Grant Agreement No.: 823783
Call: H2020-FETPROACT-2018-2020

Topic: H2020-FETPROACT-2018-01
Type of action: RIA

D5.2
WeNet's Diversity-Aware Interactions II

Revision: v.0.2

Work package	WP5
Task	T5.2, T5.3, T5.4, T5.5
Due date	28/02/2022
Submission date	28/02/2022
Deliverable lead	CSIC
Version	4.0
Authors	Nardine Osman (CSIC), Thiago Freitas (CSIC), Marco Schorlemmer (CSIC), Carles Sierra (CSIC), Ronald Chenu-Abente (UNITN), Qiang Shen (UNITN), Fausto Giunchiglia (UNITN), Bruno Rosell (CSIC)
Reviewers	Avi Segal (BGU)

Abstract	WeNet's main objective is achieving a diversity-aware, machine-mediated paradigm for social interactions. This deliverable focuses on WeNet's diversity-aware interaction model that enables online interactions while ensuring they are privacy-compliant, diversity-aware, and in more general

	terms, fulfil ethical requirements. The deliverable is divided into two main parts. The first focuses on the issue of mediating online interactions in such a way that ensures the diverse requirements of profiles are met. For this, a normative-based decentralised architecture is proposed. The second addresses the issue that arises when community members with diverse profiles interpret norms in different ways, possibly leading to unexpected behaviour in interactions, usually with norm violations that affect the individual and community experiences. For this, our initial work focuses on detecting norm violation due to the diverse interpretation of norms and providing the violator with information about the features of their action that makes this action violate a norm.
Keywords	online communities, interaction models, context, norms, privacy, norm violation detection, machine learning

Document Revision History

Version	Date	Description of change	List of contributor(s)
V1.0	14/06/2021	1st version submitted for internal review	Nardine Osman (CSIC)
V2.0	30/06/2021	Final version	Nardine Osman (CSIC)
V3.0	31/01/2022	Revised deliverable submitted for internal review	Nardine Osman (CSIC)
V4.0	28/02/2022	Final revised deliverable	Nardine Osman (CSIC)

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:*	**R**	
Dissemination Level		
PU	Public, fully open, e.g. web	✔
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to the WeNet project and Commission Services	

** R: Document, report (excluding the periodic and final reports)*

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

OTHER: Software, technical diagram, etc.

EXECUTIVE SUMMARY

WeNet's main objective is achieving a diversity-aware, machine-mediated paradigm for social interactions. This deliverable focuses on WeNet's diversity-aware interaction model that enables online interactions while ensuring they are privacy-compliant.

The deliverable is divided into two main parts. The first focuses on the issue of mediating online interactions in such a way that ensures that properties set by diverse profiles are met, with special focus on privacy. For this, we propose an architecture for the WeNet's interaction model that is based on normative systems. While normative systems have excelled at addressing issues such as coordination and cooperation, they have left a number of open challenges. The first is how to reconcile the individual goals of diverse profiles with community goals, without breaching an individual's privacy. The evolution of norms driven by individuals' behaviour or argumentation have helped take the individual into consideration. But what about the individual norms of diverse profiles that people are not willing to share with others? Then, there are the ethical considerations that may arise from online interactions, such as, how do we deal with stereotypes, biases, or racism in communities with diverse profiles, or how to avoid the abuse of community resources. Our proposal accounts for individual needs of diverse profiles while respecting privacy and adhering to the community's ethical code. We propose an architecture for normative systems that, along with community norms, introduces individual's requirements to help mediate interactions. This work is reported on in Chapter 2, which covers WP5's progress with respect to designing (T5.3) and implementing (T5.5) a diversity aware interaction model, as well as specifying the norms of WeNet's first pilot (T5.4).

The second part of the deliverable focuses on the issue that arises when different community members interpret norms in different ways due to their diverse views and beliefs, possibly leading to unexpected behaviour in interactions, usually with norm violations that affect the individual and community experiences. For this, our initial work on this topic focuses on detecting norm violation and providing the violator with information about the features of their action that makes this action violate a norm. This helps users with diverse views and beliefs to align their understanding of norms better and avoid norm violations in the future. However, we note that the work presented here is intended as a stepping stone for adapting the meaning of norms to the view of the community. We believe communities and their members evolve, and what may be considered a norm violation today might not be in the future. For example, take the norm prohibiting hate speech. Agreeing on the features of hate speech may change from one group of people to another and may also change over time. We argue that human communities do not always have one clear definition of concepts like hate speech, violation, freedom of speech, etc. The framework, as such, must adapt to the evolving diverse views of the members of its community. Ongoing work is currently developing the mechanisms for such adaptations. The work on aligning the understanding of norms is reported on in Chapter 3, which covers WP5's progress with respect to task T.2.

However, before diving into the details of WP5's progress, the deliverable opens in Chapter 1 with an overview of WP5's work and its relation with other WeNet work packages.

Co-funded by the Horizon 2020
Framework Programme of the European Union

TABLE OF CONTENTS

Co-funded by the Horizon 2020
Framework Programme of the European Union

Chapter 1

Introduction

This deliverable reports on the progress of the work in WP5 and its relation with other WeNet work packages. The first chapter provides an overview of this work and its relation with WeNet, before the concrete models and mechanisms developed in WP5 are reported on in detail in the subsequent chapters.

1.1 Advancement on WP5 Objectives

The main objective of WP5 is developing the interaction model responsible for connecting people and mediating their interactions.

The final versions of the theoretical model designed for mediating interactions between diverse profiles (**the objective of task T5.3**) along with the engine responsible for the execution of these interactions (**the objective of task T5.5**) are presented in Chapter 2. While Sections 2.2 and 2.3 present the theoretical model, Section 2.4 presents the operational model and the execution environment (the decision engine of Section 2.4.3), followed by a motivating example in Section 2.5 that highlights how the requirements of diverse profiles is taken into consideration while respecting people's privacy. In Section 2.6, we present the norms used in WeNet's first use case, where anyone can create a new request (also referred to as a task in some deliverables) that is forwarded to randomly selected people in the community and recipients are free to ignore or answer the new request. Note that specifying the norms that define the rules of interaction in WeNet's use cases is **the objective of task T5.4**. As the diversity dimensions of the profiles and the more advanced WeNet pilots were still under development at the point of writing of this deliverable and its submission at M30, the norms leveraging such diversity will be reported on in the next version of this deliverable, D5.3.

The initial work focusing on leveraging the diversity of profiles when connecting people (**the objective of task T5.1**) has been discussed in D5.1. In that work, we presented an algorithm for grouping people together, taking into consideration the diversity of their profiles. Deciding how to connect people is very much context dependent. The use case scenario is usually needed to help decide which of the profile attributes (age, gender, capabilities, personal beliefs, locations, friendships, etc.) should be taken into consideration and how. At the time of preparation of this deliverable, whose due date is M30, the use cases were not fully specified yet, and hence, their necessary profile attributes were still being agreed upon. For this reason, a generalised version of the algorithm of D5.1 was being integrated into the platform at the time of writing of this deliverable, and its results (when tested with the use cases) will be reported on in the next version of this deliverable (D5.3). The generalised version essentially accepts *any* set of attributes and calculates the diversity of the group with respect to that set. The choice of attributes for the time being will be based on some user requirements, yet to be defined when finalising the use case scenarios.

Concerning **the objective of task T5.2**, the focus is on the issues that arise when different community members with diverse views and beliefs interpret norms in different ways, usually leading to norm violations that affect the individual and community experience. For this, we have designed a machine learning mechanism that detects norm violations and provides the violator with information about the features of their action that makes this action a violation. This work is presented in Chapter 3, and it helps users with diverse views and beliefs to align their understanding of norms better so they can avoid norm violations in the future. However, this is only intended as a stepping stone for allowing the meaning of norms to continuously adapt to the evolving understanding of the diverse profiles of community members. We argue that the meaning of norms evolve with time. For example, imagine a norm that states that hate speech is not allowed. Agreeing on the features of hate speech may change from one group of people to another and may also change over time. We plan to improve the mechanism of Chapter 3 to adapt to the evolving diverse views of the members of its community. This more advanced work will be reported on in the next version of this deliverable, D5.3.

1.2 Relation to other WeNet Work Packages

WP6: The platform. WP5 is responsible for implementing the components of the WeNet platform that are concerned with mediating interactions. The platform is presented in Figure 1.1 (and described in WP6's deliverables), and the technical work packages responsible for the different components have their labels in red attached to their components. Note that WP6 is responsible for the development of the entire platform.

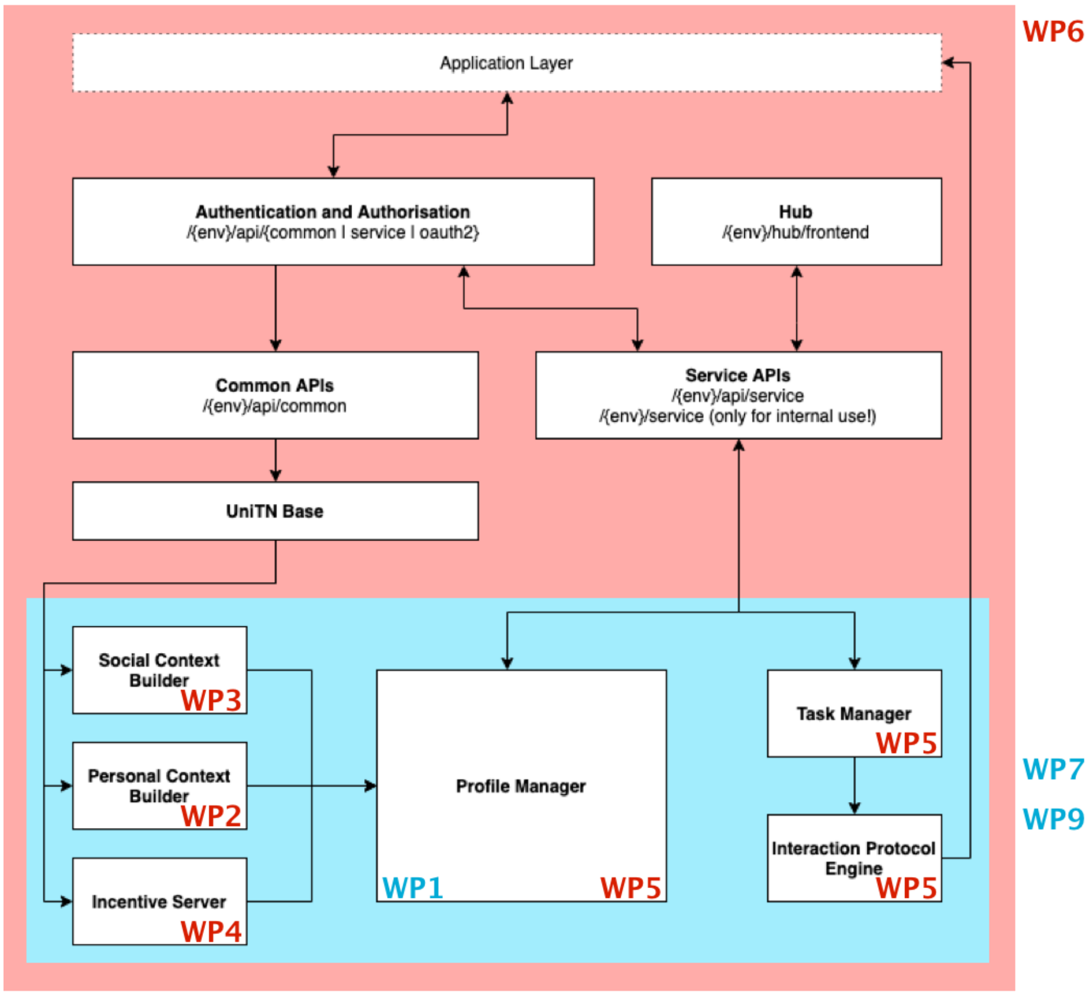

Figure 1.1: The WeNet WPs' relation to the WeNet platform

The core of WP5's work is the interaction protocol engine (IPE, also referred to as the decision engine in Chapter 2) that is responsible for mediating all interactions concerning requests (or tasks) made within a WeNet app. The requests are managed by the Task Manager component, with the help of the decision engine (IPE) to ensure every action on these requests follows the relevant norms (e.g. when can a request be created, who can receive it, etc.). As we will illustrate in Chapter 2, norms are strongly dependent on the profiles. For example, if the requester prohibits sending a request to those not in her city (specified as a norm in the requester's profile), then the decision engine (IPE) will require access to such norms as well as other data (like the person's location). As such, the Profile Manager becomes another core component that supports the decisions of the decision engine (IPE).

The design of the WP5 components, especially the profile manager and the interaction protocol engine, follows the model presented in Chapter 2 (covering tasks T5.3 and T5.5). Populating the profiles

Co-funded by the Horizon 2020 Framework Programme of the European Union

with norms is also presented in Chapter 2 (covering task T5.4).

WP2, WP3, WP4: The other technical work packages. Concerning the relation of WP5 with the other technical work packages, this is mostly materialised through the profile manager. WP2 is concerned with learning attributes concerning the individuals (e.g. information about their locations). As such, WP2 is responsible for populating certain attributes of people's profiles that deal with people's personal context. This opens the door in WP5 to design norms that make use of such attributes: e.g. only asking people nearby.

WP3 is responsible for learning people's social context, like their friendships, which answers are preferred by the requester, etc. As such, WP3 is responsible for populating certain attributes of people's profiles that deal with people's social context. Again, this opens the door in WP5 to design richer norms that can make use of such attributes: e.g. only asking close friends. Furthermore, WP3 is also concerned with ranking answers based on past experiences, and providing explanations to users on certain decisions. Both of these require access to past interaction data to learn from it. As such, WP3 also links to the task manager in order to have access to this interaction data.

WP4 is responsible for providing incentives for users to act. These incentives are personalised, and as such, heavily depend on people's profiles. In this case, and unlike the other two technical work packages (WP2 and WP3), instead of contributing to populating the profile, WP4 depends on reading profile data to learn and suggest the most appropriate incentives. Furthermore, community norms are used to control the frequency of incentives (e.g. a maximum of 3 incentivising messages can be sent per week). This constitutes WP5's other link with WP4.

WP1, WP7, WP9: The non-technical work packages. WP1 is the work package responsible for studying diversity dimensions and recommending profile attributes to be considered that would improve interactions by leveraging these diversity dimensions. As such, WP1 has been collaborating on developing the profile structure by suggesting new attributes, mostly on people's interests and their beliefs and values. With these new attributes, new norms can be put in place that would leverage the diversity (or similarity) of profiles when deciding on who to connect together.

WP7 is the work package responsible for designing the use cases, as such, they collaborate with all technical partners on the use of their tools to help design the best use case making use of those tools. In the case of WP5, the norms mediating the interactions are strongly dependent on the use case. Furthermore, designing the mechanism in charge of connecting people is also use case dependent, and the work is the result of collaboration with WP7.

WP9 is the work package responsible for dealing with ethical considerations. As such, they also collaborate with all technical partners on the potential side effects of their tools. In the case of WP5, the design of the interaction model has been done with the objective of empowering users and ensuring privacy is engineered within the system. Furthermore, norms can be used as means to address some ethical concerns. For example, imagine a norm that states that if a user has been using hate speech at least 3 times in the last month, then they will be suspended for a week; or a norm that states that if the requester is only targeting potential responders with a specific profile dimension (females, teens, Spanish, ..), then send back a warning on possible bias and exclusion; or even a norm stating that sending incentive messages should adhere to a predefined permitted frequency. We are currently analysing these norms, and some are already implemented in the system.

Some examples of profile attributes, norms, and the relation with other work packages. As illustrated above, WP5's work mostly links with other work packages through the profile. As different work packages help enrich the profile, this enriches the norms that can be specified at individual and community levels. For example, a norm restricting the delivery of messages to people nearby is not possible if information about people's location is not available in the first place. Table 1.1 presents a sample of profile attributes, examples of norms that makes use of these attributes, and the contribution of different WPs to those attributes.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Profile attribute	**Sample norm using this attribute**	**Contribution of other WPs**
Name Gender Nationality ...	*We currently do not make use of socio-demographic data, as this may introduce unwanted biases in the interactions (e.g. gender biases).*	WP6 is responsible for populating these socio-demographic attributes.
Location Busy hours ...	Send my request only to people nearby. Supress notification when I am busy.	WP2 populates these attributes on personal context.
Social relations	Send my request only to friends.	WP3 populates these attributes on social context.
Interests Values	Send my request to people with different interests. Send my request to people with similar values.	WP1 designs these attributes.

Table 1.1: Profile attributes, norms, and the contribution of other work packages

Co-funded by the Horizon 2020 Framework Programme of the European Union

Chapter 2

Empowering Users in Online Open Communities[†]

In this chapter we propose an architecture supporting *online open communities*, where by *open* communities we mean communities where previously unknown people with diverse profiles can join, possibly for a limited amount of time. Our proposal provides the communication model used in WeNet to mediate people's interactions (also refered to as the interaction model). The fundamental question that we address is "how we can make sure that the requirements of people with diverse profiles are taken into consideration by the community while their privacy is respected and the community's ethical code is not violated". The main contributions are: (i) a conceptual framework which allows to describe diverse profile at both individual and community level, including data and norms that provide information about their owner and their requirements, and (ii) a *decentralised architecture* enabling interactions that leverage the exchange of profile information among people and communities to ensure that diverse requirements are fulfilled and privacy is respected.

2.1 Introduction

The huge success of *social networks*, e.g., Facebook, Whatsapp, WeChat, has highlighted the importance of *online social relations*, where the key novelty is the possibility of enabling interactions which transcend the limitations of space and time. Thanks to social networks, it is possible for anybody to interact in real time, in writing or by talking, to virtually anybody else in the world, independently of their physical location. The implications of this success are obvious and involve a huge amplification of social relations, thus enabling the creation of large scale *online communities*. This phenomenon has been extensively studied in the literature, see, e.g., [14, 52, 21].

Following the vision described in [23], in this chapter we propose to move "from a network of computers, which in turn may be connected to people, to a network of people, whose interactions are mediated and empowered by computers" (quote from [23]). In other words, after investing in the last few decades on how the machine may be put at the service of humans in online networks (as in, for example, the Internet or the IoT), we now highlight the need to bring back the human as a critical source of providing support, and not just receiving it.

