PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

AUTOMATIC REASONING OVER CONTRACT CLAUSES

JOÃO PAULO DE SOUZA AIRES

Dissertation submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science.

Advisor: Prof. Felipe Rech Meneguzzi

Porto Alegre
2020

REPLACE THIS PAGE WITH
THE LIBRARY CATALOG
PAGE

REPLACE THIS PAGE WITH THE COMMITTEE FORMS

To Cássia

“Oxalá a minha vida seja sempre isto: O dia cheio de sol, ou suave de chuva.”
(Alberto Caeiro)

ACKNOWLEDGMENTS

Many people contributed to the process of writing this thesis and surviving to the PhD, thus thanks are to the following people.

First and foremost, to my supervisor Felipe Meneguzzi, who turned me into a scientist and had the patience to read and correct so many mistakes. If this thesis exists, it is due to his persistence in not giving up on me.

To all anonymous reviewers of AAMAS, AAAI, COIN, and AI and Law, whose feedback was essential in our path to this thesis.

To the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for funding my PhD.

To my colleagues at PUCRS, Juarez Monteiro, Ramon Pereira, Leonardo Amado, Roger Granada, and Maurício Magnaguagno, with whom I had the best conversations about anything possible and the support to deal with the challenges of the PhD.

To the members of the best band ever, Mortticia, Guilherme Wiersbicki and Lucas Zawacki for keeping my music alive.

To my fiancée Cássia Pilar Salgado, for the best partnership I could have with someone and for making me strong when I thought I was weak.

To my parents Antônia Esnarriaga de Souza and Paulo Aires and siblings Fernanda and Érica for giving me the base I can always look for.

AUTOMATIC REASONING OVER CONTRACT CLAUSES

RESUMO

Contratos escritos são utilizados para formalizar acordos envolvendo a troca de bens e serviços entre duas ou mais partes. Eles definem ações esperadas durante seu período de vigência através de normas. Tais normas seguem conceitos baseados em lógica deôntica, definindo permissões, proibições e obrigações. No entanto, conflitos entre normas podem ocorrer quando duas normas são aplicadas a um mesmo contexto tendo sentidos deônticos diferentes, como a proibição e a obrigação de uma mesma ação. Estes conflitos invalidam as normas e criam uma inconsistência para o contrato. Para evitá-los, um revisor deve ler as normas e encontrar 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, nesta tese desenvolvemos uma abordagem para identificar e classificar conflitos normativos. Transformando normas em representações vetoriais, somos capazes de extrair características relevantes de normas de forma a facilitar a identificação de conflitos normativos. Propomos quatro classes de conflitos normativos e as usamos para treinar um classificador de conflitos. Como resultado, nossa abordagem obtém acurácia superior a 99% na identificação e 78% na classificação de conflitos normativos.

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

AUTOMATIC REASONING OVER CONTRACT CLAUSES

ABSTRACT

Contracts formally represent agreements between parties and often involve the exchange of goods and services. In contracts, norms define expected behaviours from the parties using deontic statements, such as obligations, permissions, and prohibitions. However, norms may conflict invalidating themselves and producing a contract inconsistency. A conflict often arises when two or more norms are applied to the same context but have different deontic statements, such as permissions $\times$ obligations and prohibitions $\times$ obligations. The identification and resolution 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 thesis we introduce an approach to identify and classify norm conflicts between norms in contracts written in natural language. We rely on the use of sentence embeddings to represent and manipulate natural language to extract information and use it to identify norm pairs as conflicts. We propose four norm conflict classes and use them to train a norm conflict classifier that can help on the conflict cause identification. The results show that our approach achieves an accuracy higher than 99% on the identification and 78% on the classification of conflicts.

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

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

NLP – Natural Language Processing

CTD – contrary-to-duty

POS – Part-of-Speech

SVM – Support Vector Machines

ANN – Artificial Neural Networks

DNN – Deep Neural Network

CNN – Convolutional Neural Network

RNN – recurrent neural network

LSTM – Long Short-Term Memory

TF-IDF – Term Frequency - Inverse Document Frequency

CNN – Convolutional Neural Network

CBOW – continuous bag-of-words

SGD – stochastic gradient descent

CONTENTS

1. INTRODUCTION

Living in society often involves complying with a set of constraints that, at the same time, limits and structures individual behaviours. This invisible force that guides the way we interact to each other consists of predefined rules commonly known as social norms. Rules are present in every sphere of our society, such as games, corporations, institutions, and agreements. In each case, being part of a social group means accepting the specific rules created for it. The number of individuals touched by rules may vary according to the context applied. In agreements, rules apply restrictions to a minimum number of two individuals. This is a very common environment since agreements define the exchange of goods and services between individuals.

Agreements are a type of relation between individuals that involves the use of specific rules accepted within a certain amount of time. These rules describe the way each individual is expected to act during the agreement creating a mutual trust between participants. The formal way to define agreements is using contracts. Contracts are documents that define the parties, the agreement subject, what each party must comply with, and what is expected in case of violations. Each statement in a contract is defined by a clause and the rules are commonly known as norm clauses.

We use contracts in most agreements between people and enterprises [70]. Thus, numerous contracts need to be created, reviewed, read, and signed. When creating a contract, conflicts between norm clauses may arise without a careful review from contract creators. Norm conflicts often occur when a norm invalidates one or more norms by its definitions. For example, if *norm 1* states that “Company X must pay the taxes for all products.” and *norm 2* states that “Company X shall not pay the taxes from product Y”, *norm 1* invalidates *norm 2* since paying all taxes from products consists of paying the taxes from product Y too. When it happens, complying with one norm means violating the other and in a long run it may create a contract inconsistency invalidating the whole contract. One of the key aspects to consider when looking for norm conflicts is the deontic meaning in norms, which states whether a norm refers to a permission, prohibition, or obligation.

As contracts tend to be long and complex, analysing contracts for conflicts becomes a laborious task for human experts. In tasks such as creation and revision of contracts, avoiding norm conflicts requires awareness from the contract creator of norm possible interaction, which makes it difficult for long contracts with hundreds of norms. Thus, manually checking norm conflicts and resolving them demands too much human effort and is error-prone. To mitigate it, we need to be able to reason over contracts and detect such norm conflicts and their causes. Recent efforts to reason over contracts have generated multiple approaches that deal with parts of the problem of detecting such conflicts. Gao and Singh [41] developed an approach to identify business events and temporal constraints from

contracts. The approach uses machine learning and rule-based techniques to identify contract elements. Rosso *et al.* [89] developed an approach to retrieve information from legal texts using a system named JIRS. They translate norm in natural language to a symbolic language and use a series of rules to identify the conflicts. Azzopardi *et al.* [9] introduce a tool to help contract-drafters to revise contracts by allowing them to make an intelligent browsing of the contract. They use off-the-shelf natural language processing techniques to allow the tool users to browse through the clauses and get contract information faster.

In this thesis, we develop an approach to automatically classify norm conflicts in contracts by applying a new way to manipulate natural language. Our approach receives as input a raw contract written in natural language and outputs a list of norm conflicts and their types with an associated degree of confidence. The identification and classification of norm conflicts can make the contracts process faster and more reliable, which facilitates contract review. With the automatic identification of norm conflicts, contract creators can save time during the revision of contracts, and with the classification of these conflicts, they can find ways to resolve conflicts faster. Our focus in this thesis is to deal only with conflicts between norms since they are inside a specific scope that we can manipulate. General conflicts, such as discourse conflicts and contradictions involve a more complex identification process due to the range of possibilities in the use of natural language. Thus, our thesis has four contributions: (1) the formalisation of four norm conflict types; (2) a manually annotated dataset containing conflicting norm pairs and their conflict types; (3) a system able to process contracts to obtain their norm conflicts and their types; and (4) ConCon, a web-tool publicly available that uses our system to process user contracts.

This document consists of seven chapters. In Chapter 2, we introduce the concepts used in this work, such as norms, contracts, and norm conflicts. In Chapter 3, we describe the concepts of existing learning algorithms and how we use them in our approach. In Chapter 4, we describe the general task of identifying norm conflicts. In Chapter 5, we detail our approach on identifying and classifying conflicts. In Chapter 6, we describe existing work on contract reasoning and compare them to our proposal. Finally, in Chapter 7 we draw some conclusions about our research and potential future work.

2. BACKGROUND

In this chapter, we introduce the main concepts used in our thesis. In order to understand the role of norms in contracts, we describe deontic logic, which is the theoretical basis for most normative systems. Then, we narrow down the formal definition of norms, detailing their structure, components, and types. In the context of this work, we formalise norms from contracts. Therefore, to understand these documents, we explain how they are structured and used, as well as how norms are described in it. Given this context, we describe how norm conflicts arise in contracts and try to identify the causes for their occurrence as well as the way humans solve them.

In order to define a way to compare norms and identify conflicts, we introduce the concept of norm representation, which we use to reason about the information extracted.

Finally, as an essential technique in our approach, we introduce natural language processing and the methods used by us.

2.1 Deontic Logic

Logic or mathematical logic is commonly used to define what is true through reasoning. Such process relies on the verification of arguments that support an intended conclusion. The basis of a logical reasoning is through propositions, which are declarative sentences either true or false [24, Chapter 1]. Using propositions, logical methods can draw conclusions from deductive analysis that produce new knowledge from existing facts. An example of propositions are:

a. It is below 0 degrees Celsius.

b. There are clouds

c. If there are clouds and it is below 0 degrees Celsius then it is snowing.

In this example, *a* and *b* are declarative sentences and *c* is an entailment. If both *a* and *b* are true, we can deduce "it is snowing" (*d*) through *c*. This type of logical representation is known as propositional logic [53, Chapter 1]. In this type of logic, every proposition is a single unit, as in the example where a corresponds to an entire sentence. We connect propositions using logical connectives, such as "not" ($\neg$), "and" ($\wedge$), "or" ($\vee$), "implication" ($\rightarrow$), and "if and only if" ($\Longleftrightarrow$). Using such connectives, we can represent *c* as: $(a \wedge b) \rightarrow d$. Although we can use propositional logic to represent most types of sentences, if we need to make a fine-grained analysis, we need to use first-order logic (also known as predicate logic). First-order logic deals with three elements: objects, relations, and facts. Objects can

be elements such as dog, car, Jane, chair, and house. Relations can be “has father”, “is close to”, and “is red”. Finally, facts are concrete elements, such as “house is red”, “Jane has father”, and “dog is close to car”. Facts have a value either true or false. Thus, to represent this type of logic, we use relations (predicates) as capital letters followed by brackets, such as $P(X)$. Where P is the predicate that can represent, for example, “has father” and X is a variable that can be grounded using an object, such as $P(Jane)$. Thus, $P(Jane)$ can be true or false according to the facts and rules in its specific case. Predicates can have different arity depending on the type of relation they describe. For instance, consider “is red” as R. This predicate has arity of one, *e.g.* $R(house)$. A predicated representing the “is brother of” relation ($B(X, Y)$) have an arity of two. Thus, to represent the sentence “John is brother of Jane”, we can use $B(John, Jane)$. As in propositional logic, we can use first-order logic to perform deductive methods. In computer science, deductive methods are central to many artificial intelligence approaches involving agents. Agents use a knowledge base to infer new knowledge about the environment through logical inference [92, Chapter 7].

Among the branches of logic, modal logic is the one that deals with ‘modes’ of truth [53, Chapter 5]. These ‘modes’ expand the idea of propositions being either true or false by bringing the concepts of possibility, necessity, obligation, knowledge, belief, perception, among others [34]. These modes are divided into multiple types of modal logics, such as alethic, temporal, epistemic, deontic, among others. Two most used modes of truth are what **must be** (necessity) and what **may be** (possibility), respectively represented by the unary operators $\Box$ and $\Diamond$. The assertion *necessarily* P ($\Box P$, where P is some propositional variable) is true if P is true in all possible worlds. On the other hand, an assertion *possibly* P ($\Diamond P$) is true if at least one world exists where P is true. By “world” here we mean a state of affairs in which, in this case, P is true. Using both modal logic operators, it is possible to extend classical logic to express sentences such as the following ones:

- “It is possible that tomorrow it will be hot and sunny.” We can represent such sentence by $\Diamond H \wedge \Diamond S$, where H stands for “it will be hot tomorrow” and C stands for “it will be sunny tomorrow”.
- “Necessarily, if it snows, it is cold.” We can represent such sentence by $\Box(S \rightarrow C)$, where S is “if it snows” and C is “it is cold”.

To identify the value of a statement with modal logic operators, we can use the Kripke model, which defines a set of states (or worlds) and each world specify a truth value of all propositional variables. Besides the set of states, the Kripke model defines the relation between states. Figure 2.1 illustrates a Kripke model with three propositional variables (A, B, C). Each node in the graph is a world (defined by w_i) and the variables in each world are true, e.g., in the world w_1, A is true, whereas in world w_5 A and B are true. The edges describe the accessibility between the worlds. We can represent a Kripke model using a tuple $\langle W, P \rangle$, where W is the set of possible words and P is a subset of W containing an

infinite sequence of $P_1, P_2, P_3, ..., P_n$, where each P_i is a possible world in W [21, Chapter 2]. Thus, given a proposition A, a model $M = \langle W, P \rangle$, and a possible world α, we say that A is true with the following symbolism:

$\models^M_\alpha A$

For the necessity and possibility operators, we use the following rules:

- $\models^M_\alpha \Box A$ iff for every β in M, $\models^M_\beta A$
- $\models^M_\alpha \Diamond A$ iff for some β in M, $\models^M_\beta A$

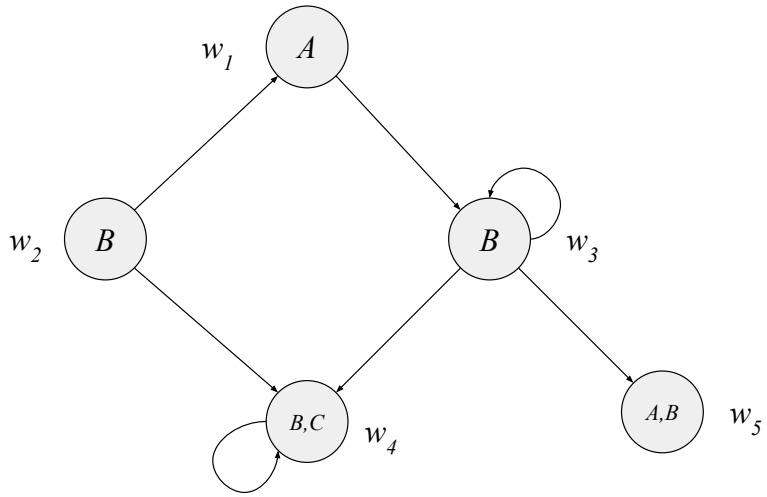


Figure 2.1 – An example of a Kripke model

Deontic logic is a type of modal logic developed to reason about “ideal” worlds from the point of view of compliance with a body of stipulations [16]. As one of the modal logics, deontic logic studies sentences composed by propositional variables and normative expressions [51, Chapter 1]. Where normative expressions are those including words, such as “obligation”, “duty”, “permission”, “right”, among others. Thus, deontic logic deals with normative notions of permission, prohibition, obligation, and right [16]. Permission is the act of allowing a certain behaviour by granting some agent to do it without sanctions. A permission can be either *stricto sensu* or *lato sensu*. When a permission is *stricto sensu*, it expresses that an act/behaviour is neither obligatory nor prohibited, *i.e.*, the agent may choose to do or not to do such permitted act. In natural language, we use modal verbs like ‘may’ and ‘can’ to express this type of permission. Whereas a *lato sensu* permission expresses that a certain act is not forbidden, *i.e.*, it grants the agent permission to perform such act. This case refers to acts that are obliged and, therefore, permitted. This type of permission is often described in natural language by “it is permitted” [93, Chapter 1]. A prohibition is a constraint that indicates an act that must not be performed. The definition of a prohibition often has a definition of a sanction in case of violation. Finally, an obligation is the act of forcing a certain act to be performed by an agent. Like prohibitions, obligations

also have a definition of sanctions for cases when the agent violates the obligation. To express the deontic meanings (permission, prohibition, and obligation) in deontic logic, we can use only **permission**. The negation of a permission is a **prohibition**, e.g., “It is not permitted to kill”, thus, it is forbidden to kill. The prohibition of not performing an act makes such act an **obligation**, e.g., “it is forbidden not to make silence in hospitals”, thus, silence becomes an obligation [103]. In more sophisticated cases, a contrary-to-duty (CTD) norm is used to enforce a second obligation if an agent violates the first one [84]. Example 2.1.1 illustrates a classic example of CTD. In the example, the first norm states that there must be no fence and in a second norm it states that, if there is a fence, it must be white. Thus, when an agent violates the first norm and uses a fence the second norm is activated describing that if it exists it must be white. The action described in the second norm is often called reparation action since it repairs a previous violation. Consider permission as P, prohibition as F, obligation as O, and an act A, we can describe the equivalences between deontic meanings as follow:

- $P(A)$;
- $F(A) = \neg P(A)$; and
- $O(A) = \neg F(A)$ or $\neg P(\neg A)$

Example 2.1.1. :

1. There must be no fence.
2. If there is a fence then it must be a white fence.
3. There is a fence.

These definitions allow us to identify other types of deontic meanings, such as indifference, commitment, and compatibility and incompatibility [103]. **Indifference** occurs when an act and its negation are both permitted, i.e., given an act A, it is $P(A) \wedge P(\neg A)$. For example, if we are on a bus, we are allowed to sit and not to sit. Therefore, indifference is more limited than permission and what is indifferent is permitted, but what is permitted is not indifferent, as well as, what is obligatory is also permitted, but not indifferent. Two acts are **compatible** if its conjunction is permitted, whereas two acts are **incompatible** if its conjunction is forbidden. For example, consider a library, you are permitted to read and think ($P(R \wedge T)$), i.e., they are compatible acts. However, you are forbidden to read and sing ($F(R \wedge S)$) at the same time, which makes these acts incompatible. If performing a certain act commits us to perform another one and the implication of both is an obligation, we have a **commitment**, i.e., $O(A \rightarrow B)$. For example, if you promise something you commit to keep your promise.

We can use deontic logic to represent normative expressions containing deontic meanings. It allows us to reason over norms and understand how their base is structured. For instance, consider Example 2.1.2, where we translate the sentences in Example 2.1.1 to deontic logic. In Example 2.1.2, we use P, F, and O to represent permission, prohibition, and obligation respectively. *Fe* stands for the fact "there is a fence" and *Fw* stands for the fact "the fence is white". In this work, we use the deontic meanings from deontic logic to identify what is the role of a norm in a contract and compare deontic meanings in order to identify conflicts.

Example 2.1.2. :

1. $O(\neg Fe)$
2. $Fe \rightarrow O(Fw))$
3. Fe

Von Wright [103] defines laws regarding the use of deontic logic. He divides them into three groups. The first group deals with the relation between permission and obligation. It consists of two laws:

1. PA is identical with $\neg(O\neg A)$, *i.e.*, $(PA) \iff \neg(O\neg A)$
2. $OA \models PA$, *i.e.*, $(OA) \models (PA)$ expresses a deontic tautology.

The first law in this group refers to the equivalence between permission and obligation. Permission of an action (A) means that there is no obligation of not performing such action. For example, if I am allowed to play piano, there must be no obligation to not play it. The second law is also referred as the axiom D in standard deontic logic. This law describes the case where if an action is obliged, it is also permitted. Consider the obligation to wear suit at work. Since wearing the suit is an obligation, everybody is permitted to use it. After defining these two laws, Von Wright describes four laws regarding the dissolution of deontic operators.

1. $OA \wedge B$ is identical with $(OA) \wedge (OB)$
2. $PA \vee B$ is identical with $(PA) \vee (PB)$
3. $(OA) \vee (OB) \models OA \vee B$
4. $PA \wedge B \rightarrow (PA) \wedge (PB)$

All laws of this group refer to the same property of dissolution. This means that if we have two propositions connected by $\wedge$ or $\vee$ connectives, and one of these propositions

is permitted or obliged, the whole expression can be reformulated to one where both propositions have the same deontic value. Finally, Von Wright describes seven laws regarding commitment in deontic logic.

1. $(OA) \wedge (OA \rightarrow B) \models OB$
2. $(PA) \wedge (OA \rightarrow B) \models PB$
3. $\neg(PB) \wedge (OA \rightarrow B) \models \neg(PA)$
4. $(OA \rightarrow B \vee C) \wedge \neg(PC) \models \neg(PA)$
5. $\neg((OA \vee B) \wedge \neg(PA) \wedge \neg(PB))$
6. $(OA) \wedge (OA \wedge B \rightarrow C \models OB \rightarrow C)$
7. $O\neg A \rightarrow A \models OA$

The first two laws of this group refers to the case where an obligation to perform a certain action leads to a second action. In this case, the second action becomes obliged too. For example, if we are obliged to enter in a room, it means we need to open the door. The action of opening the door becomes obliged too. The same example works for the second law. If we are free to perform a certain action that leads to a second one, the latter is also permitted. The third law is a variation of the first two. It describes the case where an obligatory action leads to a forbidden action. In this case, the obligatory action is also forbidden. For example, consider that we are prohibited of opening the door and obliged to enter in the room. Since we must not open the door, entering the room becomes also prohibited. This is an example of norm conflict we cover in Section 2.4. The forth law is an extension to the third, which describes the case where an obligations leads to the choice of performing to forbidden actions. As result, the obliged action becomes forbidden. To understand such law, we can use the room example and add the option of entering through the window. When both opening the door and window are forbidden, entering the room becomes forbidden too. The fifth law describes the impossibility of being obliged to choose between forbidden actions. The sixth law describes the case where doing two actions, being the first obliged, commits us to perform a third one. In this case, doing the second action alone commits us to perform the third actions. For example, if we enter the room and, to do so, open the door, it commits us to use a key. Thus, opening the door commits us to use the key. Finally, the seventh law deals with the failure of performing a certain action, which commits us to perform it. This action becomes, then, obliged.

2.2 Norms

Norms or social norms are mechanisms that regulate expected behaviours from individuals in a specific society or group. Norms can manage a variety of situations, such as property rights, forms of communication, contracts, and concepts of justice [68]. They are socially enforced and define what is expected in specific situations. Through norms, justice mechanisms can solve disputes and conflicting interests in society [86]. We express norm definitions by means of permission, prohibition, and obligation modalities from deontic logic (see Section 2.1). According to Andersen and Taylor [7, Chapter 3], norms may be either implicit or explicit. Implicit norms do not need to be stated to be understood, they can be learned by specific instructions or observation of the culture. They are commonly used for members of a group and act as a social behaviour. The violation of an implicit norm produces social punishment instead of a monetary punishment. As an example, consider some busy cities where people tend to leave one side of the escalator free for people in a hurry to have a fast access. An explicit norm is formally communicated or written and sanctions are applied in case of violation. In this case, consider transit laws that regulate how cars and pedestrians must behave. Andersen and Taylor [7, Chapter 3] describe three types of norms defined by Sumner and Keller [99], namely, folkways, mores, and laws. Norms of the folkway type are not essentially important since they are part of the social behaviours, i.e., violating them have no sanction related. Some examples of folkway norm are: dress code and proper eating. Mores are more important then folkways, they control moral and ethical behaviours and violating yields some punishment, e.g. cannibalism and incest. Laws are norms that define what is right or wrong in society, have a formal structure and violating them results in serious repercussions. Some examples of laws regulate speed limits, murder, and taxation. In society, norms can be recognised in two forms: injunctive and descriptive. Injunctive norms describe what people believe that have to be done. Whereas descriptive norms are those referring to the belief of what the majority of people is really doing [68]. For example, consider a library, people are silent and calm in a way that others tend to follow such behaviour (descriptive) as they fear sanctions if they start making noises (injunctive). Thus, while descriptive norms have a strong influence on social behaviour, injunctive norms have an impact on individuals attitudes.

Mahmoud *et al.*[68] defines a formal classification of norms that consider five types, they are: conventions, essential, regulative, constitutive, and procedural. Conventions are natural norms that do not need any enforcement to emerge. They solve coordination problems without dealing with conflict cases between an individual and the group interests, which makes such norms accepted by the majority and often expected by group members. On the other hand, essential norms deal with problems involving conflicts between individual and group interests. In this case, these norms try to solve or ease the problem in order to give

preference for the group. For example, to respect the traffic light is an essential norm. Although you may be hurry, you shall not pass over people, but let them cross the streets, which has a greater benefit to society. Besides these two norm classifications, in the context of normative multiagent systems, Caire [15] defines three other classifications. Regulative norms define prohibitions, permissions, and obligations, i.e., they define ideal and varying degrees of ideal behaviours in a system. A regulative norm expressed by an obligation may be, for example, the access to public transportation in Brazil where people must pay the fare if they are older than 06 and younger than 60. Constitutive norms are the ones that define abstract concepts, such as a piece of paper having the value of R$5 (Brazilian currency) or a certain parcel of land being the private property of an individual. Thus, this norm classification deals with money, marriage definitions, land property, among others. Finally, procedural norms are divided into objective and subjective. Objective procedural norms describe decisions and how they are applied, detailing who can make them and who can influence on them. Subjective procedural norms deal with attitudes about the way decisions should be made.

Sadat-Akhavi [93, Chapter 1] introduces two types of norm: mandatory and permissive norms. Mandatory norms impose an obligation to the norm addressee to do or not to do a certain act. Mandatory norms that impose an obligation are called obligatory norms and often describe an act that the agent must do. For example, 'Company X must buy product Y from Company Q in the next four days.' Mandatory norms that impose prohibitions are called 'prohibitive' norms, they often describe a certain behaviour that the agent must not perform.For example, 'Company X shall not use product W in the formulation of solution Z.' By contrast, permissive norms addresses to the agent the freedom to do or not to do a certain behaviour, which is different from both obligation and prohibition. For example, 'Company X may require the expiration date of product Y from Company Q.' In this example, Company X may require or not the expiration date and this freedom of choice is granted by the permissive norm.

Given these norm types and formalisation, we summarise the information in Table 2.1 bringing a relation between the types. We divide them into two macro types, namely, subjective/implicit and objective/explicit. For each macro type, we relate the equivalence between micro types, e.g., folkways and conventions are subjective/implicit norms with similar definitions. These norm types involve a social belief over what must be done, which does not necessarily mean it is enforced by the State. Punishment in these norms refer to social sanctions, such as social disapproval and ostracism. Whereas objective/explicit norms are enforced by law and have more severe sanctions, such as loss of liberty and social privileges. Both mandatory and permissive types are generic and are present in every other type, thus, we do not add them in the table.

When applied, obligatory and prohibitive norms need a support to ensure they will be complied by society members. Such support comes from sanctions that cause a detri-

Subjective/Implicit	Objective/Explicit
Folkways/Conventions	Laws/Regulatives/Constitutives
Mores/Essentials	Descriptive
Injunctive	-

Table 2.1 – Norm typology

ment over a failure to perform an obligatory action or to avoid a forbidden action [23]. Sanctions constitute the core of legal order as they emerge as a mechanism of social control [95]. Any violation of a social norm have a respective sanction applied to the violator. For example, if someone driving a car crosses the traffic light with a red signal, a monetary sanction may be applied over him. Sanctions can be both positive and negative [12]. In this case, when society members meet normative expectations, they are rewarded accordingly. This is called a positive sanction. It is the case when a child is learning social norms. As she understands and complies with norms, she receives positive sanctions, such as approval and praise [7, Chapter 3]. Whereas punishment over deviance is a type of negative sanction. The severity in negative sanctions may vary depending on the violation gravity. Sanctions have a strong influence in society as they do not need to be activated to be effective. Only the anticipation of a reward or punishment is enough for people to comply with norms.

2.3 Contracts

Contracts formalise a voluntary agreement between two or more parties that is enforceable by law. It differs from a gentleman's agreements, which is an agreement between two or more parties often made orally or written that have no enforceable mechanism other than the honour of the parties. Contracts mediate any trade between parties, whether they be explicit or implicit [47]. Contracts have three main components, which define the content and the contracts purpose, namely, *promise*, *payment*, and *acceptance* [90, Chapter 2]. 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 behavioural event, which means a commitment to do (or not to 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. Schwartz and Scott [94] describe categories of contracts. They base these categories using the agreements between society members. Agreements described in contracts often involve a seller (whether of goods or services) and a buyer. However, it may occur between individuals concerning social agreements. They occur between individuals and firms in four different categories: firm sells to firm, individual sells to individual, firm sells to individual, and individual sells to firm. The first category deals with the relation between firms, which uses

contract law to resolve disputes between firms. The second category refers to the relation between individuals governed by family law (dealing with ante nuptial agreements and divorce settlements) and real property law (dealing with home sales and leases). In category 3, they describe contracts of firms selling to individuals regulated by consumer protection law, real property law, and securities law. Finally, contracts in the fourth category describe individuals selling to firms, which involves the sale of a person's labour regulated by the laws of employment relation.

