Def. 2 says that the liberty to perform an action is the same as the absence of the obligation to avoid the action.

2.2.3 Singh's approach

Singh [30] proposes a norm representation applied to governance in organizations, which highlights aspects such as, subject, object, and context. He presents an approach for governing sociotechnical systems, which provides a norm representation for each norm. A norm representation, applied to this context, has five components: a subject, an object, a context, an antecedent, and a consequent.

- **Subject**: is the party in which the norm is focused and the one who has to perform the consequent;
- **Object**: the principal with respect to whom the norm arises;
- **Context**: the organization within whose scope the norm arises;
- **Antecedent**: expresses the conditions in which the norm is fully activated; and
- **Consequent**: expresses the conditions in which the norm is fully satisfied and thus deactivated.

Singh's approach has two components that we adapt for our work. First, the subject component, which plays an important role on the identification of a norm conflict since conflicting norms are often applied to the same party. Thus, identifying the subject in a norm representation guarantees that we only compare norms with the same subject. Second, the consequent component, which describes the norm action that must be accomplished by the subject. As conflicting norms often have different deontic meanings (obligation x prohibition), we need to compare the norm consequents to verify whether such deontic meanings refer to the same consequents. Example 2.2.2 illustrates the approach.

Example 2.2.2. :

- In case of abandonment, Company X must pay every extra cost.

(a) **Subject**: “Company X”;

(b) **Object**: “must pay every extra cost”;

(c) **Context**: the organization in which company X is part of;

(d) **Antecedent**: “In case of abandonment”;

(e) **Consequent**: “pay every extra cost”;

2.2.4 RuleML approaches

RuleML is an XML-based language created to represent different types of rules. The language is also capable of specifying queries and inferences in Web ontologies, which are devoted to knowledge representation. It maps between Web ontologies and dynamic Web behaviors of workflows, services and agents [11]. The language was created to be a canonical Web language for rules, allowing the exchange of rules between major commercial and non-commercial rule systems on the Web.

RuleML deals with different types of rules in its representation, here we cover three of them, namely reaction rules, transformation rules, and derivation rules. Reaction rules are event-condition-action-effect rules, dealing with invoking actions from a reaction to an event. Transformation rules are functional-equational rules, which represent the transformation of a rule according to a condition. Derivation rules are implicational-inference rules, a different type of transformation rule that comprises one or more conditions but derive only one conclusion for the rule. Other elements are used, such as Facts, Queries and Integrity Constraints.

When applied to contracts, the RuleML structure is organized based on the norms. Listing 2.1, extracted from [11], presents how the RuleML structure is applied to norms.

Listing 2.1 – RuleML structure

```
<Imp label="4.1" href="http://supplier.com/catalog.htm">
  <body>
    <And>
        <Atom>
            <Rel>PurchaseOrder</Rel>
            <Ind>Purchaser</Ind>
            <Ind>Supplier</Ind>
            <Var>Good</Var>
        </Atom>
        <Atom>
            <Rel>AdvertisedPrice</Rel>
            <Ind>Purchaser</Ind>
            <Ind>Supplier</Ind>
            <Var>Good</Var>
```

```
                    <Var>Price</Var>
                </Atom>
            </And>
        </body>
        <head>
            <Obligation subject="Purchaser">
                <Rel>PurchaseOrderPrice</Rel>
                <Var>Good</Var>
                <Var>Price</Var>
            </Obligation>
        </head>
    </Imp>
```

The elements in Listing 2.1 are described below:

- **<Imp>**: defines the annotated clause and indicates the clause number by using the attribute $label$;
- **<And>**: joins two predicates defined by <Atom> tag;
- **<Atom>**: indicates the predicate content of the clause;
- **<Rel>**: tag that defines the name of a predicate;
- **<Ind>**: defines the individual involved in the predicate;
- **<Var>**: refers to ground values related to predicate; and
- **<Obligation>**: tag that indicates an obligation referent to the ground values (Good and Price) of a purchase order.

LegalRuleML [1] is an extension of RuleML, implementing specific features for normative formalisms, such as, quantification of norms, defeasibility rules, deontic operators, temporal management of the rules, and temporal expressions within rules. Defeasibility rules specify that a fact is typically a consequence of another fact; deontic operators define the norm type, such as permission, prohibition and obligation; temporal management of the rules and temporal expressions within rules define information based on the time related to the norm.

The language defines new tags to represent different elements in norms. <TimeInstants> and <TemporalCharacteristics> model the events, intervals and temporal parameters that define the period of validity of the rules. <Agents> and <Authorities> are two classes for defining, respectively, the author of the rule formalization and the associated authority. Using these definitions it is possible to more specifically represent the norms in contracts. Listing 2.2 (extracted from [1]) shows the use of <TimeInstans> tag defining the temporal register of a rule. Listing 2.3 presents the tags <Agents> and <Authority>. The <Agent> tag represents the agents associated with the rule, defined by an attribute 'key'. The <Authority> tag defines the authority that determines the rule, in the example defined by 'congress'.

Listing 2.2 – LegalRuleML example

```
<lrml:TimeInstants>
  <ruleml:Time key="t1">
    <ruleml:Data xsi:type="xs:dataTime">
          1978-01-01T00:00:00
        </ruleml:Data>
  </ruleml:Time>
</lrml:TimeInstants>
```

Listing 2.3 – LegalRuleML example with Agents and Authority

```
<lrml:Agents>
  <lrml:Agent key="aut1"
        sameAs="&unibo;/person.owl#m.palmirani"/>
  <lrml:Agent key="aut2"
        sameAs="&unibo;/person.owl#g.governatori"/>
</lrml:Agents>
<lrml:Authorities>
  <lrml:Authority key="congress"
        sameAs="&unibo;/person.owl#congress"/>
        <lrml:type iri="&lrmlv;Legislature"/>
  </lrml:Authority>
</lrml:Authorities>
```

Both approaches are based on XML markup language, which is broadly used to structure and process information. However, these norm representations contemplate much information that does not help in the identification of norm conflicts. Therefore, in this work we do not use either RuleML or LegalRuleML, representing norms with components that emphasize valuable features for the conflict identification.

2.3 Contracts and their components

A contract is an agreement that two or more parties enter voluntarily, when it is useful to formalize that a certain duty comes into existence by a promise made by at least one of the parties. The creation of a contract formalizes what each party expects from the other, creating a warranty that the duties will be fulfilled [27]. Therefore, it creates legally enforceable obligations between the parties. These enforceable obligations are defined by a series of norms, which are responsible for defining any expected behavior from the parties. Contracts are used in many situations for different purposes. In any legal agreement, a contract handles the formalization of agreement conditions.

With the use of the Internet, electronic contracts arise as a new way to represent formal agreements and are increasingly explored for commercial services. An electronic contract is very similar to a traditional paper-based commercial contract, following the same rules and structure. Almost all types of contract can be represented electronically, leading to the need of managing such contracts, dealing with representation and evaluation of agreements. In this work, we deal

with contracts written in natural language, thus, the task of analyzing and evaluating norms is traditionally done by human readers. As more contracts are required to codify an increasing number of online services which span over multiple countries and different legal systems, the tasks of writing and verifying contracts by humans become more laborious, taking substantial time [10].

Contracts have three main components, which define the content and the contracts purpose, namely, *promise*, *payment*, and *acceptance* [27]. A promise in a contract represents a communication of a commitment related to a future intent. The key element in communicating a promise in a contract is a behavioral event, which means a commitment to do (or not do) something. With the promise defined, payment is the element which offers something of value in exchange for the promised. A payment may be considered as another promise, made by the other party. Finally, the acceptance is the voluntary participation that reflects each party's willingness to make commitments to the other.

Analyzing contracts from the Australian contract corpus [6] and the corpus provided by Gao *et al.* [10], we notice that most of the them are (see Figure 2.1) a semi-structured document often divided into an initial description of the parties (header) and a set of clauses describing the expected behaviors from the parties. In this work, we consider this division as a basis to extract information from contracts. The header contains information about the parties, it presents and assigns them abbreviations or equivalent names. The set of clauses in a contract determines what each party must comply with along the duration of the contract. Norm clauses are often directed to one or more parties, likewise they may describe an expected behavior from the agreement itself.

2.4 Modal Verbs and Their Deontic Meanings

As Section 2.2 describes, norm representations use concepts from deontic logic as basis to describe permissions, prohibitions, and obligations. Such elements are often represented by modal verbs in natural language norms. Modality and modal verbs are different concepts since modality describes deontic concepts and modal verbs in sentences describe modalities. For example, in the sentence: "Company X **must** pay the product taxes."; **must** is the modal verb that describes the obligation modality.

English has a set of modal verbs including: MAY, CAN, MUST, OUGHT (TO), WILL, and SHALL [22]. Moreover, *might*, *could*, *would*, and *should* are usually considered as modal verbs as well. From this set, we can map the modal verbs into deontic meanings, i.e., permission, prohibition, and obligation. Steedman [31] maps MUST and MAY to obligation and permission, respectively. The remaining of modal verbs are mapped according to English grammar rules [22]. Thus, we adopt the following mappings:

- CAN and MAY are mapped to permissions;
- MUST, SHALL, OUGHT, and WILL are mapped to obligations; and

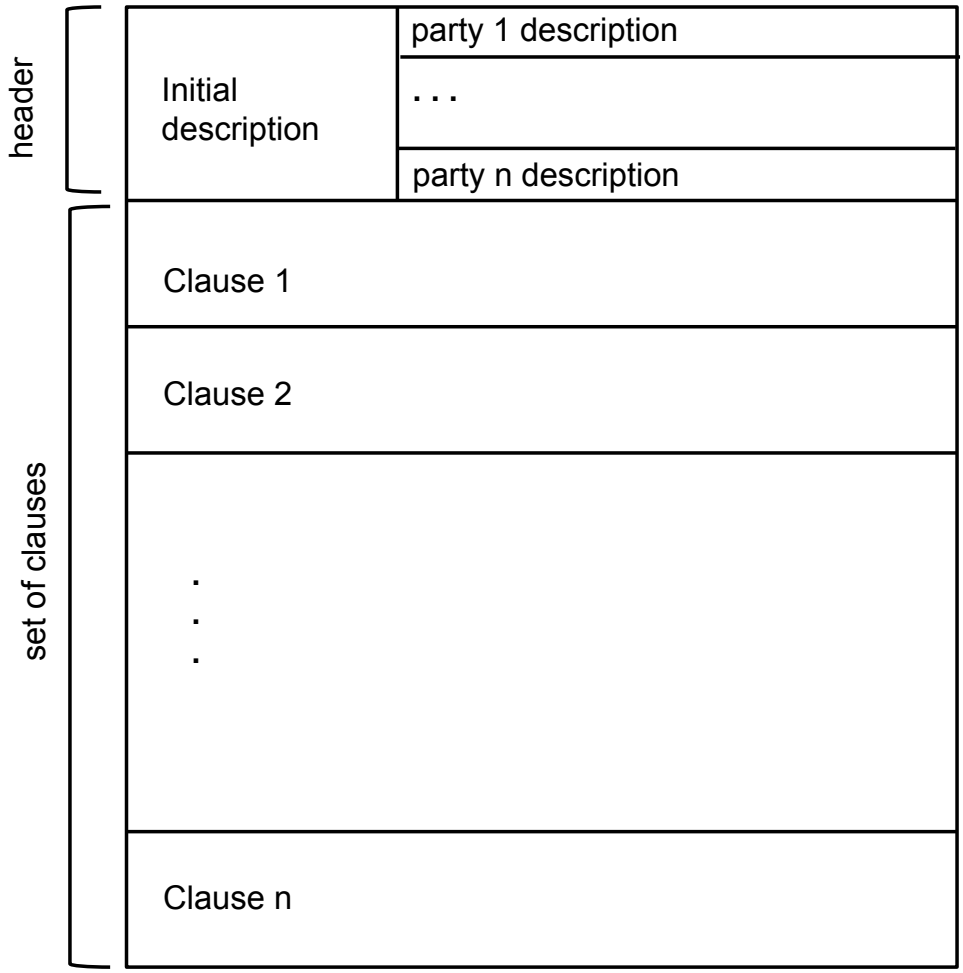

Figure 2.1 – Contract structure

- the negation of any modal is mapped to a prohibition.