This entails careful work on online social relations, which has its difficulties, one of the main issues relating to their quality. As discussed in, e.g., [49, 14], computer mediated interactions can be, and often are, less valuable in building and also in sustaining close relations. However, despite this, the possibility of developing high value online communities seems quite promising [20, 53]. Social relations then become more intense and can be applied for objectives which go far beyond the exchange of short messages or discussions about the current topic of interest; this being grounded in the fact that, in the real world, social relations allow people not only to interact but also to help one another by using their collective strengths as the means for overcoming individual weaknesses. A collective has capabilities

[†]This chapter presents a substantially extended and revised version of WeNet's architecture of D5.1 that supports the normative-based interaction model. The main changes with respect to the first version of D5.1 are: improving and extending the introduction; considering norms as part of the profile (Section 2); adding a discussion on how to build and share the profiles (Section 3), which is a critical issue when it comes to privacy; adding a description of the decision engine that is being implemented along with a discussion of some properties of our proposed system (Section 4.4); and improving the motivating example (Section 5). We also note that this version has been accepted for publication at Springer's SN Computer Science journal.

that are much more than the sum of the capabilities of any single member [53]. The intrinsic diversity of people (in their characteristics, knowledge, skills, competencies, and much more) is something that people use in their everyday life, often without even realising it. We ask a doctor for a diagnosis and a treatment when we are ill, we call up a plumber if a pipe leaks in our house, and look for someone speaking our language and Chinese if we need to sort out arrangements for a stay in Beijing. Scaling up this possibility to the whole size of the Internet would immensely enhance the ability to solve certain tasks, thanks to the help and support of third parties. And this applies to any type of need, ranging from a person who provides you a service (as in the plumber example), a need whose satisfaction requires some follow-up action in the real world, or a pure informational need (as in the Chinese example), where the need can be solved online.

Many social networks today seem to address this very problem. Current online social networks like Instagram and TikTok are based on enabling social relations with previously unknown people. Many are used for connecting people to solve specific human needs, like TaskRabbit (`www.taskrabbit.com`), Upwork (`www.upwork.com`), or PeoplePerHour (`www.peopleperhour.com`). Current ego networks [60, 50, 3, 38] already allow users to apply the support and skills of a large number of people.

However, existing solutions suffer from two issues. First, they fail to leverage the diversity of people's profiles when connecting people. Second, their use of rigid interaction protocols leave their users without much control over their data and interactions.

This chapter aims at addressing those two issues. First, we argue that the diversity of profiles must be taken into consideration when addressing people's needs. In fact, the main question that this deliverable addresses is how can we make sure that individuals with diverse needs will have their needs taken into consideration while leveraging the diversity of available profiles (that describe the different interacting entities in the system), yet ensuring that people's privacy is respected and the community's ethical code is not violated (as ethical requirements can sometimes be implemented as norms, as we illustrate in Section 2.2.2). Accommodating to people's needs and leveraging the diversity in a community, all while respecting privacy and other ethical requirements is one of the main novelties of this work. The other main novelty is in achieving the above while giving users: 1) control over their profile data, over how this data can be shared, with whom, and under which circumstances; and 2) control over how interactions are carried out within communities.

The problem of privacy online is well known and largely studied, see, e.g., [61, 36, 37, 39, 58] and has caused various studies and analyses, see, e.g., [54, 11], as well as the generation of considerable legislation, in Europe above all, but also worldwide [19, 18]. However this problem grows enormously in the case of online (open) communities, given that their enablement requires sharing information which is far more sensitive than that needed in the state of the art social networks [33].

To summarise, the main contribution of this chapter is thus the definition and articulation of an architecture enabling and supporting online open communities, with a dedicated focus on the individual and her needs, leveragin the diversity of others, and giving special attention to privacy. Towards this end, the main components of the proposed solution are:

- A conceptual framework which allows for describing individual and community profiles, including data and norms that provide information about their owner. We argue that people and communities must build their own profile which can then be used by third parties to discover the most suitable person who can help them with the task at hand. We also argue the need to empower people and communities in selecting the visibility of the profile as a trade-off between *privacy* and *openness*. On the one hand, there is a need to prevent personal information to be shared with unknown and possibly malicious people, while on the other hand, there is a need to allow for some level of personal information sharing. If nobody knows about you or about how to contact you, then no social interactions can be enabled. The solution provided is that the level of information that a person will share will depend on the *context* [10, 25, 22], e.g., the type of information itself, the goal and the people involved.

- A decentralised architecture for social networks that helps achieve the above goals by mediating social interactions through community norms. The proposed architecture empowers community members by allowing them to specify their individual rules and data that describe them, as well as to specify whom to share this information with and under what circumstances.

The chapter is structured as follows. Section 2.2 introduces our conceptual framework for profiles, which are composed of data and norms. Building rich profiles is crucial for leveraging diversity in WeNet interactions. Section 2.3 discusses profile building and profile sharing, a cornerstone for addressing privacy. Section 2.4 introduces the decentralised architecture addressing the conceptual framework (that

Co-funded by the Horizon 2020 Framework Programme of the European Union

is the communication model used for specifying interaction protocols and the execution environment for these protocols), while Section 2.5 provides a motivating example, and Section 2.6 presents the norms (referred to as sample protocols in the proposal) used in WeNet's first use case. The related work is discussed in Section 2.7 before concluding with Section 2.8.

2.2 Profile

A profile is a description of an entity, which, in turn, we take to be a person or a community. When building a software system that supports a particular social interaction, for any entity, it is fundamental to define: (i) what particular attributes are relevant for the interaction with other entities, so that their values (i.e.,*data*) should be gathered; and (ii) what rules of behaviour (i.e., *norms*) of entities affect the social interaction being modelled. The diversity of profiles in WeNet is then reflected by the diversity of people's attributes (whether shallow, like gender and age, or deep, like beliefs and competences) as well as their norms (like how much are they open to receiving and replying to help requests).

Definition 1 (Profile)**.** *A profile P is a pair of data D and a set N of norms: $P = (D, N)$*

Next, we analyse the two components of a profile in detail.

2.2.1 Data

A profile will contain an ample amount of data about that entity, information needed in order to suitably interact in a certain open community. We illustrate here the dimensions of a person's *situational context* [25], noting that a profile of any entity may consist of multiple contexts, for instance describing the entity's physical characteristics, its competences or, even an ongoing conversation. Figure 2.1 shows a small example of Ethan's situational context D. We focus on the situational context for three main reasons. The first is that it highlights the privacy issues that can be raised in relation to personal information. The second is that this information is, of course, very dynamic, thus making privacy a problem which must continuously be dealt with, with the assurance that the information provided to third parties at a certain moment of time will not hold any longer than necessary. The third is that the situational context plays a crucial role in the possibility for a certain individual to engage, within a community, in a social interaction.

A situational context is composed of four main sub-contexts, WE, WA, WO, and WI, as follows:

WE is a spatial context which captures the exact location, e.g., "Home" or "Barcelona". We refer to it as the answer to "**W**h**E**re are you?" in the case of a person and "**W**h**E**re is the community located?" for a community.

WA is a behavioural context which captures the activity, e.g., "napping". Informally, it answers the question "**W**h**A**t are you doing?" for a person and "**W**h**A**t does the community do?" for a community.

WO is a social context which captures the social relations, or the answer to "**W**h**O** are you with?" (e.g., the "family"), for a person, and "**W**h**O** do you collaborate with?" for a community.

WI is object context which captures the materiality, e.g., "smartphone" or "car". It represents the object you currently have. Informally, it answers the question "**W**hat are you w**I**th?" for a person and "**W**hat **I**nfrastructure does the community have?" for a community.

We model a person's situational context D, which we refer to as the data part of the profile, as a *knowledge graph* [8, 17, 35], which we define as the union of four smaller knowledge graphs WE, WA, WO, WI.

Definition 2 (Data)**.** *The data D, representing the context inside of the profile, is the union of the four dimensions of situational context:*

$$D = (WE \cup WA \cup WO \cup WI)$$

In this setting we define a knowledge graph as follows:

- nodes represent entities, namely anything physical, digital, conceptual, real or imaginary which is described via a set of properties, i.e., attributes and relations (e.g., MyCar, Barcelona, Ethan);

Co-funded by the Horizon 2020 Framework Programme of the European Union

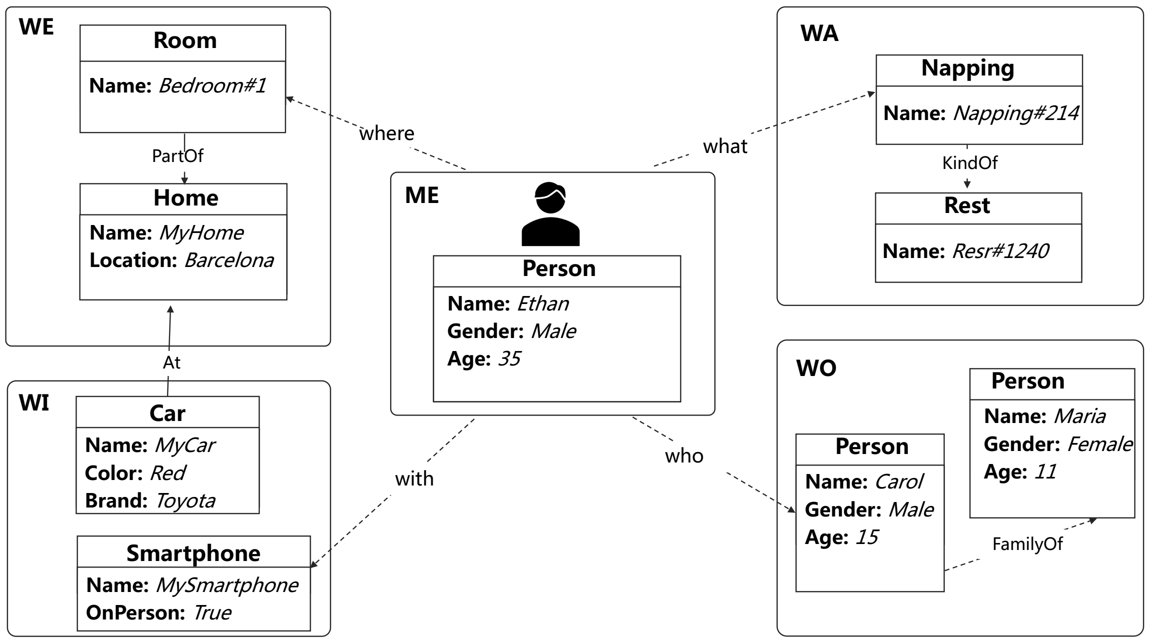

Figure 2.1: An example of situational context

- information about these nodes is represented as attributes, namely entity value pairs (e.g., Location (Barcelona), Gender(Ethan) = Male, Age(Ethan) = 35);
- links represent relations among entities, namely a limited set of pairs of entities describing how they relate (e.g., where (Ethan, Bedroom#1), who(Ethan, Carol), with (Ethan, MyCar), partOf (Bedroom#1, MyHome)).

Notice that a knowledge graph like the one defined above can be mapped one-to-one into a description logic where entities (e.g., Ethan) are instances populating concepts (e.g., Person), while attributes and relations are pairs populating, respectively, data and object properties, see, e.g., [4, 48]. This knowledge graph, in turn, can be easily represented and exported in terms of RDF triples.

We assume that a profile is continuously enriched with data coming from various sorts of streams, requiring to store the changing values of the most relevant attributes. These streams of information can be sensor data (e.g., GPS, accelerometer, giroscope, blue-tooth) which are then used to learn the various types of information stored in D. Some of this information is directly provided by the user, properly asked by the system. This topic is not described here because it is out of scope. [29, 62, 64, 9] provide a long list of concrete examples of how this can be done. From a practical point of view, D can be considered as consisting of lifelogging data [32, 7], which can be formalised as:

$$D^t(u) = \langle D^1, D^2, D^3, \ldots, D^t \rangle,\ t \rightarrow +\infty$$

where $D^t(u)$ is the data profile of user u at time t, t is growing along the user's life, and the size of streaming profile is thus continuously increasing. It is worth noticing that the problem of an ever growing profile is dealt with by implementing various forms of selective *forgetting*. The results, which are much more compact are then stored in a long term memory. Thus, again, [29, 62, 64, 9] provide examples of the kind of learning we perform over data from a two week period.

2.2.2 Norms

Norms are rules that specify behaviour at the individual and the social level. They determine what actions are acceptable, who can an individual interact with, and under what circumstances, etc. So far normative systems have mostly focused on the action, namely on 'what' can one do; here we extend this work to focus on another crucial aspect of interactions as well, namely on 'who' can one interact with, this being more and more relevant in an increasingly hyper-connected world. To achieve this, we take norms as the second component of individual and community profiles (Definition 1). Behaviour is as important in social interactions as individuals' gender, age, or relationships. For example, one individual

Co-funded by the Horizon 2020 Framework Programme of the European Union

norm can say "only seek help from people around me", while another can say "never bother me when I am napping".

Traditionally, in multiagent systems, norms have been specified through deontic operators that describe what is permitted, forbidden, or obligatory [59]. We propose a simple approach that specifies norms as *if-then* statements that specify who can perform what action, and under what condition. For instance, the above two individual norms may be specified as:

IF

$seek\_help(Person, Task)$
and $location(Person, City)$
and $friends\_around(City, List)$

THEN

$forward\_seek\_help(List, Task)$

IF

$naptime(true)$ and $notify(X)$

THEN

$suppress\_notification(X)$

The norm part of a profile is then taken to be a set of such if-then statements, and defined accordingly.

Definition 3 (Norm)**.** *A norm $n \in N$ is defined as an if-then statement: $n = \texttt{IF}\ Condition\ \texttt{THEN}\ Consequent$, where $Condition$ and $Consequent$ are expressions, or formulae, defined as follows:*

- *Each atomic formula is a formula.*
- *If C and C' are formulae, then C $\texttt{and}$ C' is a formula.*
- *If C is a formula, then $\neg C$ is a formula.*

Recall that the profile may be the profile of an individual or a community. And just like the data part of the profile, the norms part will also describe the rules of behaviour of the individual or the community, respectively. When norms are part of the individual profile, we refer to them as *individual norms*, and when they are part of the community profile, we refer to them as *community norms*. Notice how in this setting by 'community' we mean both an organisation as we have in the real world, e.g., the University of Trento, as well as an online group of people, more or less informally organised.

Community norms govern the behaviour of the community they are associated with, including its members. Any action (represented by a message exchange) in the peer-to-peer network of this community must be coherent with these norms. For instance, a norm in a mutual aid community that prohibits members from abusing the community by always asking for help and never offering help, or a norm that punishes those that do not fulfil their duties by suspending their memberships. We consider an action acceptable by the community when it doesn't violate any of the community's norms.

We say community norms can be divided into a number of categories. For example, we use *institutional* norms to refer to norms that can describe the rules of behaviour in the given community (following the concept of electronic institutions [15]). We say *ethical* norms can describe what is considered ethical and what actions are deemed unethical, and hence, unacceptable in the community. For example, imagine a norm that states that if a user has been using hate speech at least three times in the last month, then they will be suspended for a week; or a norm that states that if the requester is only targeting potential responders with a specific profile dimension (females, teens, Spanish, ..), then send back a warning on possible bias and exclusion.*Incentive* norms can help provide incentives for community members to behave in a certain way, such as encouraging benevolent behaviour, say to help maintaining the community and fulfilling its objectives. And so on.

Individual norms are rules that govern the behaviour of the individual they are associated with. They represent particular aspects of the relationship of the human with her device (mobile, tablet, computer) and with the community. For instance, a prohibition to pop-up a message during a siesta. Or an obligation to filter out messages coming from people that are not in one's vicinity. Of course, individual norms may implement certain behaviour that may not be fully aligned with the community norms. So some behaviour that is deemed 'unacceptable' or even 'unethical' by the community may be codified at this level and remain unnoticed by the community, simply because individual norms represent the individual's requirements with respect to behaviour and not that of the community. In cases of conflict between community and individual norms, community norms prevail concerning actions within the community. For example, if community norms prohibit discriminating against women, then an individual norm that

Co-funded by the Horizon 2020 Framework Programme of the European Union

asks to exclude females from a given activity will be overruled by the community's norm. However, individual norms prevail when concerning actions local to one's device.[1] For instance, while community norms may prohibit discriminating against women, one's individual norm can enforce requests coming from women to be suppressed (ignored).

Last, but not least, we note that like data, norms evolve over time. While I might accept requests to play padel from anyone today, in the future, I might change my mind and restrict receiving such requests to those made by padel professionals only.

2.3 Profile Building and Sharing

Apart from *what* is to be represented in a profile, which was presented in the previous section, there are two other fundamental questions to be addressed by a profile management system. First, *how* is the information in a profile obtained? Second, *who* has access to it? We will address the how and who in the next subsections. And we note that all of these aspects strongly influence the impact of diversity on WeNet. This is because if one builds a profile with very little information about themselves, or a profile rich with information but the information is kept private, then such information cannot be used when trying to leverage the diversity of profile for connecting people. Imagine a help request that requires a group of people to collaborate on remodelling an apartment where different competences are needed from each party so that the group can successfully fulfil a task, such as painting, plumbing, cleaning, designing, etc. If users' profiles did not have the necessary competences information in their profiles, or they are not sharing such information, then clearly, finding the right group of people that fit the required competences will be impossible. While this section (Section 2.3) describes how aspects of a profile may be shared and the following section (Section 2.4) describes how the profile may later be used by the norms mediating our interactions, Section 2.5 illustrates with a motivating example the impact of private versus shared profile information on the richness and effectiveness of interactions, i.e. the impact of leveraging the diversity of profiles in WeNet.

2.3.1 Profile Building

There are different mechanisms to obtain profile information, from simply asking the individual or the community in question (or its representative) to manually provide this information, or using sensor data that can automatically learn things (like location, busy hours, heart rate, ...), to using interaction data and learned data (e.g. observing who does one interact with often, who is usually preferred for playing padel, ...). However, as we interested in empowering users and maintaining their control, we say given the different mechanisms available for obtaining profile data, it is important to always ensure that the associated individual or community is the one deciding which of those mechanisms to use, and under what conditions. In other words, the individual or community decides how their profile is built. This is specified through *profile building rules*. For example, one individual may decide to disable all sensor data while another might permit the GPS sensor to sense its current location, and one community might only permit its president to manually provide information about it while another might permit any of its users to do so. One community may re-use, adapt, or build on top of existing norms (for example, a new social network may re-use the institutional norms of an existing social network and adapt them to their community's particular needs), whereas another might bring its members to collaboratively specify its norms. While we always stress the need for the entity in question (whether an individual or a community) to be in control, we note that how a community reaches a decision on its profile building rules is outside the scope of this chapter, which could be achieved through collective agreements or other means.