These contract types describe existing agreements in our society. They facilitate the categorisation for contract writers and lawyers. In this work, we deal with contracts written in natural language of any type. The task of analysing 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 [42].

2.4 Norm Conflicts

As contracts contain series of deontic logic statements specifying what each party is expected to fulfil, it is important that these statements are logically consistent. Any mistake in clearly specifying the norms within a contract can cause clauses to be in conflict among themselves. This is particularly true of contracts in natural language, because, not only is such language ambiguous, human writers of such contracts may overlook subtle logical conflicts. Therefore, it is important to understand how they arise and what are their configurations. Norm conflicts are the result of a collision between two or more norms due to their specifications about what ought to be done. As norms describe what is expected during a contract, they use deontic meanings (permission, prohibition, and obligation) to state how parties must behave in each situation. When we have two norms such that it is impossible to comply with all their requirements, a norm conflict arises. In this case, norms are mutually exclusive since complying with one implies non-compliance with the other, thus, they cannot exist in a legal order [93, Chapter 1].

Elhag *et al.* [33] state that when an agent performs a certain act/behaviour it is called an 'individual case'. On the other hand, a general pattern of this act/behaviour specified in a norm is called a 'generic case' (*GC*). They define that a conflict between norms occurs when, given a set of norms '*S*', there is a norm x with a generic case GC_x, another norm y with a generic case GC_y, an implication $GC_x \rightarrow GC_y$, and the deontic meaning of GC_x is incompatible with that of GC_y. This is the simplest concept of norm conflict, where the execution of a generic case implies in the execution of a second one that has a conflicting deontic meaning. Based on the definitions of Alf Ross [88], Elhag *et al.* state other

three ways to identify a norm conflict. The first one is called ‘Total-Total inconsistency’, and describes the case when we have two identical generic cases. Thus, given two norms x and y, their generic cases are equal ($GC_x = GC_y$). In such case, both agent and behaviour are the same in the generic cases and the occasion for the application of one norm is also the occasion for the other. What makes them conflict is that their deontic meanings are different, which the authors call ‘contradictory’, when one norm is a prohibition and the other is a permission, and ‘contrary’, when one norm is a prohibition and the other an obligation. Example 2.4.1 illustrates this type of conflict. The second way is called ‘Total-Partial inconsistency’, in which one norm of a pair cannot be applied without conflicting with the other, whereas the other norm, when applied, conflicts only in some cases. Using the generic case formalisation, we can say that from two norms x and y, each describing a generic case GC_x and GC_y, the Total-Partial inconsistency is an implication $GC_x \rightarrow GC_y$. Example 2.4.2 illustrates this type of conflict, where norm 1 partially conflicts with norm 2 when people are under 18 years old and norm 2 totally conflicts with norm 1. The third way to identify a conflict between norms is called ’Intersection’, and it consists of a pair of norms that may conflict in specific cases defined in both norms. It does not mean that the application of one norm conflicts with the other, but that in certain conditions of both norms they may conflict. Example 2.4.3 illustrates this conflict, in which norm 1 and 2 do not conflict except in July due to the condition in norm 1.

Example 2.4.1. :

- The receiving State shall exempt diplomatic agents from indirect taxes.
- The receiving State shall not exempt diplomatic agents from indirect taxes.

Example 2.4.2. :

1. All citizens shall vote for president.
2. Only citizens older than 18 shall vote for president.

Example 2.4.3. :

1. The payments must be done on the 5th day of the month, except in July where it must be done on the 3rd day.
2. The Company shall pay its employees on the 5th day of every month.

Sadat-Akhavi [93, Chapter 1] describes norm conflicts by defining their types, *i.e.*, what is the structure of a conflict, and their causes, *i.e.*, how they arise. In his work, Sadat-Akhavi introduces six types of norm conflicts that he divides between three groups: ‘Partial’ and ‘Total’ Conflicts; ‘Bipartite’ and ‘Multipartite’ conflicts; and ‘Supervening’ and ‘Constant’.

The first group of conflicts ('Partial' and 'Total') deals with the norm description and conditions. 'Partial' conflicts occur when two norms conflict regarding either some addressees, some dates, some places, or conditions. In such type, norms conflict only in specific cases, otherwise, they do not conflict. Example 2.4.2 illustrates this conflict type. In the example, the pair of norms are not conflicting for all citizens older than 18, however, since norm 1 is generalising the vote for all citizens, those younger than 18 cannot comply with both norms, which creates the conflict. 'Total' conflicts are those in which norms conflict for all addressees, in all places, times, and conditions. This is the most common case, as Example 2.4.1 illustrates. In the example, all definitions for both norms are conflicting since they are literally the same. When this occurs, we have an agent complying with two norms that describe the same action to be performed but with different deontic meanings, i.e., permission $\times$ obligation, permission $\times$ prohibition, or obligation $\times$ prohibition.

The second group of conflict types ('Bipartite' and 'Multipartite') deals with the number of norms involved in a norm conflict. 'Bipartite' conflicts are those involving only two norms, as examples 2.4.2 and 2.4.1 illustrate. On the other hand, 'multipartite' conflicts occur with more than two norms. In this type of conflict, two norms are not conflicting by themselves, but the existence of a third norm creates a conflict between a number of previous norms. The conflict is then characterised by the impossibility of complying with all three norms at the same type. Example 3 illustrates this conflict type, as norms 1 and 2 are not conflicting by themselves since the application of one does not interfere in the other. However, norm 3 creates a conflict between norm 1 and 2 and itself as it states that the percentage must be the same when Company X is increasing prices. As we can see, if we have only two norms from this set no conflict arises, but when they are together we have a conflict.

Example 2.4.4. :

1. Company X must increase the price of product W in 3%.

2. Company X shall increase the price of product Y in 8%.

3. When increasing product prices, Company X must apply the same percentage to all products.

The third group of norm conflict types ('Supervening' and 'Constant') deals with the meaning of norms. A 'supervening' conflict occurs when norms conflict in a given period of time and not at other periods. In such case, the concept described in a norm may change over time, which makes the norm definition conflict depending on the definition it has in a certain moment. Example 2.4.5 illustrates this conflict type. In the example, norm 1 states that Company X must follow the internal regiment, whereas norm 2 states that it must follow the national regiment. Since both regiments may change over time norms may conflict in

some periods and not conflict in others. A ‘constant’ conflict is one in which the meaning of norms do not change over time, as illustrated in examples 2.4.1 and 2.4.2.

Example 2.4.5. :

1. To define the maximum number of days employees have to return the signed contract, Company X must follow the internal regiment.
2. To define the maximum number of days employees have to return the signed contract, Company X must follow the national regiment.

Sadat-Akhavi [93, Chapter 1] also 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 2.4.1 shows a norm conflict between an obligation and a prohibition.

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” [93, Chapter 1]. Example 2.4.6 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 the moment that one tries to comply with one norm and, at the same time, is non-compliant with the other.

Example 2.4.6. :

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

The third cause for a norm conflict is when one norm prohibits a ‘necessary pre-condition’ of another norm. Suppose two actions A and B, where B cannot be performed without A having been performed before. In this case, a norm conflict arises when one norm prohibits A and another norm allows B, as Example 2.4.7 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 2.4.7. :

1. Ships flying the flag of State A shall/may render assistance to any person in danger in area X.
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 2.4.8 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 2.4.8.

Example 2.4.8. :

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

2.4.1 Norm Conflict Resolution

As explained above, norm conflicts cause an inconsistency to contracts leading to potential problems in the executions of proposed actions. In order to avoid such problems, contract writers need to resolve these conflicts by deciding what to remove, edit, or add in the contract. In this section, we show some of the alternatives often used to resolve norm conflicts in contracts.

In public international law, the resolution of norm conflicts has four traditional base rules [81].

1. *Lex posterior derogat legi priori.* (later law derogates prior law);
2. *Lex specialis derogat legi generali.* (specific law derogates general law);
3. *Lex superior derogat legi inferiori.* (superior law derogates inferior law); and
4. *Lex posterior generalis non derogat legi priori specialis.* (later general law does not derogates prior specific law)

Rule 1 "*lex posterior derogat legi priori*" refers to two laws (we refer here as norms) with time differences. In this rule, when there is a conflict between two norms, the newer one overrules the older [81]. In this case, we can consider situations involving norms describing old fashion ways of performing a certain action that may have changed over the years. Rule 2 "*lex specialis derogat legi generali*" () refers to two norms with different ranges of effect. In this case, a conflict between norm is resolved by considering the norm with the specific regulation over the more general one. Consider the situation where a norm specifies the dress code of a company and a second one describes how a certain company section must dress. The one describing the specific section will prevail over the general one since the

section must have its own needs for a specific dress code. Rule 3 "*lex superior derogat legi inferiori*" describes the case where two norms belong to different hierarchies but describe the same action. This rule resolves norm conflicts by selecting the conflicting norm that is higher in the hierarchy as the correct one. Consider, for instance, federal and state laws, federal ones will be considered in case of conflict between them. Finally, rule 4 "*lex posterior generalis non derogat legi priori specialis*" describes an exception to rule 1. This exception deals with conflicts between younger and older norms, however, in this case, if the older norm is specific the younger cannot overrule it as in rule 1. Such rule ensures that specific rules must overrule general ones in all cases.

2.5 Norm representation

In order to process natural language norms and identify conflicts between them, we aim to use norm representations that comprise the norm information needed to identify conflicts. A norm can be represented in different ways in order to be processed and reasoned about by an algorithm. In this section, we present four approaches to represent norms. First, we introduce Sergot's approach [96] in Section 2.5.1, which uses Horn clauses to represent norms. Second, we describe Benthan's [14] approach in Section 2.5.2, which uses the logic of rights to represent a norm. Third, we explain Singh's approach in Section 2.5.3, which proposes defined structures for each norm element. Finally, we introduce two approaches based on the RuleML markup language in Section 2.5.4.

2.5.1 Sergot's approach

In 1988, Sergot [96] developed a logical structure to represent norms using Horn clauses. To show how the conversion of norms into this representation, consider his example about the acquisition of British Citizenship at birth.

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.

Analysing the clause, one can establish that 'after commencement' is related to a moment in time that occurs after or the on the date on which the Act comes into force. Such clause of Act can be formalised 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. Example 2.5.1 illustrates an example of Sergot's representation.

Example 2.5.1. :
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).

2.5.2 Bentham's approach

While Sergot's Horn clause representation is one of the possible approaches to represent norms, Bentham [14] introduces an alternative approach to represent legal rules using logic. His representation follows certain conventions:

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

(2) A_p, B_p are variables for action-propositions, where p 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. For instance, consider the 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. $\neg$(Obligation(p, A_p) & Obligation(p, $\neg A_p$))
> B2. Obligation(p, q, A_p) $\Rightarrow$ 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 logical consequence of Obligation(p, q, A_p), which means that if one is obliged to perform an action on behalf of another person, such action is obliged. These two axioms can add new concepts to the ones introduced by von Wright (see 2.1). Def. 1 states 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 states that the liberty to perform an action is the same as the absence of the obligation to avoid the action.

2.5.3 Singh's approach

Singh [97] proposes a norm representation applied to governance in organisations, which highlights aspects such as the subject, object, and context of a norm. The approach aims to govern sociotechnical systems providing a formal representation for each norm. A norm representation in this context, has five components: a subject, an object, a context, an antecedent, and a consequent.

- **Subject**: is the party on which the norm is focused and the one who has to perform the consequent;
- **Object**: the action defined by the norm;
- **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.

From this Singh's approach, we highlight two components that are important in norm conflict identification. 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. Example 2.5.2 illustrates the norm representation by defining the norm elements according to the Singh' definition.

Example 2.5.2. :

- 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 organisation in which company X is part of;
  (d) **Antecedent**: “In case of abandonment”;
  (e) **Consequent**: “pay every extra cost”;

2.5.4 RuleML approaches

RuleML [44] is an XML-based language created to represent different types of rules. The language is also capable of specifying queries and inferences using knowledge representations based on Web ontologies, which are devoted to knowledge representation. The main goal of Governatori work is to turn RuleML into a canonical Web language for rules, allowing the exchange of rules between major commercial and non-commercial rule systems on the Web. Such rule representation allows the exchange of rules from different systems and tools.

In its representation, RuleML follows a hierarchical structure that consists of two main categories: Reaction rules and Transformation rules. Reaction rules describe the invocation of actions triggered as a reaction to an event. Such category concerns the way a system may react to changes in the environment and communication. Transformation rules define how a transformation must occur. Thus, it consists of a condition that, if satisfied, transform an element described in the rule to another one. In the RuleML hierarchy, transformation rules break into Derivation rules. Derivation rules allow the derivation of the information from a rule. They are a special type of transformation rule that add a conclusion when certain conditions are met. From derivation rules, the RuleML hierarchy breaks into Facts and Queries. Facts are simple pieces of true information in the environment. Queries are a type of derivation rule that has only the conditions without a conclusion. They are used to obtain the set of tuples that satisfy the condition. Finally, from queries, we have Integrity Constraint. These constraints deal with inconsistencies after an event/condition is fulfilled. RuleML can be applied to contracts to represent norms within them employing such specific structure. Listing 2.1, extracted from [44], exemplifies how we can apply RuleML to norms.

Listing 2.1 – RuleML structure

```
<lmp 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>**: connects 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>**: defines individual variables to be instantiated by ground values; and
- **<Obligation>**: tag that indicates an obligation referent to the ground values (Good and Price) of a purchase order.

LegalRuleML [8] 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, such as `<TimeInstants>`, `<TemporalCharacteristics>`, `<Agents>`, and `<Authorities>`. `<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 [8]) shows the use of <TimeInstans> tag defining the temporal register of a rule. Listing 2.3 exemplifies the tags <Agents> and <Authority>.Here, hhe <Agent> tag represents the agents associated with the rule, defined by an attribute 'key', while 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>
```

2.6 Natural Language Processing

Natural language processing (NLP) is a branch of artificial intelligence that studies methods to make computers understand and manipulate natural language [22]. Natural language is the language used by humans to communicate with each other. The first attempt to

use computers to understand and manipulate natural language is dating back to the 1940's when researchers tried to create a machine translation approach [65]. The process of building computer programs able to understand and manipulate natural language has three main problems: (1) a word level understanding, responsible for determine the morphological structure and natural of a word, such as finding its part-of-speech and meaning; (2) a sentence level understanding, regarding the order of words, grammar, and the sentence meaning; (3) a document level understanding that consists of a context and environment processing [22].

At a word level understanding, the part-of-speech of words is a basic information that gives meaning to the raw text. The part-of-speech (POS) tagging consists of the attribution of the part-of-speech for each word in a text [76, Chapter 4]. A POS describes the morphological function of a word in the text and can have several types, such as preposition, noun, pronoun, verb, and adjective. Example 2.6.1 shows a sentence in which each word is followed by a slash and its respective part-of-speech in bold.

Example 2.6.1. :

- Company X/**noun** must/**modal_verb** buy/**verb** products/**noun** until/**preposition** the/**determiner** next/**adjective** week/**noun**.

Using POS tags, we can extract more information from the text, such as named entities and syntax trees. Named entity recognition is the task of identifying the entities in a given text, such as proper names, locations, organisation, money, and time [78]. From the POS tags, several techniques involving both rule-based and machine learning can be employed to identify the named entities. Example 2.6.2 illustrates the same sentence from Example 2.6.1 with the named entities annotated.

Example 2.6.2. :

- Company X/**organisation** must buy products until the next week/**time**.

Syntax trees, also known as parse trees, are ordered and rooted trees that represent the syntactic structure of a sentence [57, Chapter 11]. Syntax trees are very useful for grammar checking and for the creation of a representation for semantic analysis. They are commonly used for information extraction and question answering tasks since from their structure one can extract and divide information from a sentence. The structural elements of a syntax tree are: S (sentence), NP (noun phrase), VP (verb phrase), and PP (preposition phrase). Every tree starts from a root S going down through noun and verb phrases until the leaves often containing the part-of-speech of the words and the words themselves. Figure 2.2 shows an example of a syntax tree generated over a sentence (I built a fence). As we can see, the root of the syntax tree is the sentence identifier and as we go down, noun and verb phrases are created from the relations between the part-of-speech of the words.

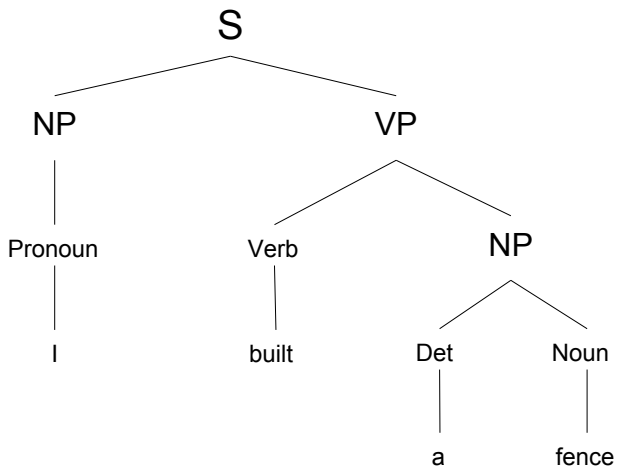


Figure 2.2 – Example of a syntax tree

Such trees represent the grammatical structure of text, and can be used to derive a logical representation of the underlying language [9], that can be ultimately be used to reason about a deontic logic representation.

3. LEARNING MODELS

Learning models are a class of models capable of learning from examples or simply using data information. When successfully trained, they can generalise for new (and unseen) examples, which makes it suitable for real-life applications. Using such models, we can avoid using an overly complex set of rules to deal with each specific case of problems. Both machine learning and deep learning have specific applications we can employ to classify and extract features from data. In what follows, we review the main types of learning models and input encoding schemes we use in our work.

3.1 Machine Learning

Machine learning is a field of artificial intelligence that deals with models designed to improve automatically through experience [75, Chapter 1]. These models, when exposed to a large amount of data, can generalise to new examples, which makes them versatile to different types of problems. This adaptive characteristic is the result of continuous update of weights during the training phase. Algorithms of this class use the weights to modify the input and obtain as output a class or value that classifies this input.

Classes of Machine Learning

Machine learning has four main classes of algorithms: supervised, semi-supervised, unsupervised, and reinforcement learning. Supervised learning is the type of learning that uses annotated data to train a model [77, Chapter 1]. During the training, the algorithm attributes a class to each example that is then verified against a gold standard base. In case of a wrong class, the algorithm updates its weights in order to correct the error. Supervised learning is often used for two types of tasks, namely, classification and prediction (also called regression). In classification, the learning algorithm tries to identify among predefined classes which one must be addressed to each input. For example, the classification of flowers by their types, which has a limited number of elements. When predicting, supervised algorithms have as output a continuous value that tries to approximate the result for a given input. We use prediction to identify what is the expected output from an input, such as the price of a house given the number of rooms and its size or when we need an approximation of how much the customer will spend in a sale given his historic. Semi-supervised algorithms deal with problems that need a large amount of labelled data but they have only part of a dataset labelled [69, Chapter 2]. In such cases, it is common to train a model using the labelled data and use the trained model to classify unlabelled data. Finally, one can use the now labelled dataset to train the model. In unsupervised learning, unlike from supervised

learning, there is no labelled data and the algorithm uses only the features from data [55, Chapter 10]. With such features, it can cluster data according to the similarities between them. This type of learning is useful when we need to identify patterns and cluster elements in a database [68]. Finally, reinforcement learning is the type of learning that uses the rewards observed in a defined environment to learn an optimal policy [92, Chapter 1]. Agents using reinforcement learning can adapt to environments by exploring the possible states and observing what each action yields as rewards. Rewards can be both positive and negative and using it as parameter the agent can find the optimal actions giving a certain state.

Machine Learning Models

There are numerous learning models for each class of machine learning and we focus on three widely used models that are directly applicable to our work. These models are often used for both classification and prediction in supervised and semi-supervised learning.

Support vector machines (SVM) is a learning model that maps the inputs to high dimensional feature space and creates a linear separation between them [25]. This linear separation is a hyperplane that separates the data according to the classes. Optimal hyperplanes have the maximum margin of separation between two classes [49]. To create the feature space where the input vectors are placed, SVM performs a nonlinear map on the data that converts it to a dot product space. Figure 3.1 illustrates how SVM creates the separation of data. In the figure, we have two types of data (green and red) and the division is made by trying to find the optimal margin that maximises the separation between the hyperplane and data on each side of it.

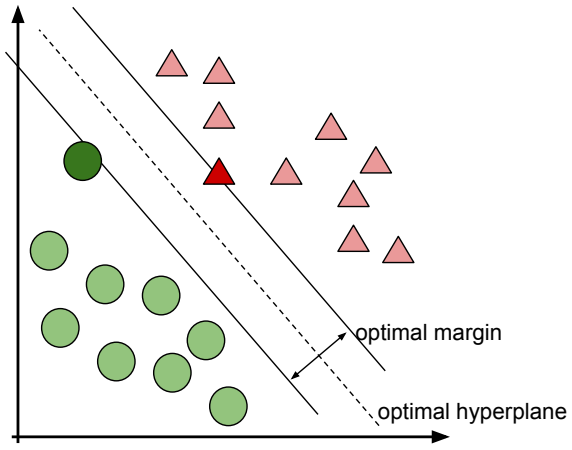


Figure 3.1 – Example of a hyperplane in a two-dimensional space. Image based on the one in Cortes and Vapnik work [25]

Artificial neural networks (ANNs) were created inspired by the mechanism employed by the human brain to process information [35, Chapter 1]. ANNs have an architecture composed by layers of fully-connected “neurones” [92, Chapter 18]. An ANN can

have three types of layers: input, hidden, and output layer [48, Chapter 1]. The input layer consists of values corresponding to the data features, i.e., the information ANN uses to approximate the input the expected output. Thus, the size of this layer is limited to the number of features. An output layer consists of the resulting values of the network. These values define what class the input belongs to or the value it represents. The number of neurones in the output layer depends on the task it is dealing. If it is a classification task, it will depend on the number of classes; if it is a regression it will depend on the number of values we are predicting. Any layer that is not an input or output layer is a hidden layer. In this layer, neurones receive the values from the previous layer and process them in order to extract information that helps in the approximation to the expected output value. The number of hidden layer, as well as the number of neurones, are variable depending on the complexity of the problem.Usually, more complex problems require a deeper network with many neurones by layer. Such bigger structure allows the network to extract different levels of information from the input data. Figure 3.2 illustrates the structure of an ANN. The simplest architecture an ANN can have is called perceptron, and it has only one neurone [87]. A perceptron can only classify linearly separable problems due to the limitation of using a single neurone that can only separate data in half. A formal definition of an ANN describes neurones as nodes or units that have connections named links. A link of two nodes i and j propagates an activation a of node i to node j. Associated with a link between nodes there is a numeric weight $w_{i,j}$. The general way to calculate the activation in a node is to apply an activation function (g) over the sum of the product between the activation from previous nodes and the weights in the links. Equation 3.1 shows how we calculate the activation a_j in node j considering n as the number of previous nodes connected to node j.

$$a_j = g(\sum_{i=0}^{n} w_{i,j} a_i) \tag{3.1}$$

Activation functions convert input signals in a node into output signals that next nodes use to propagate the information through the network. These functions allow neural networks to understand complex data patterns by adding non-linearity to the result of the linear function in a node. Non-linear functions have more than one degree and create curvatures that allow us to classify non-linear problems. The most common activation functions are Sigmoid, Tanh, ReL, and Softmax.

Sigmoid (or logistic) function converts input values into values between 0 and 1 using Equation 3.2. The function has an “S”-shaped curve centred in 0.5 and approaches 1 when the input is approaching $+\infty$ and approaches 0 when the input is approaching $-\infty$. For instance, consider the vector $x = [0.1, 0.2, 2.4, -1.7]$, when we apply the sigmoid function over the vector, we obtain $sigmoid(x) = [0.52, 0.54, 0.91, 0.15]$.

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{3.2}$$

Tanh (or Hyperbolic Tangent) function converts input values into values between -1 and 1 using Equation 3.3. It also has an "S"-shaped curve, although, unlike sigmoid, tahn is centred in 0. For instance, consider the vector $x = [0.1, 0.2, 2.4, -1.7]$, when we apply the tahn function over the vector, we obtain $tahn(x) = [0.09, 0.19, 0.98, -0.93]$.

$$tahn(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \tag{3.3}$$

ReL (Rectifier Liner or ReLU when we apply it to a unity) function preserves only the positive part of input values. Using Equation 3.4, it preserves x if it is greater than 0, otherwise returns 0. For instance, consider the vector $x = [0.1, 0.2, 2.4, -1.7]$, when we apply the ReL function over the vector, we obtain $ReL(x) = [0.1, 0.2, 2.4, 0]$.

$$rel(x) = max(0, x) \tag{3.4}$$

Finally, the softmax activation function is used to create a probability distribution over a vector of real numbers using Equation 3.5. In the Equation, consider that x is a vector with n elements, where x_i is the i^{th} element of x. This function is often used for learning models to express the probability of classes on classification tasks. For instance, consider the vector $x = [0.1, 0.2, 2.4, -1.7]$, when we apply the ReL function over the vector, we obtain $softmax(x) = [0.09, 0.09, 0.81, 0.01]$. The sum of the values in the resulting vector is always 1.0.

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} \tag{3.5}$$

The training process of an ANN is divided into two steps: feedforward and backpropagation. In the feedforward step, a neural network passes the data in the input layer through its layers up to the output layer. As data walks through the layers, nodes will activate differently transferring information about the data. This process generates an output that refers to the input class or value, depending on the learning type. The output is then compared against the expected output and, in case of a difference in their values, the error is propagated through the layers in a process called backpropagation. In the backpropagation step, the ANN updates its weights in each link between the nodes with the derivatives of the successor nodes [50].

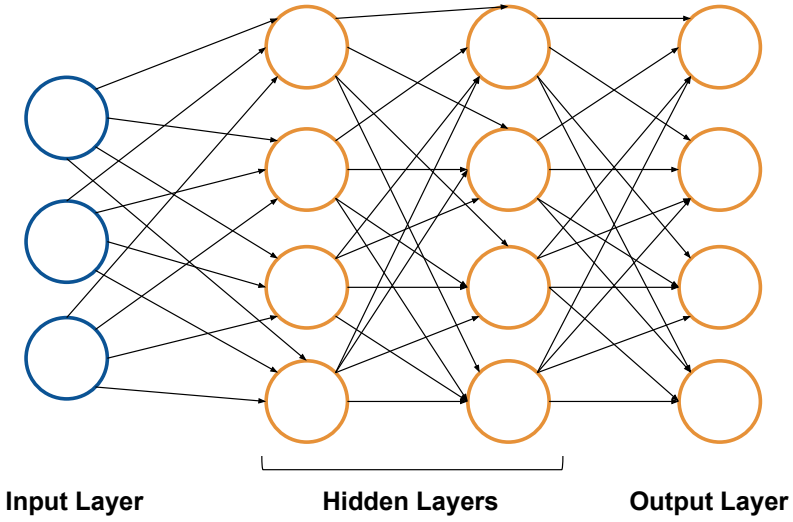


Figure 3.2 – Example of the architecture of a neural network with two hidden layers

3.2 Deep Learning Models

Deep learning is a branch of machine learning that uses hierarchical architectures to learn high-level information from the data [45]. Unlike traditional machine learning models that require a process of feature selection, deep learning models have the ability to hierarchically identify which features contain the most relevant information to differentiate input data. The simplest way to understand a Deep Neural Network (DNN) architecture is to think in a regular ANN with a large number of hidden layers following a hierarchy. The premise for the use of such number of layers is that as layers become deeper richer information can be exploited from the input, which helps in the classification [61]. The most remarkable characteristic of DNNs is that they can extract features from raw data, *i.e.*, without the need of previous feature selection. In machine learning models, one needs to perform a selection of what features from the input may result in the best-trained model. With their hierarchical structure, DNNs can extract high-level features by composing low-level ones extracted in previous layers [61]. Similar to the previous section, in this one we describe three deep learning models that have a degree of connection with our work. We start by talking about autoencoders and how they create a representation of the input data. Then, we talk about convolution neural networks, and we describe how they automatically extract information from data to improve the classification. Finally, we describe how recurrent neural networks and their variations work to deal with sequential data.

Autoencoders are a class of ANNs that you train in an unsupervised way. They are a combination of two functions, namely, encoder and decoder. Figure 3.3 illustrates the architecture of an autoencoder. The encoder turns the input data into a different representation, while the decoder turns this new representation into the original data [43, Chapter 14].