Table 2.1 illustrates the mapping between modal verbs and deontic meanings based on the associations made by Steedman [31] and complemented according to grammar rules interpreted by Palmer [22].

Table 2.1 – Mapping modal verbs into deontic meaning

Modal Verb	**Deontic Meaning**
CAN, MAY	Permission
MUST, OUGHT, SHALL, WILL	Obligation
CANNOT, MAY NOT, MUST NOT, OUGHT NOT, SHALL NOT, WILL NOT	Prohibition

Using the mapping in Table 2.1, we can identify the deontic meaning of a norm by means of its modal verb. Examples from 2.4.1 to 2.4.3 [1] present sentences and their extracted deontic meaning.

[1] Extracted from http://goo.gl/IE3q6H and http://goo.gl/E8brjp

Example 2.4.1. Purchaser **shall** also be financially responsible for all taxes and freight in connection with the New Equipment. (**Obligation**)

Example 2.4.2. Each party hereto **shall not** disclose any confidential information received by it pursuant to this Agreement without the prior written consent of the other. (**Prohibition**)

Example 2.4.3. Roxio, upon request, **may** review such agreements at any time before or after execution. (**Permission**)

2.5 Conflicts and Inconsistencies

In order to understand the concept of norm conflict, we present a series of norm conflict concepts extracted from different areas in literature, such as international law and multi-agent systems. One concept of norm conflict is the one applied to international law. It states that a conflict between two norms arises when "a party to two treaties cannot comply with its obligations in both treaties simultaneously" [12]. The sentences in Example 2.5.1 illustrate this type of conflict.

Example 2.5.1. :

- If the equipment presents an error, Customer A must send it to address X.
- If the equipment presents an error, Customer A shall send it to address Y.

In this case, both norms indicate different actions (different addresses) for the same event, making compliance with both norms impossible and invalidating them.

Although this concept is the prevailing view in international law, Vranes refines this definition, characterizing conflicts between permissions and obligations, and permissions and prohibitions [36]. A conflict between a permission and an obligation arises when the party has permission for a defined action and, simultaneously, the obligation for the same action. For example, see the sentences in Example 2.5.2.

Example 2.5.2. :

- Company A must pay the taxes.
- Company A may choose to pay the taxes.

In the example, the first norm states that the party must pay the taxes, however, the second norm permits the party to choose whether to pay the taxes or not. In the example, one norm invalidates the other producing a conflict. A conflict between a permission and a prohibition arises when the norms, simultaneously, permit and forbid the party to perform a certain action. For example, in sentences of Example 2.5.3. Both conflicts between permission x obligation and

permission x prohibition can be better understood using a logical representation. Consider the norms in Example 2.5.3, if we rewrite them logically using logical quantifiers we obtain sentences as in Example 2.5.4. In the example, norm 1 specifies that exists a product that must not be bought. On the other hand, norm 2 generalizes the possibility of buying a product for every x when x is a product. It generates a conflict, since norm 2 overlaps norm 1 creating the possibility of buying to every product.

Example 2.5.3. :

1. Company A must not buy the product X.
2. Company A may buy any product.

Example 2.5.4. :

1. $\{\exists x \mid product(x) \rightarrow \neg buy(x)\}$
2. $\{\forall x \mid product(x) \rightarrow buy(x) \vee \neg buy(x)\}$

Kollingbaum et al. [16] consider the concept of norm conflict the same as the second conflict type of Vranes. They define that a norm conflict is an interference between permissions and prohibitions. These concepts are similar to the ones presented by Sadat-Akhavi [28], which states that norm conflicts arise when one cannot comply with all requirements of two norms. It means that it is impossible to comply with both norms, since they are mutually exclusive and cannot coexist in a legal order. Thus, compliance with one norm entails non-compliance with the other.

Sadat-Akhavi [28] describes four causes for a norm conflict to arise. The first cause is when the same act is subject to different types of norms. Thus, a conflict of norms arises "if two different types of norms regulate the same act, i.e., if the same act is both obligatory and prohibited, permitted and prohibited, or permitted and obligatory". Example 2.5.5 shows a norm conflict between an obligation and a prohibition.

Example 2.5.5. :

- Norm 1: The receiving State shall exempt diplomatic agents from indirect taxes.
- Norm 2: The receiving State shall not exempt diplomatic agents from indirect taxes.

The second cause is when one norm requires an act, while another norm requires or permits a 'contrary' act. Therefore, a conflict of norms is produced if "two contrary acts, or if one norm permits an act while the other norm requires a contrary act" [28]. Example 2.5.6 illustrates the conflict. Both norms indicate different places in which a prisoner of war must be treated. Norm 1 states that it must be done in the prisoner camps, whereas Norm 2 states that it must happen in civilian hospitals. The conflict arises in the moment that one tries to comply with one norm and, at the same time, is non-complying with the other.

Example 2.5.6. :

- Norm 1: Prisoners of war suffering from disease shall/may be treated in their camps.
- Norm 2: Prisoners of war suffering from disease shall/may be treated in civilian hospitals.

The third cause for a norm conflict is when one norm prohibits a 'necessary precondition' of another norm. Suppose two actions A and B, where B cannot be performed without A been performed before. In this case, a norm conflict arises when one norm prohibits A and another norm allows B, as Example 2.5.7 shows. In the example, we consider action A as "enter area X" and action B as "render assistance to any person in danger in area X". To comply with Norm 1 one must disobey Norm 2.

Example 2.5.7. :

- Norm 1: Ships flying the flag of State A shall/may render assistance to any person in danger in area X.
- Norm 2: Ships flying the flag of State A shall not enter area X.

The fourth cause for a norm conflict to arise is when one norm prohibits a 'necessary consequence' of another norm. Suppose that one cannot perform action B without producing A as result. A conflict arises when one norm obliges B and another norm prohibits A, as Example 2.5.8 shows. If we consider action B as "replace existing rails in area X" and A as the period of time that the line in area X will be hampered, one cannot comply with both norms 1 and 2 in Example 2.5.8.

Example 2.5.8. :

- Norm 1: State A shall replace existing rails with new ones in area X.
- Norm 2: State A shall not hamper the transport of goods on the existing line in area X.

Vasconcelos et al. [34] present a concept of norm inconsistency that is similar to one type of conflict presented by Sadat-Akhavi. They state that a norm inconsistency occurs when "a substitution α can be found and unifies an obligation and a prohibition". The authors apply such concept in an agent scenario, arguing that the obligation demands that an agent performs an action that is forbidden. Example 2.5.9 shows such an inconsistency between norms.

Example 2.5.9. :

- Company A must pay every employee.
- Company A shall not pay employees that perform function X.

Given all these approaches to conflicts and inconsistencies, in this work, we decide to base our notion of conflict on the definitions provided by Sadat-Akhavi. His definitions cover the most common aspects of conflicts between natural language norms. Thus, a conflict may be of three types: (1) permission x prohibition, (2) permission x obligation, and (3) obligation x prohibition. Based on such types, we can classify a potential conflict according to these conflict types.

Since we work with contracts written in natural language, some Natural Language Processing (NLP) tasks are necessary to extract contract information. We use such information to create a norm representation that allows us to compare such norms in order to detect potential conflicts. For this representation, we want to extract norm components, such as modal verbs, party names, and norm actions.

2.6 Named Entity Recognition

Named entities are phrases that contain the names of persons, organizations and locations [32]. For example, in the sentence "*Cathy works at IBM in New York.*", we can identify three named entities, a person name (*Cathy*), an organization (*IBM*), and a location (*New York*). The identification of named entities is an important task for many Natural Language Processing (NLP) tasks, such as information extraction and relationship extraction. The task of detecting and classifying named entities in text is called Named Entity Recognition (NER) [15]. In such task, words that represent named entities in text are tagged according their classes. Using the previous example, applying NER to the sentence would result in the following annotation: "*Cathy/PERSON works at IBM/ORGANIZATION in New York/LOCATION*". Using NER to identify named entities in a sentence allows us to extract relations among the mentioned entities.

In contracts, one may find every type of named entity, such as names of persons, organizations, and locations. However, to identify the parties of a contract, person names and organizations are the main entities to consider, since contracts usually define an agreement between two or more persons, organizations, or persons and organizations. Parties in a contract are usually presented in the opening paragraphs of the contract (see Figure 2.1), which describes the parties and their characteristics. In a contract often an alias is attributed to the party, in order to easily identify it within the text. Therefore, both party name and alias must direct to the same party.

2.7 Semantic Similarity

Semantic similarity describes how connected two concepts are, i.e., how many properties they share in common [25]. In this work, we use semantic similarity to measure the distance between two norm actions. The idea is to identify norm actions that could have the same or similar meaning. Such identification is important because similar norm actions may have different deontic meanings,

i.e., one norm can be allowing a certain norm action, whereas another is forbidding a similar or equal norm action. When norm are applied to the same party, this situation describes a norm conflict.

In many fields, similarity is used to measure the distance between two or more elements. Applied to NLP, similarity can be used to measure the distance between words [23], between sentences, and between texts. The computation of semantic distance between text elements is a fundamental step to find different elements with the same meaning. Sentences in Example 2.7.1 are semantic similar as they are written in different ways but meaning remains the same.

Example 2.7.1. :

- Company X may choose to use three different products of Company Y.
- Company X may choose to use three among the available products of Company Y.