The left hand side of Figure 2.2 illustrates that the profile building rules (BR), specified by the profile owner, are responsible for building the private profile from different data sources. We note that how such data is gathered is largely beyond the scope of this chapter, referring the reader to previous work on the gathering of data [63].

2.3.2 Profile Sharing

Once a decision is made on what data to include in a profile and established the means to gather it, the remaining fundamental question is *who* is granted access to what part of the profile. As illustrated

[1]Here we talk about actions local to one's device, regardless of whether the computations behind these actions (e.g. a decision to send a notification to the user) are performed locally on the same device, executed to the cloud, or a combination of both.

Co-funded by the Horizon 2020 Framework Programme of the European Union

earlier, we require that the individual or community has full access and control over their profile. To have control over who is the profile (or parts of the profile) shared with, we require individuals and communities to define *visibility rules* that determine under what circumstances someone can see part of the individual or community profile. In this respect, an individual's take on privacy (similarly for communities) will determine how she grants access to her profile. Similar to the building rules, we note that how a community reaches a decision on its visibility building rules is outside the scope of this chapter, which may be through collective agreements or other means.

Our stance is that *privacy is not an absolute value*. In other words, not all communities have the same stance on what privacy is. For instance, consider the issue of revealing your ID number. Some community that aims at supporting the elderly might find it crucial to have the ID number of the people visiting the elderly at home. Another community that aims at organising political activities might find that revealing one's ID number is a blatant breech of their users' privacy. Additionally, we say that *privacy is fully contextual*. There is information that one may be willing to share with their family but not with their friends and even less with their foes. For example, one may be happy to share their exact location with friends, and maybe friends of friends, but not with strangers. Some may be happy to share their current city with strangers, while others wouldn't even share that. Therefore, we adopt the notion of privacy being fully contextual in the sense that it depends on the current situation as well as the objectives that one wants to achieve.

The contextuality of privacy brings up the key observation that in social relations there is always a dilemma between *privacy* and *transparency*. On the one hand, I may prefer that sensitive information is not made public to avoid its misuse, and on the other hand I want others to know everything about me that is relevant for the social interaction to help achieve my objectives. This dilemma applies also to online open communities.

Our proposed solution is that the profile elements, data and norms, can be either kept private or can be shared with others. Sharing with others does not necessarily mean making it 'public' (although that would certainly be an extreme case of sharing with others), but it means that the access to the information is granted under certain circumstances. For instance, allow my friends to know my exact location when I am making a request to meet up.

The right hand side of Figure 2.2 illustrates that the visibility rules (VR) are responsible for extracting, from the private profile, the profile data that may be shared with others in different contexts. We elaborate on the context and the contextual profiles shortly.

Note that to have complete control over a profile, building and visibility rules are both needed to be specified by the profile owner, whether an individual or a community (the black boxes of Figure 2.2).

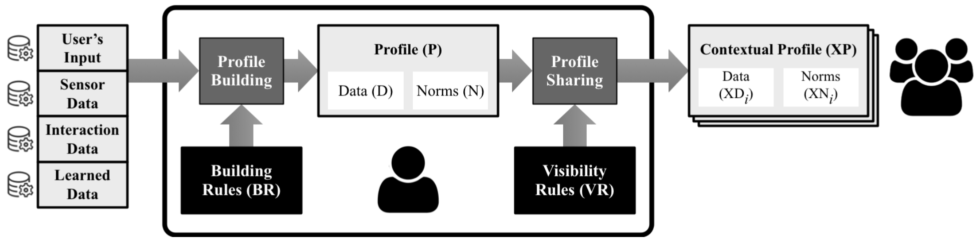

Figure 2.2: How profiles are generated and shared

Private profile

As discussed in [51] (see the related work for details), a profile will contain information about all the relevant aspects of the life of a person or a community, e.g., demographics, personality [13], competences [34], skills or investment plans, but also data which continuously change in time, even during the day, e.g., location, activities, people one is with. This *complete* set of data and norms are, by default, private to (and hence, accessible only by) the profile owner (individual or community) and the system running on the profile owner's own device (we refer to this system that is responsible for making decisions and executing actions the 'decision engine', and it is explained in further detail in Section 2.4.3). This complete and private profile is what is referred to simply as 'Profile' in Figure 2.2, and has been defined in Definition 1. For instance, if 'location("Calle Enric Granados 15, 08008 Barcelona")' is part of Alice's private profile this means that the system (decision engine) running on Alice's device has permission to use Alice's location in the reasoning, but no one else can. Private norms are those that are

Co-funded by the Horizon 2020 Framework Programme of the European Union

never shared with other entities (individual or community) or devices (e.g. 'never bother me when I am taking a nap'). Their impact on behaviour is restricted to one's own device as other devices do not have access to these norms.

Shared profile

The complete profile provides a memory of the complete description of the entity in question (individual or community). Given such a memory, a shared profile is built based on current contextual needs. This is a set of attributes and norms that can be made accessible to others, both humans and the systems (decisions enginers) running on their devices. The mechanism of building a shared profile is analogous to the one people use when meeting another, previously unknown, person and need to provide her with enough information for the task at hand (e.g. certain approaches [6] take inspiration in this model to implement semi-automatic systems for the sharing of information with others at different granularity based on their requests and requirements). Similarly, one may require to abstract the contextually relevant information from the profile and create a shared profile that will preserve privacy by hiding details that are not relevant to the current situation while still containing the information that is needed for the interaction or task. For example, share my age but not my date of birth, or share the city where I live but not the exact address.

We say shared profiles are to be shared with specific people under certain conditions that define the context of the shared profile. For example, besides sharing contact information, in certain cases users may want to share with others their preferences/interests and context-specific sensor readings like number of steps [42]. A definition is provided next.

Definition 4 (Shared Profile)**.** *A shared profile is defined as: $XP = (P', S, C)$, where P' is the part of P that will be shared, S is the set of entities (people or organisations) that are granted access to P', and C is the condition under which this access is granted.*

Note that in Definition 4, it is not necessarily the case that $P' \subseteq P$, as data may be edited before it is shared, as in editing the complete current location to only show the city. Shared profiles, as such, act as access rights, where the condition C simply specifies under what condition do the entities in S have access to the profile P' . As for notation, we note that in the remainder of this chapter, we will use XD to refer to the data part of a shared profile and XN to refer to the norms of a shared profile.

A shared profile, also referred to as contextual profile in Figure 2.2, is created by *visibility and abstraction rules* that we discuss next.

Visibility Rules and Abstraction

A visibility rule determines who can see what and when. For example, I may allow friends to have access to my exact location, while the rest may only have access to the city where I live. These visibility rules help generate the shared profile introduced above. In order to preserve privacy, and as illustrated in our location sharing example, some transformations can be applied as a set of abstraction mappings as defined in [26]. These abstraction mappings take in input an element of the input theory, in this case the knowledge graph introduced in Section 2.1.1, and produce in output a rewrite of this element which captures the desired information hiding. Formally, these mappings are theory mappings which map a given theory into a new theory satisfying the desired constraints [27]. As discussed in detail in [26], based on the theory in [27], there are only three types of abstraction mappings, defined in terms of they operate on entities, attributes and relations, as follows:

Granularity**:** the granularity operator allows for substituting object wholes with one of its object parts. This is when one wants to be more specific. The opposite holds when one wants to be more vague or general. For example, as from Figure 2.1, we can substitute a whole for a part,

Granularity(entity=MyHome)
$\Rightarrow$
entity=Bedroom#1

or, viceversa, a part with a whole,

Granularity(Home.Location = Barcelona)
$\Rightarrow$
Home.Location = Catalonia

In the first case *MyHome* is substituted with a bedroom inside the house, thus making the information more precise while, vice versa, in the second, *Barcelona* is substituted with *Catalonia*, this making the information more generic and less informative.

Generality: the generality operations allow the folding of concepts, attributes and relations towards more general or more specific notions (making them more implicit or specific, respectively). Thus for instance,

$$\text{Generality(concept = father)} \\ \Rightarrow \\ \text{concept = relative}$$

Partiality: : the partiality operation allows for the elimination of entities, attribute values and relation values from the shared profile. Thus for instance

$$\text{Partiality(Car}\{\text{Color=Red, Brand=Toyota}\}) \\ \Rightarrow \\ \text{Car}\{\text{Color=Red}\}$$

The intuition underlying these mappings is to generalise the information content of their input, thus achieving the desired level of privacy. Thus, granularity abstracts a given entity to a more general entity (in the example above, from the city of Barcelona to the region of Catalonia), generality abstracts a concept to a more general concept (in the example above, from the concept of father to the concept of relative) while, last but least, partiality, the most commonly used mappings, allows to forget some elements of the profile (in the example above, the brand of the car). While these three mappings are built in the system, it is up to the owner of the profile to define what is abstracted into what, for whom, under what conditions, etc.

2.4 Architecture and operational model

We organise this section in three parts. First we present how the profile of individuals are organised, then we do the same for communities and, finally, we conclude with a description of the decision engine that is responsible for decision making when it comes to managing behaviour. In summary, this section describes the communication model for specifying norms (task T5.3) as well as the execution environment of the communication model (task T5.5).

2.4.1 The Individual

In Figure 2.3, the schema of the peer-to-peer architecture for our proposed normative system is presented, which essentially describes WeNet's communication model. Each individual has a decision engine on her device that represents her and is used for interacting with others. In other words, individuals interact with each other through their decision engines, and their communication with their device's decision engine happen through a user interface. Whether the decision engine runs all or some of its computations locally or on a remote server is an implementation issue that usually depends on the complexity of the norms and their computational requirements.

As explained in the previous section, each individual has a profile, composed of data and norms. Additionally, each individual has building and visibility rules that define how the profile is built and with whom is it shared (and under what circumstances). As such, the decision engine on an individual's device is represented as having access to these three elements: the building and visibility rules, in addition to the individual's complete profile, which is by default private. We note that each individual has the right to access and edit its own rules and profile.

For the sake of simplicity, we leave the data sources that feed the profiles outside of Figure 2.3, as we choose to focus here on what is needed for individuals to interact with one another as opposed to building profiles. When interacting with others, individuals may decide to share some of its profile. Again, as presented in the previous section, these decisions are based on the visibility rules that specify what parts of the profile may be shared with whom and under what circumstances. The result is the decision engine on individual's device creating contextual profiles that share part of the profile in different contexts, sharing it with selected people under certain circumstances. The contextual profile is illustrated in Figure 2.3 as XP_i, which is composed of contextual data XD_i and contextual norms XN_i, and where i represents the context. Each individual usually has a number of contextual profiles.

Co-funded by the Horizon 2020 Framework Programme of the European Union

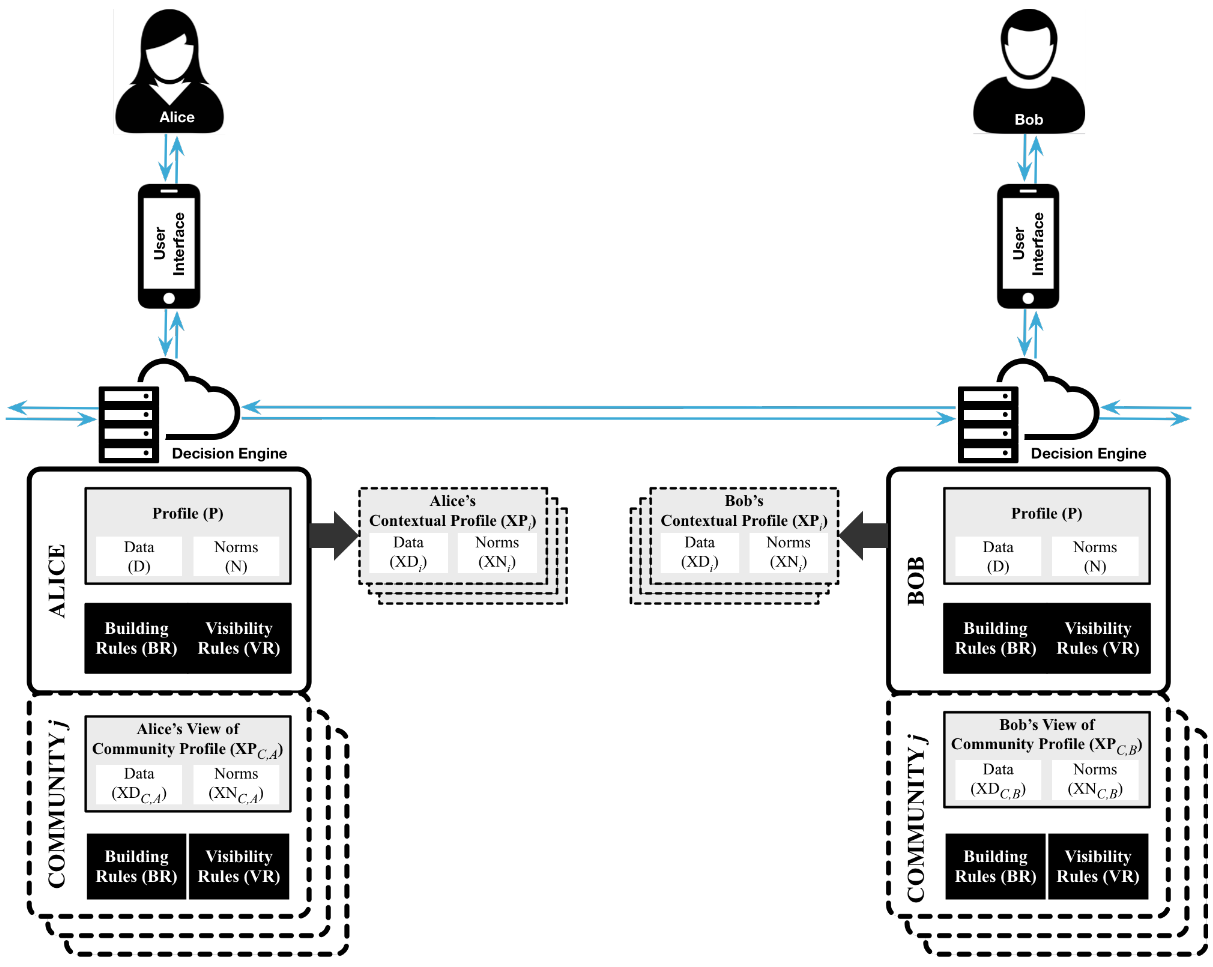

Figure 2.3: Basic (decentralised) architecture - The WeNet Communication Model

2.4.2 The Community

In addition to the individual's profile and their building/visibility rules, communities also have profiles and building/visibility rules. In other words, a community is just another entity with a profile and building/visibility rules. However, different from community members, we say any behaviour in the community should be aligned with the community norms (whereas one's behaviour does not need to be aligned with another's individual norms). As such, giving a community member access to the community's profile (or at least a selection of that profile that concerns that member's interaction in that community) is essential for ensuring that member's interaction adheres to the community's norms. The community may also provide different shared copies of its profile to different members (the $XCP_{C,i}$ in Figure 2.3, where C specifies the community and i specifies the entity that this profile is shared with). For example, a community may share some sensitive profile data with its president, but not with other members. It is the community's visibility rules that will decide what data/norms can be accessed by whom (possibly, including non-members too).

In addition to the visibility rules, building rules are used to clarify who can *edit* the original community profile and how. This is because not all members are equal. Some may be given special rights in a community that allows them to edit data/norms, and sometimes they may even be allowed to edit the rules themselves (building and visibility rules). Though we must note here that data is usually much more accessible for editing than norms, because interactions usually update community's data. For example, with Alice making a new request in the community, the community's total number of requests should automatically get updated.

We have considered centralised and decentralised approaches for implementing the community profile. In the former case, the community profile is saved in a central location. The verification process that grants access to the community profile adheres with the building and visibility rules is centralised. In the latter case, the community profile is located on those devices that communicate and coordinate their actions using distributed ledger technology [12]. Distributed ledger technologies are based on distributed, decentralized peer-to-peer networks where, unlike distributed databases, there is no need for a central administrator (blockchain is one successful example of distributed ledgers). In this chapter, we will assume the decentralised view of the system (to avoid having a centralised authority and control) and will consider that all decisions are local to each device as we illustrate in the following section (and in Figure 2.3).

Lastly, we note that one individual may be a member of more than one community, and hence the decision engine on their device will have a number of such community profiles/rules, as illustrated by Figure 2.3.

2.4.3 The Decision Engine

Every time individual or community profile and rules are being edited, we need to ensure that these actions abide by the building rules. When an action or event happens in a community (e.g. a message is received from another community member, the individual is asking to perform an action, a deadline has passed, ...), we need to ensure that responding to this action adheres to the given norms. For example, if the user is sending a message to other community members, should this message be forwarded, are there any other computations to be carried out, etc. We refer to the engine that reacts to such actions/events and responds accordingly as the *decision engine*. This is achieved by the execution environment of the WeNet communication model, which we refer to as the decision engine.

The decision engine at each device must have both a reactive and proactive behaviour.

- **Reactive Behaviour.** This allows the decision engine to react to messages received (usually representing the actions being performed), and there are two types of messages that a decision engine can receive:
  - *A message from the user interface.* When a user performs an action, it is translated into a message that is sent to its decision engine through the user interface.

    Upon the receipt of such a message, the decision engine needs to first verify that the message does not violate any of the norms (both individual and community norms).

    If the action does not violate any of those norms, then the decision engine needs to decide what to do next, usually translated into sending messages to other entities. This decision follows from the norms that the engine would have checked, and sometimes taking into account some relevant profile data.
  - *A message from another decision engine.* As with the previous case, the decision engine needs to first verify that the message does not violate any of the community norms. This rechecking upon receipt ensures that the sender's decision engine has not been manipulated to cheat. If the message violates any of the community norms, then it may either be discarded, or if the community norms require sanctioning, then the appropriate sanctions should be executed (e.g. decrease the user's reputation).

    However, if the action obeys the community norms, then the decision engine needs to decide what to do next, which is usually translated into sending messages to other entities and/or the user interface. As above, this decision takes into consideration the community and individual norms.
- **Proactive Behaviour.** This allows the decision engine to proactively perform actions as required by the norms. For example, incentivising norms might remind a user to complete their profile, if this has been neglected for some time, or remind the user of how much their contribution to their community is valued, if they haven't been active lately. A norm suppressing messages when one is sleeping might send these messages when the alarm goes off to wake the user. While external events might trigger reactive behaviour, we argue that internal events trigger proactive behaviour (e.g. reaching a timeout).

