UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

An Argumentation System for Assisting Users with Privacy Management in Online Social Networks

MASTER'S THESIS

Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

Author: Ramon Ruiz-Dolz

Tutors: Ana Maria Garcia Fornes
Stella María Heras Barberá

Experimental Director: Jose Alemany Bordera

Course 2018-2019

M'agradaría dedicar aquest treball i agrair la seua implicació en ell...

als meus pares, per facilitar-me tots els mitjans al seu abast i l'educació necessaria,

a Stella, per descobrir-me noves i interesantíssimes arees de recerca i confiar tant en mi,

a Ana, per guiar-me i donar-me l'orientació necessaria en un moment tan important,

a Jose, per prestar tota l'ajuda possible i per estar sempre quan ha sigut necessari,

i al GTI-IA, per acollir-me tan prompte i permetre'm portar el meu treball a un nivell superior.

Moltes gràcies a totes i tots.

Resum

L'ús de les xarxes socials empra gran part del temps d'oci de les persones en aquestos darrers anys. Les xarxes socials, normalment de tipus gratuït, proporcionen els seus usuaris un entorn on interactuar, xatejar i compartit tot tipus d'informació amb la resta d'usuaris. Habitualment, les xarxes socials proporcionen una serie de mecanismes de control de la privadesa del propi usuari. Aquestos mecanismes solen estar ubicats als ajustos o a la configuració del perfil. No obstant això, gran part dels usuaris desconeix l'existència d'aquestos mecanismes o directament els ignora degut a la poca importància que li solem donar a la pròpia privadesa. Aleshores, es de gran importància conscienciar aquestos usuaris sobre la rellevància de la seua privadesa, a la vegada que s'avisa de potencials disputes o violacions de privacitat que podrien ocórrer al compartir un contingut determinat a una xarxa social, abans de ser publicat. En aquest treball es proposa una forma de confrontar aquest problema. Un sistema argumentatiu capaç de raonar a l'autor d'una publicació entorn els motius pels quals no es deuría realitzar la publicació (en cas de detectar qualsevol tipus de conflicte). Aquest sistema, a més de ser desenvolupat i implementat, també serà integrat i utilitzat a una xarxa amb objectius educatius, PESEDIA.

Paraules clau: Argumentació, Persuasió, Xarxes Socials, Privadesa, Seguretat

Resumen

El uso de las redes sociales acapara gran parte del tiempo de ocio de las personas en estos últimos años. Las redes sociales, habitualmente de carácter gratuito, proporcionan a sus usuarios un entorno donde interactuar, chatear y compartir información con los demás usuarios. Normalmente, las redes sociales proporcionan una serie de mecanismos de control de la privacidad del propio usuario. Estos mecanismos suelen estar ubicados en los ajustes o en la configuración del perfil. Sin embargo, gran cantidad de usuarios desconoce la existencia de estos mecanismos o directamente los ignora debido a la poca importancia que se le suele dar a la privacidad. Es de gran importancia por lo tanto, concienciar a estos usuarios sobre la relevancia de su propia privacidad, así como avisar de potenciales disputas o violaciones de privacidad, que podrían suceder al compartir un determinado contenido en una red social, antes de ser publicado. En este trabajo se propone una aproximación para lidiar con este problema. Un sistema argumentativo capaz de razonar al autor de una publicación sobre los motivos por los que no se debería realizar dicha publicación (en caso de detectar algún tipo de conflicto). Este sistema, además de ser desarrollado e implementado, también será integrado y utilizado en una red social de carácter educativo, PESEDIA.

Palabras clave: Argumentación, Persuasión, Redes Sociales, Privacidad, Seguridad

Abstract

The use of social networks consumes most of the leisure time of the people these last years. The social networks, usually free to use, give the users an environment where interact, chat and share information with other users. Usually, the social networks provide the user with some tools that allow to control the own user's privacy. These tools are usually located in the settings or the profile configuration. However, most of the users does not know about them or directly ignores their existence due to the low concern regarding its own privacy. Therefore, it is very important to raise users awareness regarding its own privacy. In order to achieve that, it is interesting to warn the user of potential

disputes or privacy violations that may arise when sharing some specific content, before being shared. This work proposes an approximation to deal with this problem. An argumentation system able to give the reasons of why is not a good idea to share (in the case of detecting any type of violation) some specific content. In addition to the development and implementation, this system will also be integrated and used in an educational social network, PESEDIA