We train autoencoders to create a representation of data that preserves enough information so that the decoder part of the network can re-create the original data from a fixed-sized latent representation. Latent representation is a way of representing observed data (*e.g.*, text, image, and constants) from an inferred process by a mathematical model. There are two main uses for autoencoders, the first one is the use of latent representation as input for other learning models that can classify data. In this case, since the latent representation consists of significant features from raw data, it can facilitate the learning process. The second use is for dimensionality reduction. In some learning problems, input features can be too numerous. For instance, consider training an ANN to classify the flower type of flower images in RGB with 224x224 pixels each. Considering the three colour channels of RGB images, the input for such ANN would be $224 \times 224 \times 3$, which results in 150,528 input features. Such number of features, if not supported by a huge number of examples, would lead to a poor model or would make the model not to converge during training. This occurs as data sparsity becomes a problem to obtain statistical significance for the learning process, which is known as the curse of dimensionality [38]. The reduction of the number of features while preserving a smaller number of relevant ones can be done by an autoencoder. In order to reduce the input dimension, the architecture of the autoencoder is modified to have a central layer smaller than the input layer.

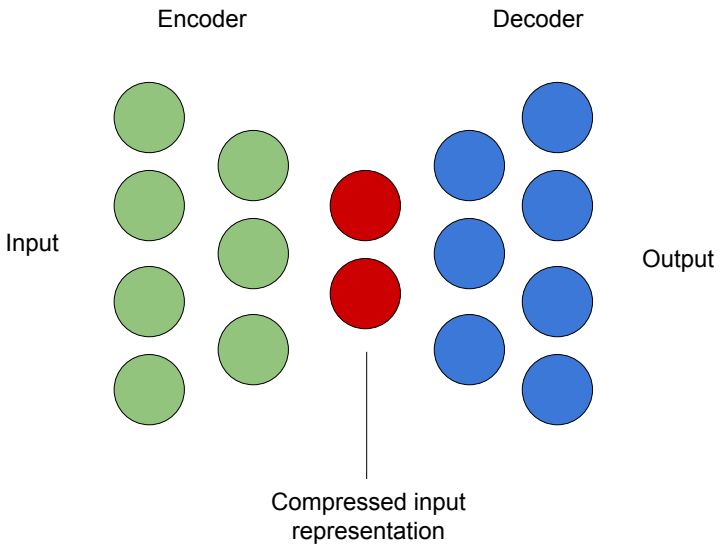


Figure 3.3 – Example of an autoencoder

Convolution Neural Networks (CNNs) are widely used to process images and videos as they are designed to receive as input multidimensional arrays, bidimensional for images and tridimensional for videos [61]. The architecture of a standard CNN has two main parts. In the first one, CNNs have two types of layers, namely, convolutional and pooling layers. Convolutional layers apply a series of filters over the input generating feature maps. A filter consists of a set of weights that modify the input. Each filter goes through the entire input multiplying its weights by the input values and the result is passed to a non-linear activation

function, such as ReL and Tanh. The feature maps generated by the filters highlight different parts of the input. Figure 3.4 illustrates a convolution, in which a feature map is created by the multiplication of the input image and the filter. Pooling layers are often employed after a convolutional layer. They reduce the dimensionality of resulting features from convolutional layers. These layers have a single filter without weights that goes through the input aiming to down-sample the size of the matrix, much in the same way resizing an image reduces its dimensions. They can be either a max pooling or a mean pooling, where the former outputs the highest value among the ones in the kernel size and the later outputs the mean value among the ones in the kernel. Figure 3.5 illustrates a max pooling over the matrix resulting in a new resized matrix. The second part of a CNN architecture is a series of fully-connected neural networks (ANNs). This part has the aim of classifying the resulting features obtained from the convolution and pooling layers. Figure 3.6 illustrates a complete CNN from the input image to the classes. In the figure, we can notice how convolution layers create feature maps as they increase the third dimension of the input. On the other hand, pooling layers decrease the size of feature maps.

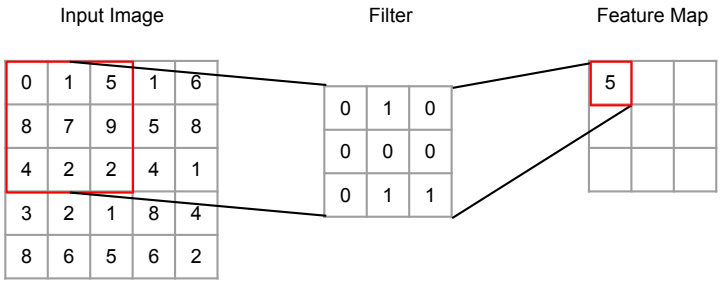


Figure 3.4 – Convolution example

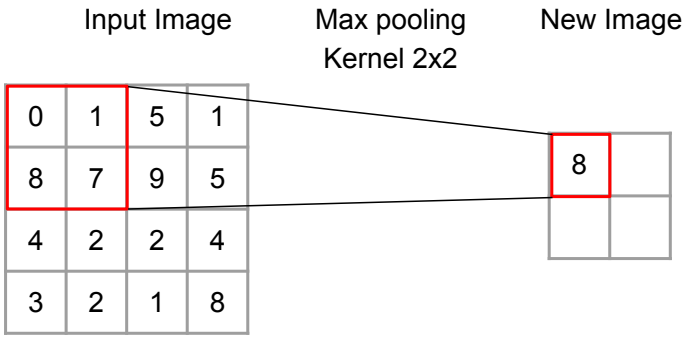


Figure 3.5 – Pooling example

Recurrent neural networks (RNNs) are a type of deep neural network that contains feedback connections between their nodes. This feedback allows this type of network to

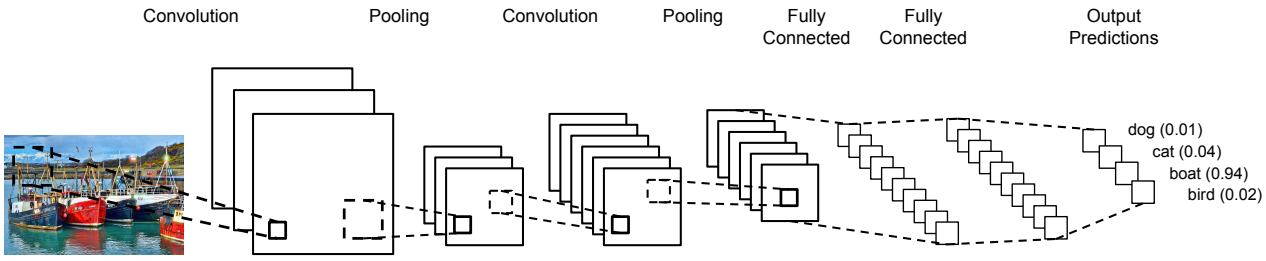


Figure 3.6 – Example of CNN. Image inspired by the one in the WILDML website http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

deal with temporal dynamics, which makes them useful for problems with dependent sequence, such as natural language processing tasks. An RNN processes an input sequence one element at a time and keeps a history with information about previous inputs in its hidden units [61]. The architecture of an RNN may vary from fully interconnected to partially connected neural nets. In fully interconnected, the input of each node is the output of all others and the network input can be any node. In partially connected networks, nodes follow a similar structure of an ANN but some of them have more than one input and output [54, Chapter 1]. Figure 3.7 illustrates both architectures where (a) is a fully interconnected network with every node connected to the others; and (b) is a partially connected network where only nodes C1 and C2 receive feedback from the second layer. When dealing with problems that have some sort of dependency between the input elements, RNNs can use the information from previous inputs to classify the subsequent ones. This is common in NLP tasks, such as text generation [100] and text annotation. Figure 3.8 illustrates how the recurrent neural network loop occurs, when unwrapped, we can see that nodes are connected as the output of one is the input to the next one.

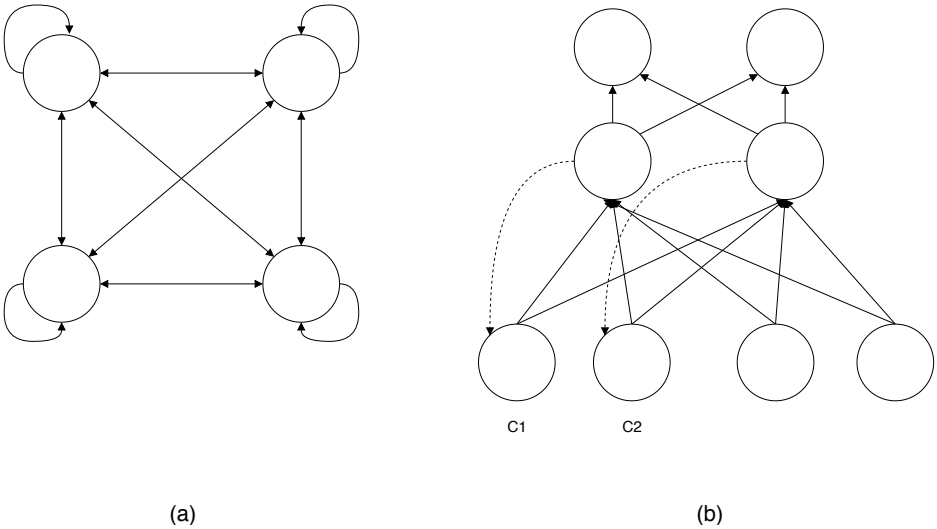


Figure 3.7 – Recurrent Neural Network architectures. (a) A Fully interconnected architecture; (b) A partially connected architecture

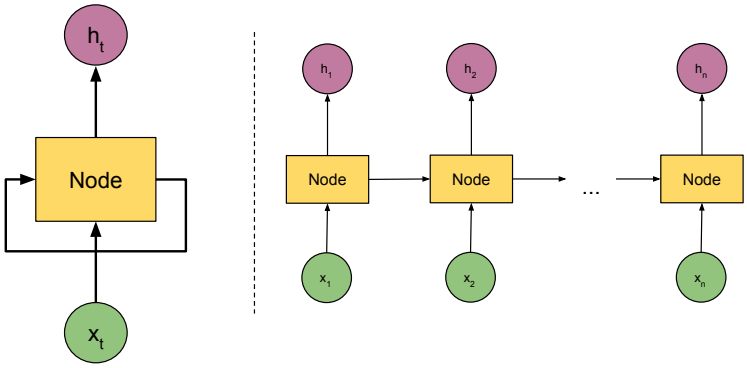


Figure 3.8 – Example of a wrapped RNN architecture on the left and an unwrapped one on the right

When dealing with long sequences, RNNs can fail to keep previous information for a long time since they give place to new ones. To solve such problem, Hochreiter and Schmidhuber [52] introduced the Long Short-Term Memory (LSTM). LSTM is a type of RNN that manages the internal information with a series of memory mechanisms. The main difference between a standard RNN and an LSTM is that the latter has a cell state that goes through the entire network. In this cell, LSTM adds and deletes new information according to the importance of it. Such mechanism ensures information to travel through the network, which helps on further decisions considering a larger context. Standard recurrent neural networks apply a simple activation in each node that assembles the information from previous nodes and the current input. Equation 3.6[1] shows how this activation occurs, where W_N represent the weights from node N and b_N the bias value. N_{t-1}, x_t are, respectively, the output from the previous node $(t-1)$ and the input of the current node. To maintain information for long series, LSTM modifies N_t to be a way to transport information all over the network. This way, it just adds new information to N_t that will be delivered to all next nodes. Additionally, LSTM applies four gates to control information that will be transmitted to the following nodes. The first gate defines what will be forgotten from the input. Equation 3.7 shows how the gate works. Similar to Equation 3.6, F_t has its own weights W_F, bias value b_F, and process the output from previous nodes h_{t-1} and current input x_t. The difference here is that we are applying a sigmoid activation function (σ) that returns values between 0 and 1. This represents how much of the previous information will be considered in the final activation as this value multiplies the input from N_{t-1}. The second gate is a sigmoid activation that defines what will be updated from the result of the third gate. Equation 3.8 shows how the activation occurs similarly as the first gate. As a third gate, we have the recurrent neural network information process already described in Equation 3.6, which we will refer as N'_t. We multiply the output from the second and third gates and add to N_t. Equation 3.9 shows how the node's activation is built using the information from the gates. Finally, in order

[1] Equation formalisation were inspired by the ones explained in https://colah.github.io/posts/2015-08-Understanding-LSTMs/

to create the output of the node, the fourth gate is a sigmoid that defines what will be used as output from the information in N_t. Equation 3.10 shows how the fourth gate obtains its value. The result of the forth gate is then multiplied by the tahn activation over the N_t, as Equation 3.11 shows. Figure 3.9 illustrates a node in the LSTM with the gates as “sigmoid” and “tahn” squares.

$$N_t = tahn(W_N \times [N_{t-1}, x_t] + b_N) \tag{3.6}$$

$$F_t = \sigma(W_F \times [h_{t-1}, x_t] + b_F) \tag{3.7}$$

$$U_t = \sigma(W_U \times [h_{t-1}, x_t] + b_U) \tag{3.8}$$

$$N_t = F_t \times N_{t-1} + U_t \times N'_t \tag{3.9}$$

$$O_t = \sigma(W_O \times [h_{t-1}, x_t] + b_O) \tag{3.10}$$

$$h_t = O_t \times tahn(N_t) \tag{3.11}$$

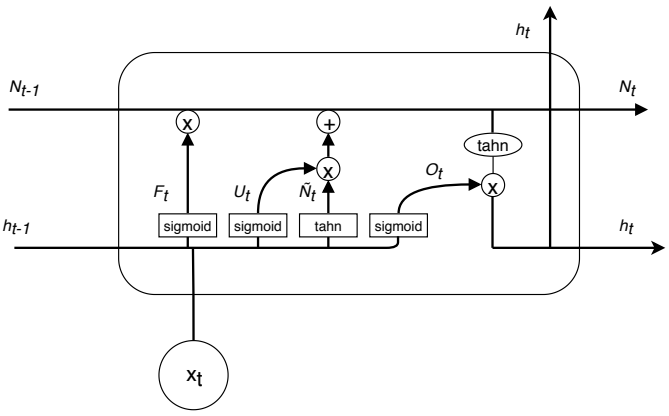


Figure 3.9 – Architecture of an LSTM node

3.3 Text Representation for Learning Algorithms

Although the use of classic NLP can achieve good results for several tasks, it relies on semi-automatic or manual pre-processing features and text representation. Dealing with complex scenarios in which text representation is not trivial, such as norm representation,

using classic NLP representations may not be enough. On the other hand, current text representations often convert natural language text into dense vectors that preserve both the syntactic and semantic meaning of texts. In this section, we introduce six approaches to represent natural language texts.

The first and most popular approach is the term-document matrix or bag-of-words. In this approach, we count the occurrences of each word in a document and create a matrix where the lines are words and the columns are documents [57, Chapter 4]. Thus, we can compare what words are in each document and differentiate each document by its words. Bag-of words is the most basic text representation for most natural language tasks, such as document classification and sentiment analysis. In order to demonstrate how bag-of-words work, consider the following documents.

1. The cat is on the bed.
2. When it rains, it pours.
3. A cat is sleeping on the bed.

In Table 3.1, we illustrate an example of bag-of-words applied to these documents. The columns of Document 1, 2, and 3 represent the bag-of-words for each document, i.e., the occurrences of each word in the rows. Using the information of Table 3.1 we can see for example that Document 2 is different from Documents 1 and 3 since words n shared by both documents have no occurrence in Document 2. As we can see, this type of text representation incorporates neither word order nor word semantics since the resulting vector has only the information of word occurrence.

Word	Document 1	Document 2	Document 3
the	2	0	1
cat	1	0	1
is	1	0	1
on	1	0	1
bed	1	0	1
a	0	0	1
sleeping	0	0	1
when	0	1	0
it	0	2	0
rains	0	1	0
pours	0	1	0

Table 3.1 – Bag-of-words matrix applied to three documents

The second approach to represent natural language is the `tf-idf` (Term Frequency - Inverse Document Frequency). In this approach, there is an effort to assign weights to words based on the frequency they appear in all documents. To calculate the value of each word, we use the frequency of the word in the document times the inverse frequency of

the same word in the other documents [57, Chapter 6]. For instance, consider the same set of documents we used in the bag-of-words example. The first step is to calculate the term frequency for each term in each document, which we did in Table 3.1. Equation 3.12) computes term frequency of a document *di* using the following information: di_{term} is the number of times *term* appears in *di*, and T_{di} the total number of terms in *di*. Given the term frequency, we need to calculate the inverse document frequency. This measure takes into consideration the presence of the term in all documents. Equation 3.13 computes the calculation. Table 3.2 shows the `idf` measure for each word in our example. As we can see, words In the equation, consider $|D|$ as the total number of documents and D_{term} the number of documents in which *term* appears. Finally, to obtain the `tf-idf`, we simply multiply `tf` and `idf` (see Equation 3.14). This approach weights the frequency of the terms within a single document by its frequency throughout the corpus. By doing this, it decreases the impact of words that occur a lot in all texts as they presumably convey less information than words that are rare in the corpus. Similarly, it increases the impact of words that are rare in the corpus, but frequent in a document should be a distinctive mark of such documents.

$$\texttt{tf} = \frac{di_{term}}{T_{di}} \tag{3.12}$$

$$\texttt{idf} = log_e(\frac{|D|}{D_{term}}) \tag{3.13}$$

$$\texttt{tf-idf} = \texttt{tf} * \texttt{idf} \tag{3.14}$$

Word	$\|D\|/D_{term}$	`idf`
the	1.5	0.17
cat	1.5	0.17
is	1.5	0.17
on	1.5	0.17
bed	1.5	0.17
a	3	0.47
sleeping	3	0.47
when	3	0.47
it	3	0.47
rains	3	0.47
pours	3	0.47

Table 3.2 – Computation of `idf` for each term

While convolutional neural networks are often used for processing images, they are suitable for processing any information in matrix form. Convolution neural networks have a unique characteristic to extract features from pixels using a series of filters. However, we can also use CNNs to process text by converting the text to a matrix-like representation that allows the convolution to extract features from it. Two existing approaches do this conversion to

	tf-idf		
Word	Document 1	Document 2	Document 3
the	0.34	0	0.17
cat	0.17	0	0.17
is	0.17	0	0.17
on	0.17	0	0.17
bed	0.17	0	0.17
a	0	0	0.47
sleeping	0	0	0.47
when	0	0.47	0
it	0	0.94	0
rains	0	0.47	0
pours	0	0.47	0

Table 3.3 – `tf-idf` matrix applied to three documents

a sentence representation and use them to classify text for sentiment analysis and syntactic categorisation. The first sentence representation, created by Zhang and LeCun [107], uses a CNN to deal with natural language processing problems. Their approach aims to, among other tasks, classify the sentiment (positive, negative, and neutral) of product reviews from Amazon. They translate a sentence into a matrix by creating a matrix representation with the text characters as lines and the alphabet as columns. Thus, given a cell $\{i, j\}$, they assign 1 when the *ith* character is equal to the *jth*, otherwise, they assign 0. Figure 3.10 illustrates their sentence representation using as an example a sentence that begins with 'above'. The resulting matrix has 1 in cells containing the same letter in both line and column (such as cells $1, 1$ and $2, 2$) and 0 otherwise. They use this binary matrix as input and during the train, the CNN extracts features relating the position of characters to the class belonging to the sentence.

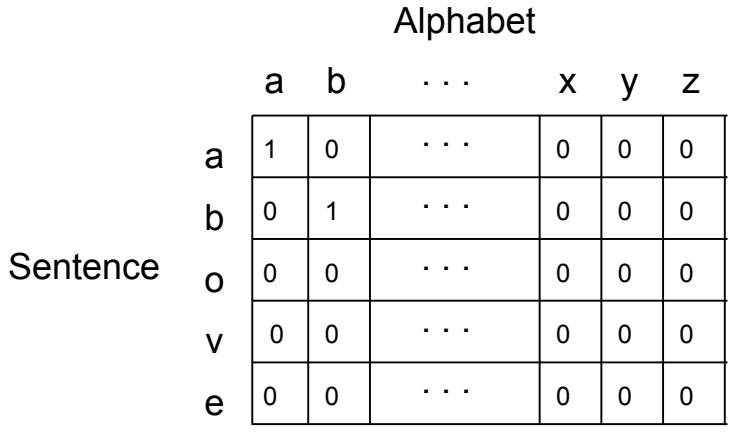


Figure 3.10 – Sentence representation by Zhang and LeCun

The second matrix-like representation of sentences is from Kim [58], which uses it to classify sentences in different natural language processing problems. Here, the represen-

tation is a matrix in which the lines are the words of a sentence and columns are the latent representations of each word. In this case, latent representations are words converted into vectors of real numbers, which may have a variable size and carries semantic information. In Kim's approach, the resulting matrix is a group of latent representation lines. Figure 3.11 illustrates this sentence representation as a matrix.

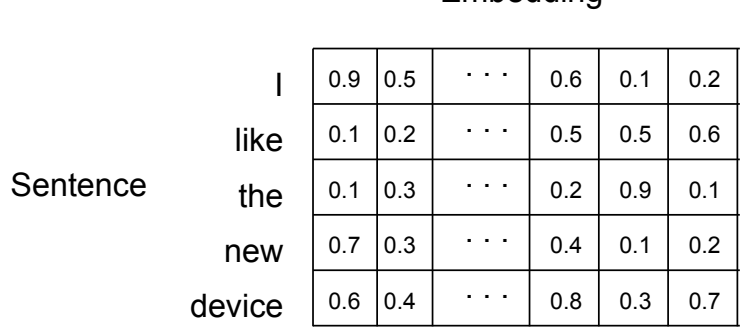


Figure 3.11 – Sentence representation by Kim

There has been much effort in recent years by researchers on Natural Language Processing to map words into low dimensional space in order to capture their lexical and semantic properties [72, 71, 82]. In order to obtain word embedding methods that convert words into n-dimensional vector representations, or `word2vec`, we can use the internal representations of neural network models, such as feed-forward models [13], and recurrent neural network (RNN) models [72]. Although much effort has been put towards training neural models for word embeddings before [13], the idea of creating shallow methods that are cheaper to train and take the advantage of much larger datasets has become ubiquitous building blocks of a majority of current state-of-the-art NLP applications [59, 108].

Continuous Bag-of-Words (CBOW) [71] is an efficient algorithm to estimate distributed word representations (`word2vec`). The main assumption of this algorithm is that words used in similar contexts have similar meaning [46]. Thus, CBOW is an artificial neural network architecture composed of three layers (input, hidden layer and output) that learns word representations by predicting a word according to its context. In order to predict the target word, the algorithm scans the corpus sequentially with a fixed-sized window, collecting a central target word and a few neighbouring words called context. The context of a target word is the average of the vectors associated with the target word inside the window, as illustrated in Figure 3.12. In the figure, consider $w(t)$ as the target word and the input vectors $w(t + | - i)$ represent the context words around $w(t)$. As each word w is represented by a vector v_w and context (C) is represented by the average of word vectors $u_{w'}$ of the context words w', the scoring function is then computed as Equation 3.15 shows.

$$s(w, C) = \frac{1}{|C|} \sum_{w' \in C} u_{w'}^{\top} v_w \tag{3.15}$$

More precisely, given a sequence containing K words, w_1, w_2, ..., w_K, the algorithm maximises the log probability of a target word given the vectors of the context words as Equation 3.16 shows.

$$\sum_{t=1}^{K} log\ p(w_t|C_t) \tag{3.16}$$

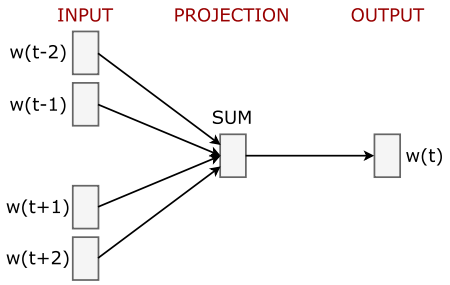


Figure 3.12 – The CBOW architecture to predict the target word based on the context.

In practice, Equation 3.16 means minimising the dot product between the target word embedding and the context words. Such optimisation is often performed via backpropagation [91] using Stochastic Gradient Descent (SGD) where each sample is a window and a loss function is defined between the target and the context vectors. Due to the size of the vocabulary, the softmax function over the scores of contexts and words should not be used for calculating the conditional probability in Equation 3.16, and instead, compute $p(w_t|C_t)$ using independent binary classifiers over words [60]. To improve generality, CBOW performs random word sub-sampling as a regularisation, deciding to discard each token with a certain probability.

Although this model does not know morphology, syntax or semantics, it can induce word representations with syntactic and semantic properties. As demonstrated by Mikolov *et al.* [73], word embeddings generated by RNN encode not only attributional similarity between words, but also linguistic similarities between pairs of words. Linguistic similarities, also referred to as relational similarities by Turney [101], can capture gender relations such as *man:woman*, *king:queen*, past tense relations such as *capture:captured*, and singular/plural relation such as *car:cars*. As such relations are observed as vector offsets between pairs of words sharing a particular relationship (*man - woman* $\approx$ *king - queen*), one could answer the analogy question "*Man is to Woman as King is to* $\langle word \rangle$", where $\langle word \rangle$ is unknown, by simply performing vector arithmetic. Thus, one should find the embedding vectors v_{King}, v_{Man}, v_{Woman} and compute $y = v_{\text{King}} - v_{\text{Man}} + v_{\text{Woman}}$, where y is expected to be the continuous space representation of the best answer. As no word might exist in the exact position y, one then has to calculate the distance between the existing embedding vectors to find the closest one. As a result, the embedding vector that is closest to y should represent the word *Queen*. Figure 3.13 illustrates the relation between word embeddings with the explained relation. In the figure, blue dots represent masculine words, whereas purple dots represent feminine

words. As a way to explain the manipulation of word embeddings, consider the points in Figure 3.13 as the following vectors: *king* = [2, 1, 2], *man* = [2, 1, 0], *woman* = [1, 2, 0], and *queen* = [1, 2, 2]. When we subtract *man* from *king*, we obtain *monarch* = [2, 1, 2] − [2, 1, 0] = [0, 0, 2], which means removing the sex property in majesty. Finally, when we add *woman* to *monarch*, we obtain the vector [0, 0, 2] + [1, 2, 0] = [1, 2, 2], which corresponds to the *queen* vector.

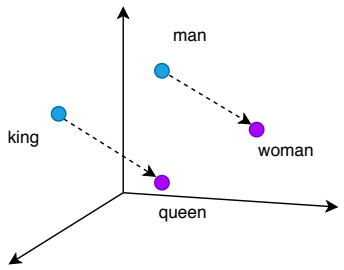


Figure 3.13 – Relation Male-Female between word embeddings

Although word embeddings work very well to capture linguistic regularities, they cannot capture the semantic between long sequences of words, such as paragraphs, sentences or documents [60]. Learning such longer representation is central to many natural language applications. Many recent approaches try to capture the semantic and syntactic properties of sentences by creating embeddings [60, 56, 80]. Among these approaches, `sent2vec` [80] is a state-of-the-art algorithm to represent sentences as embeddings. This algorithm uses an efficient unsupervised objective to train distributed representations of sentences. It is an extension of the CBOW [71] that learns the representation of sentences instead of words. According to Pagliardini *et al.* [80], the method can be interpreted as a natural extension of the word contexts from CBOW to a larger sentence context, with the sentence words being specifically optimised towards additive combination over the sentence, by means of the unsupervised objective function.

More precisely, the algorithm learns source v_w and target u_w embeddings for each word w in the vocabulary ($|\mathcal{V}|$) and embedding dimension h. The embedding generated for a sentence is defined as the average of the source word embeddings of its constituent words. Unlike CBOW, `sent2vec` does not consider only unigrams and uses n-grams to create sentence embeddings. The embedding generated for a sentence is defined as the average of the source word embeddings of its constituent words. Thus, the sentence embedding v_S for the sentence S is computed using Equation 3.17, where $R(S)$ is the list of n-grams (including unigrams) in the sentence.

$$v_S = \frac{1}{|R(S)|} \sum_{w \in R(S)} v_w \tag{3.17}$$

Similarly to `word2vec`, `sent2vec` improves generality by performing random subsampling by deleting random words once all the n-grams have been extracted. Finally, fol-

lowing Mikolov *et al.* [72], missing words are predicted from the context by using a softmax output approximated by negative sampling, which is a technique to update only a small percentage of the model's weights per training example.

As observed by Milokov *et al.* [73] and Levy and Goldberg [62], word embeddings are surprisingly good at capturing syntactic and semantic regularities in language. These regularities can be represented as vector offsets so that in the embedding space, all pairs of words that share a certain relation are related by the same offset. Using the previous example, the "monarch" relation can be learned by performing the vector operation $v_{\text{Monarch}} = v_{\text{King}} - v_{\text{Man}}$. Adding the embedding of the word *Woman* to the "monarch" embedding $y = v_{\text{Monarch}} + v_{\text{Woman}}$ should result in a vector close to the embedding representing *Queen*. Hence, given two pairs of words that share the same semantic relation $v_a : v_a^*, v_b : v_b^*$, the relation between those two words can be represented as

$$v_a^* - v_a \approx v_b^* - v_b \tag{3.18}$$

We can see from Equation 3.18 that the relation between $v_a : v_a^*$ and between $v_b : v_b^*$ generate approximately the same vector offset. As we can capture a relation between words using an offset, it is possible to capture relations from sentences likewise. Thus, consider the following three sentences:

1. The boy had a baseball in his hand.
2. The boys had baseballs in their hands.
3. The girl plays with the doll.
4. The girls play with the dolls.