The decision engine is triggered when an action is performed, and we view all actions as messages. For example, the user pressing a button is translated into a message from the user to its decision engine. Messages may be of two types, those received by decision engine from its associated user (in other words, one's actions are translated to messages that are sent to the user's own decision engine), and those received by other decision engines. Notice that here we put the restriction that no one can send messages directly to another user's decision engine, before having their message passing through their

Co-funded by the Horizon 2020 Framework Programme of the European Union

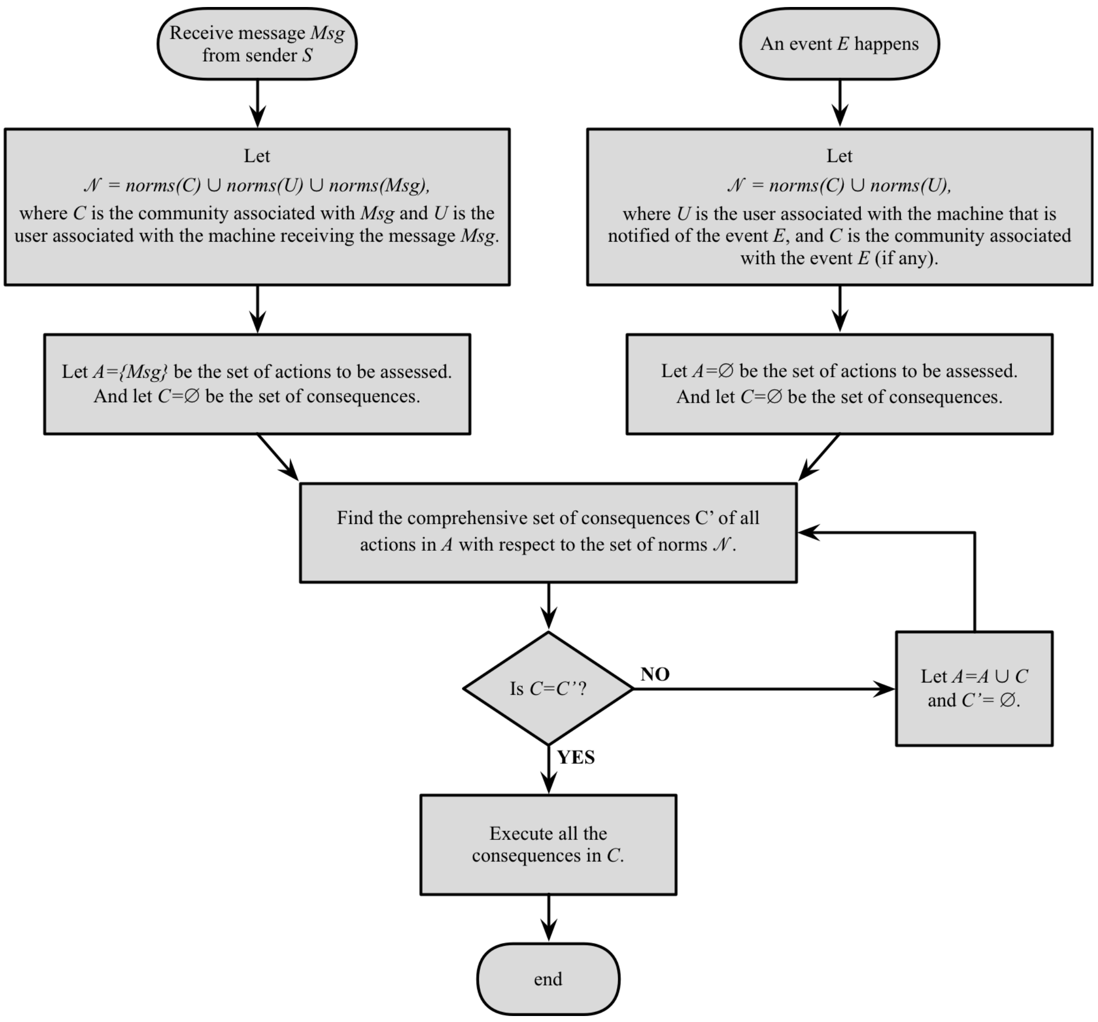
Receive message *Msg* from sender *S*
An event *E* happens
Let $\mathcal{N}$ = norms(C) ∪ norms(U) ∪ norms(Msg), where *C* is the community associated with *Msg* and *U* is the user associated with the machine receiving the message *Msg*.
Let $\mathcal{N}$ = norms(C) ∪ norms(U), where *U* is the user associated with the machine that is notified of the event *E*, and *C* is the community associated with the event *E* (if any).
Let *A*={*Msg*} be the set of actions to be assessed. And let *C*=∅ be the set of consequences.
Let *A*=∅ be the set of actions to be assessed. And let *C*=∅ be the set of consequences.
Find the comprehensive set of consequences C' of all actions in *A* with respect to the set of norms $\mathcal{N}$.
Is *C*=*C'*?
NO
Let *A*=*A* ∪ *C* and *C'*= ∅.
YES
Execute all the consequences in *C*.
end

Figure 2.4: An illustration of the decision engine's algorithm

Co-funded by the Horizon 2020 Framework Programme of the European Union

own decision engine first. Another issue to note is that we assume interactions happen in communities. As such, each message is associated with a given community.

The decision engine may also be triggered when an event happens (e.g. an alarm goes off, or a timeout is reached). In this case, the decision engine will require a list of relevant events that may trigger it and their associated communities (when applicable). For example, an alarm that marks that a community's deadline is near will be associated with that specific community, but an alarm that wakes up a person might not be associated with any community.

Figure 2.4 illustrates the behaviour of the decision engine. If triggered by receiving a message, it extracts all the norms relevant for that message, that is, the associated community norms, the individual norms associated with the user of this decision engine, and other norms that have been associated with this specific message (e.g. if one wants others to know that she is only looking for people in her vicinity, then this norm gets attached with the message). If triggered by an event, the relevant norms to be checked by the decision engine are then the individual norms associated with decision engine's user, and the norms of the community associated with the event, if any.

After compiling the set of relevant norms, the decision engine checks the norms one by one in order to see assess the consequences with respect to the triggering message or event. For example, does it need to perform some computations? Send some information back to its user? Forward the incoming message (if any) to another decision engine? Set a timer to perform some action at a later time? These consequences are usually specified by the norms. However, after compiling the complete set of consequences, and before executing them, the decision engine needs to make sure that these consequences do not have consequences themselves. As such, it goes into a loop (see the loop in Figure 2.4) to check the consequences of the consequences, and will continue to repeat this until there are no new consequences arising. When that is reached, the compiled set of consequences is executed, and the job of dealing with the triggering message or event is done. We note that the order isn't important here because we are only working with the consequences of one agent's action (that is, we are not dealing with the order of messages or actions in an interaction between more than one entity). Furthermore, our interactions happen in an asynchronous system, so sending one message a fraction of a second before another will not make much of a difference. What is important, however, is having incoherent or incoherent consequences, which we leave for future work.

Properties of the decision engine. In what follows we present a few properties of our decision engine's algorithm.

The first property is a property about the decision engine's algorithm itself, namely, its finiteness. Despite having a loop, the algorithm always comes to an end after a finite number of steps.

Property 1 (Finiteness)**.** *The decision engine's algorithm will always terminate after a finite number of steps.*

Proof Sketch. For any message/event triggering the decision engine, the decision engine will check the norms one by one and compile a set of consequences C'. The decision engine then loops to check the consequences of C'. And so on. The decision engine exits this loop when there are no more new consequences to consider.

As the set of norms is finite, it is then inevitable that the set of consequences will also be finite. With a finite set of consequences, the decision engine is guaranteed to eventually exit this loop and terminate its execution. □

The second property is a property about community behaviour. It states that with our proposed normative-based system, norms are a necessary condition for any behaviour to emerge in a community. If the set of norms is empty for a given community, then nothing can happen in that community.

Property 2 (Necessity of Norms)**.** *Norms are a necessary condition for any behaviour to emerge in a community.*

Proof Sketch. Following the algorithm of Figure 2.4, for every message or event that will trigger the norm engine, the set of norms to be evaluated will be retrieved. Now let us assume that this set of norms is empty: $\mathcal{N} = \emptyset$. The algorithm will try to go through existing norms one by one to check their relevance with respect to the triggering message/event and extract the corresponding consequences when appropriate. However, as the set of norms is empty, then there are no norms to check and the set of consequences will be empty too: $C = \emptyset$. With no consequences, the triggering message/event will result in no actions to be performed. With no actions performed, no behaviour can emerge in the community, regardless of the triggering messages and events. □

Co-funded by the Horizon 2020 Framework Programme of the European Union

The third property is about the propagation of messages in a community. It essentially states that as long as the norms require the forwarding of messages to all adjacent nodes, and there are no other norms that condition this forwarding, then any two nodes connected by a path can send messages to each other.

Property 3 (Reachability). *If there exists a norm that requires sending and forwarding messages to all neighbouring nodes unconditionally, then a message sent by node n_0 can propagate through a path $P = \{n_0, n_1, ..., n_l\}$ of any length $l \in \mathbb{N}^*$.*

Proof Sketch. Say there is a norm that states that any message to be sent shall be sent to all neighbouring nodes:

```
IF
    to_send(N,M)
    and neighbours(N,NN)
THEN
    send(N,NN,M)
```

where `to_send(N,M)` states that node `N` wants to send the message `M`, `neighbours(N,NN)` states that the set of all neighbouring nodes of `N` is `NN`, and `send(N,NN,M)` states that a message `M` is to be sent from node `N` to the set of neighbouring nodes `NN`.

Also say there is a norm that states that any received message is to be forwarded to all neighbouring nodes:

```
IF
    received(N,N',M)
    and neighbours(N',NN)
THEN
    send(N',NN,M)
```

where `received(N,N',M)` states that a message `M` has been received by `N'` from `N`.

And say there exists no other norm that conditions the above behaviour: the behaviour of sending and forwarding messages to all neighbouring nodes.

Now we show that a message sent by n_0 will propagate through a path $P = \{n_0, n_1, ..., n_l\}$ of any length $l \in \mathbb{N}^*$.

First, we note that when node n_0 sends a message, the decision engine of Figure 2.4 will send this message all neighbouring nodes of n_0, including node n_1, and that is in accordance with the consequences of the first norm presented above. As such, we show that a message propagates through a path of length 1.

Second, we note that if a message propagates through a path P of length m then it will propagate through a path P of length $m+1$. This is because if a node n_m receives a message, the decision engine will result in sending the message to all neighbouring nodes of n_m, which include the node n_{m+1}. And that is in accordance with the consequences of the second norm presented above. As such, we show that if a message propagates through a path of length m, then it will propagate through a path of length $m + 1$.

Given that a message is guaranteed to propagate through a path of length 1, and given that if a message propagates through a path of length m then it will propagate through a path of length $m + 1$, by inductions, we can then say that a message can propagate though a path of any length $l \in \mathbb{N}^*$. $\square$

There are other properties of interest that we leave for future work. For example, while Property 3 is based on the norm that all nodes will forward a message to all other neighbouring nodes (that is, the probability of forwarding a message to all neighbouring nodes is 1), it would be interesting to show that reachability decreases as the probability of forwarding a message to neighbouring nodes goes below 1. Such a property helps, for example, assess the impact of privacy on reachability. We know that norms usually make heavy use of profile data. As such, the more the data is private, then the less effective the norms can be. For example, if the address field of the majority is kept private, then a norm that only looks for people nearby won't be very successful in locating those nearby. And if reachability is affected by such norms, then reachability will certainly decrease with the increase of privacy. The proof of such interesting properties, however, will be experimental proof, where one can make use of simulations to verify the property in question. As mentioned earlier, this is left for future work.

Co-funded by the Horizon 2020 Framework Programme of the European Union

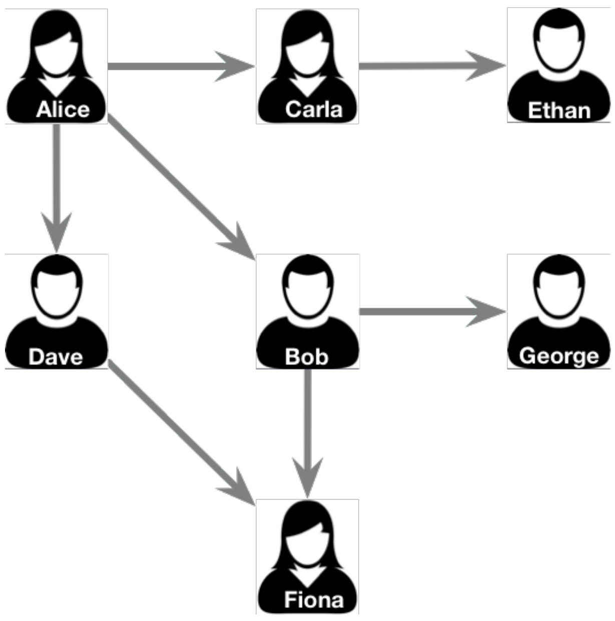

Figure 2.5: The social network of the WeNet example

2.4.4 WeNet Implementation

The decision engine has been implemented in the WeNet platform as the "Interaction Protocol Engine" component of Figure 1.1. The profiles, for both individuals and communities, are designed and implemented by the "Profile Manager" component of Figure 1.1. Both data and norms are specified in those profiles, and this data is populated by various other components, as illustrated in Figure 1.1. A sample of the community norms implemented for WeNet's first pilot, specified as part of the community profile, are presented shortly in Section 2.6. Finally, there is the "Task Manager" component that is designed to keep track of interactions so they can be analysed and used by other modules, including the decision engine (or the Interaction Protocol Manager).

2.5 A motivating Example

In this example we will specify the interaction between a number of people in an open community of mutual help. The community is inspired with the WeNet use case in mind, an open community allowing one to find help with everyday tasks, such as picking up one's child from school, finding some friends to dine with, or finding some people to play padel with. It finds help by propagating help requests through one's social network. In our specific example, we keep the community's social network very limited for the sake of simplicity. Figure 2.5 presents the social network associated with our example, where a link from node n to n' represents that n' is a friend of n. The relevant community and individual profiles associated with the nodes of Figure 2.5 are presented in Figure 2.6.

The community profile that defines a community contains data about the community (Lines 1–5 of Figure 2.6), such as the list of members of the community, the list of suspended accounts, etc. The profile also contains the norms that specify the rules of interaction. These rules help shape community behaviour, and in our example, they attempt to increase collaboration. The first rule, or community norm (Lines 6–13), restricts continuous requests for help if the requester hasn't been volunteering himself (it essentially doesn't permit 5 consecutive requests without making any offer to help). The second community norm (Lines 14–18) enforces a strict penalty on volunteers that commit to helping others and then fail to go through with their commitment, by suspending their participation in the community for 24 hours (and suspended accounts cannot make new requests for help: Lines 19–22).

In addition to community norms that govern community behaviour, individuals may also have their own norms, also saved as part of their profiles. For example, Alice has a norm that states that only people closeby (in the same city) may receive her requests (Lines 34–37). Ethan has a private norm that states that notifications are to be suppressed when he is napping (Lines 55–58). Fiona, a professional padel player, has a private norm that states that requests to play padel are to be ignored if they come

Co-funded by the Horizon 2020 Framework Programme of the European Union

```
D_community = { members(alice, bob, carla, dave, ethan, fiona);
                suspended{};
                consecutive_requests(alice,3);
                consecutive_requests(bob,5);
                ...   }
N_community = {IF
                    attempt_new_request(Requester,Request,RRNorms) and consecutive_requests(Requester,X) and X<5
                THEN
                    new_request(Requester,Request,RRNorms);
                IF
                    attempt_new_request(Requester,_,_) and consecutive_requests(X) and X=5
                THEN
                    message(Requester, "You may not request help as you first need to offer help.");
                IF
                    committed_to_offer_help(X,Request) and failed(X,Request)
                THEN
                    suspend(X) and message(Requester, "Your account is suspended for 24 hours because you failed to fulfil your
                                                    commitment.");
                IF
                    attempt_new_request(Requester,_,_) and suspended(Requester,true)
                THEN
                    message(Requester, "You may not make new requests as your account has been suspended for 24 hours.")   }
XD_community,alice = {
                suspended(alice,false);
                consecutive_requests(alice,3);
                ...   }
XD_community,bob = {
                suspended(bob,false);
                consecutive_requests(bob,5);
                ... }
...
D_alice = {   location("Calle Enric Granados 15, 08008 Barcelona");
                competency(padel,novice)   }
N_alice = {   IF
                    new_request(alice,Request,_) and city_location(alice,City)
                THEN
                    friends_in_city(City,Friends) and forward_request(alice,Request,Friends)   }
XD_alice,all = {location(alice, "Barcelona"};
                competency(alice,padel,novice)   }
XN_alice,all = {IF
                    new_request(alice,Request,_) and city_location(alice,City)
                THEN
                    friends_in_city(City,Friends) and forward_request(alice,Request,Friends)   }
D_bob = {    location("Carrer de Verdi, 32, 08012 Barcelona");
                has(car,true)   }
XD_bob,all = { location(bob, "Barcelona");
                has(bob,car,true)   }
D_carla = {   location("Passeig de Gràcia, 43, 08007 Barcelona")   }
XD_carla,all = {location(carla, "Barcelona")   }
D_dave = {    location("Pg. de Sant Joan, 152, 08037 Barcelona");
                has(car,true)   }
D_ethan = {   location("Carrer de Balmes, 197, 08006 Barcelona");
                has(car,true);
                competency(padel,professional)   }
N_ethan = {   IF
                    naptime(true) and notify(Request)
                THEN
                    supress_notification(Request)   }
XD_ethan,all = {location(ethan,"Carrer de Balmes, 08006 Barcelona");
                has(ethan,car,true)   }
D_fiona = {   location("Av. de Sarrià, 45, 08029 Barcelona");
                has(car,true);
                competency(padel,intermediate)   }
N_fiona = {   IF
                    new_request(Requester,Request) and request_type(play_padel) and ¬ competency(Requester,padel,professional)
                THEN
                    ¬ notify(fiona,Request)   }
XD_fiona,all = {location(fiona, "08029 Barcelona");
                has(fiona,car,true)   }
D_george = {  location("9 Bywater St, London SW3 4XD, UK");
                has(car,true)   }
XD_george,all = {location(george, "London"),
                has(george,car,true)   }

XN_alice,all,requestId = {
                IF
                    receive_request(alice,R) and ¬ has(X,car)
                THEN
                    ¬ notify(X,R)  }
```

Figure 2.6: WeNet example: individual and community profiles (data and norms)

Co-funded by the Horizon 2020 Framework Programme of the European Union

from novice players (Lines 64–67). In addition to individual norms, profiles hold data about the user, like their current location, their competency in padel, whether they own a car or not, etc. Not all individual profiles specify all data attributes, this is up to the user to decide what to save in their profile. It is also up to the user to decide what profile information to share and with whom. To keep the example simple, we keep the visibility rules out of Figure 2.6, and we present the shared data and norms.