Key words: Argumentation, Persuasion, Social Networks, Privacy, Security

Contents

Contents vii
List of Figures ix
List of Tables ix

1 Introduction 1
1.1 Motivation . . . 1
1.2 Objectives . . . 2
1.3 Document Structure . . . 2
2 Related Work 5
2.1 Argumentation in Artificial Intelligence . . . 5
2.1.1 Principles of Argumentation Theory . . . 5
2.1.2 Abstract Argumentation Systems . . . 7
2.1.3 Structured Arguments . . . 9
2.1.4 Complexity in Argumentation Solving . . . 10
2.2 Privacy Management in Online Social Networks . . . 11
2.2.1 Comparative of Existing Privacy Management Systems . . . 13
3 Requirements Specification 15
3.1 Introduction . . . 15
3.1.1 Purpose . . . 15
3.1.2 Scope . . . 15
3.1.3 Overview . . . 15
3.2 Overall Description . . . 16
3.2.1 Work Perspective . . . 16
3.2.2 System Functions . . . 16
3.2.3 User Characteristics . . . 17
3.2.4 Constraints and Dependencies . . . 17
3.3 Specific Requirements . . . 17
3.3.1 Functions . . . 18
3.3.2 Design Constraints . . . 20
4 Argumentation System 23
4.1 Framework Formalisation . . . 23
4.2 System Architecture . . . 24
4.2.1 Feature Extraction Module . . . 25
4.2.2 Argument Generation Module . . . 27
4.2.3 Solver Module . . . 28
4.2.4 Dialogue Module . . . 28
4.3 User-network Interaction . . . 28
4.3.1 Template Based Arguments . . . 29
4.3.2 User Responses . . . 30
5 Implementation and Integration 33
5.1 Introduction . . . 33
5.2 Technologies . . . 33

5.3 Argumentation System Implementation 34
5.3.1 Feature Extraction 35
5.3.2 Argument Generator & Argumentation Solver 35
5.3.3 Argument Template 35
5.3.4 Persuasive Statistics 36
5.4 Argumentation System Deployment 36
5.5 Network Integration 36
5.5.1 PESEDIA: An Educational OSN 37
5.5.2 Argumentation Plugin 38
5.6 System Operation 39
6 System Evaluation **41**
6.1 Argumentation System Stress Test 41
6.2 Functional Validation 43
6.2.1 Validation of content arguments generation 43
6.2.2 Validation of privacy arguments generation 45
6.2.3 Validation of risk arguments generation 46
6.2.4 Validation of trust arguments generation 46
6.3 Persuasive Evaluation 47
7 Conclusions **49**
8 Future Work **51**
Bibliography **53**

List of Figures

2.1 Example of an Argumentation Diagram 6
2.2 Argument Graph 7

4.1 Architecture of the argumentation system. 25
4.2 Structure of interaction process 29

5.1 Argumentation system structure 34
5.2 PESEDIA architecture diagram 37
5.3 Plugin structure 38
5.4 Diagram of the communication between the network and the argumentation system 39
5.5 Example of post triggering an argument in the PESEDIA network 39
5.6 Pop-up displayed to the user with a content argument of location (*You can be revealing information about where you are or where you're going.*) 40

6.1 Requests per second handled by the argumentation system 42
6.2 Time required to solve N requests at a time 42

List of Tables

2.1 Comparative of different privacy management systems 13

3.1 Function F01 18
3.2 Function F02 18
3.3 Function F03 18
3.4 Function F04 18
3.5 Function F05 18
3.6 Function F06 19
3.7 Function F07 19
3.8 Function F08 19
3.9 Function F09 19
3.10 Function F10 19
3.11 Function F11 20
3.12 Function F12 20
3.13 Design Constraint DC01 20
3.14 Design Constraint DC02 20
3.15 Design Constraint DC03 20
3.16 Design Constraint DC04 21
3.17 Design Constraint DC05 21

3.18 Design Constraint DC06 21
3.19 Design Constraint DC07 21

4.1 Template based arguments generated by our system. 30

6.1 Validation step 1 results (first part) 44
6.2 Validation step 1 results (second part) 45
6.3 Validation step 2 results 45
6.4 Validation step 3 results 46
6.5 Validation step 4 results 47

CHAPTER 1
Introduction

1.1 Motivation

One of the most interesting features related to the use of internet from the social point of view is online interaction. The Online Social Networks (OSNs) populations in Europe[1] and United States[2] have been increasing rapidly during the last years. As OSN users increase, the amount of information published on this sites also increases. Before publishing content in an OSN many factors should be taken into account. For example, a publication may contain sensitive information related to the own author or linked to other users that are involved in the publication. It is also possible that this information may be seen by undesired users. For instance, a classical dilemma regarding the privacy of the users is the multi-party privacy conflict (MPPC). A MPPC may happen in a OSN when any user publishes some content involving other users. A typical instance of this conflict is given when a user publishes an image where different users appear [36]. Some OSNs provide the option of reporting content in case of privacy violation, but that is a *posteriori* solution. Using computational argumentation techniques, as proposed in [19], it may be possible to prevent the violations before being done. To sum up, with the growing usage of OSNs it is important to raise the awareness of their users when publishing content involving or not other parties.

This work presents a new argumentation system with the objective to give users advice on what should be shared and to whom. Most of the OSNs (eg. Facebook, Instagram, Twitter, etc.) do not have any way to consider if the publication of some content can be dangerous for its users privacy before making the publication. Some other OSNs as PESEDIA[10] implement a recommendation system that warns the user before publishing the content. The main problem, as explained in [12], is that current recommendation systems don't provide a strong reasoning of why should the user take a determined action. With an argumentation system, it is possible to generate customised information that gives reasoned explanations to the users. By using an argumentation system, it is also possible to improve the effectiveness of the recommendations when trying to persuade a specific user.