Different approaches were developed to measure semantic similarity. For the English language, WordNet [2], which is a lexical database with 155,000 nouns, verbs, adjectives, and adverbs, offers six measures of similarity between words [24]. These measures are based on the structure and content of WordNet. When measuring similarity between concepts, WordNet refers to these concepts as synsets. To measure such similarity, WordNet uses a hierarchy structure. For example, the word "automobile" is more similar to "boat" than "tree" since "automobile" and "boat" share the same ancestor "vehicle". From these six available WordNet measures, three (Resnik [26], Lin [20], Jiang and Conrath [13]) are based on the information content of the least common subsumer (LCS) between two concepts. The LCS of two concepts is the most specific concept that is an ancestor of both concepts. The other three similarity measures are based on path lengths between pairs of concepts, namely, Leacock-Chodorow Similarity (*lhc*) [17], Wu-Palmer Similarity (*wup*) [37], and *path*, which is a baseline that is "equal to the inverse of the shortest path between two concepts" [24]. In this work, we make use of the *wup* measure to measure the similarity between words in sentences. It finds the depth of the LCS of the concepts, and then scales that by the sum of the depths of the individual concepts. The depth of a concept is simply its distance to the root node [24]. We use this applied to an algorithm that tries to measure the semantic similarity between sentences.

2.8 Remarks

This chapter presents several concepts in which we based our approach, it is divided into two main subjects. First, we cover concepts about the object of this work, such as contracts, norms, and deontic logic. These concepts allow us to understand which techniques are necessary to achieve our objective. Second, we present every knowledge and technique we use to identify potential conflicts. It includes the concepts of conflict, named entity recognition, and semantic similarity.

In order to describe our approach, we divide it into two chapters. The following chapter presents the detection and classification of norms in contracts. In this phase, we describe how

[2] http://wordnet.princeton.edu

we differentiate a norm sentence from a common sentence in a contract and how we classify such norm sentences according to their deontic meaning. The second chapter describes how we use the classified norms to identify potential conflicts between them.

3. DETECTING AND CLASSIFYING NORMS

Overall, our work is divided into two main phases, the creation of norm representations and the comparison of norms. The former is responsible for extracting enough information to create a norm representation, this includes the norm itself, the parties and the norm modalities. The latter uses the information to compare norms and identify potential conflicts. Figure 3.1 presents the architecture used in this work.

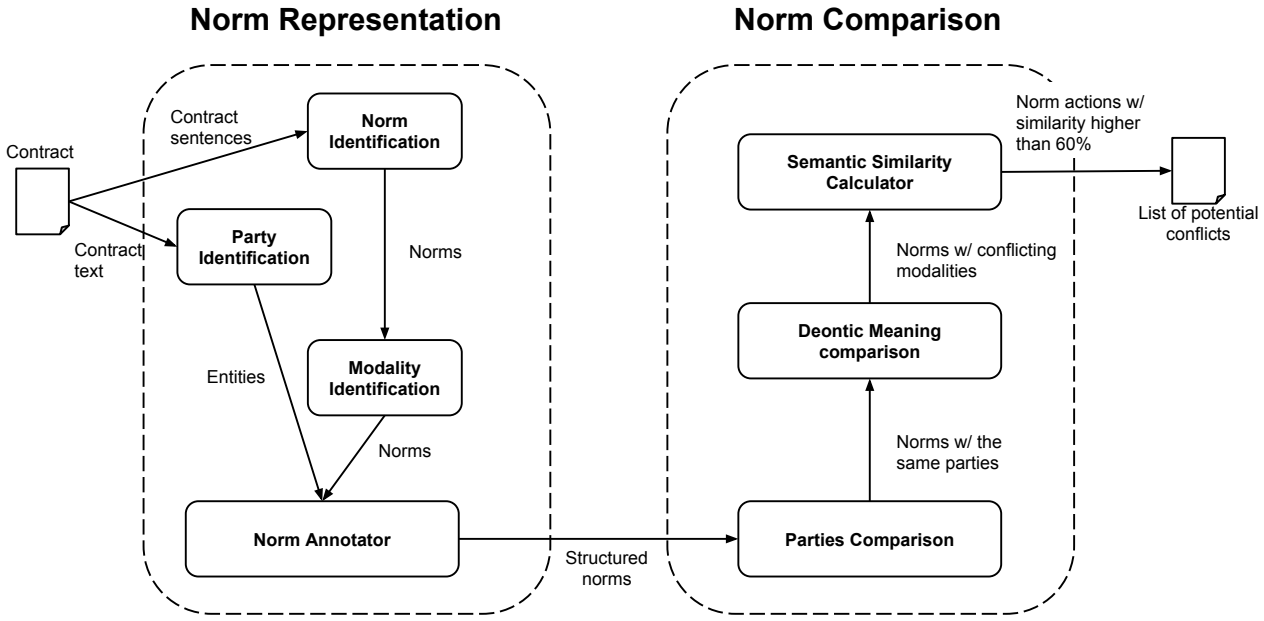

Figure 3.1 – Summary of the process of potential conflict identification

In this chapter, we focus on the first phase, which is the creation of the norm representation. This phase consists of four processes, norm, party, modality, and norm action identification.

3.1 Norm Identification

In order to detect sentences that constitute norms in contracts, we investigated two approaches, namely a Machine-learning-based and a rule-based one. Following we describe them: (1) using a data set with norm and common (non-norm) sentences to train a machine learning algorithm that detects norms and common sentences; (2) based on the studied norm representation, create a rule-based approach to recognize norms. Both approaches were tested over the Australian contract corpus [6] as test data. The corpus consists of 256 unannotated contracts, which are composed of different contract types involving different subject matters (e.g. business contracts, home leases, among others).

For this task we considered contract sentences to be of two exclusive types: norm sentences and non-norm sentences. A norm sentence usually follows a well-defined 4-component structure: an

indexing number or letter, one or more named parties, a modal verb, and a behavior description. Example 3.1.1 illustrates a typical norm sentence.

Example 3.1.1. "**9.** The **Commission must** first attempt to resolve an industrial dispute by conciliation.".

Example 3.1.2. "The Code Participants are parties to a Dispute within the meaning of clause 8.2 of the Code.".

Conversely, common sentences have a different structure, as Example 3.1.2 shows. They often have an identifier and finish with varying punctuation marks other than the period. These sentences seldom have modal verbs and often have a different structure than norm sentences.

3.1.1 Machine Learning approach

As our first approach, we use a machine learning algorithm to classify contract sentences as norm sentences and non-norms. Mitchell [21] defines that machine learning occurs when a computer program "is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". Therefore, to classify sentences as norm and common sentences, we can consider experience E as the act of learning from previously classified (as norm or common sentence) sentences; tasks T as the act of classifying new contract sentences; and the performance P as the measure of hits given tasks T, i.e., how many sentences were correctly classified (namely, accuracy).

For this approach, we manually annotated a set with 92 contracts from the Australian contract corpus. From these contracts, we created two sets: a norm sentence set with 9864 norms; and a common sentence set with 10554 sentences. These sets constitute our training and test data, which we divide 80% for train and 20% for test. For each sentence, we extracted a series of features and created a tuple with the sentence features and its intended class (norm or nonNorm). Since features involving modal verbs are the most relevant, we simply add to the feature extraction other features that can represent the context of a norm. By context, we mean the elements that compose a norm. We selected the features empirically through a context analysis. Thus, the following features are used:

- **First word**: We use this feature to identify the first element of a sentence, since norm sentences often begin with a party name;
- **Second word**: This feature reinforces the idea to extract information about the sentence context, extracting the sentence second word;
- **Last word**: As the previous features, we use the last word to identify the sentence context;
- **has(modal_verb)**: This is a binary feature that indicates whether a sentence has or not a modal verb; and

- **count(modal_verb)**: This feature indicates the number of modal verbs a sentence has, it is useful to identify whether norm sentences have more than one modal verb.

In the end, we obtain a tuple with the sentence features in the first position and the sentence class in the second position. Listing 3.1 illustrates a tuple applied to the following sentence: "Roxio may not cancel any Products for which a Service Order has been received by Adaptec." We use these tuples to train our sentence classifier.

Listing 3.1 – Tuple training example

```
(
    {
        First word: Roxio,
        Second word: may,
        Last word: Adaptec,
        has('may'): True,
        count('may'): 1
    },
    norm
)
```

3.1.2 Rule-based approach

As modal verbs are the central element that differentiates a norm sentence from a common sentence, we defined a list of modal verbs to consider in a regular expression. For this work, we considered six modal verbs, namely: may, can, must, ought, will, and shall. We use a rule-based approach that considers such modal verbs to decide whether a contract sentence is a norm sentence. The following regular expression was defined to identify norm sentences:

Listing 3.2 – Regular expression to identify a norm sentence

```
Rule = .+? modal_verb .+
where modal_verbs = (may|can|must|ought|will|shall)
```

3.2 Norm Representation

In order to identify potential conflicts between norms, we defined a norm representation based on the representations discussed in Section 2.2. In this representation, we divide a norm into three main components, namely deontic_meaning, party_name, and norm_action. Example 3.2.1 presents a norm representation applied to a real norm and Table 3.1 illustrates norm representation of real norms.

- **deontic_meaning**: Norm conflicts arise from differences between deontic meanings, such as obligation and prohibition, applied to the same context. Thus, deontic meaning is a relevant

aspect to consider in norm representations. This component may contain one of three possible values, namely permission, prohibition, and obligation, depending on the modal verb in the norm.

- **party_name**: Conflicting norms are usually applied to the same party. Therefore, a norm representation must identify which party must comply with the norm.
- **norm_action**: Two norms with different deontic meanings present a conflict when they apply the same action to the same party.

Example 3.2.1. :
At the end of the term of this Agreement, <party_name> 7UP/RC </party_name> <deontic_meaning> may <deontic_meaning> <norm_action> use the Equipment to produce Products as well as other energy drinks. </norm_action>

Table 3.1 – Representing real norms

Norm	**Norm Representation**
Purchaser shall also be financially responsible for all taxes and freight in connection with the New Equipment.	(Obligation, Purchaser, "be financially responsible for all taxes and freight in connection with the New Equipment.")
Roxio, upon request, may review such agreements at any time before or after execution.	(Permission, Roxio, "review such agreements at any time before or after execution.")
Roxio may not cancel any Products for which a Service Order has been received by Adaptec.	(Prohibition, Roxio, "cancel any Products for which a Service Order has been received by Adaptec.")

Given the norms from a contract, we need to identify each element of the norm based on the norm representation defined in Section 2.2. This task involves the identification (within the norm) of the party to which the norm is applied; the norm modality according to the central modal verb; and the action the party must comply with. This information allows us to compare a norm pair in order to identify a potential conflict. In Sections 3.3 and 3.4 these tasks are explained.

3.3 Party Identification

The first step towards the construction of a norm representation is the entity extraction, for the purpose of party identification. This task is divided into three subtasks: the identification of the contract parties, which are often defined as either person or company names; the identification of the aliases (abbreviations, nicknames), which are name simplifications often used to refer to a certain party within the contract; and the definition of a relation between the party names and their aliases, if they exist. As data for such task, we use the contract corpus provided by Gao *et al.* [10] in their work. The corpus has 2093 real contracts we use for each processing step of this approach,

including conflict identification. We use this corpus instead of the Australian contract corpus [6], since the latter consists of many contract templates that seldom have the description of the parties.