In this example, Alice is sharing her location, her level at playing padel (novice), and her norm that requires that only people in the same city may receive her request (Lines 38-43). Notice that we use 'all' in the notation for shared norms or data (e.g. $XN_{alice,all}$ or $XD_{alice,all}$) to state that this part of the profile is to be shared with everyone. Bob, Ethan, Fiona and George are sharing the city where they are located, which is extracted from their private location, and whether they have a car or not (Lines 46-47, 59–60, 68–69, and 72–73, respectively). Carla is sharing her location city (Line 49). Dave is not sharing any information about himself.

Now say Alice is looking for someone to play padel with tonight. For this specific request, she might add an additional norm, such as they must have a car to drop her off later at night. This additional norm is shared with everyone in the specific context of this request (see $XN_{alice,all,requestId}$ on lines 75–79 of Figure 2.6). The norm essentially states that whoever receives a request from Alice, they should not be notified about the request if they do not have a car.

Alice will attempt to send her request on the WeNet platform, with the new request-related norm embedded (a context-based norm that is attached to a specific request). The objective of WeNet is to start propagating her request within her social network, starting from her friends, to her friends of friends, and so on. This is achieved with each decision engine that receives the request, starting with her own decision engine, deciding whether it needs to send its user a notification about this request or not, and whether it should forward it to friends or not. Decisions of a decision engine are made by checking community norms, the individual norms of the associated user, the requester's shared norms, and any request-related norm associated with the specific request.

The steps for propagating the request and finding a volunteer is described next by the reaction of the different decision engines at the different stages of the request propagation. Note that the interaction here is asynchronous. Of course, *some* actions will happen in a specific order. For example, Alice's decision engine must first kick of the propagation of the request before other decision engines can start receiving messages and reacting to them. Or Ethan's decision engine must first receive the request from Carla's before it can react to it. However, we do not know for sure whether Carla's decision engine will receive the request before Dave's or Fiona's, for example. As such, the steps below describing the reaction of the different decision engines does not have a specific order (keeping in mind, of course, that a decision engine must receive a message before reacting to it).

- *Alice's decision engine – Initiating the propagation:* Receiving Alice's request, Alice's decision engine first checks whether it violates any community norms. As the number of consecutive requests made since Alice's last offer for help is 3 (Line 25), it fulfils the first community norm (Lines 6–13) and doesn't break any other community norm (for instance, she is not breaking the norm on Lines 19–22 because her account is not suspended —Line 24). Individual and request-related norms are also not broken. As such, Alice's decision engine decides to propagate the message to her friends Bob, Carla and Dave.

- *Carla's decision engine:* Checking the relevant norms, Carla's decision engine decides that Carla should not be notified of the request, and simply forwards the message to Carla's friends (in this simplified network, just Ethan). This is because Carla does not have any information on whether she owns a car or not, and the request-related norm requires the recipient of the request to own a car.

- *Dave's decision engine:* Dave's decision engine also suppresses sending the notification to Dave and simply forwards the request to Dave's friends (in this simplified network, Fiona). This is because Dave's location and car ownership are kept private when there is a shared norm from the requester (Alice) requiring Dave to be in Barcelona and a request-related norm requiring Dave to have a car.

- *Bob's decision engine:* Unlike Carla and Dave, Bob fulfils all requirements. He is in the same city as Alice (Barcelona) and has a car. As such, he receives a notification about Alice's request, and his decision engine forwards the request to his friends (in this simplified network, Fiona and George).

Co-funded by the Horizon 2020 Framework Programme of the European Union

- *Fiona's decision engine:* Fiona's decision engine receives the request from both Bob and Dave's decision engines, but Fiona has a private norm that ignores invitations to play padel if they come from novice players. As Alice's shared profile with Fiona states that she is novice at padel, Fiona's decision engine does not send a notification about Alice's request to Fiona.
- *George's decision engine:* George's decision engine receives the request from Bob, but as George is currently in London, his decision engine does not send him the notification about Alice's request (as it break's Alice's shared norm that requires being in the same city as Alice).
- *Ethan's decision engine:* Ethan's decision engine receives the request from Carla's, but a notification to Ethan is momentarily suppressed as Ethan is taking a nap and he has a private norm that requires suppressing notifications when napping. Finally, when Ethan wakes up from his nap, he receives Alice's request and he accepts.
- *Alice's decision engine – Finding a volunteer:* Alice's decision engine receives Ethan's acceptance, and it decides to notify Alice about this. Alice now has one volunteer to play padel with that fulfils her requirements.

For the sake of simplicity, the reader will notice that the example has been extremely simplified. For example, we do not explain how the predicate "suspend(X)" (Line 17) manages the list of suspended accounts (Line 2), or how the predicate "city_location(_)" (Line 35) extracts the city from a given location. The objective of this example is to illustrate how our proposed system ensures the interaction between people adheres to both community norms and individual ones without jeopardising people's privacy. It also illustrates the impact of private and public information (whether it was concerning data or norms) on both local and external decisions processes. For instance, we note that private individual information (data or norms) are better suited to control local behaviour, whereas shared individual information are better suited for controlling the behaviour (or decision process) on others' decision engines. For example, to see how shared norms can have an impact on other decision engines: notice that all decision engineers are aware of Alice's norm of restricting notifications to those in the same city, but only the impact of a private norm is local to the decision engine of the private norm's owner only. For example, Fiona's private norm filters the notifications sent to Fiona concerning padel requests to those that come from professional padel players. No one needs to know Fiona's restriction. And if a requester does not share their expertise on padel with Fiona, then their request will never get to Fiona, without them being aware of this.

Also note that for privacy reasons, not sharing some information assumes that the information does not exist. For instance, Dave fulfils Alice's requirements as he has a car and he is in the same city. And Dave's decision engine is fully capable of confirming this as it has access to his private data. But by notifying Dave of Alice's request, Alice can automatically deduce that Dave is in the same city (if he accepts). And as such, Dave's privacy concerning his location would be broken. For this reason, Dave's decision engine assumes that private data is not used for actions that have implications outside Dave's decision engine.

And again for privacy reasons, the community only shares the account suspension information with the account holder only: each community member can only know whether their own account has been suspended or not.

2.6 Sample Protocols: The WeNet Norms

This section presents the WeNet community norms used in the first pilot, which are the sample protocols of task T5.4. These are presented in Figure 2.7, in a simplified language for the sake of clarity and simplicity. Norm 1 states that when a task is created by a requester, the list of people to forward this to is obtained (currently a randomly selected number of people in the community), and a message is sent to all those informing them of the new task. Note that in all of the norms, logs are kept concerning any change in the state of that task (created, answered, completed, etc.). Norm 2 focuses on the people informed of the created task, the recipients. The norm simply states that every time such a message is received, the human user is to be notified automatically. Norm 3 states that when a recipient sends an answer, in the case the task hasn't been closed yet, then this answer is automatically forwarded to the task's requester. Norm 4 states that when the requester's machine receives an answer, then the user is notified about this. Norm 5 states that when a recipient decides to ignore a task, then nothing is done other than logging this info. Norm 6 states that is the requester chooses her best answer then the task is

Co-funded by the Horizon 2020 Framework Programme of the European Union

marked as closed and the author of the best answer is informed. Norm 7 states that when the machine of the best answer's author receives this message, then the user is notified. Norm 8 states that when the requester decides to ask more people, then the new list of people to be notified is obtained and a message is forwarded to the members of this list. Norms 9/10 state that when a recipient/requester reports a question/answer then this is logged in the system.

Note that these norms are very basic in the sense that they simply focus on forwarding messages without adding any conditions, such as only ask people nearby, or do not bother me during my working hours. More interesting conditions are being added to the norms of subsequent use cases. Links with other WPs are also evident in the newer version of these norms, when the norms connect to WP2's API to get the list of people nearby, or WP3's API to get freindship levels between people. These new norms will be reported on in the next version of this deliverable, D5.3

2.7 Related Work

The main idea that our proposed architecture is built upon is the profile, which is composed of the more traditional data element, as well as the more novel norms element. As such, in this section, we present the related work in the fields of both profiles and normative systems.

The issue of how to define meaningful profiles has been extensively studied in various sub-fields of Artificial Intelligence, see e.g. [30, 55]. Two are the main differences with the notion of profile presented here. The first is that, in all this previous work, what is being profiled is the *user* while in this work we profile *people*, i.e., their overall behaviour, independently of whether this behavior involves machines. As a matter of fact, the profile in Section 2.2 is mainly focused on people's everyday life properties. The second is that, in all the previous work, the profile is built by the system, largely independently from the user, for instance, in order to provide the most relevant product [40, 44]. In the work described here, the profile is built under the total control of the entity being profiled, and with the goal, not to enable a better system behaviour, but rather to be applied by other people, as a key ingredient for enabling better social interactions. It is important to point out also that some of the approaches used to represent and manage the knowledge in the profiles are based on semantic web techniques [8, 17, 24].

The notion of profile presented here shares some basic principles with the work on *contextual privacy* [46, 5]. In this work, agents are associated with a set of attributes which describe them, i.e., their *profile*. Key elements of profiles are *roles*, namely properties that characterise the way something, e.g., an agent, participates in some course of action. In this work, agents may hold multiple roles in parallel and usually hold them for some limited amount of time; thus, for instance, an agent can be at the same time a doctor and the recipient of a message. As from [46], agents interact via *communication actions* where each communication action consists of a sender, a recipient and a message, and the *context* of a communication action is the sets of roles that the involved agents have in that communication actions. Many of the ideas are common: we both profile people, rather just users, and we both have the idea of having the profile, in our case the public profile, defined in terms of the current interaction context. We also have a notion of role, where we take roles as described in [31, 43], which seems very similar to their notion. The key difference is that the work described in [46, 5] is foundational and focused on the basic principles while here we propose an approach that uses context-driven profiles to adapt what is shared to the different contexts.

The notion of (lifelong) management of personal data is discussed in [51]. This work, which is rather general and focused on basic principles, provides useful guidelines for how to store, maintain and use personal data. Of specific interest is the notion of *partial identity*, where a partial identity is the description of a person within a certain (situational) context. Thus, a person may have a partial identity at work, another when shopping, another when in vacation and so on. Furthermore, these partial identities evolve and change in time following the dynamics of the life of a person. Many of the long term issues described in this work (e.g., the minimisation of data made available to third parties) are implemented in the private profile, as implemented inside iLog, and also via the implementation of the public profile. As a matter of fact our notion of public profile can be seen as an implementation of the idea of partial identity. Related and motivated by the ideas in [51] is the work on *PPL*, for *Primelife Policy Language* [57, 2]. The idea of norms which can be circulated together with data and which can be used to define how these data should be used maps directly to the PPL notion of *sticky policies*. With respect to the general idea of sticky policies, the type of norms that we have considered in this chapter are limited to the management and circulation of the personal profile.

Another important aspect of our profiles is their inclusions of norms, allowing the proposed system to act as a normative system. Normative systems have attracted considerable attention in the multi-agent

Norm 1:	**if**	a task is created **and**
		the list of people eligible to be informed about this task is X
	then	log the created task **and**
		send a message to all people in X informing them of the new task.
Norm 2:	**if**	a message of type "new task" is received
	then	notify the user of this information.
Norm 3:	**if**	a message of type "answer" is being sent **and**
		the task is not yet closed
	then	log the message being sent **and**
		send the answer to the task's requester.
Norm 4:	**if**	a message of type "answer" is received
	then	notify the user of this information.
Norm 5:	**if**	a message of type "no answer" is being sent **and**
		the task is not yet closed
	then	log this message.
Norm 6:	**if**	a message of type "best answer" is being sent **and**
		the task is not yet closed
	then	log the message being sent **and**
		close the task **and**
		send the message to the author of the selected best answer.
Norm 7:	**if**	a message of type "best answer" is received
	then	notify the user of this information.
Norm 8:	**if**	a message of type "ask more people" is being sent **and**
		the task is not yet closed **and**
		the new list of people eligible to be informed about the task is X
	then	log the action of asking more people **and**
		send a message to all people in X informing them of the task.
Norm 9:	**if**	a message of type "report question" is being sent **and**
		the task is not yet closed
	then	log this report.
Norm 10:	**if**	a message of type "report answer" is being sent **and**
		the task is not yet closed
	then	log this report.

Figure 2.7: The community norms of WeNet's first use case

Co-funded by the Horizon 2020
Framework Programme of the European Union

systems community as one approach to maintaining the autonomy of agents while ensuring community goals and aspirations are fulfilled. Relevant work in this field is the work on electronic institutions [16] that help organise collective activities by restricting interactions to abide by some established conventions (which may be understood as norms). While normative systems have excelled at addressing issues such as coordination and cooperation [1], they have left a number of open challenges. In this chapter we deal with two such issues which are crucial in the design of open communities. The *first* is how to reconcile individual goals with community goals. A number of approaches have been studied to take the individual into consideration, such as norm synthesis techniques that would help norms evolve based on individuals' behaviour [45], or norm evolution that would allow the individuals to reason about norms through argumentation [47]. But what about individual norms that one is not willing to share with their fellow community member? The *second* is that in such an open environment the individual privacy should be protected. For instance one would not want to let others know of the blatant contradiction between community and individual norms.

Also relevant to our work is the work in agent-based simulations [41], where a theory of agent behaviour for specific contexts is needed to model agent behaviour. While behavioural models are usually used to model agents, normative models may also be used for developing a heuristic model of behaviour. However, like others, these do not provide solutions to the two issues we raise above (that is, reconciling individual goals with community goals, and protecting individual privacy in open environments).

2.8 Conclusion and Future Work

This chapter has proposed a decentralised architecture for normative systems (the WeNet communication model) that introduces individual norms, while ensuring the privacy of people. These ideas and architecture, including the decision engine of Section 2.4.3 (the execution environment of the WeNet communication model) are being developed, and continuously evolving in WeNet. The implementation integrates a pre-existing system: the *iLog system* [63], the core of the private profile component, as originally specified in [25, 24, 56]. Considerable effort has been devoted to the development of techniques for learning profile data from sensor data and human-machine interactions, and this has implemented as part of the work described in [62, 28, 56].

Our current next steps are an extension of the existing WeNet platform aimed at introducing different types of norms and corresponding different types of profiles. In fact, as illustrated in Section 2.6, norms can be used to specify the rules of interaction in a community (the interaction protocol), but also to introduce more specialised rules, such as rules specifying what is considered ethical and unethical, or rules specifying how to motivate people to act in a certain way. Working on incentives and linking them with norms is an ongoing work, which we hope to report on next. Furthermore, norms may be used as a tool to investigate how interactions may leverage people's diverse profiles. For example, an individual norm may specify whether the requester is looking for similar or diverse profiles for the request in question.

One aspect that has not been analysed in this chapter and left for future work is the conflict resolution mechanism. Having people specify their own norms will probably result in conflicting norms (or rules), and a mechanism will be needed to address such conflicts.

Last, but not least, we have illustrated with the example of Section 2.5 the impact of sharing (or not) data and norms on leveraging diversity in WeNet. Future work may formally explore the properties of our proposed system, especially when it comes to understanding private versus shared profile data and norms.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Bibliography

[1] Giulia Andrighetto, Guido Governatori, Pablo Noriega, and Leendert W. N. van der Torre, editors. *Normative Multi-Agent Systems*, volume 4. Dagstuhl Publishing, 2013.

[2] Julio Angulo, Simone Fischer-Hübner, Tobias Pulls, and Erik Wästlund. Towards usable privacy policy display & management-the primelife approach. In *HAISA*, pages 108–118, 2011.

[3] V. Arnaboldi, M. Conti, A. Passarella, and F. Pezzoni. Analysis of ego network structure in online social networks. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 31–40, 2012.

[4] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In *Handbook on ontologies*, pages 3–28. Springer, 2004.

[5] Adam Barth, Anupam Datta, John C Mitchell, and Helen Nissenbaum. Privacy and contextual integrity: Framework and applications. In *2006 IEEE symposium on security and privacy (S&P'06)*, pages 15–pp. IEEE, 2006.

[6] Igor Bilogrevic, Kévin Huguenin, Berker Agir, Murtuza Jadliwala, and Jean-Pierre Hubaux. Adaptive information-sharing for privacy-aware mobile social networks. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, page 657–666, New York, NY, USA, 2013. Association for Computing Machinery.

[7] Mark Blum, Alex Pentland, and Gerhard Troster. Insense: Interest-based life logging. *IEEE MultiMedia*, 13(4):40–48, 2006.

[8] Piero A. Bonatti, Stefan Decker, Axel Polleres, and Valentina Presutti. Knowledge graphs: New directions for knowledge representation on the semantic web (dagstuhl seminar 18371). *Dagstuhl Reports*, 8:29–111, 2018.

[9] Andrea Bontempelli, Stefano Teso, Fausto Giunchiglia, and Andrea Passerini. Learning in the wild with incremental skeptical gaussian processes. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[10] Paolo Bouquet and Fausto Giunchiglia. Reasoning about theory adequacy. a new solution to the qualification problem. *Fundamenta Informaticae*, 23(2, 3, 4):247–262, 1995.

[11] Melissa Brough, Ioana Literat, and Amanda Ikin. "good social media?": Underrepresented youth perspectives on the ethical and equitable design of social media platforms. *Social Media + Society*, 6(2):205630512092848, 2020.

[12] Daniel Burkhardt, Maximilian Werling, and Heiner Lasi. Distributed ledger. In *2018 IEEE international conference on engineering, technology and innovation (ICE/ITMC)*, pages 1–9. IEEE, 2018.

[13] Gokul Chittaranjan, Jan Blom, and Daniel Gatica-Perez. Mining large-scale smartphone data for personality studies. *Personal and Ubiquitous Computing*, 17(3):433–450, 2013.

[14] Jonathon N. Cummings, Brian S. Butler, and Robert E. Kraut. The quality of online social relationships. *Communications of the ACM*, 45(7):103–108, 2002.