Sentences 1 and 3 are examples of sentences in the single form, whereas sentence 2 and 4 are in the plural form. The relations between sentences 1 and 2 and 3 and 4 are the plural form. Therefore, in the same way, we manipulate word embeddings, we can create an offset from sentence embeddings generated by the `sent2vec` algorithm. Consider that sentence 1 to be the vector $v_1 = [0.0, 0.7, 0.8]$, sentence 2 to be the vector $v_2 = [0.1, 0.7, 1.8]$, sentence 3 to be the vector $v_3 = [-0.1, 0.3, 0.6]$, and sentence 4 to be the vector $v_4 = [0.2, 0.2, 1.5]$. Thus, when if we perform the subtraction of sentence embeddings from sentences 1 and 2, the resulting embedding represents the plural form relation $p = [0.1, 0.7, 1.8] - [0.0, 0.7, 0.8] = [0.1, 0.0, 1.0]$. To obtain a consistent representation of such relation, we need to take the average of many resulting embedding subtractions. Let us consider that the average offset of the plural relation is the vector $p = [0.2, 0.0, 1.1]$. Now, if we sum our plural offset to a sentence embedding in a single form, we can obtain a sentence embedding in the plural form. For example, if we take the sentence embedding of sentence 3 and sum it to our plural offset, the result would be the embedding

$p_{v_4} = [-0.1, 0.3, 0.6] + [0.2, 0.0, 1.1] = [0.1, 0.3, 1.7]$. In our small example, the closest vector to p_{v_3} is v_4, which corresponds to the plural form of sentence 3.

The use of text representations depends on the problem. Simple problems that do not rely much on semantic information can use both bag-of-words and `tf-idf`. They do not require a large corpus and can represent information about the words in a text. On the other hand, to solve problems requiring semantic information at a word-level, such as machine translation and syntactic parsing, one can use `word2vec` (or a word embedding variation). This type of representation needs a large corpus to capture the semantics of each word and create an identification of word relatedness. Variations in the text size, such as sentences and whole documents, may be solved by using both `sent2vec` and `doc2vec`. As in the `word2vec`, both representations need a large body of text to train. While representations such as `word2vec`, `sent2vec`, and `doc2vec` can be input for learning algorithms such as LSTM, matrix-like text representations can be used as input for CNNs. Representations of such type may not require a large corpus since the content of the matrix is often text characters, such as the presence of a letter or word. For some cases, the values in a matrix can be a word embedding, which requires a previous representation of words.

4. AUTOMATIC REASONING OVER CONTRACTS

In this chapter, we describe the task of automatic reasoning over contracts. Our focus on this thesis is to reason over norms in contracts to identify conflicts between them. In our approach, we deal with contracts written in natural language, which requires the use of specific techniques. Since computers need an amenable representation to process natural language, we need to convert it to an approximation representation, as the ones we discuss in Section 3.3. Using natural language processing techniques introduced in Section 2.6, we are able to process natural language in contracts to automate the process of norm conflict identification introduced in Section 2.4. We divide this chapter into four sections; the first one describes the problem we are dealing with in this thesis as well as the questions we answer with our approach. In the second section, we start to detail the task of identifying and classifying conflicts between norms. The first step towards this is the process of norm identification. The third section describes how we can represent norms from a contract. Finally, the fourth section describes the norm conflict identification and classification process.

4.1 Norm Conflict Identification

Contracts use norms to state expected behaviours from parties and determine sanctions when individuals do not comply with them. As norms follow deontic statements to describe permissions, obligations, and prohibitions, when poorly described, norms may arise conflicts between their definitions. Such conflicts result in the impossibility of complying with a norm without violating the other, as we describe in Section 2.4. The identification of norm conflicts requires contract drafters, lawyers, and the contract parties to manually identify them by comparing every norm against each other. As the number of contracts increases with online contracts and their complexity vary according to the theme, identifying norm conflicts becomes a time-consuming and error-prone task. Recent approaches propose the automation of parts of the contract processing, such as contract structure manipulation [11, 18, 17], information retrieval from contracts [37], and even norm conflict identification [36]. In this thesis, we introduce an approach that identifies and classifies norm conflicts in contracts, leading to the following statement:

> **Thesis statement:** We aim to automatically identify norm conflict and their types in contracts written in natural language, as well as the nature of these conflicts.

This leads to two main questions we answer in this thesis:

1. How can we represent norms in order to **effectively** compare them and identify conflicts?

2. Is it possible to classify norm conflicts in order to improve the process of resolving conflicts?

We developed approaches to identify and classify norm conflicts. These approaches follow different strategies different strategies. Most approaches involving norm conflicts have three main modules: (1) Norm Identification, (2) Norm Representation, (3) Norm Identification/Classification. Figure 4.1 illustrates the architecture of an approach to identify conflicts. We detail each of these modules in the following sections.

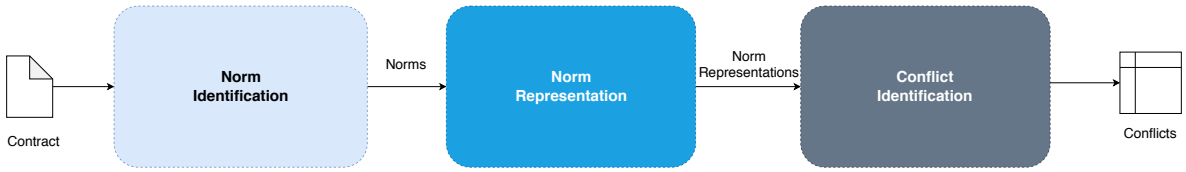


Figure 4.1 – Common architecture of a norm conflict identification approach

4.2 Norm Identification

Norm identification is the process of extracting norm clauses from contracts using information extraction techniques. Dealing with natural language requires us to usually work with raw data, which, in this case, is raw text. Although contracts are semi-structured documents (see Section 2.3), their text has no annotation. Thus, in order to identify the elements of a contract, such as the parties, dates, subjects, and norms, we need to process the text. Example 4.2.1 illustrates the raw text with one norm in bold.

Example 4.2.1. :

> In a December 1998 contract, Schoeller Plast Industries GmbH, with Ifco's consent, transferred all rights and duties from the 1997 supply agreement to SWS, effective December 31, 1998. The latter at that time was operating under the name Schoeller Plast AG. In October 2001, MTS Ecologistics Administration GmbH was merged into Ifco. As a consequence of this, Ifco inherited the logistics business that was previously carried on by MTS Ecologistics Administration GmbH. The parties intend to use this contract to produce a new wording of all rights and duties from previous contracts by and between them. **2.2 Two basic production paths shall be provided: production from recycled material, and production from new material.**

Norms in natural language often follow patterns to make them easier to understand for readers, and two key patterns are the following: the presence of verbs conveying a deontic meaning, and a well-defined structure much along the lines of our formal definition in

Section 2.5. The first one is the deontic meanings aspect. Since norms describe deontic meanings, such as permissions, obligations, and prohibitions, they often have modal verbs in their syntactic structure. In natural language, modal verbs have the role of representing deontic meanings, *e.g.*, 'must' and 'shall' define obligations, whereas 'may' defines permission. The negation of modal verbs often define prohibitions. Example 4.2.2 illustrates the use of modal verbs with their respective deontic meaning.

Example 4.2.2. :

- **Obligation**: You **must** pay product taxes.
- **Prohibition**: The state **shall not** use public cars for food transportation.
- **Permission**: Buildings **may** be either red or blue.

Besides using modal verbs to define deontic meanings, norms use other expressions, such as 'obliged', 'permitted', and 'forbidden'. As we can see, the definition of a deontic meaning is a central concept that defines norms. The second pattern in norms refer to their defined structure, *i.e.*, we can frame their elements into defined classes. Other than deontic meanings, norms always deal with a subject, an object, an antecedent, and a consequent [97]. Norms define a subject that must perform or avoid the action defined in the object. In order for a norm to be fully activated, the antecedent conditions must hold. Finally, the norm is fully satisfied when the conditions in the consequent hold, which deactivates it. The following list describes each element in a norm.

- **Norm**: In case of delay, Delivery Company must pay the delivery cost.
- **Subject**: Delivery Company
- **Object**: pay the delivery cost
- **Modal Verb**: must
- **Deontic Meaning**: Obligation
- **Antecedent**: In case of delay
- **Consequent**: pay the delivery cost.

When using norms to identify conflicts, we select only part of these elements, they are: Subject, Object, Deontic Meaning, and Antecedent. By using these four elements, we can check to whom the norm is applied (Subject), what is intended in each norm (Object), the deontic force over the Object (Deontic Meaning), and the existence of any condition to the execution of the norm (Antecedent).

4.3 Norm Representation

Norms describe what is expected from contractual parties in the most diverse situations. Within norm definitions, we may often find a party name, a modal verb carrying a deontic meaning (permission, obligation, or prohibition), an action, a consequent, and an antecedent. Our challenge here is to compare such information between norms to identify conflicts, as discussed in Section 2.4. Since our goal is to automate such process, the best way to automatically compare norms is to create a representation that allows the computer to perform it. As we discussed in Section 4.3, various authors proposed formal representations to norms. Moreover, in Section 3.3, we discussed how we can represent text to use it as input to learning models. These two representations differ in their formats and goals. One can use formal norm representations to implement logical approaches that involve the comparison of elements and information retrieval On the other hand, we use general text representation as a computational encoding that use a latent representation to describe text. Using such knowledge, we can represent norms in a way that the comparison between them favours the identification of conflicts.

We need to use a mathematically closed representation of natural language in order to use machine learning approaches to process contracts. The most basic way to represent norms is using the natural language form of it. On the other hand, basic forms of text representation can better represent norms for some specific tasks. For instance, bag-of-words can be useful when trying to identify norms in a contract text. Since norms often use modal verbs, the indication of them in a sentence may be enough to perform identification. As described in Section 4.3, most norm representations consist of using the norm deontic meaning followed by the subject and object. The proposed representations can make direct comparisons between norms and allow us to draw logical conclusions from them. However, such norm representations have a common flaw that is the need of someone to translate norms in natural language to them. In order to automate a process of such representations, we need a mechanism to perform the conversion from natural language to these representations.

The state-of-the-art form of natural language representation is the use of text embeddings. This representation uses the power of neural networks and massive data to create a vector representation to text. In order to generate such representation, neural networks take into consideration the context in which the text is applied. Such context provides information about the text meaning and resemblance to other pieces of text. Commonly used in natural language processing tasks, text embeddings can also represent norms as discussed in Chapter 3. Such representation dismisses the need for a specific identification of each norm element as it will automatically identify them by its learning process.

4.4 Conflict Identification

What makes a conflict between two norms to arise may vary according to the context related to both norms. Thus, an automatic norm comparison aiming to identify conflicts must consider the meaning of the words and their relation. However, a direct comparison between norm elements can also be insufficient to detect conflicts. Since natural language have multiple ways to be structured preserving the same meaning, we must take into consideration both syntactic and semantic information.

As we described in Section 2.4, norm conflicts have many causes and types. The most basic kind of norm conflict regards the existence of two norms with opposite deontic meanings describing the same subject, object, antecedent, and consequent. Example 4.4.1 illustrates the occurrence of this type of norm conflict with the modal verbs in bold, respectively an obligation and prohibition. In order to identify this type of conflict, we just need to check if what surrounds the modal verb is equal while the deontic meaning is different. This is quite simple if we just compare the text of both norms and check the deontic meaning from modal verbs. The same conflict case can vary from being explicit as the example shows or more complex as Example 4.4.2 illustrates. In this case, we must consider that the complexity is in the fact that although they describe the same scenario, each one uses different words expressing the same meaning. Such complexity makes conflict analysis harder than just a comparison of items. Now it is necessary to know the meaning of each word in its specific context and compare them semantically. A similar situation may occur when both norms have the same deontic meaning but dealing with conflicting objects. In this case, what arises the conflict is the possibility of performing or avoiding conflicting objects at the same time. Example 4.4.3 shows how this conflict can emerge.

Example 4.4.1. :

1. Shoe boxes **must** be red.
2. Shoe boxes **cannot** be red.

Example 4.4.2. :

1. Company D must pay the bills by the 30th of September.
2. Regarding its bills, Company D shall not pay them in the last day of the month.

Example 4.4.3. :

1. Company D car must be in the parking lot number 1.
2. The car of Company D shall use parking lot number 3.

Conditions play an important role on norms as they define when norms activate and exceptions to rules. They can cause conflicts when what is defined under the condition conflicts with a general norm. In this case, we must consider that the condition may be satisfied in a specific situation arising the conflict. Example 4.4.4 shows an example of conflict aroused by the object under the condition. In the example, there is no conflict in most of the years, however, when it is a leap-year, a conflict arise as Company E will be able to pay the bills at the 15th day of the month, which violates norm 1. Identifying this type of conflict requires the identification of conditions and their objects in order to compare them and check conflicting objects.

Example 4.4.4. :

1. Company E shall pay all your bills at the 12th day of the month.
2. If leap-year, Company E shall pay bills at the 15th day of the month.

The classification of norm conflicts into types can help us to identify the causes that led the conflict to occur. Once we know the causes of norm conflicts, we can resolve them by changing what is creating the conflict. The automatic classification of norm conflicts can help contract drafters to quickly identify an alternative to resolve conflicts.

5. NORM CONFLICT CLASSIFICATION

In this thesis, we introduce four main contributions. The first one is an approach to detect and classify norm conflicts in contracts. The second is the definition of four norm conflict types, which allows one to differentiate conflicts. The third contribution is a manually annotated dataset containing norm conflict cases according to the defined types. Finally, we built a public available web tool containing our approach to identify conflicts. We summarise all these contributions in this Chapter.

The detection and classification of norm conflicts depends on a series of sub-tasks that allow us to manipulate norms in natural language. We defined a three-module architecture to process natural language contracts and return its norm conflicts classified. Our approach is the result of a series of experiments to answer our research questions. Figure 5.1 illustrates our approach's architecture. In the following sections, we detail each module of our architecture.

The classification of norm conflicts relies on previous definitions of such classes of conflicts. We introduce four norm conflict types that allow us to cluster norm conflicts according to their causes of conflict. These four types also describe norm conflicts with different levels of complexity on their identification.

As part of the creation of an approach capable of detecting and classifying norm conflicts, we need a set of annotated data that describes each norm conflict case. In the absence of an existing dataset, we decided to manually annotate norm conflict cases to make possible a learning process of conflict types.

In order to make it available to the public, the resulting approach became part of a web tool called ConCon. Such web tool has as main goal to provide free access to our norm conflict identifier as well as to allow users to annotate conflicting cases.

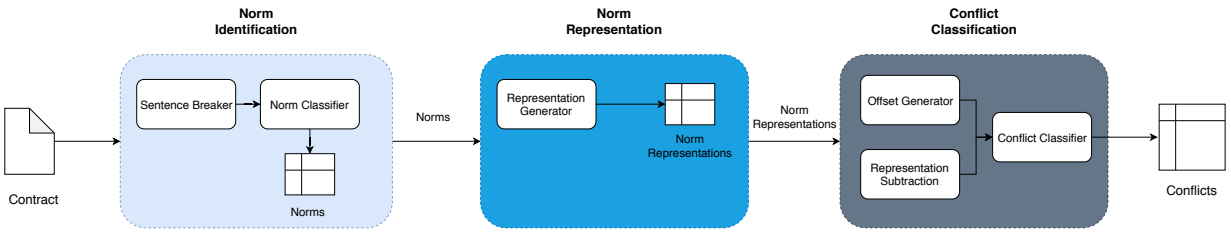


Figure 5.1 – Norm conflict classifier architecture

5.1 Norm Conflict Types

Automating the identification of conflicts between norms is the key process to make contract writing and revision faster. By identifying such cases in contracts, the contract writer

can prioritise detailed revision of those norms identified as being most likely to be in conflict. In order to further help contract authors, we want to indicate what causes for a conflict to occur within a contract, instead of simply detecting conflicting clauses in a contract. While there are various typologies for norm conflicts [6, 102, 79, 85], in the case of conflicts in contracts, one specific typology stands out by relating the deontic representation of norms within clauses to the possible types of conflicts [104]. However, in order to diagnose the specific nature of the conflict and amend clauses to eliminate them, contract writers often need more information than the deontic modalities of two clauses that are in conflict. Thus, we leverage the typology of Akhavi [93] to identify four conflict types that can facilitate the task of eliminating existing conflicts by finely defining the *nature* of the conflicts. The four types are: *deontic-modality*, *deontic-structure*, *deontic-object*, and *object-conditional*.

5.1.1 Formalisation of Norm Conflict Types

In order to formalise the four conflict types, consider a contract $\mathcal{C}$ and a set of norms $\mathcal{N} = \{n_1, n_2, ..., n_n\}$ where $\mathcal{N} \subseteq \mathcal{C}$ and n_i is the i^{th} norm in the contract. In Section 4.2, we describe six elements of a norm, they are: Subject, Object, Modal Verb, Deontic Meaning, Antecedent, and Consequent. To make it easier to formalise our conflict types, we concentrate norm elements into four main elements, as follows: Party, Deontic Meaning, Action, and Condition. Here, we convert Antecedent into Condition, and Consequent into Action, which is also describing Object. Subject becomes Party, and Modal Verb is now described by Deontic Meaning since in our analysis we will need only the deontic meaning information. In our formalisation, $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ stands for the set of parties in $\mathcal{C}$ and $\mathcal{D} = \{d_1, d_2, d_3\}$ stands for the set of deontic meanings consisting of Permission, Prohibition, and Obligation. $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ stands for the set of actions declared in $\mathcal{C}$, and $Cond = \{c_1, c_2, ..., c_n\}$ is the set of conditions described in norms from $\mathcal{N}$. We add a second index to each element to describe the party, deontic meaning, action, and condition as part of a norm. For example, given a norm n_i, we represent the action a_q expressed in such norm as $a_{q,i}$, where q is the action identifier and i is the norm identifier. Thus, we can classify each conflict type.

Deontic-modality conflict type represents the simplest conflict case. It arises when two norms (n_i and n_j) refer to the same parties ($p_{k,i}$ and $p_{k,j}$), express the same action ($a_{q,i}$ and $a_{q,j}$), but with different deontic meanings ($d_{r,i} \neq d_{t,j}$). In norms written in English, the deontic meaning is often represented by a modal verb, such as *must*, *shall*, *ought*, *etc.* Thus, *deontic-modality* conflict type consists of norms where modal verbs indicate different deontic meanings, subsuming most conflict types identified by Vranes and Akhavi [104, 93]. Example 5.1.1 illustrates a pair of norms, in which this conflict type can occur. Here, **may** indicates a permission whereas **shall not** indicates a prohibition.

Example 5.1.1. :

1. The Specifications **may** be amended by the NCR design release process.
2. The Specifications **shall not** be amended by the NCR design release process

Deontic-structure conflict type is similar to *deontic-modality* conflict since it involves two norms (n_i and n_j) with different deontic meanings ($d_{r,i} \neq d_{t,j}$) but with a different natural language structure. In this case, it can refer to the same subject (*i.e.*, ($a_{q,i}$ and $a_{q,j}$), ($p_{k,i}$ and $p_{k,j}$), and ($c_{u,i}$ and $c_{u,j}$)) using different expressions. Example 5.1.2 illustrates this conflict type.

Example 5.1.2. :

1. All inquiries that Seller receives on a worldwide basis relative to Buyer's air chamber "Products" as specified in Exhibit III, **shall** be directed to Buyer.
2. Seller **may not** redirect inquiries concerning Buyer's air chamber "Products".

This example contains two norms with different sentence structures involving the use of different word and word-structure referring to the same subject. The conflict arises because in the first norm there is an obligation towards the Seller that inquiries must be sent to Buyer, whereas the second norm prohibits the Seller to send such inquiries.

Deontic-object conflict type indicates cases where both norms (n_i and n_j) have the same deontic meaning ($d_{r,i}$ and $d_{r,j}$) but different overall natural language structures. In this particular case, the conflict arises due to the definitions about the norm actions ($a_{q,i}$ and $a_{w,j}$) or specification details ($c_{u,i}$ and $c_{o,j}$). The example below illustrates the case.

Example 5.1.3. :

1. Autotote shall make available to Sisal one (1) working prototype of the Terminal by **May 1**, 1998.
2. Autotote shall make available to Sisal one (1) working prototype of the Terminal by **June 12**, 1998.

In this case, although the action itself is the same in both norms, the conflict arises because their date definition differs in such a way as to allow the fulfilment of norm 2 while violating norm 1. Example 5.1.4 exemplifies a different (and more direct) instance of the same type of conflict. Specifically, we have a conflict because the actions are mutually exclusive.

Example 5.1.4. :

1. CoPacker will assume no costs of transportation and handling for such rejected Products.

2. CoPacker shall assume all costs of transportation and handling rejected Products.

Object-conditional conflict type between a pair of norms (n_i and n_j) occurs when a condition or exception in one norm ($c_{u,i}$) conflicts with the action expressed in the second norm ($a_{q,j}$). In this case, the deontic meanings, parties, and actions can be either the same or different. This type of conflict concerns specific examples where a condition is involved, and the following example illustrates the case.

Example 5.1.5. :

1. The Facility shall meet all legal and administrative code standards applicable to the conduct of the Principal Activity thereat.
2. Only if previously agreed, the Facility ought to follow legal and administrative code standards.

In this example, we have two norms that have the same deontic meaning, but the condition in the second one produce a conflict since one can comply to it and, if it was not previously agreed, do not follow legal and administrative code standards for facilities.

5.2 Norm Conflict Dataset

The task of classifying norm conflicts according to their types involves the need for a dataset containing examples of each conflict type. Such examples consist of pair of norms containing conflicting elements that represent a specific conflict type. As far as we have researched, there is no available corpus containing actual norm conflicts. Thus, we decided to manually annotate a set of norm conflicts according to the types we defined and described in the previous session.

By creating a dataset with norm conflict information and type specifications, we can build intelligent systems able to improve the identification and solution of norm conflicts. As we publish such dataset, new research can be conducted that may expand the area in new directions. Therefore, these efforts can lead to further advancements on the automatic contract processing area.

In a previous effort to create a dataset containing norm conflicts, we manually annotated a set of 111 norm pairs as conflicts [5]. Thus, in order to manually annotate norm conflicts and their types, we defined a three-step process. The first step consists of manipulating and annotating the existing dataset with norm conflicts. The second step consists of manually creating norm conflicts from real contracts. Finally, the third step consists of labelling each norm conflict according to the defined types. Figure 5.2 illustrates the process of norm conflict annotation in the second and third step.

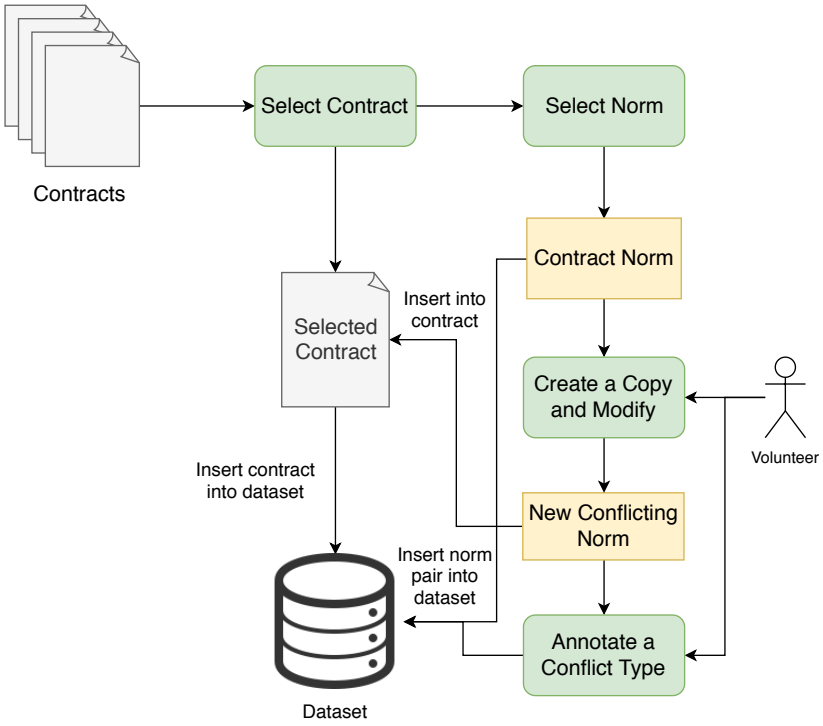


Figure 5.2 – Process of norm conflict annotation

The first step to create our dataset is to adapt the existing dataset to the specifications we aim to achieve in the new one. To perform such adaptation, we analysed each conflicting norm pair in order to check its conflicting type according to our type definitions. Thus, we selected one of the four types of conflict (*deontic-modality*, *deontic-structure*, *deontic-object*, and *object-conditional*) for each pair of norms marked as conflicting. By annotating the dataset, we observed that it includes only two types of conflicts (*deontic-modality* and *deontic-structure*). Since the main goal of this first dataset was to evaluate only conflict identification, we focused our effort on simple and structural conflicts.

Once we have the annotation of our existing dataset, we proceeded to the next steps. The creation of new conflicting norms should contain examples from the other two types as well. Hence, in the second step, we selected human volunteers to perform the annotation of new conflicting examples. We use our web-based tool (see Section 5.8). To do so, we open a section for annotation in our web-tool (see Figure 5.3 a)). Our annotation system randomly selects norms within the contract corpus (see Figure 5.3 b)). The instructions to volunteers were just the definition of our conflict types and how they should use our interface. Besides, they were instructed to freely edit the selected norm in order to create a conflict with the original norm, following one of the four types. We did not evaluate or post-processed created conflicts since we assume volunteers correctly generated them.

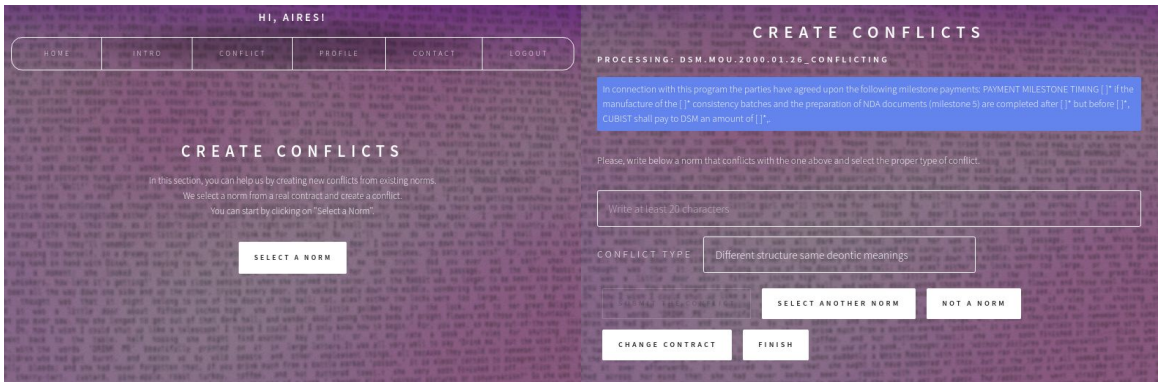
As Figure 5.2 shows, the creation of norm conflicts consists of a selection of norms from real contracts. At first, our system selects a random contract and a random norm from this contract to display to the volunteer. The selection of contracts follows a defined plan. We select the contract and check its norms using our norm identifier. We defined a minimum

number of five norms to select a contract. Thus, given the number of norms, we established a maximum number of conflicts to the contract, *i.e.*, the number of conflicts volunteers may create can be equal or smaller than the number of norms in the contract minus one. Then, when selecting a contract, we prioritise contracts that are still missing conflicts. However, if the volunteer faces difficulties to create conflicts, she can change the contract or request a different norm from the same contract.

Once the volunteer feels confident about a certain norm, he starts to write in a text field a new norm that must conflict to the selected one. This step is depicted in Figure 5.3(b) and can be better described with Example 5.2.1. In the example, the new norm generates a direct conflict with the existing one. After finishing the new norm, the volunteer starts the third step of our annotation process, which is the annotation of a type for the created conflict. In our interface, we provide a drop-down menu that contains the four conflict types (*deontic-modality*, *deontic-structure*, *deontic-object*, and *object-conditional*). The volunteer then selects the one that better fits to the norm created. In our example, the *deontic-object*is the best type for the norm pair since the conflict occurs due to a modification in both the object structure and the deontic meaning of the norm. We then insert the new norm into the contract as one of its original norms. The conflicting norm pair becomes part of a separated dataset containing only conflicting and non-conflicting norm pairs.