To extract parties from contracts we need to understand the contract structure. Since contracts are semi-structured documents, one may find the parties' definition, most of the time, in the beginning of a contract. Among the analyzed contracts, we identify several ways in which parties are described, however, one particular pattern is commonly used. Consequently, we base our entity extraction on this pattern, as Example 3.3.1 illustrates [1].

Example 3.3.1. THIS AGREEMENT is made by and **between** *Lucent Technologies Inc.*, a Delaware corporation, acting through its Microelectronics Group, having an office at Two Oak Way, Berkeley Heights, New Jersey 07922 ("Lucent") **and** *CD Radio Inc.*, a Delaware corporation, having its principal place of business at 2175 K Street NW, Washington, DC, 20037 ("CD Radio").

As Example 3.3.1 shows, entities are presented using a simple pattern, which starts with the word "between" and ends with the end of the sentence. The sequence of words after "between" defines the first entity, it extends until the word "and", which separates the first entity from the second one. We consider only contracts with exactly two parties, i.e., contracts that describe the exchange of goods or services between either two persons or two companies. Using this pattern, we create a regular expression that extracts this block of text from contracts. From this block of text, we extract the entities. First, we use a list [2] with 2526 company names in order to find entities in the block of text. If the entity is not in the list, we use the NLTK part-of-speech tagger to annotate the text, and then we extract the sequence of elements tagged as proper nouns. In Example 3.3.1, the names in italic represent the party names.

However, in many cases entities are represented by aliases within the contract. This information is often present in the same opening block of text that defines the parties in the contract. Within the description of each party, surrounded by parenthesis and, in some cases, double quotes, the name that will be used in the contract is defined. In this case, we apply a simple regular expression to extract the nicknames and abbreviations, when they exist. Using Example 3.3.1, we extract:

- Entities: {Lucent Technologies Inc., CD Radio Inc.}
- Abbreviations: {Lucent, CD Radio}

To ensure that the alias indicates the party full name, we apply the regular expression in the stretch that defines the party. For example, from "between" to "end", which describes the first party, if the regular expression matches an element, we address it as the alias to the first party.

Once we have the contract parties, we can identify them in a norm, as in Example 3.3.2.

Example 3.3.2. :

[1] Extracted from http://goo.gl/vCHiUI
[2] Extracted from http://goo.gl/YC5Utv

- <Party1>Lucent</Party1> shall acknowledge all orders in writing.

- <Party2>CD Radio</Party2> shall test, review, and approve the work in Phase 1.

3.4 Modality Identification

The identification of modalities is an important step in the process of detecting potential conflicts. Based on the modal verb in a norm, we divide the sentence into subject and action since the party (subject) presented before the modal verb is intended to comply with the action specified after the modal verb, as Example 3.2.1 in Section 3.3 presents. Besides, identifying the deontic meaning of a modal verb in a norm allows us to compare norms and detect deontic conflicts between two norms.

In order to identify modal verbs, we define a list with six verbs, namely: can, may, must, ought, shall, and will. For each norm we search whether it has one of the listed items. When identified, we check whether there is a negation after the verb, such as "may not". In this case, we compare what we found with dictionary entries indicating the deontic modality based on the modal verb with or without the negation. The dictionary follows the rules we defined in Section 2.4, Table 3.2 illustrates it. At the end of the process we obtain an annotated norm according to its deontic meaning, as the Example 3.4.1 presents.

Example 3.4.1. <Party1>Lucent<Party1> <Obligation>shall</Obligation> acknowledge all orders in writing.

Table 3.2 – Modal verb dictionary

Word	Deontic Modality
can	permission
may	permission
shall	obligation
must'	obligation
will	obligation
ought	obligation
shall not	prohibition
cannot	prohibition
may not	prohibition
will not	prohibition
must not	prohibition
ought not	prohibition

3.5 Implementation and Evaluation

In this section, we describe how we implemented and evaluated our norm identification approaches. First, we present the results obtained in the norm identification. Second, we detail the party identification results. Finally, we present the results obtained in the modality identification.

3.5.1 Norm Identification

In our first approach to create a norm detector, we use a machine learning algorithm to classify a contract sentence as norm or common sentence. In order to process the texts in the Australian contract corpus we used the Natural Language Toolkit[3] (NLTK). NLTK is a toolkit developed in the Python language that contains many methods for text processing, such as text manipulation, tokenization and annotation. It provides a part-of-speech tagger and a series of corpora and classifiers, which facilitates text classification.

First, to evaluate how the features impact on the final result we tested different features sets. We defined such features on Section 3.1. As we assume that features involving modal verbs are more relevant to the task, we began testing them, and after we added the other features. Comparing the features, in Table 3.3 we present the results obtained from the mean of ten executions of the classifier with each set of features.

Table 3.3 – Features and respective F-measure

Features	**F-Measure**
{*has(modal_verb)*, *count(modal_verb)*}	0.80
{*has(modal_verb)*, *count(modal_verb)*, *(firstword)*}	0.85
{*has(modal_verb)*, *count(modal_verb)*, *(firstword)*, *(secondword)*}	0.85
{*has(modal_verb)*, *count(modal_verb)*, *(firstword)*, *(secondword)*, *(lastword)*}	0.87

In order to test the learning algorithm, we shuffle the sentences in the data sets and divide them with 80% of the sentences comprising the training set and the remaining 20% for the test set. We compare a series of classifiers that allow us to choose the best one for the task of classifying contract sentences. Using the annotated contracts from the Australian contract corpus [6], we compare three classifiers based on their F-score for the task. The classifiers are Naïve Bayes, Decision Tree and Maximum Entropy. This comparison is shown in Table 3.4, which presents an arithmetic mean for ten executions of each classifier. As Table 3.4 shows, Naïve Bayes and Maximum Entropy obtain the best results. However, we chose to use Naïve Bayes as our classifier, since it has a superior F-score.

To test our rule-based approach to detect norms, we used contracts from the Australian contract corpus [6]. We performed a test with the same test set used in the machine learning

[3] http://www.nltk.org/

Table 3.4 – Comparison between classifiers

Classifier	Accuracy	Precision	Recall	F-Measure
Naïve Bayes	**0.86**	0.80	**0.95**	**0.87**
Decision Tree	0.70	0.65	0.77	0.70
Maximum Entropy	0.84	**0.81**	0.86	0.83

approach. As result, we obtained 79% of precision, 98% of recall, and 87% of F-measure in this task.

3.5.2 Comparing Results for Norm Identifier

To create a norm identifier, we tested two different approaches (a machine learning based and a rule-based one) and obtained similar results. Although machine learning is a better choice to generalize the norm recognition, when applied to contracts, a simple regular expression seems to be enough, given the corpus we used in this work. Therefore, we decided to use the rule-based approach for the task of recognizing norms. We obtain a faster processing, since using the machine learning approach we spend more time in extracting features and then classify the sentences.

3.5.3 Party Identification

In order to evaluate our party identifier, we executed the party identifier in 100 contracts from the manufacturing type [4]. As result, we obtained an accuracy of 60% considering the whole task of identifying party names, aliases, and their relation. The errors arise from differences with the patterns we use, such as: contracts with three parties (4 errors), exemplified in Example 3.5.1; party descriptions that do not use “between”/“and” pattern (10 errors), exemplified in Example 3.5.2; aliases pattern does not match (17 errors); and other errors related to the link between party name and alias.

Example 3.5.1. :

Entered into this 6th day of August 1997 (hereinafter "Effective Date") by and between **JPI PHARMACEUTICA INTERNATIONAL**, a division of Cilag AG International Zug, a company duly organized and existing under the laws of Switzerland, having its principal office in CH-6300 Zug, Kollerstrasse 38, Switzerland (hereinafter referred to as "JPI")

and

JANSSEN PHARMACEUTICA Inc., 1125 Trenton-Harbourton Road, Titusville, NJ 08560, USA (hereinafter referred to as "JANSSEN US") (JPI and JANSSEN US collectively referred to herein as "JANSSEN")

[4] Extracted from http://contracts.onecle.com/type/47.shtml

and

Alkermes Controlled Therapeutics Inc. II, a company organized and existing under the laws of the Commonwealth of Pennsylvania, having its principal office at 64 Sidney Street, Cambridge, MA 02139-4136, U.S.A. (hereinafter referred to as "ACT II").

Example 3.5.2. :
BETWEEN

(1) Minnesota Mining and Manufacture Company ("3M") and 3M Innovative Properties Company ("3M IPC"), both having a principal office at 3M Center, Building 275-3E-10, St. Paul, MN 55144-1000, USA).

(2) Sepracor Inc. ("SEPRACOR"), having a principal office at 111 Locke Drive, Marlborough, MA 01752.

3.5.4 Modality Identification

To evaluate the modality identification, we ran it over 150 norms in 6 real contracts of manufacturing type. As result, we obtained an accuracy of 97% in the identification. The errors are related to sentences that have more than one modal verb and thus mistake the identifier. Example 3.5.3 illustrates a norm that has two modal verbs, which in fact describes two norms in one sentence. The final norm representation element, namely act, is easily identified by getting the stretch of text after the modality identification. Thus, we simply detect every action specification as the act itself, as the Example 3.5.4 shows.

Example 3.5.3. Except as otherwise provided in this Agreement, the AAI Agreements **shall** remain unamended and **shall** continue in full force and effect.

Example 3.5.4. <Party1>Lucent<Party1> <Obligation>shall</Obligation> <Act>acknowledge all orders in writing</Act>.

3.5.5 Evaluation Conclusions

The results show that we are able to extract contract information even so we do not obtain high values in all evaluations. In the norm identification, we obtain good results that can guarantee the detection of a good number of norms from contracts. In the party identification, the results are lower, which can impact in the final result. However, in the modality identification we obtained a high accuracy, which indicates that we can identify most of the deontic modalities in norms. The errors we obtain in each step will reflect in the second phase since it uses the extracted information to compare norms and identify potential conflicts.

3.6 Remarks

In this chapter, we present the first phase of our approach. To do so, we divide the phase in three identification steps. First, we present two approaches to detect norms, which is the main element of this work. Both approaches obtain an F-measure of 87%. Second, we present an approach to detect parties based on a rule-based approach. This approach obtains an accuracy of 60%. Finally, we present an approach to identify the modality of norms in which we check the modal verb within a norm in a dictionary that indicates the modality based on the modal verb. For such approach, we obtain an accuracy of 97%.

In the following chapter, we use the norm representation to compare pair of norms and identify potential conflicts between them. We perform a comparison between the norm components. To identify whether two norm actions are similar, we develop a semantic similarity algorithm. The algorithm compares two norm actions and return a value that we use to define whether the norm actions are equal or similar.