[15] Mark d'Inverno, Michael Luck, Pablo Noriega, Juan Rodriguez-Aguilar, and Carles Sierra. Communicating fiopen systems. *ARTIFICIAL INTELLIGENCE*, 186:38–94, 7 2012.

Co-funded by the Horizon 2020 Framework Programme of the European Union

[16] Mark D'Inverno, Michael Luck, Pablo Noriega, Juan A. Rodriguez-Aguilar, and Carles Sierra. Communicating open systems. *Artificial Intelligence*, 186:38–94, 2012.

[17] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In *SEMANTiCS (Posters, Demos, SuCCESS)*, 2016.

[18] European Commission. White paper on artificial intelligence: A european approach to excellence and trust, 2020.

[19] European Parliament. General data protection regulation (regulation (eu) 2016/679): Legislative act, 2016.

[20] Samer Faraj, Sirkka L. Jarvenpaa, and Ann Majchrzak. Knowledge collaboration in online communities. *Organ. Sci.*, 22(5):1224–1239, 2011.

[21] Fernback and J. Beyond the diluted community concept: a symbolic interactionist perspective on online social relations. *New Media & Society*, 9(1):49–69, 2007.

[22] F. Giunchiglia, V. Maltese, and B. Dutta. Domains and context: first steps towards managing diversity in knowledge. *Journal of Web Semantics, special issue on Reasoning with Context in the Semantic Web*, pages 53–63, 2012.

[23] Fausto Giunchiglia. A diversity-aware internet, when technology works for people, 2020. `https://ec.europa.eu/digital-single-market/en/news/diversity-aware-internet-when-technology-works-people`.

[24] Fausto Giunchiglia, Enrico Bignotti, and Mattia Zeni. Human-like context sensing for robot surveillance. *International Journal of Semantic Computing*, 12(01):129–148, 2017.

[25] Fausto Giunchiglia, Enrico Bignotti, and Mattia Zeni. Personal context modelling and annotation. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 117–122. IEEE, 2017.

[26] Fausto Giunchiglia and Mattia Fumagalli. Teleologies: Objects, actions and functions. In *ER-International Conference on Conceptual Modeling*, pages 520–534. ICCM, 2017.

[27] Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artificial intelligence*, 57(2-3):323–389, 1992.

[28] Fausto Giunchiglia, Mattia Zeni, and Enrico Big. Personal context recognition via reliable human-machine collaboration. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 379–384. IEEE, 2018.

[29] Fausto Giunchiglia, Mattia Zeni, Elisa Gobbi, Enrico Bignotti, and Ivano Bison. Mobile social media usage and academic performance. *Computers in Human Behavior*, 82:177–185, 2018.

[30] M. Golemati, A. Katifori, C. Vassilakis, G. Lepouras, and C. Halatsis. Creating an ontology for the user profile: Method and applications. In *Research Challenges in Information Science (RCIS 2007)*, page 407–412, 2007.

[31] Nicola Guarino and Christopher Welty. Evaluating ontological decisions with ontoclean. *Communications of the ACM*, 45(2):61–65, 2002.

[32] Cathal Gurrin, Alan F. Smeaton, and Aiden R. Doherty. Lifelogging: Personal big data. *Found. Trends Inf. Retr.*, 8(1):1–125, June 2014.

[33] Mark Hartswood, Marina Jirotka, Ronald Chenu-Abente, Alethia Hume, and Fausto Giunchiglia. Privacy for peer profiling in collective adaptive systems. *Privacy and Identity Management for the Future Internet in the Age of Globalisation*, pages 237–252, 2015.

[34] Terrence Hoffmann. The meanings of competency. *Journal of European Industrial Training*, 1999.

[35] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs, 2020.

Co-funded by the Horizon 2020 Framework Programme of the European Union

[36] David J. Houghton and Adam N. Joinson. Privacy, social network sites, and social relations. *Journal of Technology in Human Services*, 28(1-2):74–94, 2010.

[37] Zhenhui Jiang, Cheng Suang Heng, and Ben C. F. Choi. Research note —privacy concerns and privacy-protective behavior in synchronous online social interactions. *Information Systems Research*, 24(3):579–595, 2013.

[38] Jure Leskovec and Julian J. Mcauley. Learning to discover social circles in ego networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 539–547. Curran Associates, Inc., 2012.

[39] Sam Levin. Facebook told advertisers it can identify teens feeling 'insecure' and 'worthless'. *The Guardian*, 1 May 2017.

[40] R. Logesh, V. Subramaniyaswamy, and V. Vijayakumar. A personalised travel recommender system utilising social network profile and accurate gps data. *Electronic Government, an International Journal*, 14(1):90–113, 2018.

[41] C. M. Macal and M. J. North. Tutorial on agent-based modelling and simulation. *Journal of Simulation*, 4(3):151–162, Sep 2010.

[42] M.D.L Martinez-Villaseñor, M Gonzalez-Mendoza, and N Hernandez-Gress. Towards a ubiquitous user model for profile sharing and reuse. *Sensors*, 12(10):13249–13283, 2012.

[43] Claudio Masolo, Laure Vieu, Emanuele Bottazzi, Carola Catenacci, Roberta Ferrario, Aldo Gangemi, Nicola Guarino, et al. Social roles and their descriptions. In *KR*, pages 267–277, 2004.

[44] V. Mishra, P. Arya, and M. Dixit. Improving mobile search through location based context and personalization. In *2012 International Conference on Communication Systems and Network Technologies*, pages 392–396, 2012.

[45] Javier Morales, Maite López-Sánchez, and Marc Esteva. Using Experience to Generate New Regulations. In *Proc. of IJCAI '11*, pages 307–312, 2011.

[46] Helen Nissenbaum. Privacy as contextual integrity. *Wash. L. Rev.*, 79:119, 2004.

[47] N Oren, Michael Luck, Simon Miles, and T Norman. *An Argumentation Inspired Heuristic for Resolving Normative Conflict*, pages 0000 – 0000. Unknown Publisher, 2008.

[48] Jeff Z Pan and OWL Working Group. Owl 2 web ontology language document overview: W3c recommendation 27 october 2009, 2009.

[49] M. R. Parks and L. D Roberts. Making moosic: The development of personal relationships online and a comparison to their offline counterparts. *Social and Personal Relationships*, 15:517–537, 1998.

[50] Brea L Perry, Bernice A Pescosolido, and Stephen P Borgatti. *Egocentric network analysis: Foundations, methods, and models*, volume 44. Cambridge University Press, 2018.

[51] Andreas Pfitzmann and Katrin Borcea-Pfitzmann. Lifelong privacy: Privacy and identity management for life. In *IFIP PrimeLife International Summer School on Privacy and Identity Management for Life*, pages 1–17. Springer, 2009.

[52] Robert Plant. Online communities. *Technology in Society*, 26(1):51–65, 2004.

[53] Ognjen Scekic, Daniele Miorandi, Tommaso Schiavinotto, Dimitrios I. Diochnos, Alethia Hume, Ronald Chenu-Abente, Hong-Linh Truong, Michael Rovatsos, Iacopo Carreras, Schahram Dustdar, and Fausto Giunchiglia. Smartsociety – a platform for collaborative people-machine computation. In *SOCA*, 2015.

[54] Laura Schelenz, Karoline Reinhardt, and Daniela Gjuraij. Developing a conceptual and ethical framework of diversity: Deliverable 9.1 for the project wenet- the internet of us, 2019.

[55] Silvia Schiaffino and Analía Amandi. Intelligent user profiling. In Max Bramer, editor, *Artificial Intelligence An International Perspective: An International Perspective*, pages 193–216. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

Co-funded by the Horizon 2020
Framework Programme of the European Union

[56] Qiang Shen, Stefano Teso, Wanyi Zhang, Hao Xu, and Fausto Giunchiglia. Multi-modal subjective context modelling andrecognition. In *2020 International Workshop Modelling and Reasoning in Context (ECAI Workshops)*, 2020.

[57] Slim Trabelsi, Jakub Sendor, and Stefanie Reinicke. Ppl: Primelife privacy policy engine. In *2011 IEEE International Symposium on Policies for Distributed Systems and Networks*, pages 184–185. IEEE, 2011.

[58] Zeynep Tufekci. Facebook's surveillance machine. *The New York Times*, 2018.

[59] Harko Verhagen, Martin Neumann, and Munindar P. Singh. Normative multiagent systems: Foundations and history. In Amit Chopra, Leendert van der Torre, Harko Verhagen, and SerenaEditors Villata, editors, *Handbook of Normative Multiagent Systems*, page 3–25. College Publications, 2018.

[60] Stanley Wasserman, Katherine Faust, et al. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[61] Wanhong Xu, Xi Zhou, and Lei Li. Inferring privacy information via social relations. In *IEEE International Conference on Data Engineering Workshop*, 2008.

[62] Mattia Zeni, Enrico Bignotti, and Fausto Giunchiglia. Combining crowdsourcing and crowdsensing to infer the spatial context. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 28–33. IEEE, 2018.

[63] Mattia Zeni, Ilya Zaihrayeu, and Fausto Giunchiglia. Multi-device activity logging. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 299–302. ACM, 2014.

[64] Mattia Zeni, Wanyi Zhang, Enrico Bignotti, Andrea Passerini, and Fausto Giunchiglia. Fixing mislabeling by human annotators leveraging conflict resolution and prior knowledge. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(1):32, 2019.

Co-funded by the Horizon 2020
Framework Programme of the European Union

Chapter 3

Detecting Norm Violation in Online Communities[‡]

This chapter addresses the issue that arises when community members with diverse views and beliefs interpret norms in different ways (or understand norms differently), possibly leading to unexpected behaviour in interactions, usually with norm violations that affect the individual and community experiences. To address this issue, our ongoing research work proposes a framework capable of detecting norm violations and providing the violator with information about the features of their action that makes this action violate a norm. This helps people understand norms better, and hence, align their diverse understanding of a norm (norm alignment). We build our framework using Machine Learning, with Logistic Model Trees as the classification algorithm. Since norm violations can be highly contextual, we train our model using data from the Wikipedia online community, namely data on Wikipedia edits. We choose to work with Wikipedia data simply because WeNet use case data were not available yet at the time of conducting this research. As such, our work is then evaluated with the Wikipedia use case where we focus on the norm that prohibits vandalism in Wikipedia edits.

3.1 Introduction

The aligned understanding of a norm is an essential process for the interaction between different agents (human or artificial) in normative systems. Mainly because these systems take into consideration norms as the basis to specify and regulate the relevant behaviour of the interacting agents [9]. This is especially important when we consider online communities in which different people with diverse profiles are easily connected with each other. In these cases, misunderstandings about the community norms may lead to interactions being unsuccessful. Take for example a norm that formally specifies that vandalism actions are not permitted. People might not agree on what vandalism actions are. Thus, the goals of this research are: 1) to investigate the challenges associated with detecting when a norm is being violated by a certain member, usually due to a misunderstanding of the norm; and 2) to inform this member about the features of their action that triggered the violation, allowing the member to change their action to be in accordance with the understanding of the community, thus helping the interactions to keep running smoothly. In summary, the proposed mechanisms aims at aligning the understanding of a norm for interacting community members with diverse views and beliefs. To tackle these goals, our main contribution is to provide a framework capable of detecting norm violation and informing the violator of why their action triggered a violation detection.

The proposed framework is using data, that belongs to a specific community, to train a Machine Learning (ML) model that can detect norm violation. We chose this approach based on studies showing that the definition of what is norm violation can be highly contextual, thus it is necessary to consider what a certain community defines as norm violation or expected behavior [2, 4, 16]. Note that while norms are symbolically specified, the meaning of a norm is not always clear, such as what is vandalism or what is hate speech. This is why logic is not enough to detect violation in these cases and why ML approaches are needed to learn the meaning of such norms and detect their violation.

[‡]This chapter has been accepted and presented at the International Workshop on Coordination, Organizations, Institutions, Norms and Ethics for Governance of Multi-Agent Systems (COINE), co-located with AAMAS 2021.

Co-funded by the Horizon 2020 Framework Programme of the European Union

To investigate norm violations, this work is specifically interested in norms that govern online interactions, and we use the Wikipedia community as a testbed, focusing on the article editing actions. This area of research is not only important due to the high volume of interactions that happen on Wikipedia, but also for the proper inclusion and treatment of diverse people in these online interactions. For instance, studies show that, when a system fails to detect norm violations (e.g., hate speech or gender, sexual and racial discrimination), the interactions are damaged, thus impacting the way people interact in the community [8, 11].

Previous works have dealt with norms and normative systems, proposing mechanisms for norm conflict detection [1], norm synthesis [12], norm violation on Wikipedia [17, 3] and other online communities, such as Stack Overflow [5] and Reddit [4]. However, our approach differs mainly in three points: 1) implementing an ML model that allows for the interpretation of the reasons leading to detecting norm violation; 2) incorporating a taxonomy to better explain to the violator which features of their actions triggered the norm violation, based on the results provided by our ML model; and 3) codifying actions in order to represent them through a set of features acquired from previous knowledge about the domain, which is necessary for the above two points. Concerning the last point, we note that our framework does not consider the action as is, but a representation of that action in terms of features and the relation of those features to norm violation (as learned by the applied ML model). For the Wikipedia case, we represent the action of editing articles based on the categorization introduced in [17], with features such as: the measure of profane words, the measure of pronouns, and the measure of Wiki syntax/markup (the details of these are later presented in Section 3.5.2).

To build our proposed framework, this work investigates the combination of two main algorithms: 1) the Logistic Model Tree, the algorithm responsible for classifying an article edit as a violation or not; and 2) the K-Means, the clustering algorithm responsible for grouping the features that are most relevant for detecting a violation. The information about the relevant features is then used to navigate the taxonomy and get a simplified taxonomy of these relevant features.

Our experiments describe how the ML model was built based on the training data provided by Wikipedia, and the results of applying this model to the task of vandalism detection in Wikipedia's article edits illustrate how our approach can reach a precision of 78,1% and a recall of 63,8%. Besides, the results also show that our framework can provide information about the specific group of features that affect the probability of an action being considered a violation, and we make use of this information to provide feedback to the user on their actions.

The remainder of this chapter is divided as follows. We first open with a note on the relation of diversity to this work in Section 3.2. Section 3.3 then presents the basic mechanisms used by our proposed framework. Section 3.4 describes our framework, while Section 3.5 presents its application to the Wikipedia edits use case, and Section 3.6 presents our experiment and its results. The related literature is presented in Section 3.7. We then describe our conclusions and future work in Section 3.8.

3.2 A Note on Diversity

As illustrated in Chapter 2, people's profiles in WeNet are composed of norms that govern one's behaviour as well as attributes that describe features of the person in question. These attributes are divided into a number of categories (see Table 1.1): socio-demographic attributes (e.g. gender, nationality, education, etc.), attributes on personal context (e.g. relevant locations, like home, work, current location, etc.), attributes on social context (e.g. friendships), attributes on competences and interests (e.g. cooking, sports, music, etc.), and attributes about beliefs and values (e.g. personality, psychosocial profile, etc.). In this chapter, the focus is on the diversity of the interpretation of certain norms: how different people understand norms in different ways. For example, what is considered vandalism may change from one person to another, or even one community to another (e.g. graffiti); what is considered hate speech may also change from one person to another; and what is considered nudity that must be banned in a community may as well change with people, context, and communities (e.g. images of women breastfeeding, images of nudity in art, etc.).

In a way, the understanding of norms that ban vandalism, hate speech, nudity or other concepts define a person's beliefs and/or values. In some cases, there might be a relation between socio-demographic attributes (e.g. nationality, education, etc.) and these beliefs and values, although studying such relations is outside the scope of this work. This research focuses solely on how a diverse understanding of norms may lead to norm violations that affect the community's interaction experience.

In the work presented in this chapter, we focus on detecting when a norm violation happens, and we inform the user what features of their action led to this violation, as an approach to align people's understanding of a norm and help improve their actions. However, the work presented here is intended as a stepping stone for adapting the meaning of norms to the view of the community. We strongly believe that communities and their members evolve, and what may be considered a norm violation today might not be so in the future. For example, imagine a norm that states that nudity is not allowed. Agreeing on the features of nudity, for example, may change from one group of people to another and may also change over time. We argue that human communities do not always have one clear definition of concepts like nudity, hate speech, vandalism, freedom of speech, etc. The framework, as such, must have a mechanism to *adapt* to the diverse views of the members of its community, which may change over time. Our next deliverable, D5.3; will report on this ongoing work in further detail.

3.3 Background

This section aims to present the base concepts upon which this work is built. We first start with the description of the taxonomy which we intend to use to formalise a community's knowledge about the features of the actions. Next, we describe the ML algorithms applied to build our framework. First, the Logistic Model Tree (LMT) algorithm, which is used to build the model responsible for detecting possible vandalism; and second, the K-Means algorithm, responsible for grouping the features of the action that are most relevant for detecting violation.

3.3.1 Taxonomy for Action Representation

In the context of our work, an action (executed by a user in an online community) is represented by a set of features. Each of these features describes one aspect of the action being executed, i.e., the composing parts of the action. The goal of adopting this approach is to equip our system with an adaptive aspect, since by modelling an action as a set of features allows the system to deal with different kinds of actions (in different domains). For example, we could map the action of participating in an online meeting by features, such as: amount of time present in the meeting; volume of message exchange; and rate of interaction with other participants. Besides, in the context of norm violation, the proposed approach can use these features to explain which aspects of an action were problematic.

Defining an action through its features gives information about different aspects of the action that might have triggered a violation. However, it is still necessary to find a way to present this information to the violators. The idea is that this information must be provided in a human-readable way, allowing the users to understand what that feature means and how different features are related to each other. With these requirements in mind, we propose the use of a taxonomy to present this data. This classification scheme provides relevant information about concepts of a complex domain in a structured way [7], thus handling the requirements of our solution.

We note that, in this work, the focus is not on *building* a taxonomy of features. Instead, we assume that the taxonomy is provided with their associated norms. Our system uses this taxonomy, navigating it to select the relevant features. The violator is then informed about the features (presented as a subsection of the larger taxonomy) that triggered the violation detected by our model.

3.3.2 Logistic Model Tree

With respect to the domain of detecting norm violations in online communities, interpreting the ML model is an important aspect to consider. Thus, if a community is interested in providing the violator with information about the features of their action that are indicative of violation, then the proposed solution needs to work with a model that can correctly identify these problematic features.