Example 5.2.1. :

- **Existing norm**: In addition, the site may allow users to search the USF product catalogue and gain access to the price of the items, but not the ones in the Customer's standard Ordering Guide.
- **Volunteer norm**: Users are forbidden to have access to products details, such as price and USF.



a) b)

Figure 5.3 – Annotation section in our web-tool

The result of our annotation process is a dataset containing a total of 228 conflicting norms including the existing 111 conflicts from the previous dataset in addition to a total of

11,329 non-conflicting sentence pairs. Table 5.1 details the number of conflicts of each type in our new dataset, where we can see that the proportion of *deontic-modality* conflict type is the highest one. This occurs because it is the simplest type of conflict one can create. On the other hand, creating a conditional cases is rather difficult since one needs to find a plausible condition that fits the norm context.

Conflict Type	# of Elements	% in dataset
deontic-modality	97	42%
deontic-structure	61	27%
deontic-object	30	13%
object-conditional	40	18%

Table 5.1 – Number and proportion of each conflict type in the new dataset

5.3 Identifying Norms

The first step towards norm conflict identification and classification is to identify which sentences in a contract contain deontic statements (norms). In Section 4.2, we saw that the task of identifying norms relies on the identification of specific elements that norm sentences have. We detail this process in two sections. The first section describes the creation of a dataset containing the annotation of norm and non-norm sentences. The second section describes our experiments to create a sentence classifier able to identify norms.

5.3.1 Contract Sentences Dataset

We use a linear classifier to identify norm sentences among contract sentences. In order to train the classifier, we need a dataset with annotated examples of both norm and non-norm sentences. Since there is no such dataset available, we decided to manually annotate a series of sentences from different contracts as being either norms or not.

We made the annotation using a script that randomly selects sentences from a set of contracts and asks the user to annotate them as norms or non-norms. Figure 5.4 illustrates the process of sentence annotation. Our base of contracts consists of 164 contracts from the manufacturing type extracted from the Onecle repository[1]. As a result, we obtained a total of 1,209 contract sentences from 22 different contracts. From these sentences, 691 are norm sentences and 518 are non-norm sentences. We use them as train and test sets in our experiments.

[1] https://contracts.onecle.com/type/47.shtml

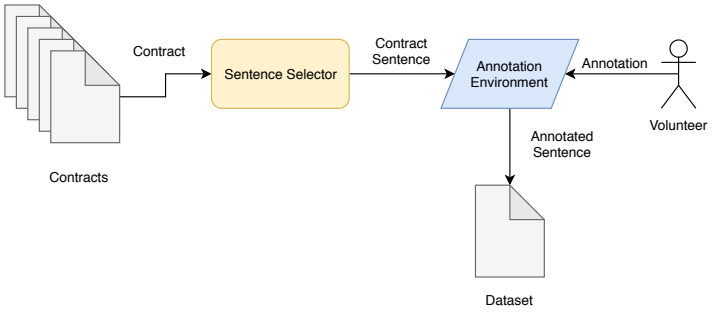


Figure 5.4 – Process of contract sentence annotation

5.3.2 Experiments

The task of identifying norm sentences in contracts is binary and, therefore, we consider contract sentences to be of two exclusive types: norm sentences and non-norm sentences. Although most norms use modal verbs to express deontic meanings, we cannot use a simple rule to identify norms as they can present other ways to express the same meanings. Considering the need for a flexible approach that can be adapted to generalise over different expressions, we decided to use a linear classifier. We trained a sentence classifier using the Support Vector Machines (SVM) learning algorithm with our manually annotated dataset. SVM is often used to train models with datasets containing few training examples, multiple features, and binary classification. The learning algorithm creates a hyperplane that tries to find the best division between two classes [49]. In order to train the SVM, we turn each sentence into a bag-of-words representation [46], which represents the frequency of words in a sentence from a fixed dictionary. Using this representation, the SVM learns from the frequency each word appears in a class.

To evaluate our sentence classifier, we divided our manually annotated dataset into train and test set. We use an 80/20 division, which results in 967 sentences in the train set and 242 sentences in the test set. Both sets are balanced according to the number of elements for each class, *i.e.*, 551 norm sentences and 416 non-norm sentences in the train set, and 137 norm sentences and 105 non-norm sentences in the test set. To compare the SVM with other linear models, we test the same dataset with two other classifiers: Perceptron and Passive Aggressive. Perceptron is a well-known linear model, which can be better explained as a neuron in a neural network [74]. It processes the input by multiplying it using a set of weights. The result goes to an activation function, which defines the input class. Passive Aggressive [26] is a linear model that has its name based on its weight update rule that, in each round, can be passive, when the hinge-loss result of its update is zero, and aggressive, when it is a positive number. Table 5.2 shows the results for each classifier.

As we can see, SVM obtains the best result for the task with an accuracy of 90%. The passive aggressive algorithm has similar accuracy and has the best precision in com-

Classifier	Prec.	Rec.	F-Score	Acc.
Perceptron	0.89	0.88	0.88	0.87
Pass. Agr.	**0.92**	0.88	0.90	0.89
SVM	0.88	**0.94**	**0.91**	**0.90**

Table 5.2 – Results for sentence classifier

parison to the others. However, since SVM obtains a better overall result, we use it as our sentence classifier.

5.4 Representing Norms

Once we have identified norm sentences from contracts using our sentence classifier, as part of our approach, we need to create a norm representation that allows us to process them. The representation of norms is central to our work since the way we represent them defines how we will compare them to identify norm conflicts. Among the existing representations of norms and general text described in Section 2.5 and Section 3.3, we decided to use one that we can obtain without manual translation and that carries syntactic and semantic information. Thus, we use the `sent2vec` [80] algorithm to convert norm sentences into latent representations. `sent2vec` is a state-of-the-art sentence embedding algorithm, which, similarly to the word embedding algorithm (CBOW) [71], converts sentences into dense vectors. These vectors preserve both syntactic and semantic information from sentences. Using such embeddings, we can perform arithmetic operations on them and perform natural language manipulations. `sent2vec` provides pre-trained models over the Wikipedia sentences dataset[2], where sentences are divided into unigrams and bigrams. For the unigram pre-trained model, `sent2vec` produces vectors with 600 positions containing real values. Whereas using the bigram pre-trained model, `sent2vec` produces vectors with 700 positions containing real values. The algorithm preserves the meaning of sentences according to the these values position over the vector, which differentiate and approximate sentences depending on their similarity.

In order to understand what happens to the norm sentences, consider the following set of norms:

1. Company X must pay the product taxes.
2. Company X shall not pay the taxes for product Y.
3. Company Y may use room B on the weekends.

Figure 5.5 illustrates the representation of norms 1, 2, and 3 described as latent_norm1, latent_norm2, and latent_norm3, respectively. When we plot these sentences,

[2]https://github.com/google-research-datasets/wiki-split

they will follow a certain pattern according to their meaning, as described in Section 3.3. Figure 5.6 illustrates the set of norms in a plot. Dashed lines indicate the distance between latent_norm1 and the other representations. As we can see, the representations of norms 1 and 2 share more similarities semantically, therefore, they are close to each other. Whereas norm 3 is far from both norms as it describes a different subject.

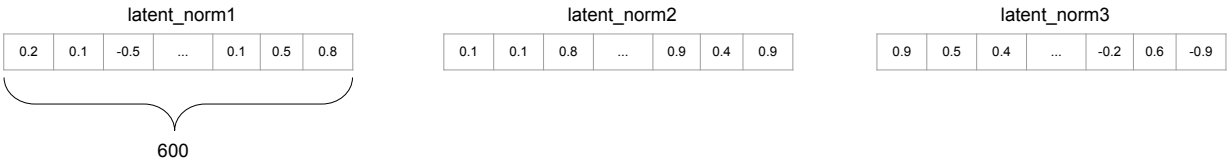


Figure 5.5 – Norm representation example

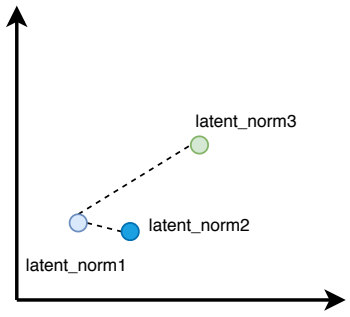


Figure 5.6 – Plotting norm representations

5.5 Classifying Norm Conflicts

As our third module, once we have identified and represented norms from contracts, we use norm representation to identify and classify norm conflicts. This is the central module of our approach. Here, we use the information contained in the representations to classify norm conflicts. In order to describe the process that took us to our final approach, we dived this section into four subsections. In the first subsection, we describe the types of norm conflicts we defined for this work. In the second one, we explain how we create our annotated dataset of norm conflicts. In the third one, we describe norm conflict identification, which was our initial goal. Finally, we describe the classification of norm conflicts and the evaluation of our approach.

5.5.1 Norm Conflict Identification

Initially, in this research, we tried to find more efficient ways to identify conflicts. In this process, we developed two approaches to this subject.

CNN Identifier

Our first approach was a convolution neural network that classifies norm pairs as conflicts or non-conflicts. To create the norm conflict identifier, we trained a CNN using norm pairs from the dataset built in our previous work [3]. The premise for this approach was that conflicting norms tend to be similar in that usually both norms refer to the same party/parties with similar actions, and only the modal tone of the sentence differs. Consequently, we rely on training examples that consist of binary images created from each pair of norms denoting the distance between these norms. Thus, we created a pair-of-norms representation using a matrix to denote similar characters in both norms. Given two norms α and β, our matrix consists of the characters from α in its lines and the characters from β in its columns, as Figure 5.7 illustrates. Given a cell {i, j}, we assign 1 to it when the i^{th} character of α is equal to the j^{th} character of β and 0 otherwise. We limit the length of both norms to 200 characters, which is the mean length of norms from our dataset and truncate overlong sentences to give the matrix a squared shape. Using this representation, we train a CNN to generate a model to classify norm pairs as norm conflicts and non-norm conflicts. The network architecture consists of two convolutional layers followed by a max pooling layer and two fully-connected neural networks. Each convolutional layer has 32 kernels that are responsible for extracting features from the input image.

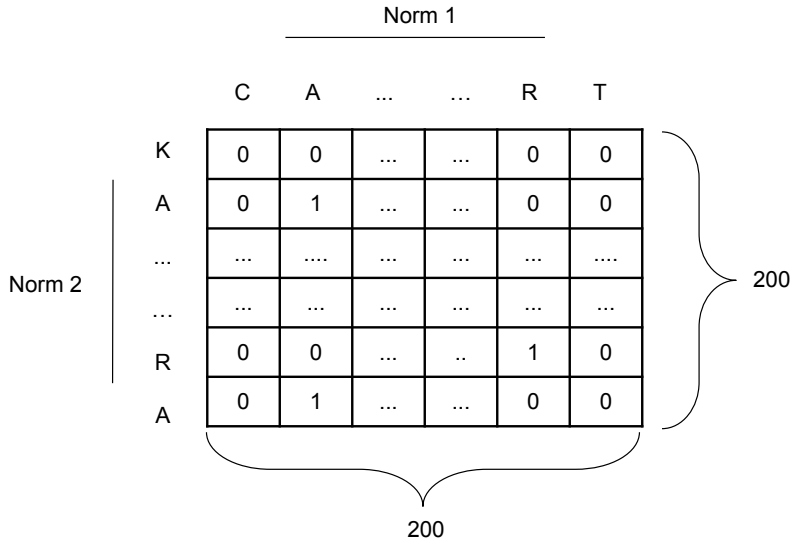


Figure 5.7 – Norm pair representation in our approach

To evaluate the CNN norm conflict identifier, we used a 10-fold cross-validation step dividing the dataset into train, validation, and test. In our previous dataset, we have a total of 111 norm pairs with conflicting norms and 204,443 conflict-free norm pairs, therefore, the first step is to create a balanced dataset. By having a balanced dataset, we ensure that the learning model will not have inconsistencies during the training, such as biasing the learning by focusing on the non-conflicting features. In this case, using oversample and undersample techniques are not simple since we are dealing with text, which makes it difficult to automatically produce consistent data. Thus, we reduced the number of non-conflicting norm pairs to 111, which gives us a total of 222 samples. Each fold has 10% of the data, which is around twenty-two samples, eleven of each class. In each round, we use eight folds to train, one to validate, and one to test. To prevent overfitting, we use the early stopping technique that monitors the accuracy in the train and validation set. When the accuracy in the validation set starts to decrease and the training accuracy keeps increasing, overfitting is detected, resulting in the termination of the training phase. We show the accuracy results for each fold and the mean accuracy overall in Table 5.3. In order to compare the results to existing work, we use the rule-based approach from Aires *et al.* [4]. Using the results obtained in our previous approach over the same dataset, we compare to the ones in this approach. Table 5.4 shows that using the CNN approach we have an improvement of 6% in the overall accuracy.

Fold	0	1	2	3	4	5	6	7	8	9	**Mean**
Accuracy	0.85	0.85	0.76	0.95	0.85	0.76	0.71	0.95	0.95	0.80	0.84

Table 5.3 – Results for the norm conflict identifier

Approach	**Accuracy**
Rule-based [4]	78%
CNN	**84%**

Table 5.4 – Comparison between CNN approach and Rule-based approach in norm identification

Embedding Identifier

Our second approach to identify norm conflicts uses sentence embeddings to represent norms. Given the representations and a vector offset, we identify conflicts by comparing the subtraction of norm representations with the vector offset. Figure 5.8 illustrates how we process contracts to identify norm conflicts. We divide our architecture into three main modules. In the first module (Figure 5.8 (A)), we convert norms from a contract into sentence representations using the Sent2Vec algorithm [80]. In the second module, we generate an offset representing the concept of conflict between norms. As explained in

Section 3.3, offsets can concentrate on vectors the relation between natural language elements. A simple way to understand offsets is to consider the superlative relation between words, such as 'great' and 'greatest'. If we subtract the representation of both words, we obtain a resulting vector of their difference, which, in this case, is the superlative relation ($v_{super} = v_{great} - v_{greatest}$). Now, if we perform the same subtraction on similar examples, such as small/smallest and big/biggest, we will obtain similar vectors as result as Equation 5.1 shows. When we take the average vector of all subtractions of the same relation, the resulting vector offset will consist of a strong representation of this relation.

$$v_{biggest} - v_{big} \approx v_{smallest} - v_{small} \tag{5.1}$$

We can see from Equation 5.1 that the relation between $v_{big} : v_{biggest}$ and between $v_{small} : v_{smallest}$ generates approximately the same vector offset. Therefore, we can learn a $v_{conflict}$ vector offset that corresponds to a relation between pairs of embeddings, allowing us to use matrix operations to find similar vectors. Assuming that the linguistic regularities of word embeddings can be extended to sentence embeddings, a vector offset containing the relation between two sentences should hold. Thus, a vector representing the conflict between two sentences (norms) can be learned as a vector offset:

$$v_{\text{conflict}} = \frac{1}{|\mathcal{P}|} \sum_{(n_1,n_2)\in\mathcal{P}} v_{n_1} - v_{n_2} \tag{5.2}$$

where $\mathcal{P}$ represents the set of all norm pairs that contain conflicts, and v_{n_1} and v_{n_2} are the vector embeddings of each norm of the pair.

In order to learn the vector offset (v_{conflict}), we extract pairs of norms (n_1, n_2) from our previous created dataset with known conflicts [5], where n_1 conflicts with n_2. We compute the embedding (v_{n_1}, v_{n_2}) of each norm pair using the Sent2Vec library[3] and a pre-trained model from the Wikipedia English corpus, and use these embeddings to compute a conflict vector (v_{conflict}) as the average offset of all pairs of conflicts in training data, as illustrated in Figure 5.8(B).

After building a vector offset representing conflicts in contracts, our third module consists of using it and the produced representation of norms to identify norm conflicts in contracts. We identify a conflict between two norms by comparing their distance in the latent representation to the learned offset against a threshold following Equation 5.3.

$$\mathcal{C}(v_{n1}, v_{n2}, v_{conflict}, t) \begin{cases} \top & \textit{if } d(v_{conflict}, v_{n2} - v_{n1}) < \lambda \\ \bot & \textit{otherwise} \end{cases} \tag{5.3}$$

[3]https://github.com/epfml/sent2vec

Our conflict identifier $\mathcal{C}$ takes four parameters: a pair of norm embeddings (v_{n1} and v_{n2}); an offset vector representation ($v_{conflict}$) containing the average value of differences between conflicting norms; and a threshold (λ) that we use to classify the distance of a vector to the $v_{conflict}$ as a conflict or not. We consider a conflict ($\top$) when the difference between $v_{conflict}$ and the subtraction between v_{n1} and v_{n2} is smaller than λ.

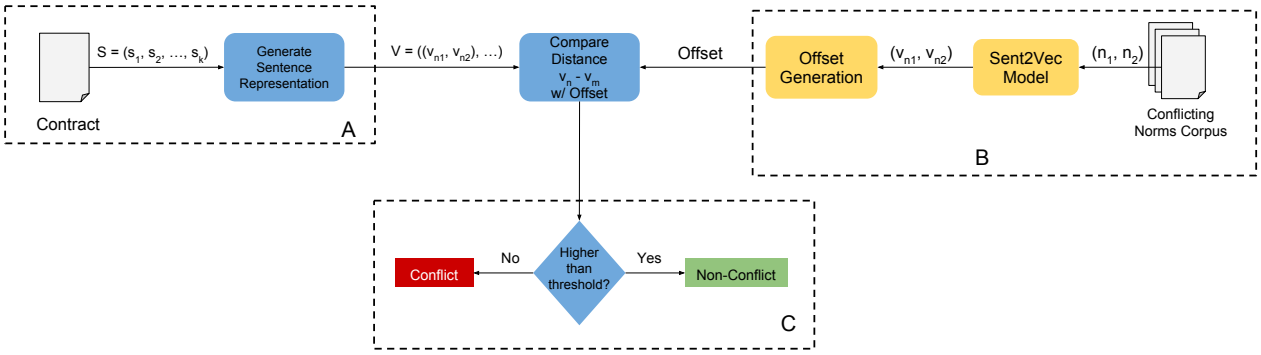


Figure 5.8 – Architecture of our embedding approach to identify norm conflicts

To measure the performance of our approach, we use a 10-fold cross-validation over the norm conflict dataset. Since the generation of the offset uses only the conflicting norms, we divide both sets of conflicting and non-conflicting norms into 10 folds. From the conflicting set, we use 9 folds to generate the offset and the remaining one to test. The test set is then a concatenation between the conflicting fold with its corresponding non-conflicting. We use the same size of conflicts and non-conflicts, so the total number of samples is 222 (111 conflicts + 111 non-conflicts). Each fold has a total of 11 samples and the generation of the offset is made using 99 conflicting samples. Our test set consists of 22 balanced samples. We use the accuracy to measure our results.

In order to compare the results using different distance metrics, we decided to use euclidean distance and cosine similarity. The Euclidean distance (d_{euc}) computes the similarity between two vectors, as shown in Equation 5.4, where y' is the offset vector representation generated by the pair of sentences from the contract. The cosine similarity (d_{cos}) measures the cosine of the angle between two vectors from the same space. Thus, vectors with the same orientation have a similarity of 1, whereas opposite vectors have a -1 similarity. Equation 5.5 illustrates how we calculate the cosine similarity. In order to obtain positive values (from 0 to 2), we use the cosine distance, which can be obtained by applying $1 - d_{cos}$.

$$d_{euc}(v_{conflict}, y') = ||v_{conflict} - y'||^2 \tag{5.4}$$

$$d_{cos}(v_{conflict}, y') = \frac{v_{conflict} \cdot y'}{||v_{conflict}||\ ||y'||} \tag{5.5}$$

We evaluate our approach using different distance metrics and pre-trained models. We use euclidean and cosine distance in both unigram and bigram models. To see what is the best threshold to divide conflicts from non-conflicts, we vary its value according to the distance metric applied. Table 5.5 shows the accuracy results using euclidean distance in both unigram and bigram. We vary the threshold (λ) from 0 to 3 based on the maximum distance non-conflict cases obtained during tests. As we can see, we get the best results for all folds using the unigram model and a threshold of $\lambda = 2$.

To generate a better visualisation of the results considering the variation of the threshold, we created the heat map shown in Figure 5.9 with the accuracy for both unigram and bigram models using the cosine similarity. In this case, we vary the threshold value (λ) from 0 to 2 as they are, respectively, the minimum and maximum values when using the cosine similarity. As Figure 5.9 illustrates, we obtain the best results using a threshold between $\lambda = 0.4$ and $\lambda = 0.8$. This range of threshold shows the best separation between conflicting and non-conflicting cases. The boundaries (0 and 2) obtain poor accuracy since can only correctly classify half of the test samples. Overall, we obtain the best result using a bigram model with a threshold of $\lambda = 0.8$. When comparing the distances used, we can see that both have similar results with opposite models. The Euclidean distance obtains better results using the unigram model, whereas the cosine distance obtains better results using the bigram model.

Table 5.6 compares the results of our approach with related work on the dataset we used in our experiments. We use the best result obtained by the mean of folds, which is the euclidean distance with unigram model and a threshold of $\lambda = 2$. Although using just the distance between embedding vectors, our approach overcomes approaches using rules [4] and deep learning models [3].

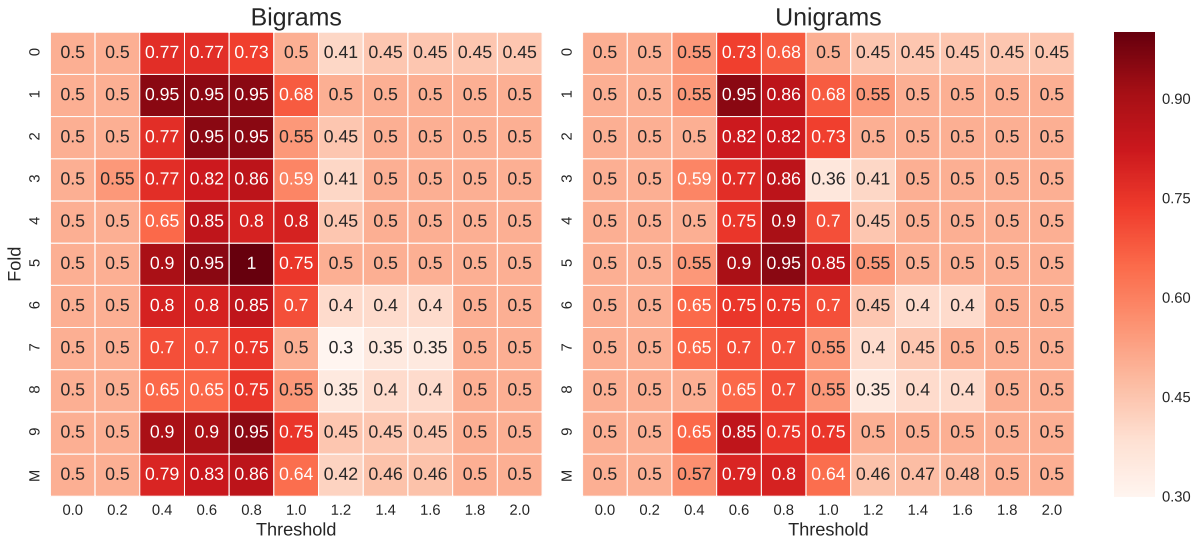


Figure 5.9 – Accuracy obtained using the cosine distance for different folds and thresholds. “M” represents the Mean value for the specific threshold values for the Mean.

	Threshold Unigram				Threshold Bigram			
Fold	0	1	2	3	0	1	2	3
0	0.5	0.77	**0.95**	**0.95**	0.5	0.86	0.82	0.50
1	0.5	0.77	**0.95**	0.91	0.5	0.91	0.86	0.59
2	0.5	0.91	**0.95**	0.82	0.5	0.91	0.91	0.64
3	0.5	0.73	**0.91**	**0.91**	0.5	0.86	0.82	0.50
4	0.5	0.85	**0.95**	0.85	0.5	**0.95**	0.85	0.50
5	0.5	0.90	**1.00**	**1.00**	0.5	0.90	0.95	0.50
6	0.5	0.85	**0.95**	0.90	0.5	**0.95**	0.85	0.55
7	0.5	0.70	**0.95**	0.80	0.5	0.90	0.85	0.55
8	0.5	0.75	**0.90**	0.85	0.5	**0.90**	0.70	0.55
9	0.5	0.90	**0.95**	0.90	0.5	**0.95**	0.85	0.50
Mean	0.5	0.81	**0.95**	0.89	0.50	0.91	0.85	0.54

Table 5.5 – Results obtained using both unigram and bigram models with euclidean distance

Approaches	**Accuracy**
Rule-based approach [4]	0.78
CNN approach [3]	0.84
Offset Conflict	**0.95**

Table 5.6 – Comparison between accuracies from our previous work and the embedding approach

5.5.2 Norm Conflict Classification

The identification of conflicts is a binary classification problem, where a pair of norms is classified either as a conflict or non-conflict, which our previous work has addressed with various degrees of success [4, 3]. Nevertheless, previous research has never attempted to identify the nature of the conflicts it identifies. We address this limitation by expanding the binary classification to a multi-class classification task, where we classify a pair of norms according to a predefined conflict typology or as non-conflict. Unlike conflict identification, in conflict classification, we want not only to identify whether a norm pair is a conflict but also to classify its type. Each pair of norms containing a conflict can be classified into one of the four types of conflicts we describe in Section 5.1: *deontic-modality*, *deontic-structure*, *deontic-object*, or *object-conditional*.

In our work on conflict identification [1] (see Section 5.5.1), we use an offset model to capture the notion of normative conflict in latent space. We create the offset model by learning the average distance between the embeddings of pairs of conflicting norm sentences. Using this learned model, the approach consists of measuring the distance between norm pairs and comparing this distance to the learned offset model. When such distance is below a manually set threshold, the approach indicates a potential conflict. In order to improve this identification and add the classification of norm conflicts, we developed ap-

proaches to eliminate the need of manually setting a threshold. In what follows, we developed two approaches for norm conflict identification, *i.e.*, the classification of unseen pairs of norms as conflict or non-conflict, and two further approaches to classify the *type* of conflict that occurs between norm pairs. Before identifying or classifying norms, we transform each norm written in natural language within a contract into a vector representation that can capture its semantic information. In the process of representing norms, we use the Sent2Vec algorithm in the same way as applied to the norm identifier approach [1]. Therefore, each norm in a contract is converted into its corresponding sentence embedding before classification.

In order to perform experiments over the use of embeddings, we employ them in two ways. First, we use sentence embeddings to represent the sentences that comprise norms in a contract. We call such semantic representation of sentences *norm embeddings* and represent them as E_n. Second, we generate an embedding representing the comparison between two norms using various embedding arithmetic operations and call them *comparative embeddings* and represent them as E_{conc}.

Conflict Identification by Distance Between Embeddings

Our first class of approaches to identify norm conflicts is based on generating comparative embeddings by computing the distance between norm embeddings E_n and using these distances as a semantic representation of the presence or absence of norm conflicts (*i.e.* conflicts and non-conflicts). We developed two distance-based approaches to find the comparative embedding corresponding to the conflict between norms, both of which are based on finding the mean value (centroid in embedding space).

The first approach consists of concatenating norm embeddings of norm pairs representing conflicts and non-conflicts and computing the mean of this extended embedding space, which we call E_{conc}. Thus, given two norms $n1$ and $n2$, we compute $E_{conc} = E_{n1} \cdot E_{n2}$ and, for all norm pairs in a set C of norm pairs (either conflicting or non-conflicting), we generate a centroid embedding using Equation 5.6 by replacing E_{op} by E_{conc}. The second approach consists of computing an offset embedding by subtracting the embeddings of each norm pair $E_{off} = E_{n1} - E_{n2}$ and then computing a centroid embedding (in the same embedding space as the original norms) using Equation 5.6 by replacing E_{op} by E_{off}. We illustrate both concatenation and subtraction embedding manipulations in Figures 5.10.

$$E_{cent} = \frac{\sum_{i=1}^{C} E_{op}[i]}{C} \tag{5.6}$$

Figure 5.11 (a) illustrates this idea, where small blue dots represent the embeddings generated by the concatenation or offset of a pair of norms representing a conflict, small red dots represent the embeddings generated by the concatenation or offset of a pair

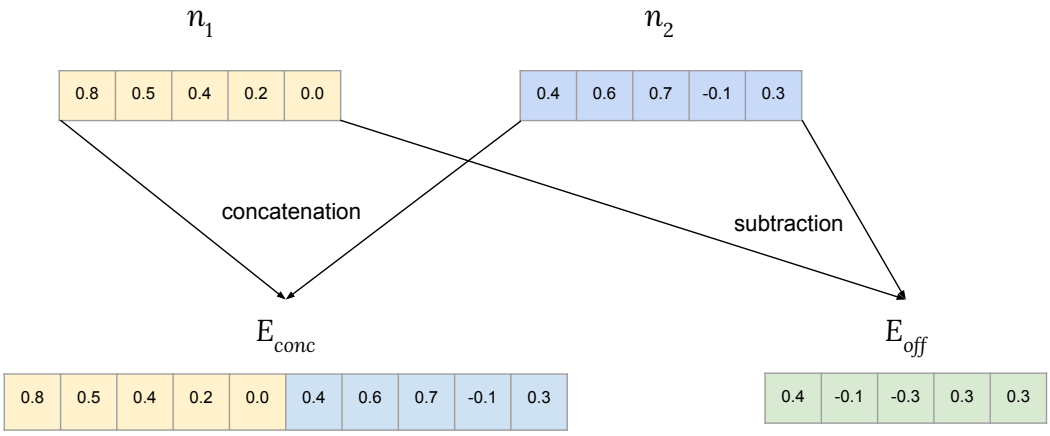


Figure 5.10 – Manipulation over embeddings. On the left, the embedding resulting from a concatenation. On the right, the embedding resulting from the subtraction of two norm embeddings.

of non-conflicting norms, the large blue and red dots correspond to the mean of all small blue dots and small red dots, respectively, and the green dot is the embedding generated from a pair of norms that we want to classify.