4. DETECTING AND CLASSIFYING POTENTIAL CONFLICTS

Given the norm representation pattern we proposed in Chapter 3, in this chapter we present the second phase of our approach. Here we want to use such representation to compare different norms and detect potential conflicts between them. From these conflicts, we want to classify them according to the deontic meaning they present.

To detect a potential conflict, we compare two norms from the same contract and verify whether the conditions specified in Section 2.5 are satisfied. Such conditions, applied to a pair of norms, are:

- Both norms are applied to **the same party**;
- Both norms have **conflicting deontic meanings**; and
- Both norms refer to **the same act**, i.e., the same consequent in which the party must fulfill.

In order to classify a potential conflict, we perform a comparison between the deontic meanings in norm pairs, thus, based on the deontic conflict it arises, we indicate a conflict type. In this work, we deal with three conflict types, which we defined based on the conflict types proposed by Sadat-Akhavi [28]. Such types are:

(1) Permission x Prohibition

(2) Permission x Obligation

(3) Obligation x Prohibition

Following the relation between modal verb and deontic meaning, which we describe in Section 2.4, we are able to distinguish between the three types. For example, a norm pair can be classified as a potential conflict of type (2) if one norm has "can" as modal verb and the other has "must" as modal verb.

The following sections describe how we achieve our objectives. Section 4.1 describes the algorithm we use to measure the semantic similarity between norm actions. Finally, Section 4.2 describes the evaluations we performed to obtain our final results.

4.1 Computing Semantic Similarity

Two norms constitute a potential conflict when they have different deontic modalities but the same action, as Example 4.1.1 shows. However, as usual in natural language, norm actions can be written differently, and yet have the same meaning. This brings the need for a method that measures the semantic similarity between two norm actions. Example 4.1.2 illustrates two norms

that have norm actions with the same meaning but written in different ways. Thus, given two norms applied to the same party, we extract the norm actions and calculate the semantic similarity between them. Based on this measure, we decide whether a norm pair must be considered a potential conflict.

Example 4.1.1. :

- Purchaser must <act>pay the product taxes</act>.
- Purchaser shall not <act>pay the product taxes</act>.

Example 4.1.2. :

- Purchaser must <act>pay the product taxes</act>.
- Purchaser shall not <act>pay the taxes applied to the product</act>.

To measure the semantic similarity between two norm actions, we use Algorithm 4.1. This algorithm takes two parameters σ_1 and σ_2, the word-vector corresponding to the norm actions of two sentences, and assumes that the number of words in σ_1 is smaller than σ_2. The comparison of these sentences consists of iterating over the indexes of each word in both word-vectors (Line 2). If both vectors have the exact same word in the same position we add 1 to the final score (Line 3). Otherwise, we compare the word in the first vector to every other word in σ_2 (Line 6). If the same word is found in a different position, we add 0.7 to the final similarity score (Line 8) representing a penalty for the different positions of the same word. If the same word is not found in a different position, the algorithm tries to compare the similarity between the synonyms of both words being compared (Lines 11 and 12) keeping the highest semantic similarity between them. We calculate the semantic similarity for individual words using the WUP [37] measure provided by WordNet, which generates a score that represents how semantic similar two word senses are based on the depth of the two senses in the taxonomy and their most specific ancestor node (the SIMILARITY function in Line 13). After iterating over all words, we add the highest value to the final score (Line 15). Finally, we normalize the similarity score by the mean length of both sentences (Line 16).

In summary, the algorithm has three main cases: first, add 1 to the final score when in both norm actions, in the same position, there are the same words (see Figure 4.1); second, add 0.7 to the final score when there are same words in both norm actions but in different positions (see Figure 4.2); third, find the highest value of semantic similarity between a word in the first norm action and the words of the second one (see Figure 4.3). Examples 4.1.3 and 4.1.4 illustrate the application of the semantic similarity algorithm over norm actions. In both cases, the resulting similarity is 60%.

Example 4.1.3. :

Semantic Similarity: 0.60

- Cubist shall <act>at all times upon reasonable notice and during normal business hours have the right to inspect the Facility to ascertain compliance with GMPs.</act>

Algorithm 4.1 – Calculates the semantic similarity between two norm actions.

Require: $|\sigma_1| \leq |\sigma_2|$

procedure COMPUTE_SIMILARITY(σ_1, σ_2)
  for ind_1 in σ_1 **do** ▷ Iterate over the indexes of σ_1
    if $\sigma_1[ind_1]$=$\sigma_2[ind_1]$ **then** $sim \leftarrow sim + 1$
    else
      $sim_{max} \leftarrow -\infty$
      for ind_2 in σ_2 **do**
        if $\sigma_1[ind_1]$=$\sigma_2[ind_2]$ **then**
          $sim_{max} \leftarrow 0.7$
          break
        else
          $s_1 \leftarrow$ synonyms of σ_1
          $s_2 \leftarrow$ synonyms of σ_2
          $sim_{1,2} \leftarrow \max_{s_1,s_2}(\text{SIMILARITY}(s_1, s_2))$
          $sim_{max} \leftarrow \max(sim_{max}, sim_{1,2})$
      $sim \leftarrow sim + \max(0, sim_{max})$
  return $sim/\text{MEAN}(len(\sigma_1), len(\sigma_2))$

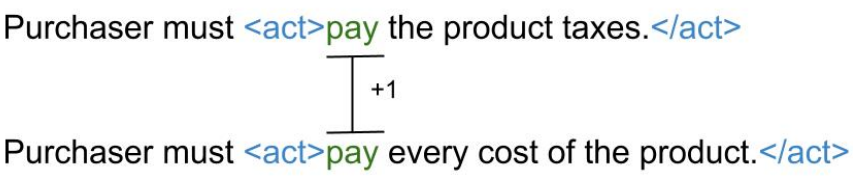

Figure 4.1 – Case 1: Same words in the same position

- Cubist may <act>be, at all times and upon reasonable notice and normal business hours, denied an inspection of the Facility to ascertain compliance with GMPs.</act>

Example 4.1.4. :

Semantic Similarity: 0.60

- AorTech shall <act>use only approved suppliers listed on AorTech's approved supplier list when purchasing such material.</act>

- AorTech may <act>use unapproved suppliers listed on AorTech's unapproved supplier list when purchasing such material.</act>

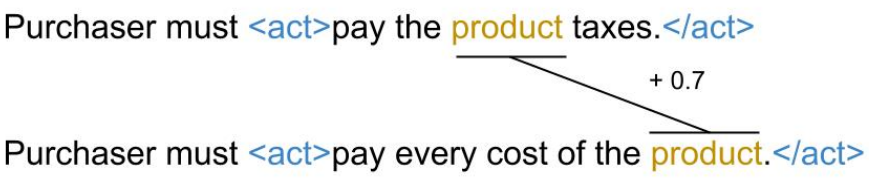

Figure 4.2 – Case 2: Same words in different positions

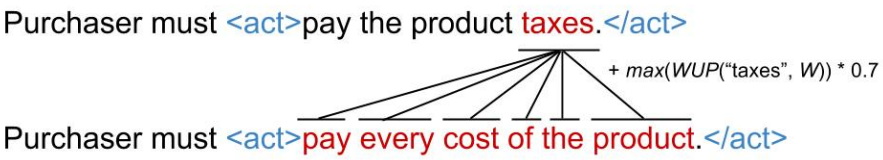

Figure 4.3 – Case 3: Different words and different positions

Once calculated, the semantic similarity between acts allows us to detect potential conflicts between norms with the same parties, conflicting deontic meanings, and actions with, at least, 60% of semantic similarity. Figure 4.4 illustrates the detection of a potential conflict. It consists of the comparison between each component of the norm representation, namely the parties, the deontic meanings, and the norm actions. As we describe in this section, to calculate the semantic similarity, we only consider the norm action, i.e., the portion of text right after the modal verb. We do not consider elements before the modal verb, such as conditions and further details about the norm. Such information often describes the conditions in which a norm is activated. Sentence 4.1.5 illustrates a conditional norm. Although we believe this is a relevant information to be considered, identifying such conditional terms demands time and a whole study about the context. Therefore, for this work we decided not to consider such information, since most norms do not present conditional terms.

Example 4.1.5. In the event that Y, Company X must pay the taxes.

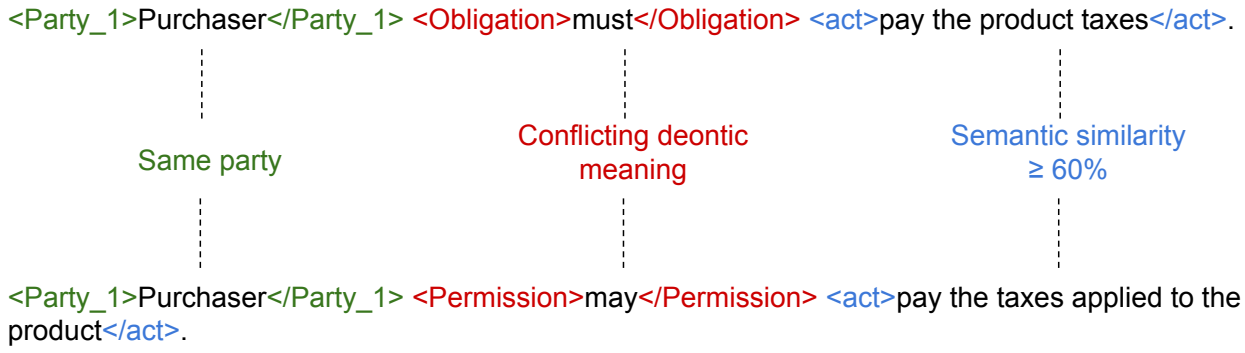

Figure 4.4 – Demonstration of a potential conflict detection

4.2 Implementation and Evaluation

4.2.1 Semantic Similarity Algorithm

To evaluate our semantic similarity algorithm, we compare our results to the ones obtained by Li *et al.* [19] in their work. They propose an algorithm to compute the similarity between sentences, which considers the semantic information and word order information implied in the sentences. In their work, they present a table with 16 sentence pairs that are used to show the results obtained by

the algorithm. We use this sentence pairs to compare our results with the results of their algorithm. Besides, in order to evaluate how much the WUP measure improves our algorithm, we compare our results with an approach that uses the same algorithm but, instead using WUP measure, it only compares if the elements are equal without considering the semantic similarity between them. The results are presented in Table 4.1.