In this work, we are interested in supervised learning, which is the ML task of finding a map between the input and the output. Several algorithms exist that implement the concepts of supervised learning, e.g., artificial neural networks and tree induction. We are most concerned with the ability of these algorithms to generate interpretable outputs, i.e., how the model explains the reasons for taking a certain decision. As such, the algorithm we chose that contains this characteristic is the tree induction algorithm.

The ability to interpret the tree induction model is provided by the way a path is defined in this technique (basically a set of *if-then* statements), which allows our model to find patterns in the data, present the path followed by the model and consequently provide the reasons that lead to that conclusion.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Although induction trees have been a popular approach to solve classification problems, this algorithm also presents some disadvantages. This has prompted Landwehr et al. [10] to propose the Logistic Model Tree (LMT) algorithm, which adds logistic regression functions at the leaves of the tree.

In logistic regression, there are two types of variables: the independent and the dependent variables. The goal is to find a model able to describe the effects of the independent variables on the dependent ones. In our context, the output of the model is responsible for predicting the probability of an action being classified as norm violation.

Dealing with odds is an interesting aspect present in logistic regression, since the increase in a certain variable indicates how the odds changes for the classification output, in this case the odds indicate the effect of the independent variables on the dependent ones. Besides, another important aspect is the equivalence of the natural log of the odds ratio and the linear function of the independent variables, represented by equation 3.1:

$$ln(\frac{p}{1-p}) \quad \leftarrow \quad \beta_0 + \beta_1 x_1 \tag{3.1}$$

where ln is the logarithm of the odds ratio, p [0,1] is the probability of an event occurring. β represents the parameters of the model, in our case the weights for features of the action. After calculating the natural logarithm, we can then use the inverse of the function to get our estimated regression equation:

$$\hat{p} \quad \leftarrow \frac{\epsilon^{\beta_0+\beta_1 x_1}}{1+\epsilon^{\beta_0+\beta_1 x_1}} \tag{3.2}$$

where $\hat{p}$ is the probability estimated by the regression model.

With these characteristics of logistic regression, we can see how this technique can be used to highlight attributes (independent variables) that have relevant influence over the output of the classifier probability.

Landwehr et al. [10] demonstrate how neither of the two algorithms described above (Tree Induction and Logistic Regression) is better than the other. Thus, to tackle the issues present in these two algorithms, LMT adds to the leaves of the tree a logistic regression function.

Figure 3.1 presents the description of a tree generated by the LMT algorithm. With a similar process as the standard decision tree, the LMT algorithm obtains a probability estimation as follow: first, the feature is compared to the value associated with that node. This step is repeated until the algorithm has reached a leaf node, when the exploration is completed. Then the logistic regression function determines the probabilities for the class, as described by equation 3.2.

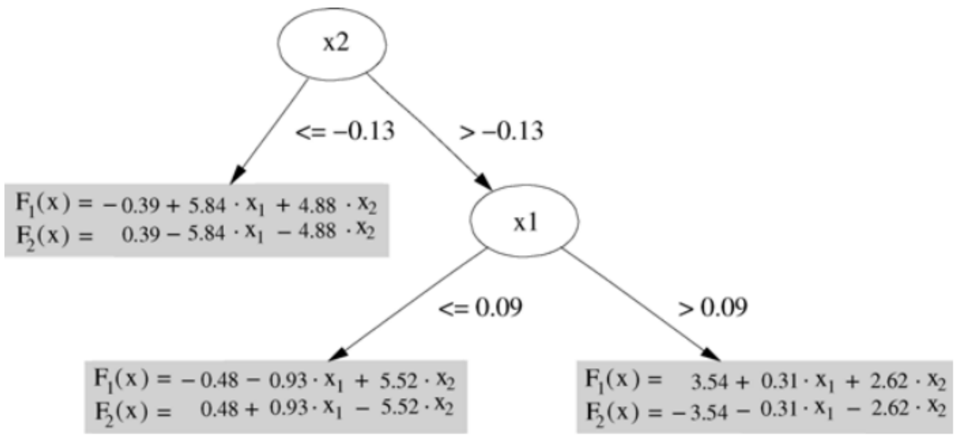
x2
<= −0.13
> −0.13
F1(x) = −0.39 + 5.84 · x1 + 4.88 · x2
F2(x) = 0.39 − 5.84 · x1 − 4.88 · x2
x1
<= 0.09
> 0.09
F1(x) = −0.48 − 0.93 · x1 + 5.52 · x2
F2(x) = 0.48 + 0.93 · x1 − 5.52 · x2
F1(x) = 3.54 + 0.31 · x1 + 2.62 · x2
F2(x) = −3.54 − 0.31 · x1 − 2.62 · x2

Figure 3.1: An example of a tree built by the LMT algorithm[10]. X_1 and X_2 are features present in the dataset. F_1 and F_2 are the equations found by the logistic regression model, describing the weights for each feature present in the training dataset.

3.3.3 K-Means Clustering Method

K-Means is a clustering algorithm with the goal of finding a number K of clusters in the observed data, attempting to group the most 'similar' data points together. This algorithm has been used successfully

Co-funded by the Horizon 2020 Framework Programme of the European Union

in different applications, such as feature learning [15] and computer vision [20]. To achieve this goal, K-Means clusters the data using the nearest mean to the cluster center (calculating the squared Euclidean distance), thus reducing the variance within the group [14].

In this work, the K-Means algorithm can be used to group the features that may indicate an action as violation (we use the features' weights multiplied by their input values as indication of relevance for the classification probability). First, after detecting a possible violation, the ML model provides the K-Means algorithm with the set of features present in the logistic regression and their associated values (the input multiplied by the weight). Then, based on the values of these features, the algorithm is responsible for separating the features of an entry, marked as violation, in two groups (using the centroid of each cluster): 1) those that our model found with highest values, i.g., the most relevant for the vandalism classification; and 2) those with the lowest values, e.g., less relevant for the vandalism classification. Lastly, the output of K-Means informs the framework which are the most relevant features for detecting violations (i.e. the first group), which the framework can then use to navigate the taxonomy and present a selected simplified taxonomy of relevant features to the violator.

3.4 Framework for Norm Violation Detection (FNVD)

This section presents the main contribution of our work on norm alignment for people with diverse profiles: the framework for norm violation detection (FNVD). The goal of this framework is to be deployed in a normative system so that when a violation is detected the system can: 1) enforce the norms by, say, prohibiting the action; and 2) inform the user of what caused this violation allowing him to align his understanding of the norm to that of the community.

The main component of our framework is the machine learning (ML) algorithm behind the detection of norm violations, specifically the LMT algorithm of Section 3.3.2. An important aspect to take into consideration, when using this algorithm, is the data needed to train the model. In our work, the community must provide the definitions of norm violations through a dataset that exemplifies actions that were previously labeled as norm violations. Thus, here we are using data provided by Wikipedia, gathered using Mechanical Turk [13].

After defining the data source, our proposed approach essentially 1) collects the data used to train the LMT model; 2) trains the LMT model to detect possible violations and to learn the action's features relevant to norm violations; and 3) when violations are detected, according to the LMT model's results, then the action responsible for the violation is rejected and the violator is informed about the features of their action that triggered the model output. Furthermore, in both cases (when actions are labelled as violating norms or not), we suggest that the framework collects feedback from the members of the community, which can then be used as new data to retrain the ML model. This is important as we strongly believe that communities and their members evolve, and what may be considered a norm violation today might not be in the future. For example, imagine a norm that states that hate speech is not allowed. Agreeing on the features of hate speech may change from one group of people to another and may also change over time. Clearly, human communities do not always have one clear definition of concepts like hate speech, violation, freedom of speech, etc. The framework, as such, must have a mechanism to adapt to the views of the members of its community, as well as adapt to the views that may change over time. While we leave the adaptation part for future work, we highlight its need in this section, and prepare the framework to deal with such adaptions, as we illustrate in Figure 3.2.

To further clarify how our framework would act to detect a norm violation when deployed in a community, it is essential to explore the diagram in Figure 3.2. Step 0 represents the training process of the LMT model, which is a fundamental part of our approach because it is in this moment that the rules for norm violation are specified. Basically, after training the model, our framework would have identified a set of rules that describe norm violation. We can portray these rules as a conjunction of two elements: 1) the tree that is built by the LMT algorithm on top of the collected data; and 2) the weights presented in the leaves of the tree. These weights are the parameters of the estimated regression equation that defines the probability of norm violation (depicted in Equation 3.2). With the trained LMT model, the system starts monitoring every new action performed in the community (Step 1). In Step 2, the system maps the action to features that the community defined as descriptive of that action, which triggers the LMT model to start working to detect if that action is violating (or not) any of the norms. Step 3 presents the two different paths that can be executed by our system. If the action is detected as violating a norm (Condition 1), then we argue that the system must execute a sequence of steps to guarantee that the community norms are not violated: 1) the system does not allow the action to persist (i.e., action is not executed); 2) the system presents to the user information about which action features were the most rel-

Co-funded by the Horizon 2020 Framework Programme of the European Union

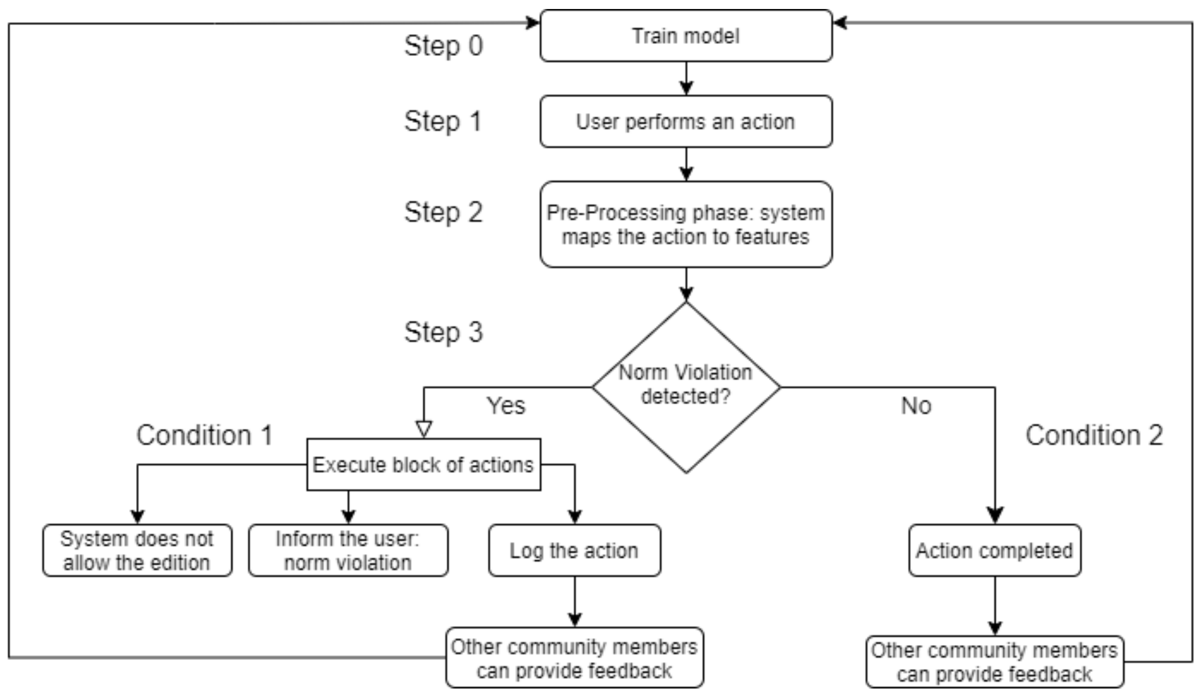
Step 0
Train model
Step 1
User performs an action
Step 2
Pre-Processing phase: system maps the action to features
Step 3
Norm Violation detected?
Yes
No
Condition 1
Condition 2
Execute block of actions
System does not allow the edition
Inform the user: norm violation
Log the action
Action completed
Other community members can provide feedback
Other community members can provide feedback

Figure 3.2: How the framework works when deployed in an online community.

evant for our model to detect the norm violation, and the taxonomy of the relevant features is presented; 3) the action is logged by the system, allowing other community members to give feedback about the edit attempt, thus providing the possibility of these members flagging the action as a non-violation. The feedback collected from the users can later be used to continuously train (Step 0) our LMT model (future work). However, if the executed action is not detected as violating a norm (Condition 2), then the system can proceed as follows. The action persists in the system (i.e., action is executed), and since any model may incorrectly classify some norm violation as non-violation, the system allows the members of the community to give feedback about that action, providing the possibility of flagging an already accepted action as a violation. Getting people's feedback on violations that go unnoticed by the model is a way to allow the system to adapt to new data (people's feedback) and update the definitions of norm violations by continuously training the LMT model (Step 0).

To obtain the relevant features for the norm violation classification (Condition 1), we use the K-Means algorithm. In our context, due to the estimated logistic regression equation, the LMT model provides the weights for each feature multiplied by the value of these features for the action. This indicates the influence of the features on the model's output (i.e., the probability of an action being classified as norm violation). With the weights and specific values for the features, the K-Means algorithm can group the set of features that present the highest multiplied values, which are the ones we assume that contribute the most for the probability of norm violation. Then, by searching the taxonomy using the group of relevant features, our system can provide the taxonomy structure of the features that trigger norm violation, this is useful due to the explanatory and interpretation characteristics of a taxonomy. The aim of providing this information is to clarify to the member of the community performing the action, what are the problematic aspects of their action as learned by our model.

3.5 Use Case

We focus on the problem of detecting vandalism in Wikipedia article edits as a way to help develop this foundational research work on norm alignment. This use case is interesting because Wikipedia is an online community where norms such as 'no vandalism' may have different interpretations by different people (diverse views and beliefs). In what follows, we first present the use case's domain, followed by the taxonomy used by our system, and finally, an illustration of how our proposed framework may be applied to this use case.

Co-funded by the Horizon 2020 Framework Programme of the European Union

3.5.1 Domain

Wikipedia [18] is an online encyclopedia, containing articles about different subjects in any area of knowledge. It is free to read and edit, thus any person with a device connected to the internet can access it and edit its articles. Due to the openness and collaborative structure of Wikipedia, the system is subject to different interpretations of what is the community's expectation concerning how content should be edited. To help address this issue, Wikipedia has compiled a set of rules, the Wikipedia norms [13], to maintain and organize its content.

Since we are looking for an automated solution for detecting norm violations by applying machine learning mechanisms, the availability of data becomes crucial. Wikipedia provides data on what edits are marked as vandalism, where vandalism annotations were provided by Amazon's Mechanical Turk. Basically, every article edit was evaluated by at least three people (Mechanical Turks) that decided whether the edit violates the 'no vandalism' norm or not. In the context of our work, the actions performed by the members of the community are the Wikipedia users' attempts to edit articles, and the norm is *"Do not engage in vandalism behavior"* (which we refer to as the 'no vandalism' norm). It is this precise dataset that we have used to train the model that detects norm violations. We present an example of what is considered a vandalism in a Wikipedia article edit, where a user edited an article by adding the following text: *"Bugger all in the subject of health ect."*

3.5.2 Taxonomy Associated with the 'No Vandalism' Norm

An important step in our work is to map actions to features and then specify how they are linked to each other. We manually created a taxonomy to describe these features by separating them in categories that describe their relation with the action.[2] In this work, we consider the 58 features described in [17] and 3 more that were available in the provided dataset: LANG_MARKUP_IMPACT, the measure of the addition/removal of Wiki syntax/markup; LANG_EN_PROFANE_BIG and LANG_EN_PROFANE_BIG_IMPACT, the measure of addition/removal of English profane words. In the dataset, features ending with _IMPACT are normalized by the difference of the article size after edition. The main objective of this taxonomy is to help our system present to the violator an easy-to-read explanation of the reasons why their article edit was marked as violating a norm by our model, specifying the features with highest influence to trigger this violation.

To further explain our taxonomy approach, we present in Figure 3.3 the constructed taxonomy for Wikipedia's 'no vandalism' norm. We observe that features can be divided in four main groups. The first is user's direct actions, which represent aspects of the user's article editing action, e.g., adding a text. This group is further divided in four sub-groups: a) written edition, which contains features about the text itself that is being edited by the user; b) comment on the edition, which contains features about the comments that users have left on that edition; c) article after edition, which contains features about how the edited article changed after the edition was completed; and d) time of edition, which contains features about the time when the user made their edition. The second group is the user's profile, general information about the user. The third is the page's history, how the article changed with past editions. The last group is reversions, which is essentially information on past reversions.[3] In total, these groups have 61 features, but due to simplification purpose, Table 3.1 only presents a subset of those features.

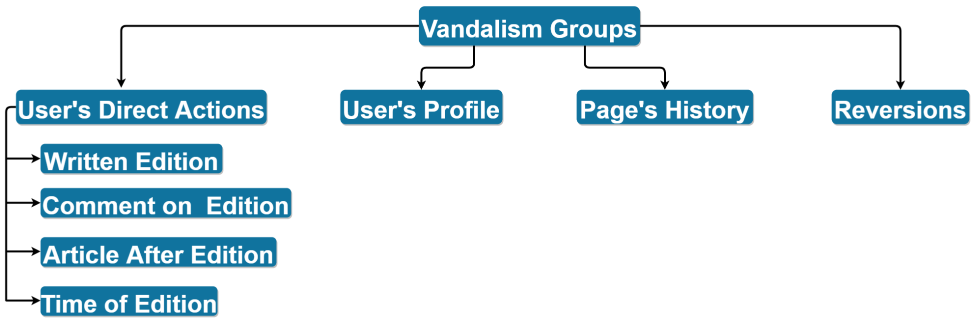

Figure 3.3: Taxonomy associated with Wikipedia's 'no vandalism' norm.

[2]For the complete taxonomy, the reader can refer to `https://bit.ly/3sQFhQz`
[3]A reversion is when an article is reverted back to a version before the vandalism occurred.

Table 3.1: Example of Features present in the taxonomy groups.

Group	Features
Written Edition	LANG_ALL_ALPHA; LANG_EN_PRONOUN
Comment on Edition	COMM_LEN; COMM_LEN_NO_SECT
Article After Edition	SIZE_CHANGE_RESULT; SIZE_CHANGE_CHARS
Time of Edition	TIME_TOD; TIME_DOW
User's Profile	HIST_REP_COUNTRY; USER_EDITS_DENSITY
Page's History	PAGE_AGE; WT_NO_DELAY
Reversions	HASH_REVERTED; HASH_IP_REVERT

3.5.3 FNVD Applied to Vandalism Detection Online

It this section, we first describe an example of how our framework can be configured to be deployed in the Wikipedia community. First, the community provides the features and the taxonomy describing that feature space (see Figure 3.3). Then, our framework trains the LMT model to classify norm violations based on the data provided (Step 0 of Figure 3.2), which must contain examples of what that community understands as norm violation and regular behaviour.