In order to identify a conflict, we calculate the distance between the comparative embedding (either E_{off} or E_{conc}) of the norm pair and the centre of the embeddings (E_{cent}) representing conflicts (d_1 in Figure 5.11) and the distance between the embedding of the pair and the centre of the embeddings representing non-conflicts (d_2 in Figure 5.11). Finally, the pair of norms is identified as a conflict if the distance to the centroid representing conflicts is smaller than the distance to the centroid representing non-conflicts.

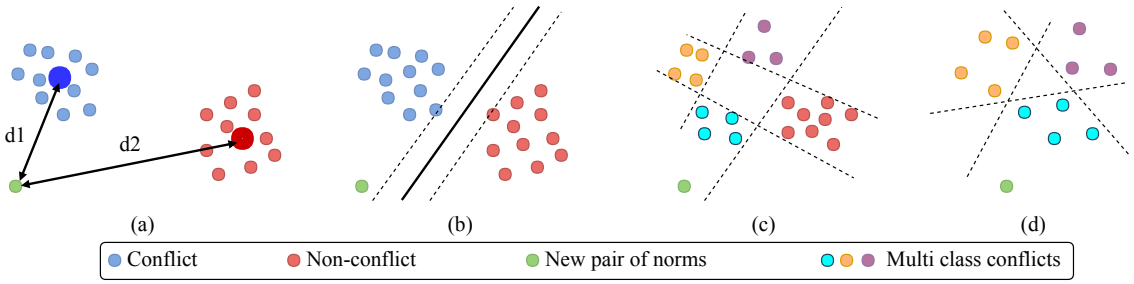


Figure 5.11 – Approaches using binary classification (a) and (b) and multiclass classification (c) and (d).

Conflict Identification by SVM Classification

In our second approach to identify conflicts, we train a Support Vector Machine [27] (SVM) using either the concatenation E_{conc} or the offset E_{off} of embeddings representing pairs of norms. The main idea of this approach is that an SVM creates a hyperplane that tries to maximize the margin between conflicting and non-conflicting norms.

Figure 5.11 (b) illustrates this idea, where blue dots represent embeddings of conflicting pairs of norms, red dot represent embeddings of non-conflicting pairs of norms, the green dot represents the embedding of an unseen pair of norm that we want to classify, and the line between blue and red dots represent an optimal hyperplane, while the dashed line represents the margin of the SVM.

Conflict Classification Using Non-Conflict Norms and Types of Conflicts

The classification of norm conflicts consists of the identification of a conflict type to a conflicting pair of norms. Therefore, we developed an SVM to classify norm pairs into five classes (four conflicts and a non-conflict class) containing either the concatenation E_{conc} or the offset E_{off} of embeddings representing pairs of norms.

Figure 5.11 (c) illustrates the idea of this approach, where red dots represent non-conflicting pairs of norms, yellow, purple, and light blue represents different types of conflicting pairs of norms, and the green dot represents the embedding of an unseen pair of norms that we want to classify. Dashed lines represent the possible hyperplanes that separate the classes.

Conflict Classification Using Only Conflicts

Our dataset (see Section 5.2) is unbalanced in terms of conflicting and non-conflicting pairs of norms, *i.e.*, the number of pairs representing non-conflicting norms is larger than the number of pairs representing types of conflicting norms. Due to this unbalanced nature, we remove the non-conflicting pairs of norms and test the performance of an SVM classifier when trying to divide only conflicting pairs of norms. Thus, this approach classifies the type of conflict (*deontic-modality*, *deontic-structure*, *deontic-object*, or *object-conditional*) of an unseen pair of conflicting norms. This approach shed light on what types are the hardest and the easiest to classify. In order to do that, we train an SVM classifier using either the concatenation E_{conc} or the offset E_{off} of embeddings representing pairs of norms.

Figure 5.11 (d) illustrates this approach, where yellow, purple and light blue represent the embeddings of different types of conflicting pairs of norms, the green dot represents the embedding of an unseen pair of norms that we want to classify, and the dashed lines represent the separation of each class from the other classes.

5.6 Experiments

In this section, we describe our experiments over the approach to identify and classify norm conflicts in contracts. We divide our experiments in two moments. The first one

describes the selection of non-conflicting pairs to perform a fair set of samples to evaluate our approach. The second one describes the techniques we use to classify norm conflicts.

5.6.1 Selecting Non-Conflicting Norm Pairs

As our dataset contains a larger number of non-conflicting pairs of norm when compared to the number of conflicting pairs of norms, we select a subset of the non-conflict pairs to avoid biasing the model towards a non-conflict identification in our experiments. We sample non-conflicting pairs using three different criteria: (a) the closest pairs to the centroid of conflicts, (b) the farthest pairs to the centroid of conflicts, and (c) random selection of non-conflicting pairs. The idea here is to verify the importance of the distance between non-conflicting pairs and conflicting pairs when generating the non-conflict centroid and training the SVM.

To select non-conflicting pairs that are related to the conflicts, we first generate E_{cent} for conflicting pairs. Having this centroid representing conflicts, we can calculate and rank the distance for each norm pair that represents a non-conflict. A E_{cent} representing non-conflicts is generated using this rank. Thus, an unseen pair of norms can be classified by comparing its distance with the E_{off} of both conflict and non-conflict centroids.

In order to generate (a) the closest pairs to the conflicting pairs, we select a subset with the top n ranked pairs representing non-conflicting norms, *i.e.*, pairs with the smallest distance to the centroid of conflicts, where n is the number of pairs representing conflicts in the dataset. Conversely, we select (b) the farthest pairs to the conflicting pairs as a subset with the bottom n ranked pairs representing non-conflicting norms, *i.e.*, pairs with the highest distance to the centroid of conflicts. Finally, we perform (c) random selection of non-conflicting pairs, creating a subset containing n randomly selected pairs.

5.6.2 Classification of Norm Pairs

We measure the performance of our approaches using a k-fold cross-validation using our extended conflict dataset. All our experiments use 10 folds, where 9 folds compose the train set and the remaining fold the test set. Each set contains an equal number of pairs representing conflicts and non-conflicts. We measure the performance in terms of Accuracy ($\mathcal{A}$), Precision ($\mathcal{P}$), Recall ($\mathcal{R}$) and F-measure ($\mathcal{F}$). The configuration of each experiment is as follows.

Conflict Identification by Distance Between Embeddings: This experiment identifies an unseen pair of norms by checking if the distance between the offset of both norms (E_{off})

and E_{cent} of conflicts is smaller than E_{off} and E_{cent} of non-conflicts. In our experiments, we test the Euclidean and Cosine distances.

Conflict Identification and Classification by SVM Classification: For experiments with the Support Vector Machine, we use an off-the-shelf implementation of a linear kernel for norm identification and the implementation of Crammer and Singer [27] for multiclass classification from *scikit-learn*[4] toolbox. We do not optimise any parameter, using the default values from the toolbox (L2 penalty, Square hinge loss, C=1.0). Training step consists of learning an equal number of embeddings representing conflicts and non-conflicts. The test step predicts classes for a norm embedding of unseen pairs of norms.

5.7 Results

Using the extended version of our dataset, we compute the Accuracy ($\mathcal{A}$), Precision ($\mathcal{P}$), Recall ($\mathcal{R}$) and F-measure ($\mathcal{F}$) for all developed approaches. We report the best scores for each approach in Table 5.7, where:

CI-Distance: describes the conflict identification (CI) using the Cosine distance to measure the distance between the concatenated E_n of unseen pair of norms (E_{conc}) and the E_{cent} of conflicts and non-conflicts. This approach maximises the distance between centroids of conflicts and non-conflicts by using E_n of non-conflicting norms that are the farthest to the E_{cent} representing conflicts.

CI-SVM: describes the conflict identification (CI) using an SVM classifier trained with the concatenation of E_n (E_{conc}) representing conflicting and non-conflicting norms.

CC-All: describes the conflict classification (CC) of unseen pairs of norms as non-conflict or any of the four types of conflicts. The SVM is trained using E_{conc} of conflicting and non-conflicting norms.

CC-Conf: describes the conflict classification (CC) of unseen pairs of norms using into one of the four types of conflicting norms, where embeddings are generated by the offset of each norm pair (E_{off}).

Approach	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$
CI-Distance	0.93	0.94	0.93	0.93
CI-SVM	**0.99**	**0.99**	**0.99**	**0.99**
CC-All	**0.78**	**0.78**	**0.79**	**0.78**
CC-Conf	0.65	0.65	0.65	0.60

Table 5.7 – Performance summary, where 'CI' denotes Conflict Identification and 'CC' denotes Conflict Classification.

[4] http://scikit-learn.org

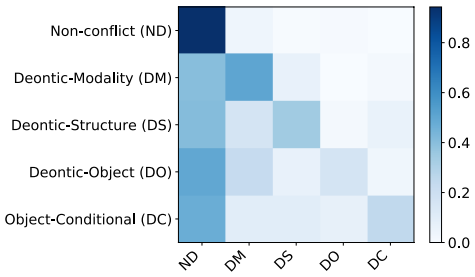


Figure 5.12 – Confusion matrix of the *CC-All* model.

As we can see in Table 5.7, our intent to remove the use of a threshold can be achieved by using an SVM classifier, which achieved 99% of accuracy. Our approach that uses the distance between an unseen pair of norms and the centroids of conflicting and non-conflicting norms also achieves the state-of-the-art results. Unlike our previous approach [1] that uses the offset between embeddings, this approach shows that the concatenation of both embeddings is more promising for conflict identification. When testing the conflict classification using a multiclass SVM, we see that the accuracy decreases to 78% when considering non-conflicts in the classification. Since our dataset is unbalanced, *i.e.*, pairs of norms are not equally distributed into classes, the results may be biased towards the non-conflict class, which contains half of the pairs of norms. Analysing the distribution of correct classifications using the confusion matrix generated using the results for all folds and illustrated in Figure 5.12, we note that most of the corrected classified pairs are non-conflicting norms.

In order to test how our approach performs over the conflict types alone, we remove the non-conflicting pairs and test the *CC-Conf* model, showing that the model achieves 65% of accuracy. We generate the confusion matrix for this model to see how pairs of norm are classified. Figure 5.13 shows that the model tends to classify most conflicts as *deontic-modality*. Observing Table 5.1, we can see that more than half of the pairs containing conflicts (52%) are classified as *deontic-modality*, which may introduce a bias towards this class.

The working example below shows how we manipulate norms from the contract to the classification of the norm conflicts. Consider the contract between Solectron and NCR[5]. This is one of the contracts in which a volunteer included hand-crafted norms meant to conflict with existing ones. The first step in our approach is to break the contract into sentences and classify them to identify those that are norms. From these sentences, we select the following two sentences that are conflicting and show how we transform them from a textual representation into one amenable to machine learning in our approach:

[5]You can find it here: https://github.com/JoaoPauloAires/potential-conflict-identifier/blob/master/code/data/manufacturing/solectron.mfg.1998.04.27.shtml

1. To this end, subject to any confidentiality agreements Solectron may have, Solectron will both inform and provide a commercially reasonable opportunity for acquisition of new and emerging Solectron and industry technology.

2. Solectron can choose not to inform new industry technologies produced by them.

Once we have all sentences, we convert them to a bag-of-words format (word count representation) to create a numerical representation of them. Our bag-of-words is a vector with 3753 positions where each position refers to a word. To represent a sentence, the vector is filled with the number of occurrences of each word in their defined position. The representations of our selected sentences become a sparse vector containing mostly zeros and a few counts of the words.

1. $[0, 0, 0, 0, ..., 1, ..., 3, ..., 0, 0, 0]$

2. $[0, 0, 0, 0, ..., 1, ..., 1, ..., 0, 0, 0]$

We use these vectors as input to a trained Support Vector Machines model that outputs whether the sentence is a norm or not. Given the classification, we select those identified as norms. In our example, both selected sentences are norms.

The next step is to convert the natural language version of the sentences into an embedding that preserves both the syntactic and semantic information from sentences. We use the pre-trained model of the Sent2Vec embedding. This model receives as input the sentences in natural language and outputs a dense vector with 600 dimensions. Our selected sentences are represented as follows:

1. $[0.021708, -0.026507, -0.059721, ..., -0.022589, -0.14392, -0.12648]$

2. $[0.25004, 0.16867, 0.17345, ..., 0.12928, -0.35085, 0.11649]$

Once we have the Sent2Vec representation of both sentences, to check if they are conflicting, we use a Support Vector Machines model that processes the concatenation of them. Thus, first we concatenate them to build the input. The concatenation of both representations results in a vector with 1200 dimensions as follows:

- $[0.021708, -0.026507, -0.059721, ..., -0.022589, -0.14392, -0.12648, ..., 0.25004, 0.16867, 0.17345, ..., 0.12928, -0.35085, 0.11649]$

Using the concatenation as input to the SVM model, we obtain the class of conflict the norm pair belongs. Then, to calculate the confidence of this result, we check the distance from the input to the class margin. The closest to the margin the smaller the confidence. To obtain a value between 0 and 1, we apply the sigmoid function over the distance. In our example, the norm pair represents a *deontic-structure* conflict.

Approach	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$
CI–CNN	0.83	0.87	0.83	0.85
CI–Embedding	0.74	0.98	0.48	0.63
CI-SVM	**0.99**	**0.99**	**0.99**	**0.99**
CC–CNN	0.63	0.59	0.64	0.61
CC-All	**0.78**	**0.78**	**0.79**	**0.78**

Table 5.8 – Comparing our approach to the state-of-the-art for both identification and classification.

5.7.1 Comparing with Previous Work

In order to compare this approach to the previous ones, we use our extended dataset to obtain the result from existing approaches. We divide tasks into identification (CI) and classification (CC) and evaluate the approaches that can fit to them. For the identification task, we can use both CNN [3] and embedding [1] approaches. Table 5.8 shows the direct comparison between the results of existing approaches and our best result (CI-SVM) for the task. As we can see by the results, our approach surpasses existing ones applied in our dataset.

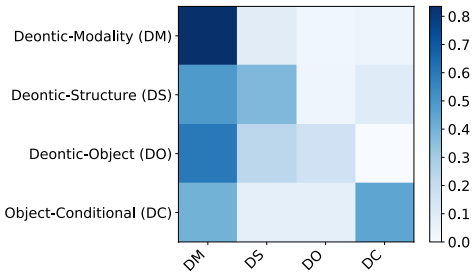


Figure 5.13 – Confusion matrix of the *CC-Conf* model.

For the classification task, we can only use the CNN approach since the embedding approach [1] can only be used to identify conflicts. We modified the last layer of the CNN to perform a multiclass classification. Table 5.8 shows a comparison between approaches, where $\mathcal{A}$ stands for accuracy, $\mathcal{P}$ stands for precision, $\mathcal{R}$ stands for recall, and $\mathcal{F}$ stands for F-Measure. As we can see, our new approach obtains higher results for both tasks when compared to the previous ones. This can be explained by our combination of embedding representation and the use of an SVM instead of depending on a threshold to decide whether a norm pair is a conflict or not.

5.8 ConCon

The task of norm conflict identification and resolution in a contract is commonly performed by humans that understand the contract and are able to modify it. This is often a task for contract creators, which are responsible for defining the terms of a contract. However, dealing with contractual norms can be complex as they describe specifications and conditions that may conflict with other definitions. Identifying each case may take time and lead to contract inconsistencies due to unidentified conflicts.

One way to facilitate the process of identifying norm conflicts is by automating or semi-automating it. The automation can provide indications of potential norm conflicts without the assistance of contract creators. By using such indications, we shorten the time spent in the verification of contracts. One can create different levels of automation that may include the automation of norm conflicts resolution, which would free humans from the task. However, the creation of a system that automates these processes demands a huge volume of norm conflict examples, which can be hard to find.

As a way to semi-automate the task of norm conflict identification, we propose, in this thesis, a norm conflict classifier. Our goal is to provide hints about potential conflicts and their types to contract creators. With such information, identifying and resolving norm conflicts becomes faster and simpler. In order to make it available to the public, we converted our norm conflict classifier into a web-tool called ConCon (see ConCon's home screen in Figure 5.14). ConCon provides an interface that allows users to upload contracts and obtain indications of potential conflicts with a degree of confidence. We make it publicly available through http://lsa.pucrs.br/concon. Our goals with such web-tool are two-fold: (1) allow people to use our norm conflict classifier; and (2) obtain annotated data with the help of users.

In order to use our web-tool, users must register by providing basic information, such as username and email, and accept our terms of use. We use this information to track contracts provided by users as well as contributions made by them. Once registered, each user will have a profile to store and manage uploaded contracts, and the permission to use our conflict identifier. In the "Conflict" section, we invite users to upload personal contracts to check norm conflicts or test our models by selecting a contract from our repository. After uploading a contract, users must select one of our models to process the contract, as shown in Figure 5.15. Currently, we have three models that perform norm conflict identification in three different ways. The aim is to allow users to check which one yields the best performance on finding conflicts. We store all processed contracts in the "Profile" section, where users can test different models over the contracts or remove them. By processing a contract with one of the models, ConCon displays a new screen containing a drop-down menu and a table with two columns. The drop-down menu consists of all norm conflict occurrences detected

by the model in the contract associated to a confidence score. In the table, ConCon displays the contract sentences in both columns. By clicking on one of the options in the drop-down menu, the norm pair annotated as a conflict will be highlighted and centred on the screen. Figure 5.17 illustrates the process of displaying norm conflicts in ConCon. The top bar is a drop-down menu containing all potential conflicts. We display the norm ids (these ids refer to the identification in our database) followed by a confidence rate given by our models. Such confidence is the model output to the conflicting class. Thus, giving the model we applied, we have different ways to calculate the confidence. In the CNN approach, we calculate the confidence by taking the softmax output from the network. In the approaches using only the distance (*e.g.* CI-Distance), we scale the distance in a value between 0 and 1 to extract the confidence. When the distance is 0, we have 100% of confidence. SVM can indicate the distance of a sample from the class margin. Using such distance, we apply the following formula to calculate the confidence: $1 - sigmoid(distance)$ When selecting one of the items in the drop-down menu, the pair of norms become read and are centralised on the screen.

ConCon's three models have specific ways of processing contract norms. They are the result of three different techniques of conflict identification developed along the MSc and Ph.D. The three models are: the rule-based model, the CNN model, and the 'embedding' model. The rule-based model is the result of my master degree [4], which uses a closed norm structure and consider a few norm conflict cases (precisely the first and second norm causes defined by Sadat-Akhavi [93], check Section 2.4 for more details). The CNN model is the result of one of our experiments during this Ph.D. [2], which we detail in Section 5.5.1. Finally, the 'embedding' model is the result of our final approach described throughout this chapter. As a way to improve our dataset, our second goal with ConCon is to ask users to help us with their knowledge to create a set of conflicting norms. Users contribution is two-fold: (1) for each conflict our models identify, users can mark it as mislabelled; (2) if users find a non-identified conflict, they can select both norms in our interface and mark them as conflicts. Figure 5.16 shows the options users have to correct conflict identification.

Regarding public access to our web-tool, we have received a few visits so far. A total of 16 people have registered as users, but only 8 have uploaded a contract. We believe the low public access has to do with the need to register to use. Thus, as future work, we aim to provide basic access without this requirement. Then, if people get interested, they can register to have full access. Such modification will increase access to the tool.

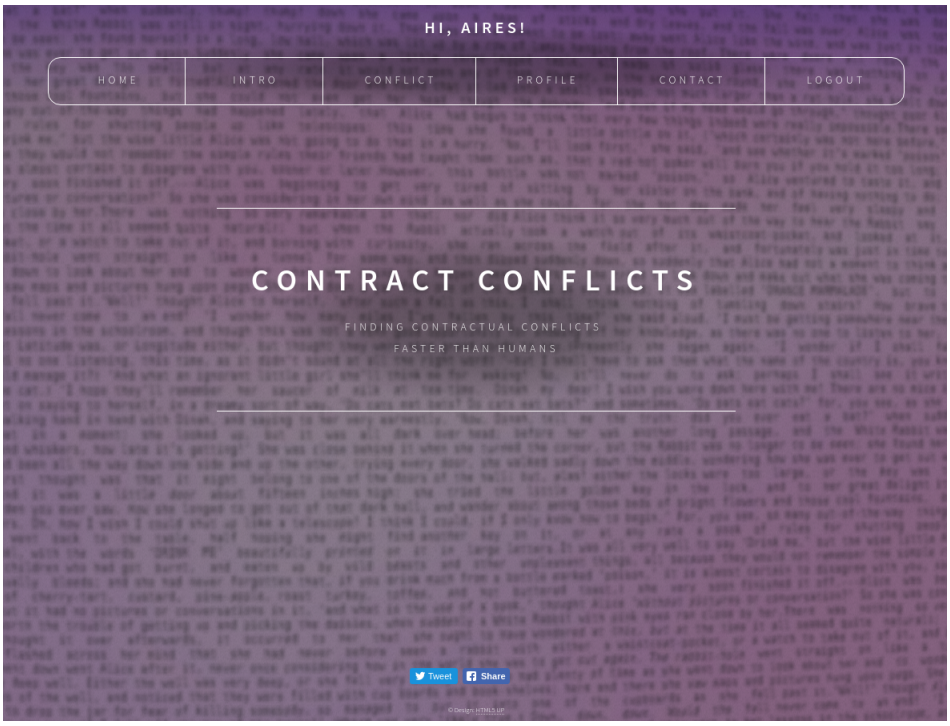


Figure 5.14 – ConCon's home screen

Figure 5.15 – ConCon screen for contract upload

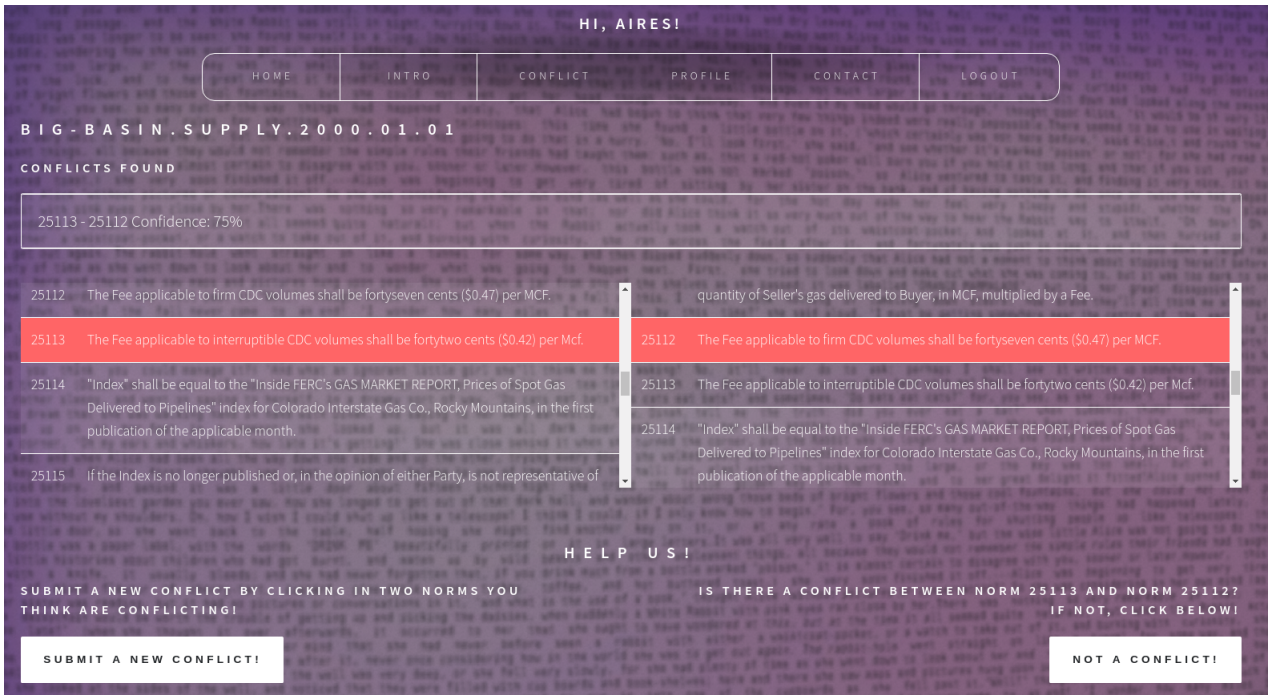


Figure 5.16 – Users can use both the ‘Not a Conflict!’ button in the bottom right of the screen to remove a mislabelled conflict and the ’Submit a new Conflict!’ button in the bottom left of the screen to add a non-identified conflict

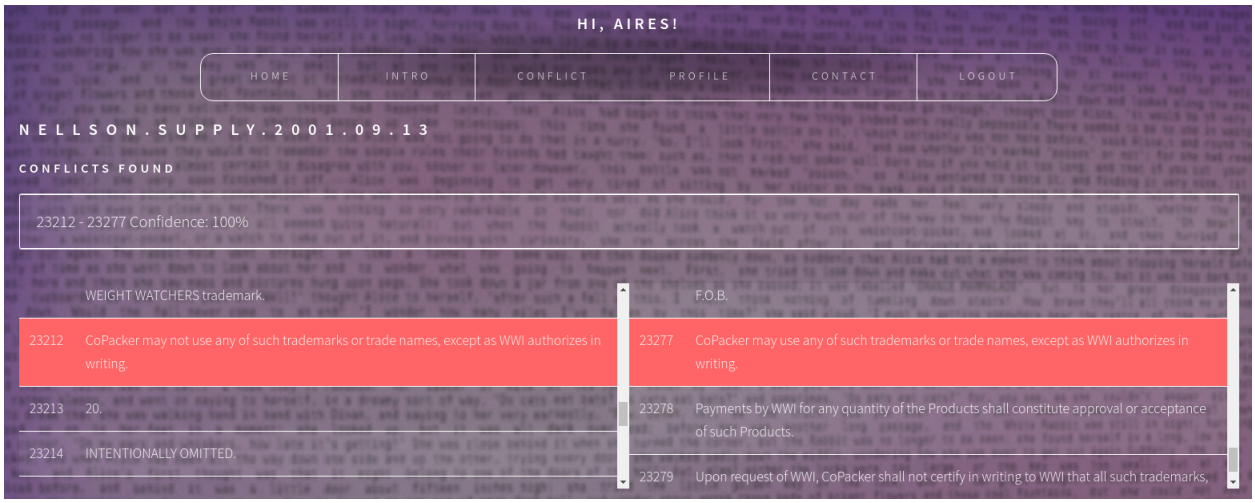


Figure 5.17 – Conflict identification screen

6. RELATED WORK

In this chapter, we revise existing work that perform tasks related to the ones we use in our thesis. The process of norm conflict identification involves a series of tasks (see Chapter 4), such as information extraction, text classification, and conflict identification. We analyse and compare existing approaches in each of these tasks and contrast to how we decided to deal over them. We divided this chapter into four sections according to the tasks and subjects related to our work. In each section, we describe work that perform different techniques to solve the tasks.

6.1 Information Extraction in Contracts

In this section, we discuss existing work on information extraction in contracts. Information extraction in contracts refers to the automatic identification of contract elements, which are the basis for more specific tasks, such as norm conflict identification. Elements such as party names, norms, dates, and modal verbs are central to start reasoning over a contract.