As the results show, when applied to common sentences some discrepancies arise among the algorithms. Our approach presents good results in some sentence pairs, such as ("I like that bachelor.", "I like that unmaried man."), in which the result is superior to the other algorithms. However, when calculating the similarity between ("Red alcoholic drink.", "An English dictionary.") it results a much higher similarity than the others, showing that our algorithm also considers some unnecessary aspects. In comparison to Simple_Similarity, which does not use WUP similarity measure, we obtained significant results when applied to sentence pairs that have similar meanings but different phrasal construction. As an example, the sentence pair ("Red alcoholic drink.", "A bottle of wine.") in which Similarity_WUP obtains 0.31 of similarity while Simple_Similarity obtains only 0.14. Since Simple_Similarity does not take into consideration the synonyms of a word, sentence pairs like the previous example receive lower similarity. In the sentence pair ("A glass of cider.", "A full cup of apple juice."), Similarity_WUP obtains 0.5 while Simple_Similarity obtains 0.34, which shows that Simple_Similarity limits its computation to the equality of the words.

Although our approach presents some deficiencies, it is much simpler to implement than the one presented by Li *et al.* and slightly better than a simple comparison between words. For these reasons, we use the Algorithm 4.1 in our approach to compute the semantic similarity between sentences.

4.2.2 Potential Conflict Identifier Evaluation

The evaluation of a system that identifies potential conflicts must be performed over a corpus containing real conflicts. However, as long as we searched for a corpus with real normative conflicts, we did not find a corpus with such characteristics. Thus, we decided to manually create norm conflicts using a set of real norms as a basis. For such task, we created a system that assists a user to insert conflicts given a set of norms. Using the created conflicts in contracts, we execute our potential conflict identifier over the altered contracts. This section presents the way how we implement the system used to create conflicts and how we use the resulting conflicts to evaluate our potential conflict identifier.

Conflict Insertion

To insert conflicts into real contracts, we created a system that has as input a set of contracts and every contract is associated with a set of norms. The set of contracts consists of 146 contracts of manufacturing type, which make part of the corpus created by Gao *et al.* [10]. For each

Table 4.1 – Similarity comparison between three algorithms: **Similarity_WUP**, which is our approach that uses the WUP similarity measure; **Simple_Similarity**, which is the approach that does not use the WUP similarity measure; and **Li *et al.***, which is the approach proposed by Li *et al.* [19]

Sentence Pair	**Similarity_WUP**	**Simple_Similarity**	**Li *et al.***
("I have a pen.", "Where do you live?")	0.08	0	0
("Red alcoholic drink.", "An English dictionary.")	0.43	0.25	0
("I have a pen.", "Where is ink.")	0.32	0.14	0.12
("I have a hammer.", "Take some apples.")	0.37	0.24	0.12
("Canis familiaris are animals.", "Dogs are common pets.")	0.34	0.34	0.36
("Red alcoholic drink.", "Fresh apple juice.")	0.49	0.25	0.42
("I have a hammer.", "Take some nails.")	0.34	0.24	0.5
("I like that bachelor.", "I like that unmaried man.")	0.72	0.62	0.56
("Red alcoholic drink.", "A bottle of wine.")	0.31	0.14	0.58
("Red alcoholic drink.", "Fresh orange juice.")	0.45	0.25	0.61
("It is a dog.", "It is a log.")	0.85	0.8	0.62
("A glass of cider.", "A full cup of apple juice.")	0.5	0.34	0.67
("It is a dog.", "That must be your dog.")	0.38	0.32	0.73
("Dogs are animals.", "They are common pets.")	0.43	0.34	0.73
("It is a dog.", "It is a pig.")	0.85	0.8	0.79
("John is very nice.", "Is John very nice?")	0.68	0.68	0.97

contract, we extract its norms using the norm identifier (described in Section 3.1), and from the set of norms, we select norms applied to one of the contract parties. Then, the user may use the system to create conflicts by duplicating an existing norm and altering it in order to conflict with the existing one.

We selected two volunteers who had no prior contact with or knowledge about the algorithms for conflict identification.. For the insertion, we ask each volunteer to create different types of conflict. We divided the conflict insertion into two types, namely modal alteration and modal and structure alteration. A modal alteration consists of, given the duplicate of an existing norm, altering the modal verb in order to change the deontic meaning and create a conflict based on the defined types ((1), (2), and (3)). A modal and structure alteration consists of altering both the modal verb and the norm action structure from a duplicate of an existing norm. In structure alteration, the user must modify the words and their order without missing the norm action meaning, as Example 4.1.2 illustrates. We use the modal alteration type to test the potential conflict identifier accuracy on the task of identifying conflicts between norms that have the same party, same norm actions, but

different deontic modalities. Therefore, the user simply modifies the modal verb without changing the norm action structure, creating a conflict illustrated by Example 4.1.1. On the other hand, we use the modal and structure type to test the potential conflict identifier on the task of identifying conflicts between norms that have the same party, different norm actions (but with the same meaning), and different deontic modalities. Thus, the user is intended to modify the structural organization of the norm.

The process of conflict creation is conducted by the user that selects an option for each process. First, the user selects a random contract, which the system selects randomly from the contract corpus. Following, the user may choose either to get another contract or get a random norm from the contract. Getting a random norm, the user must copy such norm and make the necessary modifications. From this point, the user may choose either to start all over again, get another norm from the same contract, or finish the conflict insertion. At any point, the user can come back to a previous step in the conflict insertion. Figure 4.2.2 illustrates the process in which the user is submitted to create a new conflict.

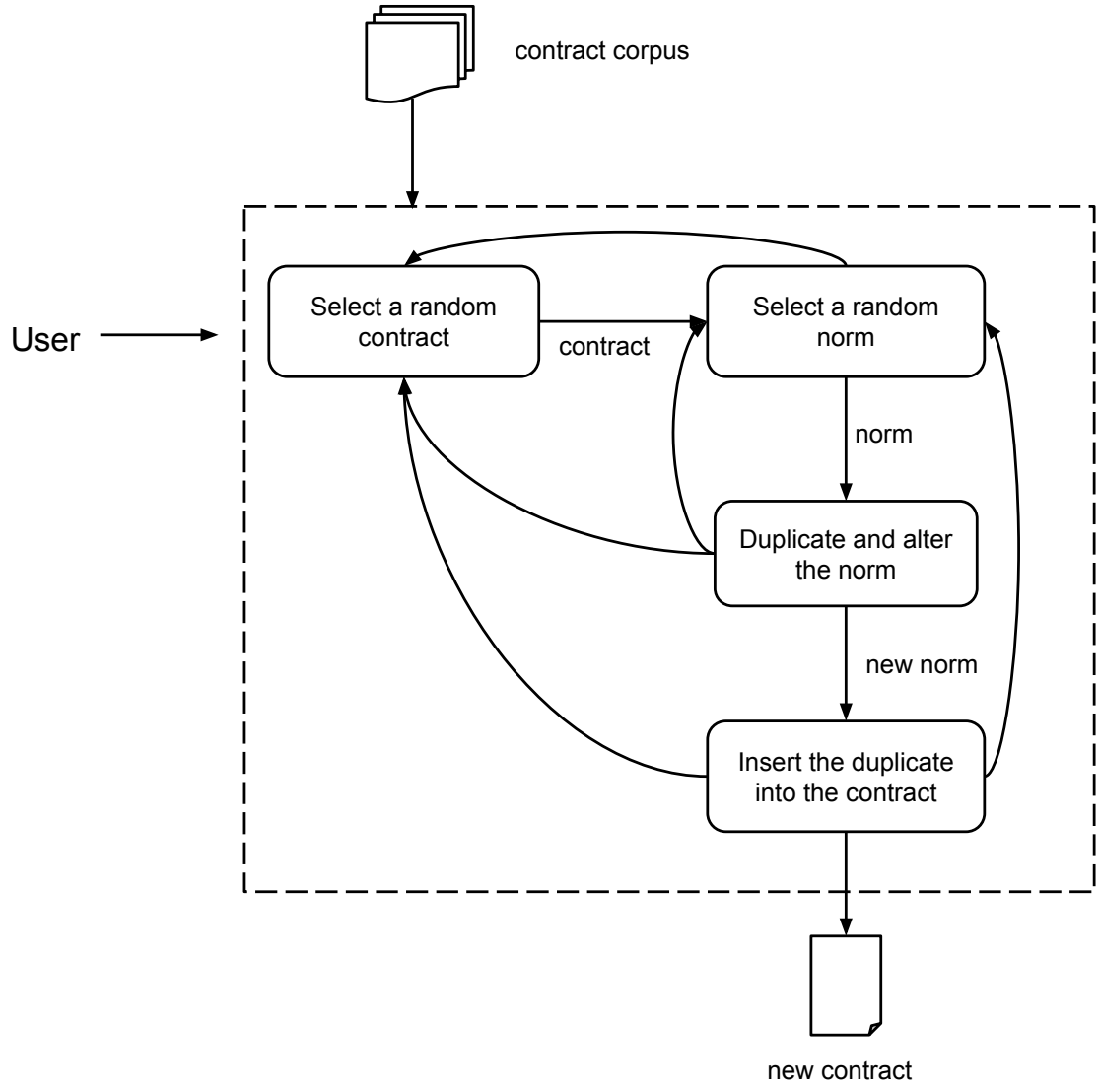

Figure 4.5 – Scheme of the conflict insertion system

To the first volunteer, we asked to create conflicts of the modal alteration type. The volunteer created 94 conflicts in 10 different contracts, totaling 13 conflicts of type (1), 36 conflicts of type (2), and 46 conflicts of type (3). To the second volunteer, we ask to create conflicts of the modal and structure type. Since it demands more time and English language knowledge, the second volunteer created only 17 conflicts in 6 different contracts, totaling 2 conflicts of type (1), 6 conflicts of type (2), and 4 conflicts of type (3). From these conflicts, we execute the potential conflict identifier over the 16 contracts.

Evaluation

We consider the evaluation according to each type of conflict insertion, then we make a final round considering all conflicts. Table 4.2 shows the results we obtained with the first type of conflict. As the results show, we obtain a relevant accuracy of 79% for the task. This accuracy, as well as the others, is a pounded meaning that sums the product of each contract accuracy by the total number of conflicts in the contract, and then divide it by the total number of conflicts. Given $n\_contracts$ as the total number of contracts, $conflicts_{(i)}$ as the total number of conflicts in contract i, $accuracy_{(i)}$ as the accuracy obtained in contract i, and $n\_conflicts$ as the total number of conflicts overall contracts, Equation 4.1 shows how we calculate the pounded meaning. Among the results, we identified 11 conflicts of type (1), 32 of type (2), and 31 of type (3), in addition to two extra potential conflicts that were not in the conflict set. We can explain the errors by considering the error rate in each subprocess that constitute the norm structure. Consequently, either an error in the norm identifier or in the party identifier may reflect on the final result. Since this set of conflicts is made of simple modifications in the modal verb, we expected good results. For this reason, we tested our potential conflict identifier over the second set of conflicts, which is made of modifications in modal verb and in the norm action structure. Table 4.3 presents the results we obtained. We obtain a lower accuracy (70%) if compared with the previous set of conflicts, since in this one we have differences in the norm actions. Although the set has a small number of conflicts, the results show that in some contracts we identified all inserted conflicts. This confirms that the semantic similarity works to identify norm actions with the same meaning. Finally, Table 4.4 presents the accuracy when we consider all conflicts without distinction. Therefore, we obtain an overall accuracy of 78%, which we consider an important result given the whole necessary process to obtain such potential conflict identifier.

$$accuracy = \frac{\sum_{i=1}^{n\_contracts} conflicts_{(i)} * accuracy_{(i)}}{n\_conflicts} \tag{4.1}$$

4.3 Remarks

This chapter presents the second phase of our approach. In such phase, we use the norm representation proposed in the first phase to compare norm elements and identify potential

Table 4.2 – Results of the potential conflict identifier execution over the first set of conflicts.