In the context of vandalism detection on Wikipedia, the relevant actions performed by the members of the community are the attempts to edit Wikipedia articles. Following the diagram in Figure 3.2, when a user attempts to edit an article (Step 1), our system will analyse this edit. We note here that our proposed LMT model does not work with the action itself, but the features that describe it. As such, it is necessary to first find the features that represent the performed action. Thus, in Step 2, there is a pre-processing phase responsible for mapping actions to the features associated to the norm in question. For example, an article about Asteroid was edited with the addition of the text *"i like man!!"*. After getting this edition text, the system can compute the values (as described in [17]) for the 61 features, which are used to calculate the vandalism probability. For brevity reasons, we only show the values for some of these features:

1. LANG_ALL_ALPHA, the percentage of text which is alphabetic: 0,615385;
2. WT_NO_DELAY, the calculated WikiTrust score: 0,731452;
3. HIST_REP_COUNTRY, measure of how users from the same country as the editor behaved: 0,155146.

After calculating the values for all features, the LMT model can evaluate if this article edit is considered 'vandalism' or not. In the case of detecting vandalism (Condition 1 of Figure 3.2), the system does not allow the edition to be recorded on the Wikipedia article, and it presents to the violator two inputs. The first is the set of features of their edit that have the highest influence on the model's decision to detect the vandalism. To get this set, after calculating the probability of vandalism (as depicted in Equation 3.2), the LMT model provides the features that present a positive relationship with the output. These 'positive features' are then used by K-Means to create the group with the most relevant ones (Table 3.2 presents an example of this process). The second input is the selected part of the taxonomy related to chosen set of features, providing further explanation of those features that triggered the norm violation. Additionally, the system will log the attempt to edit the article, which eventually may trigger feedback collection that can at a later stage be used to retrain our model.

Table 3.2: List of features that positively affects the probability of vandalism detection. Total Value is the multiplication between the feature's values and the features' weights. The most relevant features, as found by K-Means, are marked with an (*).

Features	Total Value
WT_NO_DELAY*	1.08254896
HIST_REP_COUNTRY*	0.899847
LANG_ALL_ALPHA*	0.7261543
HASH_REC_DIVERSITY	0.15714292
WT_DELAYED	0.12748878
LANG_ALL_CHAR_REP	0.12
HIST_REP_ARTICLE	0.093548

The features WT_NO_DELAY, $HIST\_REP\_COUNTRY$ and $LANG\_ALL\_ALPHA$ were indicated by K-Means as the most relevant for the classification of vandalism. With this information, our

Co-funded by the Horizon 2020 Framework Programme of the European Union

framework can search the taxonomy for the relevant features and then automatically retrieve the simplified taxonomy structure for these three specific features, as shown in Figure 3.4. The user is informed of the problematic features that triggered norm violation, their meaning, and the simplified taxonomy under which they fall.

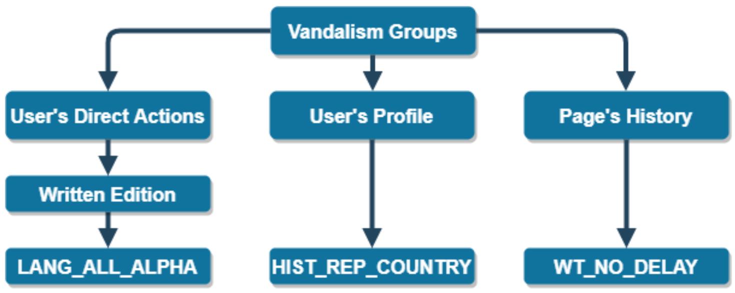

Figure 3.4: Taxonomy for part of the features that were most relevant for the vandalism classification. These features are then presented to the user with a descriptive text.

However, in case the system classifies the article edit as 'non-vandalism' (Condition 2 of Figure 3.2), the Wikipedia article is updated according to the user's article edit and community members may provide feedback on this new article edit, which may later be used to retrain our model (as explained in Section 3.4).

3.6 Experiments and Results

The goal of this section is to describe how the proposed approach was applied for detecting norm violation in the domain of Wikipedia article edits, with an initial attempt to improve the interactions in online communities. Then, we demonstrate and discuss the results achieved.

3.6.1 Experiments

Data on vandalism detection in Wikipedia articles [17] were used for the experiments. This dataset has 61 features and 32,439 instances for training (with 2,394 examples of vandalism editions and 30,045 examples of regular editions). The model was trained with WEKA [19] and evaluated using 10 folds cross-validation.

3.6.2 Results

The first important information to note is how the LMT model performs when classifying vandalism in Wikipedia editions. In Figure 3.5, it is possible to see the model that was built to perform the classification task.[4] The tree has four decision nodes and five leaves in total. Since the LMT model uses logistic regression at the leaves, the model has five different estimated logistic regression equations, each of these equations outputs' the probability of an edition being a vandalism.

The LMT model correctly classifies 96% of instances in general. However, when we separate the results in two groups, vandalism editions and regular editions, it is possible to observe a difference in the model's performance. For the regular editions, the LMT model achieves a precision of 97,2%, and a recall of 98,6%. While for vandalism editions, the performance of the model drops, with a precision of 78,1% and a recall of 63,8%. This decrease can be explained by how the dataset was separated and the number of vandalism instances, which consequently leads to an unbalanced dataset. In the dataset, the total number of vandalism instances is 2,394 and the other 30,045 instances are of regular editions. A better balance between the number of vandalism editions and regular edition should improve our classifier, thus in the future we are exploring other model configurations (e.g., ensemble models) to handle data imbalance.

The influence of each feature on determining the probability of a norm violation is provided by the LMT model (as assumed in this work, feature influence is model specific, meaning that a different model

[4]Trained model available at: `https://bit.ly/3gBBkwP`

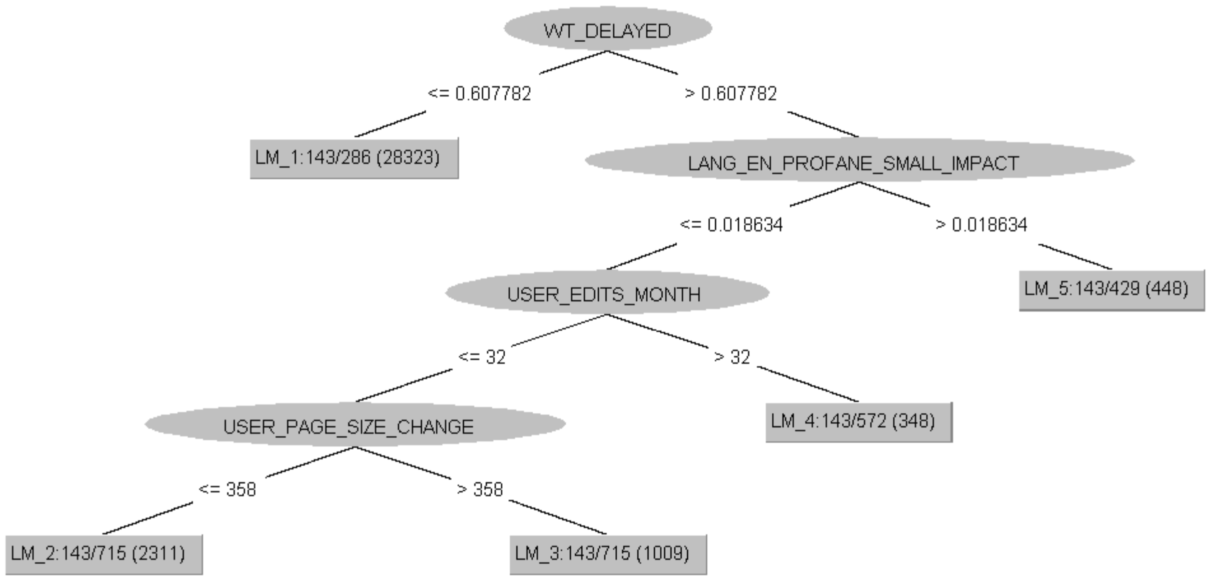

Figure 3.5: The built model for the vandalism detection, using Logistic Model Tree.

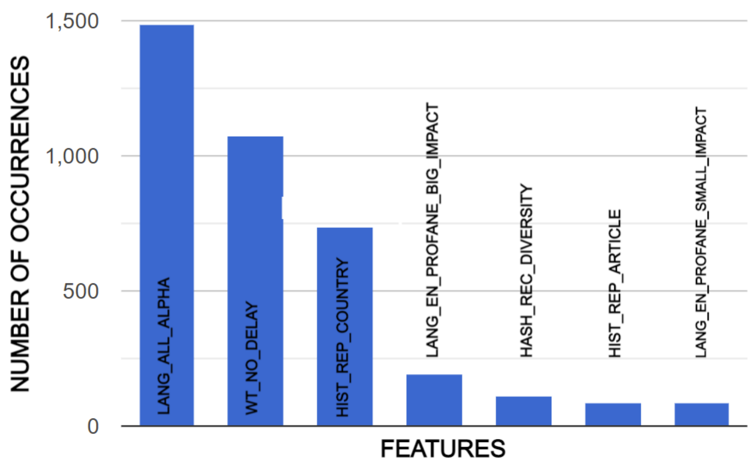

Figure 3.6: Number of occurrences of relevant features in vandalism detection.

can find a different set of relevant features). The graph in Figure 3.6 shows the number of times a feature is classified as relevant by the built model. Some features appear in most of the observations, indicating how important they are to detect vandalism. Future work shall investigate if this same behavior (some features present in the actions have more influence than other features to define the norm violation probability) can be detected in other domains.

"LANG_ALL_ALPHA" (describing the percentage of text which is alphabetic) recurrently appears as relevant when vandalism is detected. This happens because this feature presents, as estimated by the LMT model, a positive relationship with the norm violation, meaning that when a vandalism edition is detected, this feature is usually relevant for the classification.

3.7 Related Work

In this section, we present the most relevant works related to that reported in this chapter. Specifically, we reference the relevant literature that uses ML solutions to learn the meaning of a violation, then use that to detect violations in online communities. In addition to the specific works presented below, it is also worth to mention a survey that studies a variety of research in the area, focusing on norm violation

Co-funded by the Horizon 2020 Framework Programme of the European Union

detection in the domains of hate speech and cyberbullying [2].

Also investigating norm violation in Wikipedia but using the dataset from the comments on talk page edits, Anand and Eswari [3] present a Deep Learning (DL) approach to classify a comment as abusive or not. Although the use of DL is an interesting approach to norm violation detection, we focus on offering interpretability, i.e., providing features our model found as relevant for the detection of norm violation. While the DL model in [3] does not provide such information.

The work by Cherian et al. [5] explores norm violation on the Stack Overflow (SO) community. This violation is studied by analyzing the comments posted on the site, which can contain hate speech and abusive language. The authors state that the SO community could become less toxic by identifying and minimizing this kind of behavior, which they separate in two main groups: generic norms and SO specific norms. There are two important similarities between our works: 1) both studies use labeled dataset from the community, considering the relevant context; and 2) the norm violation detection workflow. The main difference is that we focus on the interpretation of the reasons that indicate a norm violation as detected by our model, providing information to the user so they can decide which specific features they are changing. This is possible because we are mapping the actions into features, while Cheriyan at al. [5] work directly with the text from the comments, which allows them to focus on providing text alternatives to how the user should write their comment.

Chandrasekaran et al. [4] build a system for comment moderation in Reddit, named Crossmod. Crossmod is described as a sociotechnical moderation system designed using participatory methods (interview with Reddit moderators). To detect norm violation, Crossmod uses a ML back-end, formed by an ensemble of classifiers. Since there is an ensemble of classifiers, the ML back-end was trained using the concept of cross-community learning, which uses data from different communities to detect violation in a specific target community. Like our work, Crossmod uses labeled data from the community to train the classifiers and the norm violation detection workflow follows the same pattern. However, different from our approach, Chandrasekaran et al. [4] use textual data directly, not mapping to features. Besides, Crossmod do not provide to the user information on the parts of the action that triggered the violation classifier.

Considering another type of ML algorithm, Di Capua et al. [6] build a solution based on Natural Language Processing (NLP) and Self-Oganizing Map (SOM) to automatically detect bullying behavior on social networks. The authors decided to use an unsupervised learning algorithm because they wanted to avoid the manual work of labeling the data, the assumption is that the dataset is huge and by avoiding manual labelling, they would also avoid imposing a priori bias about the possible classes. This differs from our assumptions since we regard the data/feedback from the community as the basis to deal with norm violation.

One interesting aspect about these studies is that they are either in the realm of hate speech or cyberbullying, which can be understood as a sub-group of norm violation by formalizing hate speech and cyberbullying in terms of norms that a community should adhere to. Researchers are interested in these fields mainly due to the damage that violating these norms can cause in the members of an online community, and due to the available data to study these communities.

3.8 Conclusion and Future Work

The proposed framework, combining machine learning (Logistic Model Trees and K-Means) and taxonomy exploration, is an initial approach on how to detect norm violations and how to explain such a violation to users with diverse views in order to align their views with that of the community. To study norm violation, our work used a dataset from Wikipedia's vandalism edition, which contains data about Wikipedia article edits that were considered vandalism.

The framework described in this work is a first step towards detecting vandalism, and it provides relevant information about the problems (features) of the action that led to vandalism. Further investigation is still needed to get a measure of how our system would improve the interactions in an online community. The experiments conducted in our work show that our ML model has a precision of 78,1% and a recall of 63,8% when classifying data describing vandalism.

Future work is going to focus on the use of feedback from the community members to continuously train our ML model and align the community's view to theirs, as explained in Section 3.4. The idea is to apply an online training approach to our framework, so when a community behaviour changes, that would be taken to indicate a new view on the rules defining the norm, and our ML model should adapt to this new view.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Throughout this investigation, we have noticed that the literature mostly deals with norm violation that focus either on hate speech or cyberbullying. We aim that our approach can be applied to other domains, thus we are planning to explore domains with different actions to analyse how our framework deals with a different context (since these domains would have a different set of actions to be executed in an online community).

Co-funded by the Horizon 2020
Framework Programme of the European Union

Bibliography

[1] J. P. Aires, J. Monteiro, R. Granada, and F. Meneguzzi. Norm Conflict Identification using Vector Space Offsets. In *IJCNN*, pages 1–8, 2018.

[2] Areej Al-Hassan and Hmood Al-Dossari. Detection of Hate Speech in Social Networks: a Survey on Multilingual Corpus. In *6th International Conference on Computer Science and Information Technology*, pages 83–100, 2019.

[3] M. Anand and R. Eswari. Classification of Abusive Comments in Social Media using Deep Learning. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pages 974–977, 2019.

[4] Eshwar Chandrasekharan, Chaitrali Gandhi, Matthew Wortley Mustelier, and Eric Gilbert. Crossmod: A Cross-Community Learning-Based System to Assist Reddit Moderators. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW), November 2019.

[5] Jithin Cheriyan, Bastin Tony Roy Savarimuthu, and Stephen Cranefield. Norm Violation in Online Communities–A Study of Stack Overflow comments. *arXiv preprint arXiv:2004.05589*, 2020.

[6] M. Di Capua, E. Di Nardo, and A. Petrosino. Unsupervised Cyber Bullying Detection in Social Networks. In *ICPR*, pages 432–437, 2016.

[7] Kirk Dean Fiedler, Varun Grover, and James T.C. Teng. An Empirically Derived Taxonomy of Information Technology Structure and Its Relationship to Organizational Structure. *Journal of Management Information Systems*, 13(1):9–34, 1996.

[8] Kishonna L. Gray. Gaming out online: Black Lesbian Identity Development and Community Building in Xbox Live. *Journal of Lesbian Studies*, 22(3):282–296, 2018.

[9] Andrew JI Jones, Marek Sergot, et al. On the Characterisation of Law and Computer Systems: The Normative Systems Perspective. *Deontic logic in computer science: normative system specification*, pages 275–307, 1993.

[10] Niels Landwehr, Mark Hall, and Eibe Frank. Logistic Model Trees. *Machine learning*, 59(1-2):161–205, 2005.

[11] Lavinia McLean and Mark D Griffiths. Female Gamers' Experience of Online Harassment and Social Support in Online Gaming: A Qualitative Study. *International journal of mental health and addiction*, 17(4):970–994, 2019.

[12] Javier Morales, Michael Wooldridge, Juan A Rodríguez-Aguilar, and Maite López-Sánchez. Off-Line Synthesis of Evolutionarily Stable Normative Systems. *Autonomous agents and multi-agent systems*, 32(5):635–671, 2018.

[13] Martin Potthast and T. Holfeld. Overview of the 1st International Competition on Wikipedia Vandalism Detection. In *CLEF*, 2010.

[14] Pradeep Rai and Shubha Singh. A Survey of Clustering Techniques. *International Journal of Computer Applications*, 7(12):1–5, 2010.

[15] S. Y. Rubaiat, M. M. Rahman, and M. K. Hasan. Important Feature Selection Accuracy Comparisons of Different Machine Learning Models for Early Diabetes Detection. In *International Conference on Innovation in Engineering and Technology*, pages 1–6, 2018.

Co-funded by the Horizon 2020
Framework Programme of the European Union

[16] Leendert van der Torre. Contextual Deontic Logic: Normative Agents, Violations and Independence. *Annals of mathematics and artificial intelligence*, 37(1-2):33–63, 2003.

[17] Andrew G West and Insup Lee. Multilingual vandalism detection using language-independent & ex post facto evidence. In *CLEF Notebooks*, 2011.

[18] Wikipedia contributors. Wikipedia — Wikipedia, the free encyclopedia, 2021. [Online; accessed 22-Feb-2021].

[19] Ian H. Witten, Eibe Frank, and Mark A. Hall. The WEKA Workbench. In *Data Mining: Practical Machine Learning Tools and Techniques*, pages 403–406. Morgan Kaufmann, 2011.

[20] Xin Zheng, Qinyi Lei, Run Yao, Yifei Gong, and Qian Yin. Image Segmentation Based on Adaptive K-Means Algorithm. *EURASIP J Image Video Process*, 2018(1):1–10, 2018.

Co-funded by the Horizon 2020
Framework Programme of the European Union