Contracts are semi-structured documents (see Section 2.3), therefore, unlike raw text, we have landmarks that can help us to identify general contract information, such as parties description and contract summary. However, identifying specific elements, such as parties, norm clauses, and content lines may require sophisticated techniques. Curtotti and McCreath [28] developed an approach to classify components of contract texts. They defined 32 classes of components that represent 'lines' in contracts. Such components are structural elements, such as clause headings, content lines, and metadata, such as parties, dates, emails, and addresses. Their approach consists of three main steps. First, they use a hand-coded tagger to label contract lines that were later manually corrected. Second, they use a hand-coded feature extractor to extract features from the labelled data. Finally, they use the extracted features as input to a machine learning algorithm. Thus, using both rule-based and machine learning algorithms, they create a model to classify contract 'lines' as components. As data for their experiments, they use the Australian Contract Corpus [29]. This dataset consists of 256 contracts, a total of 42,910 sentences, and a vocabulary of 14,217. In their experiments, they selected 30 contracts from the corpus. They annotated this set of contracts and use them as a train/test set. As result, they obtain an average accuracy of 83.48% in their multiclass task. In our approach, we identify norms from contracts using a machine learning approach. The main difference between our work and theirs is that they focus on specific contract components that break norm clauses into smaller elements.

Gao *et al.* [42] developed Contract Miner: a tool to detect service exceptions in contracts. In their work, the authors describe service exceptions as exceptions that define a condition applied to norm clauses. When the condition holds, it activates the norm. Exceptions follow a pattern of representation in natural language with expressions such as: “in case of”, “if”, “in the event of/that”, among others. Example 6.1.1 illustrates a service exception in a norm. The goal of Gao *et al.* in this work is to identify what follows the exception pattern, *i.e.*, what is in italics in Example 6.1.1. To these service exceptions, the authors employ linguistic patterns and parsing techniques. Their process consists of four main steps. In the first step, they preprocess the contract to remove HTML tags and other noises and break the text into sentences. As a follow-up step, they extract sentences containing exception patterns using the linguistic rules. In the third step, they use an off-the-shelf natural language parser to build noun phrases (see Section 2.6) from the selected sentences. Finally, they identify those noun phrases corresponding to service exceptions. Trying to improve their rules, they formulated a conjunction rule that considers that if a noun phrase is connected with an exception phrase by a conjunction, the noun phrase is likely to be an exception phrase. To identify noun phrases, they created an algorithm that defines a pipeline to process the sentences. In their experiments, they use a dataset containing 2,647 contracts obtained online through the Onecle repository[1]. They manually selected contracts from different categories and organise them accordingly. These contracts were not previously annotated, thus, they selected five contracts from the same category and manually annotated the service exceptions in them. From such annotation, they obtained 24 matching sentences (containing service exceptions). They evaluated their approach comparing different linguistic patterns and the use or not of the conjunction rule. As result, their approach obtains an F-Measure of 90%. Identifying service exceptions is important when trying to identify norm conflicts. Our alternative in such case is to use a sentence embedding that agglutinates all the information in a dense vector. Besides, we do not have a specific evaluation of service exception identification in our approach.

Example 6.1.1. :

- **In case of** *any defect in Serviced Products*, Fujitsu shall, at Fujitsu’s option, 1) rework the applicable Serviced Products, or 2) issue a credit to FASL.

Gao and Singh [41, 40] produced two approaches for the identification of norms and their elements in natural language contracts. They identify occurrences of business events and temporal constraints, and identify norms and classify them in such contracts, as follows. In their first work [41], they developed an approach to identify business events and their temporal constraints from contracts. These events are purchases, deliveries, bill payments, bank interest, among others. Example 6.1.2 (extracted from Gao and Singh) illustrates some of these events. Their approach consists of five steps. In the first one, they preprocess

[1] https://www.onecle.com/

the contract texts removing HTML tags and breaking it into sentences. Second, they filter the contract sentences to remove those referring to definitions and postal codes. Third, they parse and prune the resulting sentences from the second step to generate candidate business events and temporal constraints. Given these candidates, they apply a machine learning algorithm using specific features from these sentences to identify true events and temporal constraints. As an extra step, their fifth step consists of identifying hidden event topics by using a topic modelling approach. This approach tries to infer the topic given the text information. To create a dataset, they use a series of contracts from the online repository of contracts Onecle. In their experiments, they obtain an f-measure of 89% in the event identification and 90% in temporal constrains extraction. In our approach, we avoid the use of parsing and learning algorithms to extract elements by relying on the representation of sentence embeddings.

Example 6.1.2. :

- 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 a licensed product.

In their second work, Gao and Singh [40] developed an approach to extract normative relationships from contracts. The authors define as normative relationships six main types: commitments (both practical and dialectical), authorisations, powers, prohibitions, and sanctions. These six norm types differ regarding their relationship between the subject and the object defined in the norm. Example 6.1.3 (extracted from the authors work) illustrates two of these types. In the authorisation type, we can see that the subject has the right (authorised) to use the test results internally. In the prohibition type, we see that the subject shall not (prohibition) issue a press release that mentions Sharp without previous consent. Their approach follows a similar path as the previous ones having four main steps. First, they preprocess the text to obtain only valid sentences from the contract. Second, they compute the candidate sentence using a set of modal verbs and other signal words. From these candidate sentences, they extract features using an existing language parser. The parser extracts part-of-speech, phrase chunks, and dependencies between tokens. Finally, they train a machine learning algorithm to classify norms using the extracted features. In their approach, they use both rule-based and machine learning approaches as, according to them, it improves the final result. For this work, they use the same set of contracts from the previous work. In this particular case, they manually annotated 66 norm sentences in a random contract. They obtain an average f-measure over all normative relationships of 84%. In our approach, we use a classifier to select norm sentences from contract sentences. We show that using simple text representation we can attain considerable results. Unlike the

authors, we do not need to previously classify norms as we compare all norms against each other.

Example 6.1.3. :

- **authorisation**: Danger will have the right to use the test results internally for product management and planning purposes.
- **Prohibition**: Danger shall not issue a press release mentioning Sharp without Sharp's prior consent, which shall not be unreasonably withheld or delayed.

Wyner *et al.* [105] introduce a controlled language to convert legal natural language into a machine-parsable form. By controlled language, the authors refer to an annotated language that a machine can manipulate and understand. The proposed controlled language describes a defined number of language features described in the following list.

- Deontic modalities: The donations **must** be labelled...
- Conditional rule statements: **If** you pay before the date..., (**then**) you must assure...
- Main verbs: You are not required **to test**...
- Semantic roles, *e.g.* Agents and Themes: **You** shall not move **blood components**.
- Lists: ... unless you meet **the following conditions: (A)...; (B)...**
- Exceptions: **Except on Fridays**, you must pay taxes.
- Anaphora to various referential types, *e.g.* NPs, lists, and exceptions: You must move blood components according to the required under paragraph **202.131**.

The authors use an NLP parser and a series of rules to extract each of these elements. They simplify complex terms into simpler ones, extract part-of-speech tags, noun, verb, and prepositional phrases. Sentences are then simplified and normalised using indicators that highlight relations between clauses, *e.g.* subordination and list structures. Finally, they perform anaphora resolution by linking expressions to their antecedent definitions. To test their approach, they use a passage from the US Code of Federal Regulations. It refers to the US Food and Drug Administration and contains two sentences. As result, they obtained an annotated passage according to their definitions. Since this is a partial study, their results are still preliminary. Once established, we can use this approach to contrast ours in the conflict identification task.

More recently, Chalkidis *et al.* [18] proposed an approach to extract contract elements automatically. They define eleven contract elements to be extracted: Contract title, contracting parties, start, effective, termination dates, contract period, value, governing law,

jurisdiction, legislation refs, and clause headings. To identify such elements, they define zones where elements can be found. Given these zones, they use machine learning algorithms combined with hand-crafted features, part-of-speech (POS) tags, named entity recognition, and embeddings to identify the elements. In order to train their algorithms, the authors created a dataset of around 3,500 English contracts annotated. In order to evaluate their approach, they compare a series of different learning algorithms. As results, they obtain an f-measure of 80% in a macro-average among all contract elements. Although these elements can help to automate daily tasks for contract drafters, to identify conflicts we are interested in identifying specific information from norms.

A similar work by Chalkidis and Androutsopoulos [17] modify the work of Chalkidis *et al.* [18] to use deep neural networks, such as LSTM and BILSTM over the same zones as the previous work. Their work focus on the identification of the same set of elements defined by Chalkidis *et al.*. They test over different input formats involving pre-trained words, POS tags, and token-shaped embeddings. The authors use the same dataset as Chalkidis *et al.* [18] in their experiments. Using deep neural networks, they obtain an f-measure of 87% in the macro-average among all contract elements.

Pinheiro and Meneguzzi [83] developed an approach to automatically extract and formalise contract components. The authors define six norm components and three contract components to extract. The norm components are party, action, condition, subject, object, and deontic modality. Contract components are title, parties, and parties alias. They use a syntactic and semantic parser to process contractual norms and extract the targeted objects. In order to evaluate their work, the authors manually annotated 15 contracts. They annotated a total of 1217 norms and 3855 sentences. As result, they obtained an average f-measure of 77% considering all elements.

The approaches surveyed in this section allow us to compare and contrast with ours. Curtotti and McCreath create a classifier to identify contractual structures from natural language text. On the other hand, Gao and Singh have three approaches, (1) detection of business events and their temporal constraints, (2) identification of norm exceptions, and (3) identification and classification of norms. Wyner *et al.* propose a controlled language to represent contract components. Chalkidis *et al.* use machine and deep learning to detect specific elements in a contract based on contract zones. Finally, Pinheiro and Meneguzzi extract contract and norm elements from contracts using a syntactic and semantic parser. The main difference between our approach and theirs is that, instead of relying on regular expressions and machine learning models to identify specific elements, we use embeddings to represent legal text. Using such embeddings, we rely on the vector representation of natural language that is the result of training a model to convert sentences into dense vectors (see Section 3.3). These vectors contain both syntactic and semantic information from sentences, which facilitates the manipulation and classification of contract elements.

6.2 Legal Text Classification

The task of classifying contract texts is important to improve contract drafters daily activities. From classified information, one can develop contract querying systems, correction systems to improve writing, as well as norm conflict warnings. The following work describe approaches that try to classify contract elements in order to facilitate for contract drafters and readers to manipulate contracts.

Gabbard *et al.* [39] developed a system that helps lawyers on the contract drafting process by alerting about missing contract components and giving feedback to improve contract quality. The authors define a four-step approach. In the first step, they preprocess the text to extract textual features using a natural language processing tool. Second, the resulting features are the input to a hybrid approach with machine learning and knowledge-engineered rules to define the type of contract. Finally, a verification module checks the components of the contract against a gold standard and give scores according to the presence of them. The authors manually annotated a set of contracts for their task. They performed some empirical experiments, but they are inconclusive.

Lippi *et al.* [66] developed a tool called CLAUDETTE to identify potentially unfair clauses in online contracts. They define the occurrence of an unfair where there is a significant imbalance in the parties rights and obligations, which cause a detriment of the consumer. In order to classify such clauses, they selected eight categories: Arbitration, Unilateral change, Content removal, Jurisdiction, Choice of law, Limitation of liability, Unilateral termination, Contract by using. To represent sentences from the contract, they use the bag-of-words technique. Such representations become the input to machine learning algorithms to classify full sentences or paragraphs that contain unlawful clauses. To train their approach, they use a manually annotated corpus containing 50 contracts with more than 12,000 sentences. As result, they obtained an f-measure score of 80% using an ensemble of five classifiers.

Chalkidis *et al.* [19] describe an approach to detect both obligation and prohibition norms in contracts. The authors define six classes to represent all possible cases: None, Obligation, Prohibition, Obligation list intro, Obligation list item, and Prohibition list item. By list intro and item, they refer to cases where the norm is broken into items, which makes it multiple norms in a single description. They use deep learning algorithms as BILSTM with self-attention model and hierarchical LSTM to detect the deontic meanings. In order to convert the legal text as input to the learning algorithms, they used pre-trained 200-dimensional word embeddings and 25-dimensional POS tag embeddings using Word2Vec [71]. As data, they randomly selected 100 English service agreements, which resulted in 6,385 training, 1,595 development, and 1,420 test contract sections. In their results, they obtain an f-

measure score higher than 80% for all classes with a macro-average of 89% and a micro-average of 95%.

Gabbard *et al.* describe an approach to identify missing elements in a contract and help lawyers in the process of creating contracts. Lippi *et al.* uses learning algorithms to identify potential unfair clauses in contracts. Chalkidis *et al.* detects obligation and prohibition norms in contracts. All these approaches are examples of automation of contract processing, which can be correlated to our work. The main difference between them and our research is the task. Our classification is focused on the identification and classification of norm conflicts according to types. From these work, the one by Chalkidis *et al.* could be included as part of ours since we could use their work to identify norms in contracts.

6.3 Norm Conflict Identification in Multi-Agent Systems

In our work, we propose an approach to identify and classify norm conflicts in contracts. Although this is quite new for contracts written in natural language, in multi-agent systems, many approaches deal with norm conflicts. In order to analyse how they solve this problem in this scenario and check for possible solutions in ours, in this section we describe some existing work in this topic.

Figueiredo and Silva [30] propose an algorithm for identification of conflicts between norms. Their goal is to create agents in a multi-agent system that identify conflict cases between norms and values. They understand value as concepts about desirable end states or behaviours. To specify actions and norms, they use a language named Z. 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, they define some particular cases, such as action refinement, which occurs when a super action has one or more subactions. Conflicts arise when a norm states an obligation to the agent to execute a super action and all its subactions demote an important value to the agent. When a norm states a prohibition to execute a super action and at least one of its subactions promotes an important value to the agent. Finally, they describe 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 created an algorithm to identify conflicts between norms formalised in Z language.

Vasconcelos *et al.* [102] introduce an approach to deal with normative conflicts in multi-agent systems. They developed mechanisms for detection and resolution of normative

conflicts. They use a formal representation of norms with constraints, and introduce formal definitions of normative conflicts and defining how they can be resolved. Moreover, they propose mechanisms for 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 norm constraints, they build a normative representation. For the task of detecting a conflict, they define 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 with 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.

Li *et al.* [64] propose an approach to detect conflicts in composite institutions. They define a 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 behaviours, without considering that the participants will comply with such normative behaviours. 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 policies 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 policies. To detect conflicts, they define an event-driven model in which events derive from the actions of participants in the system. Thus, a composite institution is conflicting if exists a composite trace that is inconsistent for two corresponding synchronised 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.

Approaches involving the detection of normative conflicts in agent scenarios can help us to identify similarities between certain formalisation. Among the described work, Figueiredo and Silva and Vasconcelos *et al.* propose an algorithm to identify norm conflicts. Vasconcelos *et al.* propose a method for resolving normative conflict, which is a challenge for natural language norm conflicts. Li *et al.* propose an approach to detect conflicts in a specific case of composite institutions.

6.4 Norm Conflicts in Natural Language

The identification of norm conflicts in contracts written in natural language involves the conversion of natural language to a representation that allows the computer to process and find such conflicts. Existing approaches have worked to convert norms into formal representations and make it possible to identify conflicts between them.

Rosso *et al.* [89] propose an approach to retrieve information from legal texts. Their approach uses JIRS[2] (Java Information Retrieval System), a system that measures distances between sentences using n-grams, to develop a solution for three problems: passage retrieval in treaties, patents, and contracts. In the first problem, they want to answer questions from treaty documents. Given a question about the content of the treaty, they use JIRS to measure the distance between the question and the text in the treaty, thus, they can rank the best answers to each question by their similarity. To the second problem, they develop an approach to help patent creators identify similar patents. As in the first problem, given a set of patents and a new one, they use JIRS to measure how similar the new patent is to existing ones. To the third problem, they develop an approach to identify conflicts between norms in contracts. They divide the process of conflict identification into three steps. First, they translate every norm in contract to a formal contract language ($\mathcal{CL}$ [36]), which they call Contract /Language clauses. Second, they analyse the clauses using a model checker performed by the contract analysis tool *CLAN* [37]. From the identified conflicts, they use JIRS to translate the sentences from $\mathcal{CL}$ to natural language. In order to evaluate their conflict identifier, they translate a set of 10 norm clauses into their formal contract language. Given the translated norms, they empirically evaluate the resulting conflicts identified by their approach.

Azzopardi *et al.* [9] propose a tool to provide intelligent contract editing to lawyers by using off-the-shelf NLP techniques. They extract information from clauses to make intelligent browsing among the clauses. The authors use a mix of regular expressions and named entity recognition to identify the clauses parties. If they fail in the identification, the tool user can specify the parties of the contract. Once identified the parties, the user can navigate through the norms in which the parties are mentioned. As an extension, the tool allows the user to browse through other existing contracts to find cross-references. In order to make the connection between natural language and a formal language, the authors propose a deontic logic representation that allows them to identify conflicts. To translate the natural language to the deontic logic representation, they use syntax trees and rely on the noun and verb phrases to identify clause elements. With the translated language, they use the following norm conflict definitions to identify conflicts:

- Permission and prohibition of the same action;

[2] http://sourceforge.net/projects/jirs/

- Obligation and prohibition of the same action;
- Obligation of two mutually exclusive actions; and
- Obligation and permission of mutually exclusive actions.

As gold standard, they manually-tagged 5 contracts by converting elements into their proposed representation. They evaluate their work in two ways: (1) testing the English to deontic logic representation over a set of contracts; (2) obtaining feedback from specialists that used the tool. In the evaluation (1), they selected random contracts from the Australian Contract Corpus [28]. In the results, they obtained 62% of F-measure.

In a follow-up work, Azzopardi *et al.*[10] introduce a formal approach to represent contracts in natural language in incomplete domains. Their algorithm uses the same deontic logic representation from the previous work to deal with contracts containing unparsed parts. By translating the natural language to the deontic logic, they can manipulate norms in order to detect conflicts. In order to test their approach, they introduce a controlled contract as a case of study. They test such approach in a controlled scenario using a conflict identifier over a parsed contract with unknown elements. As result

Aires *et al.* [4] developed a rule-based system to identify norm conflicts in contracts. They use rules to identify norms from contracts and then break them into three elements: party name, modal verb, and norm actions. Once they have all norms and their components, they compare the components in order to identify conflicts. In order to identify cases where norm pairs describe the same action but with different meanings, they developed an algorithm to calculate the semantic similarity. To evaluate their work, they manually annotated the norm conflicts from real contracts. As result, they obtained an accuracy of 78%.

Norm conflicts can often be mistaken as contradictions. Li *et al.* [63] introduce an approach to detect contradictions in natural language. An example of contradiction can be the existence of two sentences, such as (1) Vehicles are in a crowded street.; and (2) Vehicles are in an empty street. These sentences contradict themselves since “crowded” and “empty” have opposing meanings describing the same situations. In order to detect contradictions, they create a contradiction-specific embedding to convert the natural language to a latent representation that focuses on the differentiation of contradicting terms. To perform the detection of contradictions, they train a semantic relation representation learning model using their embedding as input. As data for such training, the authors automatically generated a set containing 1.9 million contrasting pairs and 1.6 million paraphrase pairs. Their best model obtains an accuracy of 82% on the detection of contradictions. Although the general concept of contradiction may also describe conflicts, the specific task of norm conflict identification differs from contradiction detection. Neither a norm conflict identifier can detect contradictions nor a contradiction detector can identify norm conflicts. As we describe in this thesis, norm conflict identification depends on specific characteristics of norms that we only

find in norms, such as deontic meaning and law terms. Therefore, general contradictions do not help on improving our norm conflict dataset, neither the classification of norm conflicts.

Among the existing work involving the identification of normative conflicts in natural language, most of them involve the use of a controlled language. Rosso *et al.* manually translate the natural language to a formal contract language that makes conflict detection. Using the same logic, Azzopardi *et al.* propose an automatic approach to translate the natural language to a deontic logic representation that allows them to identify certain types of conflicts. Our work has two main differences when compared to Rosso *et al.*. First, our work tries to identify normative conflicts dealing directly with natural language, whereas in their work they use the approach proposed by Fenech *et al.* [37], which uses a single contract that has its norms *manually translated* into the controlled language $\mathcal{CL}$. Second, to identify norm conflicts, *CLAN* uses a series of predefined rules, whereas in our approach we rely on learning algorithms over natural language embeddings that convert norms into dense vector automatically extracting syntactic and semantic information that helps on norm conflict classification. The proposed contract representation by Azzopardi *et al.* has varied goals. In their evaluation, the results show that they still do not have a concrete approach. Their conflict identification, thus, is a description of how they can use their representation to this end. However, they did not check it separately, which prevents us to compare our approach directly to theirs. Aires *et al.* [4] uses a rule-based approach to identify norm conflicts. We compare this work against ours showing that the use of recent techniques can improve the identification and surpass limitations of rule-based approaches.

7. CONCLUSION

The automation of contract processes is a subject that has been obtaining attention recently. A large number of approaches develop techniques to process contract information and make the manipulation and evaluation easier for contract drafters. Among these approaches, the automatic identification of normative conflicts in contracts has the potential to accelerate the process of contract review. In this thesis, we have addressed the identification of normative conflicts and the classification of them into conflict types. By applying a series of techniques of text representation and learning models, we obtain the state-of-the-art in both tasks. In this chapter, we describe our thesis in a general way dividing it into four sections. First, we summarise our work along with the chapters of the thesis by pointing out the main elements in each of them in Section 7.1. From the summary, in Section 7.2, we detail the contributions of our work. Then, in Section 7.3, we describe the limitations identified in our approach and the potential future work. Finally, we conclude by discussing the main aspects of our work, the techniques applied, and the impact it has in Section 7.4.

7.1 Thesis Summary

The identification of norm conflicts in contracts is a vastly discussed problem in the literature [93, 20, 67]. Resolving conflicts depends on a manual analysis of the contract, its norms, and meanings. In this PhD thesis, we developed an approach to classify conflicts between norms in contracts written in natural language. The process of creating a norm conflict classifier demands the use of natural language processing techniques and a study of the structure and elements of norms.

In our research to find ways to automatically classify norm conflicts, we describe, in Chapter 3, learning models that can process natural language and classify it and techniques to represent natural language to use as input to these models. Learning models have the property of finding patterns in data by using processing data features. We cover the most common models, such as Support Vector Machines, Convolutional Neural Networks, and Recurrent Neural Networks. Each model has its specific characteristics that we can employ to different problems. Since our work deals with the manipulation of natural language, we discuss the ways to represent text for processing by learning algorithms. The state-of-the-art approaches involve converting natural language into latent representations, which carry syntactic and semantic information in a vector structure.

To understand the problem of norm conflict classification, in Chapter 4, we introduce the main modules that one can usually employ to deal with the problem. We divide norm conflict classification into three main modules. The first module processes the raw text

from contracts to identify contract sentences referring to norms. Since the goal is to classify norm conflicts, first, we need to identify norms within the text. Then, in the second module, it is necessary to create a representation of these norms. This module applies techniques to convert natural language norms into a representation that allows a computer to manipulate it to find conflicts. Finally, the third module applies a technique to compare norms to identify and classify conflicts between norms. This module can involve the use of rule and machine learning-based approaches that find patterns among the elements of norms indicating conflicts.

Once we have defined the general process of classifying norm conflicts, in Chapter 5, we describe our approach. Using the three-module architecture, we detail the steps towards the conflict classifier. In the first module, the norm identifier, we employ a sentence classifier able to differentiate a non-norm sentence from a norm sentence. We manually annotated a public available dataset containing examples of norm and non-norm sentences to train our sentence classifier. As our second module, we use the state-of-the-art Sent2Vec [80] algorithm to convert sentences into latent representations. These representations can preserve the main features of norms into vectors containing real-valued vectors. Finally, in our third module, we perform a series of manipulations over the representations, and, using a learning model, we train a conflict classifier. The classifier is able to classify norm pairs into five classes: non-conflict, *deontic-modality*, *deontic-structure*, *deontic-object*, and *object-conditional*. In order to obtain data to train our conflict classifier, we manually annotated, with the help of volunteers a public available dataset containing norm and non-norm sentences classified according to the conflict types.

Finally, in Chapter 6, we discuss the recent research related to our contribution by contrasting them to our approach. We divide these approaches into four sections according to their specificity. First, we describe approaches dealing with information extraction over contracts. These approaches focus on the selection of specific elements from contracts, such as party names, norm exceptions, and norm relationships. In the second section, we describe approaches to classify elements of the contract, such as abusive norms, and the quality of written contracts. The third section contains approaches of norm conflict identification in multi-agent systems. We compare how these approaches can be related to the identification of norm conflicts in natural language contracts. Finally, in the last section, we bring approaches that identify norm conflicts in contracts. We show how they manipulate natural language to the task, and we point the main differences between these approaches and ours.

7.2 Contributions

During the process of developing an approach to classify norm conflicts, we produced contributions that we detail in this section. This thesis has four main contributions: (1) the formalisation of four norm conflict types (2) the creation of an approach that receives a raw contract and outputs norm conflicts and their types; (3) the construction of a dataset containing the classification of real normative conflicts; and (4) ConCon: a publicly available web-tool that performs norm conflict classification over users' contracts. Beyond these contributions, during this thesis, (5) we have published the results of our approaches as well as (6) the result of side projects. Thus, we divide this section into six subsections to better describe each one of these contributions.

7.2.1 Norm Conflict Types

Although identifying norms conflicts can make a review of contracts faster, the classification of conflicts can facilitate their resolution. As part of this thesis, we defined four conflict types that describe causes that may have led to the norm conflict. The four types are: *deontic-modality*, *deontic-structure*, *deontic-object*, and *object-conditional*. *deontic-modality* describes conflicts between norm pairs where the only conflicting element is the deontic meaning, *i.e.*, norm conflicts with norms having a single difference: prohibition $\times$ obligation; obligation $\times$ permission; and permission $\times$ prohibition. *deontic-structure* is an extension to *deontic-modality* in which the conflict still occurs due to different deontic meanings, but norms have a different way of representation in natural language, *i.e.*, they describe the same context using different words and structure. This type indicates a case where norms do not seem similar by their words but by their meaning, and, by expressing different deontic meanings, they are conflicting. *deontic-object* describes conflict cases where both norms express the same deontic meaning and the conflicting part is the objects described by them, *i.e.*, the action description of one norm conflicts with the action of the other. Finally, *object-conditional* describes norm conflicts involving a condition or exception in one of the norm pairs. In this type, norms may no be conflicting directly, however, when the condition holds, it may cause a conflict. Using these types, contract creators can find solutions to conflicts by checking their types. Each type suggests a cause for the conflict, which implies a solution to them.

7.2.2 Norm Conflict Classifier

As we have discussed in our thesis summary, this thesis describes the process of creating a norm conflict classifier. Our norm conflict classifier consists of three modules that allow the processing of raw contracts without any further information as input. Its main goal is helping contract creators in the process of contract revision by suggesting norm conflicts in the text and their causes. Unlike most approaches, our norm conflict classifier uses a norm representation that depends solely on the raw text of norms. This representation allows us to rely on the algorithm's efficiency on the selection of important elements instead of manually selecting them. Thus, our approach simplifies the identification and classification by automating some of the processes that could involve rules and lead to misclassifications. Our classifier is publicly available at https://github.com/mir-pucrs/norm-conflict-classification.

7.2.3 Norm Conflict Dataset

The process of creating a norm conflict classifier depends on obtaining a reliable dataset containing annotated examples of norm conflicts and non-conflicting norms. In our approach, we introduce four types of conflicts that must be part of the dataset. To create such a set of annotated data, we had the help of volunteers that manually created conflicting norms to existing ones in contracts following the four indicated types. Our final dataset consists of 228 norm conflicts and 11,329 non-conflicting sentence pairs. The dataset is publicly available at https://github.com/mir-pucrs/norm-conflict-classification-dataset.

7.2.4 Contract Conflicts Tool

As a result of our thesis, we created ConCon: a web-tool that identifies and classifies norm conflicts. ConCon is publicly available and serves both for people to check their contracts to help us on correcting misclassified conflicts. The main goal of our tool is to make it easy to access our norm conflict classifier as well as generate annotated data to improve the training of our classifier. ConCon consists of a series of approaches to identify and classify norm conflicts. Users can select one of the many models we developed and test them over real contracts. Moreover, users can correct mistakes made by our models, which becomes annotated data. Everyone can access ConCon at http://lsa.pucrs.br/concon.

7.2.5 Publications on Norm Conflict Classification

During the PhD, we published a series of papers regarding our advances on the norm conflict identification/classification. Our first approach described in Section 5.5.1, which uses a convolutional neural network to identify norm conflicts was published in two venues:

- João Paulo Aires and Felipe Meneguzzi. A deep learning approach for norm conflict identification. In Proceedings of the Sixteenth International Conference on Autonomous Agents and MultiagentSystems (AAMAS), pages 1451–1453, 2017.
- João Paulo Aires and Felipe Meneguzzi. Norm conflict identification using deep learning. In AAMAS'17 Workshop on Coordination, Organisation, Institutions and Norms in Multi-AgentSystems, COIN 2017, pages 194–207, 2017

The following approach also described in Section 5.5.1, which identifies norm conflicts by manipulating latent representations of norms had a publication in IJCNN-18:

- João Paulo Aires, Roger Granada, Juarez Monteiro, and Felipe Meneguzzi. Norm conflict identification using vector space offsets. In Proceedings of the 2018 International Joint Conference on Neural Networks, IJCNN'18, pages 337–344, 2018.