	Contract	Conflicts	Identified	Accuracy
	roth-mfg-2006-11-26	14	11	0.78
	chiron.mfg.2003.10.02	8	8	1.0
	ibm.mfg.1999.04.15	7	5	0.71
	johnson.mfg.2000.06.01	7	5 + 1 extra	0.71
	southeast.mfg.2004.24	6	3 + 1 extra	0.5
	nellson.supply.2001.09.13	11	9	0.81
	medica.mfg.2003.05.12	17	14	0.82
	usfoodservice.distrib.2003.01.27	12	9	0.75
	foamtec.mfg.1998.01.30	2	1	0.5
	cardinal.mfg.2004.02.13	10	9	0.9
Total	10	94	74	**0.79**

Table 4.3 – Results of the potential conflict identifier execution over the second set of conflicts.

	Contract	Conflicts	Identified	Accuracy
	hershey.mfg.1998.03.13	4	1	0.25
	astrazeneca-manufacturing-2006-06-28	4	1	0.25
	aortech.mfg.1998.12.23	2	2	1.0
	cypress-mfg-2005-10-06	3	3	1.0
	cinram.mfg.1999.11.04	1	1	1.0
	dsm-capua.mfg.2000.06.22	4	4	1.0
Total	6	17	12	**0.70**

Table 4.4 – Overall results of the potential conflict identifier.

	Contract	Conflicts	Identified	Accuracy
Total	16	111	86	**0.78**

conflicts. To compare norm actions, we propose an algorithm that calculates the distance between two sentences. Based on the distance measure, we consider whether two norm actions are similar or equal. Thus, if a pair of norms has the same party, similar or equal norm actions, and conflicting deontic meanings, we identify it as a potential conflict. To evaluate our approach, we present a method of conflict insertion in which volunteers insert into contracts norms that conflict with the existing ones. As result, we obtain an accuracy of 78% overall created conflicts.

The following chapter presents a series of works that have similar approaches to ours. We divide these works into three categories, namely contract annotation, information extraction from contracts, and normative conflicts. For each category, we compare the works with ours, describing their approach and the main differences.

5. RELATED WORK

In this Chapter, we present works that have similar approaches to ours. We divide them into three categories according to their work proposals. First, we talk about contract annotation, which is the task of identifying contractual elements. Second, we present works about information extraction from contracts, which is the task of identify relevant information in a contract. Finally, we present works related to the task of identifying norm conflicts in contracts, which consists of different approaches applied to multi-agent systems and contracts written in natural language.

5.1 Contract Annotation

Since our work deals with the identification of the contract structures, in this section we present two approaches that have this end. First, we present the Curtotti and McCreath's work, then we present the Athan *et al.* work.

Curtotti and McCreath [5] present an approach for annotating contracts using machine learning and rule-based techniques. The aim of the work is to classify the components of sentences in contracts according to their structure. To classify the elements, 32 classes were defined based on the most significant contract structures. Classes such as CLAUSEMATTER, CLAUSEHEAD, PARTIEHEAD and HEAD are used to annotate a sentence. To extract data for machine learning, a hand-coded tagger was created and its outcome was manually corrected. As data, they use the Australian Contract Corpus [6] with 256 contracts, containing 42910 sentences and a vocabulary of 14217 words. 30 contracts are randomly select and then divided into three sets, one for train and other two for test. Using different classifiers to compare the results, they obtain 0.86 of F-score in CLAUSEMATTER class that has more relevance. Example 5.1.1 shows two contract sentences annotated by the Curtotti and McCreath classifier. The classification indicates that the sentence is a numbered one and is a clause matter.

Example 5.1.1. :

- (233, '#NUMBEREDLINE# (b) #CLAUSEMATTER# #ENDSEMI#', '(b) owned, controlled or provided by a Party or a Collaborator;')
- (277, '#NUMBEREDLINE# (d) #CLAUSEMATTER#', '(d) information which is rightfully known by the recipient Party (as shown by its written record) prior to the date of disclosure to it hereunder; or')

Athan *et al.* [1] present an approach for annotating legal elements in contracts. The aim of the work is to improve the modelling of semantic norms, providing an expressive XML standard for modelling normative rules that satisfies legal domain requirements. They define a series of tags to represent normative elements, such as the agents and authorities of a norm, also the time and

event that occur in a norm. Besides, they propose a new way to tag norm elements, allowing the creation of structured contracts. These tags carry relevant information for contracts representation, what improves their machine-readability. Such a representation is crucial for many tasks involving contract information extraction.

Both efforts introduce new ways to annotate contracts in order to structure data for text processing. In our work, we want to detect norm sentences among the contract sentences and identify the parties enrolled to the contract. The use of these works associated with ours could result in a more accurate information extraction. One can use it to identify other information than norms and parties.

5.2 Information Extraction from Contracts

In this section, we discuss three research ideas that are relevant in that they present different types of information extraction from contracts, selecting important elements for contract analysis. We use these works as basis to formulate our approach and compare our method with theirs.

In the first one, Gao and Singh [10] present an approach for extracting exceptions within norms in contracts, which arise when a norm defines a condition for not be applied. The system, named Enlil, uses natural language processing and machine learning to process contracts and to discover service exceptions or contingency conditions within contracts. The process starts with contracts being segmented into sentences, then based on pre-defined patterns they extract a set of sentences that refer to exceptions. These patterns are formed by a sequence of words, such as "in (the) event of", "in (the) event that" and "in (the) case that". Example 5.2.1 (extracted from Gao and Singh's work [10]) illustrates the occurrence of the pattern "in the event" in a norm. With the extracted sentences and using a natural language parser, noun phrases are created. From these noun phrases, the ones that correspond to exceptions, based on the patterns and exception's characteristics, are identified. A corpus with 2,647 contracts from Onecle repository [1] is used as data for processing. As result, Enlil gets an F-score of 0.9 in classifying contracts using a manually annotated corpus.

Example 5.2.1. :

- Each Party will notify the other Party promptly *in the event* a Party receives **an accusation of infringement pertaining to Licensed Product**.

Gao and Singh [8] develop a hybrid approach for extracting business events and their temporal constraints from contracts. The aim is to create a system that receives a contract as input and returns its events and temporal constraints. The approach is divided into three steps: the first one is to extract the business events, which are an occurrence of significance to a service

[1] http://contracts.onecle.com

engagement, this occurrence is present in contract clauses. To extract the clauses, they select the sentences from contracts and filter them using modal verbs, which is a type of auxiliary verb that represents modality, which is likelihood, ability, permission, and obligation. They build grammar trees to extract sentences with named entities from a set of candidate sentences. Example 5.2.2 presents sentences (extracted from Gao and Singh's work [8]) that represent business events. The second step is the event clustering, using Latent Dirichlet Allocation [4] they extract clusters of events in the same context. Finally, to extract temporal constraints they first select sentences that contain temporal information.

Then, knowing that temporal constraints are often expressed in prepositional phrases (PP), they extract all PP from sentences. With PP, they build a feature vector for each temporal constraint candidate. Using different machine learning algorithms they obtain an F-score of 0.89 for event extraction and 0.9 for temporal constraints. Table 5.1 presents temporal varieties in Service Contracts.

Example 5.2.2. :

- Each party shall be licensed under those rights of the other party;
- 3M shall notify SEPRACOR without undue delay; and
- 3M may increase the supply price for licensed product.

Table 5.1 – Varieties of temporal information in service contracts, extracted from Gao and Singh's work [8]

Classification	**Example**
Time point	on Friday
Frequency	at the beginning of every month
Constraint	before the next payment date.

Gao and Singh [9] create an approach for extracting and classifying norms in contracts. First, they extract norms structure based on Singh's formulation [30]. This formulation defines that a norm has a fixed structure: a type, a subject, an object, an antecedent and a consequent. Singh also defined six main norm types, namely, commitments (which is divided into practical and dialectical), authorizations, powers, prohibitions, and sanctions. The process starts by the extraction of norms from contracts. Using a modal verb filter and then extracting a feature vector, they create a classifier to extract norms from contracts. Second, using 868 norm sentences extracted from real manufacturing contracts, they labelled each norm sentence according to its norm type, creating a gold standard. To extract norms structure, the authors generate an heuristic for each structure. Applying different machine learning algorithms to classify norm types, based on the gold standard set, they obtain an F-score of 0.84. Table 5.2 presents the main features they use in their work.

These works present complex approaches dealing with norms and contracts. They are such relevant for applications related to contract processing. The main difference between those works

Table 5.2 – Classification Features, extracted from Gao and Singh's work [9]

Feature	**Example**
Subject contains organization	Motorola; Google
Clause signal	if; unless
Modal verb	may; should
Main verb expresses an event	deliver; perform
Main verb expresses a state	have; be
Main verb has physical impact	produce; pay
Main verb has social impact	terminate; approve
Negation present	not; neither
Only present	only
Dialectical commitment signal	warrants; understands
Practical commitment signal	agrees to
Authorization signal	the right to $\langle$ physical$\rangle$
Power signal	the right to $\langle$ social$\rangle$
Prohibition signal	must not
Sanction signal	responsible for breach

and ours is the information they try to extract. Although we also deal with norm and contract process, our objective is to identify potential conflicts between norms. To this end, we use similar techniques as the ones presented by them but we go in a different direction, dealing with normative properties, such as permissions, obligations, and prohibitions.

5.3 Normative Conflicts

This section presents works related to the identification of normative conflicts. We present a series of approaches that involve the identification of norm conflicts in multi-agent systems.

Figueiredo and Silva [7] present an algorithm for identifying conflicts between norms. Their goal is to create agents in a multi-agente system that identify conflict cases between norms and values. They understand a value as concepts about desirable end states or behaviours. To specify actions and norms they use the Z language. They define possible conflict cases, such as when the norm states an obligation to the agent to execute an action that demotes an important value to the agent or when a norm states a prohibition to the agent to execute an important action that promotes an important value to the agent. Moreover, some particular cases are defined, such as action refinement, which occurs when an superaction has one or more subactions. The conflicts arise when a norm states an obligation to the agent to execute a superaction and all its subactions demote an important value to the agent. When a norm states a prohibition to execute a superaction and at least one of its subactions promotes an important value to the agent. Finally, they present conflict cases for action composition, which is when a composed action is a series of actions. The conflicts arise when a norm obligates some composed action and at least one of the actions demotes an important value to the agent. It also occurs when a norm states a prohibition to execute a

composed action and at least one of the actions promote an important value to the agent. As a result of the work, the authors produce an algorithm to identify conflicts between norms formalized in Z language.