Recently, our web-tool was accepted for demonstration in AAMAS-19.

- João Paulo Aires, Roger Granada, and Felipe Meneguzzi. Concon: A contract conflict identifier. In Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems, 2019.

Finally, we published in AAMAS-19 our last idea of using norm latent representations for norm conflict classification:

- João Paulo Aires, Roger Granada, Juarez Monteiro, Rodrigo Coelho Barros, and Felipe Meneguzzi. Classification of contractual conflicts via learning of semantic representations. In Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor, editors, Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS'19, Montreal, QC, Canada, May 13-17, 2019, pages 1764–1766. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

The last findings of this thesis have been recently submitted to the Intelligent Systems and Technology journal (TIST).

7.2.6 Publications on Side Projects

As part of a research group, we have published in different areas. In the area of image processing, we have published three papers focusing on action recognition.

- **João Paulo Aires**, Juarez Monteiro, Roger Granada, Rodrigo Coelho Barros, and Felipe Meneguzzi. Temporal Regions for Activity Recognition. In: International Conference on Artificial Neural Networks (ICANN), 2017, Alghero. International Conference on Artificial Neural Networks (ICANN), 2017.
- **João Paulo Aires**, Juarez Monteiro, Roger Granada, Felipe Meneguzzi, and Rodrigo Coelho Barros. Improving Activity Recognition using Temporal Regions. In: Symposium on Knowledge Discovery, Mining and Learning, 2017, Uberlândia. 5th Symposium on Knowledge Discovery, Mining and Learning (KDMiLe), 2017.
- Juarez Monteiro, **João Paulo Aires**, Roger Granada, BARROS, R. C., and Felipe Meneguzzi. Virtual Guide Dog: An Application to Support Visually-Impaired People through Deep Convolutional Neural Networks. In: International Joint Conference on Neural Networks (IJCNN), 2017, Anchorage. International Joint Conference on Neural Networks (IJCNN), 2017.

In the area of automated planning, we have published four papers on the task of goal recognition.

- Leonardo Rosa Amado, **João Paulo Aires**, Ramon Fraga Pereira, Maurício Cecílio Magnaguagno, Roger Granada, and Felipe Meneguzzi. LSTM-Based Goal Recognition in Latent Space. In: Workshop on Planning and Learning (PAL-18), 2018, Stockholm. ICML / IJCAI / AAMAS 2018 Workshop on Planning and Learning, 2018.
- Leonardo Rosa Amado, Ramon Fraga Pereira, **João Paulo Aires**, Maurício Cecílio Magnaguagno, Roger Granada, and Felipe Meneguzzi. Goal Recognition in Latent Space. In: International Joint Conference on Neural Networks (IJCNN), 2018, Rio de Janeiro. 31st International Joint Conference on Neural Networks (IJCNN), 2018.
- Leonardo Rosa Amado, **João Paulo Aires**, Ramon Fraga Pereira, Maurício Cecílio Magnaguagno, Roger Granada, and Felipe Meneguzzi. An LSTM-Based Approach for Goal Recognition in Latent Space. In: Workshop on Plan, Activity, and Intent Recognition (PAIR@AAAI), 2019, Honolulu. The AAAI 2019 Workshop on Plan, Activity, and Intent Recognition (PAIR), 2019.
- Leonardo Rosa Amado, **João Paulo Aires**, Ramon Fraga Pereira, Maurício Cecílio Magnaguagno, Roger Granada, Gabriel Paludo Licks, and Felipe Meneguzzi. LatRec:

Recognizing Goals in Latent Space. In: International Conference on Automated Planning and Scheduling, 2019, Berkeley. Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS), 2019.

Finally, in the area of natural language processing, we have published an approach to the task of sentiment analysis.

- **João Paulo Aires**, Carlos Padilha, Christian Vahl Quevedo, and Felipe Meneguzzi. Deep Learning Approach to Classify Aspect-Level Sentiment using Small Datasets. In: International Joint Conference on Neural Networks (IJCNN), 2018, Rio de Janeiro. 31st International Joint Conference on Neural Networks (IJCNN), 2018.

7.3 Limitations and Future Work

The contributions of this thesis involve different aspects of the processing of norm conflict classification. Each module in our approach uses a specific application of techniques that solves the problem, however, other methods could be used to obtain similar or better results. In order to focus on the main issues to be addressed, we had to limit our scope of work, which results in several opportunities for future work. In this section, we describe the main limitation of our work and the ideas for potential future work.

7.3.1 Limitations

Dataset

In this thesis, we employ classifier models in two tasks, in the identification of norm sentences and the classification of norm conflicts. Both cases involve the use of a manually annotated dataset. As explained in Chapter 3, learning models use annotated data to identify data patterns in its learning process. The datasets we use in our approach have a limited size due to the number of volunteers and the difficulty of the annotation task. Although we ensure that the learning models are generalising to the data in the dataset by using regularisation methods, since we are dealing with natural language, it still can fail to generalise to examples that drastically differ from the ones in the dataset. We can manipulate natural language in many ways that, yet obvious for us, may mislead learning models. A larger dataset could cover some outlier examples and improve the generalisation of our classifier. We created ConCon, our web-tool, as an alternative to obtain annotated data from real contracts and have better training of our models.

Conflict Types

In our analysis to define norm conflict types, we selected a range of common norm conflict causes based on the definitions in the literature [93, 104]. As a result, we defined the four types we use to classify norm conflicts. However, there are other norm conflict causes that we do not cover in this thesis. Norm conflicts may arise from other normative concepts, such as contrary-to-duty cases where a norm is only activated when another one is violated. The activated norm may conflict with existing ones and the contrary-to-duty should be modified. Due to the limitation of time and volunteers to create conflicts, we decided to define four types that cover the most common causes.

Norm Representation

The use of Sent2Vec to create a norm representation allows us to manipulate the meanings of natural language expressions helping us on the classification of norm conflicts. Although this was the state-of-the-art approach of natural language representation by the time we started our work, further experiments could have been employed. In our experiments, we use the pre-trained models to Sent2Vec that consist of Wikipedia text from a diverse range of articles. A Sent2Vec trained over real contracts could generate a better representation of norms that could improve our results. By taking the pre-trained model and fine-tuning it using a dataset of contracts, the comparison of norms could be refined to a more precise level.

7.3.2 Future Work

Once we have discussed the limitations in our approach, we can propose ways to extend it. Our approach is the first step for the automation of contract processes. Each module in our approach can be improved to consider a wider range of possibilities. This approach can also be part of a bigger system that makes a general analysis of contracts. We selected four ways one can extend our approach.

7.3.3 Improvement on Norm Representation

By the time developed the approach used of this thesis, Sent2Vec [80] was the state of the art in terms of natural language representation. Immediately before the submission of this thesis, the BERT (Bidirectional Encoder Representations from Transformers) [31] proved to surpass the Sent2Vec representation [98, 32, 106], which can represent natural language

for specific tasks making the representation powerful to deal with corner cases and context specifications. We aim to use BERT as future work in order to test how it can improve the results of our work.

Resolution of Norm Conflicts

Our approach is the first one to propose a classification for norm conflicts in contracts. We introduce four types that describe causes for conflicts to occur. Once we have such a definition, the next step would be the automatic resolution of norm conflicts. Thus, using our classification, a separate approach could search for ways to resolve the conflict. The idea is to modify, delete, or add a norm to resolve the conflict. Such an approach would involve the study of norm conflict resolution and text generation techniques.

7.3.4 Extension of Conflict Types and Dataset

As discussed in our limitations section, our proposed set of norm conflict types was limited to the most common ones. Similarly, the dataset we use contain a limited number of examples of these conflict types. Thus, one way to improve our approach is to list all possible conflict types and creating a robust dataset containing a diversity of conflict examples. By creating new conflict types, we ensure the identification of corner cases and improve the classification that can lead to a better norm conflict resolution. Increasing the dataset improves the learning model capacity of generalising to more cases. Such a task would demand human effort to annotate a large number of conflicts.

7.3.5 Automatic Processing of Contracts

The identification and classification of norm conflicts are only one of many processes one can automate in contracts. An extension of our approach could be its integration to a system that unifies the automation of other contract processes. As we have detailed in Chapter 6, there is great effort to automate the process such as the identification of unfair clauses, missing information in contracts, among others. The unification of these efforts could create a single architecture able to process raw contracts facilitating the manual analysis of contract creators.

7.4 Conclusions

In this thesis, we introduced an approach to identify and classify norm conflicts in contracts written in natural language. We demonstrated that norm conflict causes can be formalised into specific types introducing four conflict types. By using a state-of-the-art text representation algorithm to manipulate norms, we show that it is possible to detect conflicts between norms using operations over such representations. Our resulting approach is an effort to integrate different areas of artificial intelligence and law. The manipulation of natural language contracts demanded techniques of the natural language processing area. On the other hand, creating a classifier to assign the conflict type of a norm conflict required the techniques of the machine learning area. The classification of norm conflicts makes the task of reviewing contracts faster by indicating potential modifications to resolve norm conflicts. By creating a publicly available web-tool, we make it possible for the public to access our contributions and make their analysis and contributions to our approach.

REFERENCES

[1] Aires, J. P.; Granada, R.; Monteiro, J.; Meneguzzi, F. “Norm conflict identification using vector space offsets”. In: Proceedings of the International Joint Conference on Neural Networks, 2018, pp. 337–344.

[2] Aires, J. P.; Meneguzzi, F. “A deep learning approach for norm conflict identification”. In: Proceedings of the Sixteenth International Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2017, pp. 1451–1453.

[3] Aires, J. P.; Meneguzzi, F. “Norm conflict identification using deep learning”. In: Proceedings of the Workshop on Coordination, Organisation, Institutions and Norms in Multi-Agent Systems, 2017, pp. 194–207.

[4] Aires, J. P.; Pinheiro, D.; de Lima, V. S.; Meneguzzi, F. “Norm conflict identification in contracts”, *Artificial Intelligence and Law*, vol. 25–4, Aug 2017, pp. 397–428.

[5] Aires, J. P.; Pinheiro, D.; Meneguzzi, F. “Norm dataset: Dataset with norms and norm conflicts”. Source: https://doi.org/10.5281/zenodo.345411, May 2019.

[6] Alchourrón, C. E. “Conflicts of norms and the revision of normative systems”, *Law and Philosophy*, vol. 10–4, Nov 1991, pp. 413–425.

[7] Andersen, M.; Taylor, H. “Sociology: Understanding a Diverse Society, Updated”. Cengage Learning, 2007, 691p.

[8] Athan, T.; Boley, H.; Governatori, G.; Palmirani, M.; Paschke, A.; Wyner, A. “Oasis legalruleml”. In: Proceedings of The International Conference on Articial Intelligence and Law, 2013, pp. 3–12.

[9] Azzopardi, S.; Gatt, A.; Pace, G. J. “Integrating Natural Language and Formal Analysis for Legal Documents”. In: Proceedings of the Conference on Language Technologies and Digital Humanities, 2016, pp. 32–35.

[10] Azzopardi, S.; Gatt, A.; Pace, G. J. “Reasoning about partial contracts”. In: Proceedings of the Legal Knowledge and Information Systems: The Twenty-Ninth Annual-Ninth Conference-Ninth, 2016, pp. 23–32.

[11] Azzopardi, S.; Pace, G. J.; Schapachnik, F. “Contract automata with reparations.” In: Proceedings of the Legal Knowledge and Information Systems: The Twenty-Seventh Annual-Seventh Conference,-Seventh Krakow,-Seventh, 2014, pp. 49–54.

[12] Baldwin, D. A. “The power of positive sanctions”, *World Politics*, vol. 24–1, Oct 1971, pp. 19–38.

[13] Bengio, Y.; Ducharme, R.; Vincent, P.; Janvin, C. “A neural probabilistic language model”, *Journal of Machine Learning Research*, vol. 3, Mar 2003, pp. 1137–1155.

[14] Bentham, J.; Hart, H. L. A.; Rosen, F. “Of laws in general”. Athlone Press London, 1970, 342p.

[15] Caire, P. “A Normative Multi-Agent Systems Approach to the Use of Conviviality for Digital Cities”. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, chap. 18, pp. 245–260.

[16] Carmo, J.; Jones, A. J. I. “Deontic Logic and Contrary-to-Duties”. ordrecht: Springer Netherlands, 2002, chap. 4, pp. 265–343.

[17] Chalkidis, I.; Androutsopoulos, I. “A deep learning approach to contract element extraction”. In: Proceedings of the Legal Knowledge and Information Systems: The Thirtieth Annual Conference, Luxembourg, 2017, pp. 155–164.

[18] Chalkidis, I.; Androutsopoulos, I.; Michos, A. “Extracting contract elements”. In: Proceedings of the 16th International Conference on Articial Intelligence and Law, 2017, pp. 19–28.

[19] Chalkidis, I.; Androutsopoulos, I.; Michos, A. “Obligation and prohibition extraction using hierarchical rnns”. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2018, pp. 254–259.

[20] Chalkidis, I.; Kampas, D. “Deep learning in law: early adaptation and legal word embeddings trained on large corpora”, *Artificial Intelligence and Law*, vol. 27–2, Jun 2019, pp. 171–198.

[21] Chellas, B. “Modal Logic: An Introduction”. Cambridge University Press, 1980, 295p.

[22] Chowdhury, G. G. “Natural language processing”, *Annual Review of Information Science and Technology*, vol. 37–1, Jan 2003, pp. 51–89.

[23] Cooter, R. “Prices and sanctions”, *Columbia Law Review*, vol. 84–6, Jun 1984, pp. 1523–1560.

[24] Copi, I.; Cohen, C. “Introduction to Logic”. Macmillan, 1990, 654p.

[25] Cortes, C.; Vapnik, V. “Support-vector networks”, *Machine Learning*, vol. 20–3, Sep 1995, pp. 273–297.

[26] Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; Singer, Y. “Online passive-aggressive algorithms”, *Journal of Machine Learning Research*, vol. 7, Dec 2006, pp. 551–585.

[27] Crammer, K.; Singer, Y. “On the algorithmic implementation of multiclass kernel-based vector machines”, *Journal of Machine Learning Research*, vol. 2, Mar 2002, pp. 265–292.

[28] Curtotti, M.; McCreath, E. “Corpus based classification of text in australian contracts”. In: Proceedings of the Australasian Language Technology Association Workshop, Melbourne, Australia, 2010, pp. 18–26.

[29] Curtotti, M.; McCreath, E. “A corpus of australian contract language: Description, profiling and analysis”. In: Proceedings of the 13th International Conference on Artificial Intelligence and Law, 2011, pp. 199–208.

[30] da Silva Figueiredo, K.; da Silva, V. T. “An algorithm to identify conflicts between norms and values”. In: Proceedings of the Coordination, Organisations,Institutions and Norms in Multi-Agent Systems, 2013, pp. 259–274.

[31] Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. “BERT: pre-training of deep bidirectional transformers for language understanding”, *CoRR*, vol. abs/1810.04805, Oct 2018, pp. 16.

[32] Du, J.; Qi, F.; Sun, M. “Using BERT for word sense disambiguation”, *CoRR*, vol. abs/1909.08358, Sep 2019, pp. 5.

[33] Elhag, A. A. O.; Breuker, J. A. P. J.; Brouwer, B. W. “On the formal analysis of normative conflicts”, *Information & Communications Technology Law*, vol. 9–3, Jul 2000, pp. 207–217.

[34] Emerson, E. A. “Temporal and modal logic.”, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, vol. 995–1072, Dec 1990, pp. 5.

[35] Fausett, L. “Fundamentals of Neural Networks: Architectures, Algorithms, and Applications”. Prentice-Hall, Inc., 1994, 480p.

[36] Fenech, S.; Pace, G. J.; Schneider, G. “Automatic conflict detection on contracts”. In: Proceedings of the International Colloquium on Theoretical Aspects of Computing, 2009, pp. 200–214.

[37] Fenech, S.; Pace, G. J.; Schneider, G. “CLAN: A tool for contract analysis and conflict discovery”. In: Proceedings of the Automated Technology for Verification and Analysis, 7th International Symposium, 2009, pp. 90–96.

[38] Friedman, J. H. “On bias, variance, 0/1—loss, and the curse-of-dimensionality”, *Data Mining and Knowledge Discovery*, vol. 1–1, Mar 1997, pp. 55–77.

[39] Gabbard, J.; Sukkarieh, J. Z.; Silva, F. “Writing and reviewing contracts: don’t you wish to save time, effort, and money?” In: Proceedings of the 15th International Conference on Artificial Intelligence and Law, 2015, pp. 229–230.

[40] Gao, X.; Singh, M. P. “Extracting normative relationships from business contracts”. In: Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems, 2014, pp. 101–108.

[41] Gao, X.; Singh, M. P. “Mining contracts for business events and temporal constraints in service engagements”, *IEEE Transactions Services Computing*, vol. 7–3, Nov 2014, pp. 427–439.

[42] Gao, X.; Singh, M. P.; Mehra, P. “Mining business contracts for service exceptions”, *IEEE Transactions Services Computing*, vol. 5–3, Nov 2012, pp. 333–344.

[43] Goodfellow, I.; Bengio, Y.; Courville, A. “Deep Learning”. MIT Press, 2016, 773p.

[44] Governatori, G. “Representing business contracts in *ruleml*”, *International Journal of Cooperative Information Systems*, vol. 14–2-3, Jul 2005, pp. 181–216.

[45] Guo, Y.; Liu, Y.; Oerlemans, A.; Lao, S.; Wu, S.; Lew, M. S. “Deep learning for visual understanding: A review”, *Neurocomputing*, vol. 187, Apr 2016, pp. 27 – 48.

[46] Harris, Z. S. “Distributional structure”, *Word*, vol. 10–2-3, Dec 1954, pp. 146–162.

[47] Hart, O.; Holmström, B. “The theory of contracts”. Cambridge University Press, 1987, chap. 3, pp. 71–156.

[48] Haykin, S. “Neural Networks: A Comprehensive Foundation”. Prentice Hall PTR, 1998, 842p.

[49] Hearst, M. A. “Support vector machines”, *IEEE Intelligent Systems*, vol. 13–4, Jul 1998, pp. 18–28.

[50] Hecht-Nielsen, R. “Theory of the backpropagation neural network”. In: Proceedings of the International Joint Conference on Neural Networks, 1989, pp. 593–605.

[51] Hilpinen, R. “Deontic Logic: Introductory and Systematic Readings”. D. Reidel, 1981, 200p.

[52] Hochreiter, S.; Schmidhuber, J. “Long short-term memory”, *Neural Computation*, vol. 9–8, Nov 1997, pp. 1735–1780.

[53] Huth, M.; Ryan, M. “Logic in Computer Science: Modelling and Reasoning About Systems”. Cambridge University Press, 2004, 442p.

[54] Jain, L. C.; Medsker, L. R. “Recurrent Neural Networks: Design and Applications”. CRC Press, Inc., 1999, first ed., 416p.

[55] James, G.; Witten, D.; Hastie, T.; Tibshirani, R. “An Introduction to Statistical Learning: With Applications in R”. Springer Publishing Company, Incorporated, 2014, 426p.

[56] Joulin, A.; Grave, E.; Bojanowski, P.; Douze, M.; Jégou, H.; Mikolov, T. “Fasttext.zip: Compressing text classification models”, *CoRR*, vol. abs/1612.03651, Aug 2016, pp. 13.

[57] Jurafsky, D.; Martin, J. H. “Speech and Language Processing”. Prentice-Hall, Inc., 2009, 1032p.

[58] Kim, Y. “Convolutional neural networks for sentence classification”. In: Proceedings of the Conference on Empirical Mehods in Natural Language Processing, 2014, pp. 1746–1751.

[59] Lai, S.; Xu, L.; Liu, K.; Zhao, J. “Recurrent convolutional neural networks for text classification”. In: Proceedings of the Association for the Advancement of Artificial Intelligence, 2015, pp. 2267–2273.

[60] Le, Q.; Mikolov, T. “Distributed representations of sentences and documents”. In: Proceedings of the 31st International Conference on International Conference on Machine Learning, 2014, pp. II–1188–II–1196.

[61] LeCun, Y.; Bengio, Y.; Hinton, G. “Deep learning”, *Nature*, vol. 521–7553, May 2015, pp. 436–444.

[62] Levy, O.; Goldberg, Y. “Linguistic regularities in sparse and explicit word representations”. In: Proceedings of the 18th Conference on Computational Natural Language Learning, 2014, pp. 171–180.

[63] Li, L.; Qin, B.; Liu, T. “Contradiction detection with contradiction-specific word embedding”, *Algorithms*, vol. 10–2, Aug 2017, pp. 59.

[64] Li, T.; Balke, T.; De Vos, M.; Satoh, K.; Padget, J. “Conflict detection in composite institutions”. In: Proceedings of the International Workshop on Agent-based Modeling for Policy Engineering, 2012, pp. 75–89.

[65] Liddy, E. D. “Natural language processing”. Encyclopedia of Library and Information Science, 2001, chap. 1, pp. 30–55.

[66] Lippi, M.; Pałka, P.; Contissa, G.; Lagioia, F.; Micklitz, H. W.; Sartor, G.; Torroni, P. “Claudette: an automated detector of potentially unfair clauses in online terms of service”, *Artificial Intelligence and Law*, vol. 27–2, Jun 2019, pp. 117–139.

[67] MacMahon, P. “Conflict and Contract Law”, *Oxford Journal of Legal Studies*, vol. 38–2, Apr 2018, pp. 270–298.

[68] Mahmoud, M. A.; Ahmad, M. S.; Mohd Yusoff, M. Z.; Mustapha, A. “A review of norms and normative multiagent systems”, *The Scientific World Journal*, vol. 2014, Jul 2014, pp. 23.

[69] Marsland, S. “Machine Learning: An Algorithmic Perspective, Second Edition”. Chapman & Hall/CRC, 2014, 457p.

[70] Meneguzzi, F.; Miles, S.; Luck, M.; Holt, C.; Smith, M. “Electronic contracting in aircraft aftercare: A case study”. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track, 2008, pp. 63–70.

[71] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. “Efficient estimation of word representations in vector space”, *CoRR*, vol. abs/1301.3781, Jun 2013, pp. 12.

[72] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; Dean, J. “Distributed representations of words and phrases and their compositionality”. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems, 2013, pp. 3111–3119.

[73] Mikolov, T.; Yih, W.; Zweig, G. “Linguistic regularities in continuous space word representations”. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2013, pp. 746–751.

[74] Minsky, M.; Papert, S. “Neurocomputing: Foundations of research”. In: *Neurocomputing: Foundations of Research*, 1988, chap. Perceptrons, pp. 157–169.

[75] Mitchell, T. M. “Machine Learning”. McGraw-Hill, Inc., 1997, 432p.

[76] Mitkov, R. “The Oxford Handbook of Computational Linguistics”. OUP Oxford, 2005, 806p.

[77] Mohri, M.; Rostamizadeh, A.; Talwalkar, A. “Foundations of Machine Learning”. The MIT Press, 2012, 412p.

[78] Nadeau, D.; Sekine, S. “A survey of named entity recognition and classification”, *Lingvisticæ Investigationes*, vol. 30–1, Aug 2007, pp. 3–26.

[79] Nikiforakis, N.; Noussair, C. N.; Wilkening, T. “Normative conflict and feuds: The limits of self-enforcement”, *Journal of Public Economics*, vol. 96–9, Oct 2012, pp. 797–807.

[80] Pagliardini, M.; Gupta, P.; Jaggi, M. “Unsupervised learning of sentence embeddings using compositional n-gram features”. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2018, pp. 528–540.

[81] Paulson, S. L. “On the status of thelex posterior derogating rule”, *Liverpool Law Review*, vol. 5–1, Mar 1983, pp. 5–18.

[82] Pennington, J.; Socher, R.; Manning, C. D. “Glove: Global vectors for word representation”. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1532–1543.

[83] Pinheiro, D.; Meneguzzi, F. “Understanding contracts in natural language”, Master’s Thesis, Programa de Pós-Graduação em Ciência da Computação, 2019, 70p, escola Politécnica.

[84] Prakken, H.; Sergot, M. “Contrary-to-duty obligations”, *Studia Logica*, vol. 57–1, Jul 1996, pp. 91–115.

[85] Rauhut, H.; Winter, F. “Types of normative conflicts and the effectiveness of punishment”, *Social Dilemmas, Institutions, and the Evolution of Cooperation*, vol. 1–1, Jan 2017, pp. 239–258.

[86] Richardson, H. S. “Specifying norms as a way to resolve concrete ethical problems”, *Philosophy & Public Affairs*, vol. 19–4, Sep 1990, pp. 279–310.

[87] Rosenblatt, F. “The perceptron: A probabilistic model for information storage and organization in the brain”, *Psychological Review*, vol. 65–4, May 1958, pp. 386–408.

[88] Ross, A. “On Law and Justice”. Berkeley and Los Angeles: University of California Press, 1959, 576p.

[89] Rosso, P.; Correa, S.; Buscaldi, D. “Passage retrieval in legal texts”, *The Journal of Logic and Algebraic Programming*, vol. 80–3-5, May 2011, pp. 139–153.

[90] Rousseau, D. M.; McLean Parks, J. “The contracts of individuals and organizations”, *Research in Organizational Behavior*, vol. 15–6, Jan 1993, pp. 1–43.

[91] Rumelhart, D. E.; Hinton, G.; Williams, R. J. “Learning representations by back-propagating errors”, *Nature*, vol. 323–6088, Oct 1986, pp. 533–538.

[92] Russell, S.; Norvig, P. “Artificial Intelligence: A Modern Approach”. USA: Prentice Hall Press, 2009, 1152p.

[93] Sadat-Akhavi, S.-A. “Methods of Resolving Conflicts Between Treaties”. Brill Nijhoff, 2003, 228p.

[94] Schwartz, A.; Scott, R. E. “Contract theory and the limits of contract law”, *The Yale Law Journal*, vol. 113–3, Apr 2003, pp. 541–619.

[95] Schwartz, R. D.; Orleans, S. “On legal sanctions”, *U. Chi. L. Rev.*, vol. 34, Feb 1966, pp. 274.

[96] Sergot, M. “Representing Legislation as Logic Programs”. USA: Oxford University Press, Inc., 1988, chap. 10, pp. 209–260.

[97] Singh, M. P. “Norms as a basis for governing sociotechnical systems”, *ACM Transactions on Intelligent Systems and Technology*, vol. 5–1, Jan 2014, pp. 21:1–21:23.

[98] Souza, F.; Nogueira, R. F.; de Alencar Lotufo, R. “Portuguese named entity recognition using BERT-CRF”, *CoRR*, vol. abs/1909.10649, Sep 2019, pp. 8.

[99] Sumner, W.; Keller, A. “Folkways: A Study of the Sociological Importance of Usages, Manners, Customs, Mores, and Morals”. Ginn, 1906, 700p.

[100] Sutskever, I.; Martens, J.; Hinton, G. “Generating text with recurrent neural networks”. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, 2011, pp. 1017–1024.

[101] Turney, P. D. “Similarity of semantic relations”, *Computational Linguistics*, vol. 32–3, Sep 2006, pp. 379–416.

[102] Vasconcelos, W. W.; Kollingbaum, M. J.; Norman, T. J. “Normative conflict resolution in multi-agent systems”, *Autonomous Agents and Multi-Agent Systems*, vol. 19–2, Oct 2009, pp. 124–152.

[103] von Wright, G. H. “Deontic logic”, *Mind*, vol. 60–237, Jan 1951, pp. 1–15.

[104] Vranes, E. “The definition of ‘norm conflict’ in international law and legal theory”, *European Journal of International Law*, vol. 17–2, Apr 2006, pp. 395–418.

[105] Wyner, A.; Nazarenko, A.; Lévy, F. “Towards a High-Level Controlled Language for Legal Sources on the Semantic Web”. Springer International Publishing, 2016, chap. 9, pp. 92–101.

[106] Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K. Q.; Artzi, Y. “Bertscore: Evaluating text generation with BERT”. In: Proceedings of the 8th International Conference on Learning Representations, 2020, pp. 43.

[107] Zhang, X.; LeCun, Y. “Text understanding from scratch”, *CoRR*, vol. abs/1502.01710, Mar 2015, pp. 10.

[108] Zou, W. Y.; Socher, R.; Cer, D. M.; Manning, C. D. “Bilingual word embeddings for phrase-based machine translation”. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1393–1398.