An approach for normative conflicts in multi-agent systems is presented by Vasconcelos *et al.* [33]. They present mechanisms for detection and resolution of normative conflicts. They develop a formal representation of norms with constraints, presenting formal definitions of normative conflicts and defining how they can be resolved. Moreover, they present mechanisms to adoption and removal of norms in order to avoid conflicts in global normative states. For norm representation, they use a tuple $\langle v, t_d, t_a, t_e \rangle$, where v receives one of the three deontic concepts, obligation, permission and prohibition. t_d, t_a and t_e define respectively, the time when v is introduced, the time when it is activated and when it expires. Using first-order logic terms to represent the agents and the roles of the agents and formulae to represent norms constraints, they build a normative representation. For the task of detecting a conflict, they define a conflict as an action that is simultaneously prohibited and permitted/obliged, and its variables have overlapping values. To resolve conflicts they manipulate the constraints associated to the norms' variables, removing any overlap in their values. In norm adoption, they use a set of auxiliary norms to exchange by the ones applied to the agent. In norm removal, they remove a certain norm and all curtailments it caused, bringing back a previous form of the normative state. This work is extremely complete and brings many intuitions for the use of first-order logic in norms representation.

Li *et al.* [18] present and approach to detect conflict in composite institutions. They define composite institution as a type of cooperative institution in which member institutions are treated individually, which makes a new composite institution not an individual institution but a common governance scope of all individual institutions. An institution is a set of policies that encourage specific normative behaviors, without considering that the participants will comply with such normative behaviors. In such scenario, conflicts may arise when agents of an individual institution interact with the composite institution, since the policies of a composite institution may conflict with the polices of an individual institution. They implement the computational model of institutions using *AnsProlog*, which facilitates the reasoning, verification, and validation of the institution and its polices. To detect conflicts, they define an event-driven model in which events derive from actions of participants in the system. Thus, a composite institution is in conflict if exists a composite trace that is inconsistent for two corresponding synchronized models. The inconsistency occurs when in corresponding states a fluent (denoting a normative property, such as permission, prohibition, and obligation) is true in one and false in the other. Besides, they present an approach to detect conflict between institution policies applying a similar approach that determine conflict traces and detect conflicts.

Figueiredo and Silva's work [7] present an algorithm for normative conflicts detection using first-order logic. They use Z language to formalize the conflict types and then identify them between norms. The same context is used by Vasconcelos *et al.* [33], which uses first-order logic to resolve norm conflicts. They use different algorithms for detecting and solving conflicts. Li *et al.* present an

approach that also uses a logic language to detect conflicts, however, they apply in the context of institutions, which have their own participants and a set of rules. To detect conflicts, they propose a series of rules that consider normative properties applied to similar states.

These works have a lot in common, but differ in fundamental ways both from our work and among themselves. The major difference between these works and ours is that we work with conflicts at the natural language level. Such property led us to create different approaches to identify potential conflicts, such as information extraction and semantic similarity. Therefore, although our work has the same global objective of the presented ones, the approaches we use to achieve our objective and the contribution we leave are completely different.

6. CONCLUSION

This work presented an approach to identify potential conflicts between norms in contracts. Our contribution was given in four main aspects: first, we created a norm identifier that allows us to distinguish norm sentences from common sentences in a contract; second, we created a party identifier that identifies the parties and their aliases in contracts with only two parties; third, we proposed a representation to norms with three components, namely party name, modal verb, and norm action, such representation helps with norm comparison to conflict identification; finally, we presented an algorithm that measures the semantic similarity between sentences. During this work, we also created a set of data that can be available for other researchers, such as the sets of norm sentences and common sentences used in the norm identification phase and the small contract corpus with conflicting norms used to evaluate our approach.

To achieve our objective, we divided our approach into two phases. In the first phase, we used several techniques to extract information from contracts, such as norms, parties, and norm elements. With such information, we created a norm representation that characterizes the most relevant features of a norm. In the second phase, the extracted information was used to compare different norm structures and identify potential conflicts between them. As our main result, the potential conflict identifier obtains an accuracy of 78% when executed over 16 contracts containing norm conflicts. This result represents the application of our approach in a defined corpus of contracts, in which contracts have been modified in order to contain norm conflicts. As long as we searched, we did not find works that deal with conflicts between norms in natural language, therefore, we consider that this is a good first result. The potential conflict identifier is a first step to the automation of contract analysis since it allows a computer to review and indicate potential conflicts within a contract. Additionally, it provides an automatic function for a task that is usually made by humans and thus error prone and time consuming.

As future work, we plan to generalize our approach in order to contemplate other contract structures. The current approach deals with contracts with at most two parties and follows a certain pattern of parties description. As an alternative, we can either create a more general rule for the party identification or invest in a machine learning approach to recognize parties and aliases together with named entity recognition techniques. Besides, we plan to improve the semantic similarity algorithm, since it considers extra aspects that are often unnecessary to the similarity calculus, such as extra synonyms that often do not have much in common with the word but are still compared. Finally, we want to make available a richer conflict corpus, larger and with a better variation of study cases of conflicts, which can be used to get more expressive results.

6.1 Publications

We have submitted our partial reports to different events in order to obtain peer reviews for our work. Among these events are the Association for the Advancement of Artificial Intelligence (AAAI-15) conference and the International Joint Conference on Artificial Intelligence (IJCAI-15). We have received good reviews and important suggestions. Recently, our work have been accepted as a short paper, as follows:

- João Paulo Aires, Vera Lucia Strube de Lima, and Felipe Meneguzzi. Identifying potential conflicts between norms in contracts. In International Workshop on Coordination, Organisations, Institutions and Norms in Multi-Agent Systems, 2015.

BIBLIOGRAPHY

[1] Tara Athan, Harold Boley, Guido Governatori, Monica Palmirani, Adrian Paschke, and Adam Wyner. Oasis legalruleml. In *Proceedings of ICAIL*, pages 3–12, 2013.

[2] Robert Axelrod. An evolutionary approach to norms. *The American Political Science Review*, 80(4):pp. 1095–1111, 1986.

[3] Jeremy Bentham, Herbert Lionel Adolphus Hart, and Frederick Rosen. *Of laws in general*, volume 177. Athlone Press London, 1970.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[5] Michael Curtotti and Eric Mccreath. Corpus based classification of text in australian contracts. In *Proceedings of the Australasian Language Technology Association Workshop, Melbourne, Australia*, pages 18–26, 2010.

[6] Michael Curtotti and Eric C. McCreath. A corpus of australian contract language: Description, profiling and analysis. In *Proceedings of the 13th International Conference on Artificial Intelligence and Law*, ICAIL '11, pages 199–208, New York, NY, USA, 2011. ACM.

[7] Karen da Silva Figueiredo and Viviane Torres da Silva. An algorithm to identify conflicts between norms and values. In *Coordination, Organisations,Institutions and Norms in Multi-Agent Systems*, pages 259–274, 2013.

[8] Xibin Gao and Munindar P. Singh. Mining contracts for business events and temporal constraints in service engagements. *Services Computing, IEEE Transactions on*, 2013.

[9] Xibin Gao and Munindar P. Singh. Extracting normative relationships from business contracts. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 101–108, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[10] Xibin Gao, Munindar P. Singh, and Pankaj Mehra. Mining business contracts for service exceptions. *IEEE T. Services Computing*, 5(3):333–344, 2012.

[11] Guido Governatori. Representing business contracts in *ruleml*. *International Journal of Cooperative Information Systems*, 14(2-3):181–216, 2005.

[12] C. Wilfred Jenks. The conflict of law-making treaties. *BYIL*, 30:401, 1953.

[13] Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, 1997.

[14] Andrew J. I. Jones and Marek J. Sergot. Deontic logic in the representation of law: Towards a methodology. *Artif. Intell. Law*, 1(1):45–64, 1992.

[15] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.

[16] Martin J Kollingbaum, Timothy J Norman, Alun Preece, and Derek Sleeman. Norm conflicts and inconsistencies in virtual organisations. In *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, pages 245–258. Springer, 2007.

[17] Claudia Leacock and Martin Chodorow. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.

[18] Tingting Li, Tina Balke, Marina De Vos, Ken Satoh, and Julian Padget. Conflict detection in composite institutions. In *International Workshop on Agent-based Modeling for Policy Engineering (AMPLE 2012)*, pages 75–89.

[19] Yuhua Li, David McLean, Zuhair Bandar, James O'Shea, and Keeley A. Crockett. Sentence similarity based on semantic nets and corpus statistics. *IEEE Trans. Knowl. Data Eng.*, 18(8):1138–1150, 2006.

[20] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 296–304, 1998.

[21] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[22] Frank Robert Palmer. *Mood and modality (2nd Edition)*. Cambridge University Press, 2001.

[23] Mike Paterson and Vlado Dančík. Longest common subsequences. In Igor Prívara, Branislav Rovan, and Peter Ruzička, editors, *Mathematical Foundations of Computer Science 1994*, volume 841 of *Lecture Notes in Computer Science*, pages 127–142. Springer Berlin Heidelberg, 1994.

[24] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet::similarity: Measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL–Demonstrations '04, pages 38–41, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[25] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(1):17–30, Jan 1989.

[26] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 448–453, 1995.

[27] Denise M. Rousseau and Judi McLean Parks. *The contracts of individuals and organizations*, volume 15. JAI PRESS LTD, 1993.

[28] Ali Sadat-Akhavi. *Methods of resolving conflicts between treaties*, volume 3. Martinus Nijhoff Publishers, 2003.

[29] M. Sergot. Machine intelligence 11. chapter Representing Legislation As Logic Programs, pages 209–260. Oxford University Press, Inc., New York, NY, USA, 1988.

[30] Munindar P. Singh. Norms as a basis for governing sociotechnical systems. *ACM Trans. Intell. Syst. Technol.*, 5(1):21:1–21:23, January 2014.

[31] Mark J Steedman. Verbs, time, and modality. *Cognitive science*, 1(2):216–234, 1977.

[32] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[33] Wamberto W. Vasconcelos, Martin J. Kollingbaum, and Timothy J. Norman. Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 19(2):124–152, 2009.

[34] Wamberto Weber Vasconcelos, Martin J. Kollingbaum, and Timothy J. Norman. Resolving conflict and inconsistency in norm-regulated virtual organizations. In *6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14-18, 2007*, page 91, 2007.

[35] G. H. von Wright. *Deontic Logic*, volume 60 of *New Series*. Oxford University Press on behalf of the Mind Association, 1951.

[36] Erich Vranes. The definition of 'norm conflict' in international law and legal theory. *European Journal of International Law*, 17(2):395–418, 2006.

[37] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.