This work is framed in the national project *"Agentes Inteligentes para Asesorar en Privacidad en Redes Sociales"* (TIN2017-89156-R) of I+D+I funded by the *Ministerio de Ciencia, Innovación y Universidades* and proposed by the Computing Technology - Artificial Intelligence (GTI-IA) group, member of the Valencian Research institute for Artificial INtelligence (VRAIN). One of the main objectives of this project is to propose and develop an argumentation framework able to persuade human users with the exchange of argu-

[1] https://www.statista.com/statistics/271430/social-network-penetration-in-the-eu/
[2] https://www.statista.com/study/40227/social-social-media-usage-in-the-united-states-statista-dossier/

ments, but also taking into account users' preferences and values related to their privacy. Therefore, this work can also be considered the first step taken towards achieving this objective.

1.2 Objectives

The main purpose of this work is to develop a persuasive argumentation system able to help users with their privacy management. Therefore, this system must be a complete tool that performs a huge number of varied tasks. In order to be able to reach the purpose of this work we need to define the following objectives.

The first objective consists on acquiring knowledge in computational argumentation, the basis of this whole work. Since it is a domain that has not been learned during the master courses, a deep study of the most important concepts and ideas will be performed. The second pillar of this work is privacy management. An study of the existing tools that assist users with privacy management will also be done. Since the purpose of this work is quite innovative, it will not be possible to use an existing tool. Despite this, some important ideas and concepts can be learnt in order to properly define our new system.

Once having settled up all the knowledge basis of this work, we need to identify and define all the requirements that our argumentation system will have. This is the initial step for carrying out the following tasks. The design of an argumentation framework to deal with potential privacy disputes in OSNs and its implementation is the core objective of this work. This system must also be integrated in a real OSN in order to be used with human users in real contexts.

Finally, the system will be tested and validated before using it. A metric for evaluating the persuasiveness of our system will also be proposed.

To sum up, the main objectives of this work are:

1. Analyse and study the most important concepts of computational argumentation and the argumentation frameworks existing in the literature.
2. Study and compare the most relevant privacy management systems.
3. Identify the requirements of the system being developed in this work.
4. Propose a new value based argumentation framework to deal with potential privacy disputes in OSNs.
5. Implement and integrate the argumentation system in a real online social network.
6. Test and validate the operation of the system.
7. Define the evaluation of the persuasiveness of the system to be carried out when gathering enough statistically representative data.

1.3 Document Structure

This document is structured in eight different chapters. In Chapter 2, a review over the main concepts of computational argumentation and the main privacy management techniques are described. Chapter 3 contains the formal specification of requirements defined before carrying out this work. In Chapter 4, the new argumentation framework proposed is defined, the architecture of the system is depicted and the interaction with

human users is also explained. Chapter 5 contains all the detailed information regarding the implementation of the system and its integration in a real online social network. There is also a brief description of how does the system behave once it is fully operative. In Chapter 6, the process of testing and validation are described and the evaluation metrics proposed are defined. The most important conclusions reached while carrying out this work can be found in Chapter 7. Finally, in Chapter 8, the open lines of research and improvements over the system developed in this work are proposed.

CHAPTER 2
Related Work

This section is divided into two main blocks. The first block is focused on reviewing the basis of argumentation in the artificial intelligence field. An analysis over the theoretical definition of argumentation, and a review of the most classical computational argumentation approaches is done here. The second block is focused on studying the existing methods of privacy management in the OSN domain. With the analysis of existing systems and methods, it will be easier to contextualise the purpose of this work.

2.1 Argumentation in Artificial Intelligence

In this section we perform a theoretical review of the existing concepts of argumentation in artificial intelligence [30]. Three main points are covered in this section: (i) the most basic notions of argumentation theory, (ii) the concept of abstract argument and the most relevant frameworks, and (iii) the definition of structured arguments with its new approaches to argumentation systems. To close the section we also explained the complexity of computational argumentation in order to emphasise on the computational problems that may arise when designing and implementing an argumentation system.

2.1.1. Principles of Argumentation Theory

As stated in [39], four important tasks are carried out by argumentation. These tasks are identification, analysis, evaluation and invention. Identification is the task of obtaining the two main components of an argument from a given text, the premises and the conclusion. Analysis is the task of finding implicit premises or conclusions from a given argument in order to be able to evaluate it correctly. It is common to do not communicate all the premises or some conclusions of a determined argument if those premises or conclusions can be implicitly understood. Even though they are not explicitly expressed, it is important to make them explicit in the analysis task before evaluating the complete argument. Evaluation is the task of determining whether an argument is weak or strong in a given context. Finally, the task of invention is to create new arguments in order to prove a specific conclusion.

In order to help undertaking these tasks, some techniques have been proposed in the literature. Argument diagramming is an important tool to help with the task of analysis and evaluation. As can be observed in Figure 2.1, an argumentation diagram is a graph where nodes are propositions and edges represent inferences. These diagrams are really helpful when finding implicit premises or conclusions but can be also useful when determining the value of an argument.

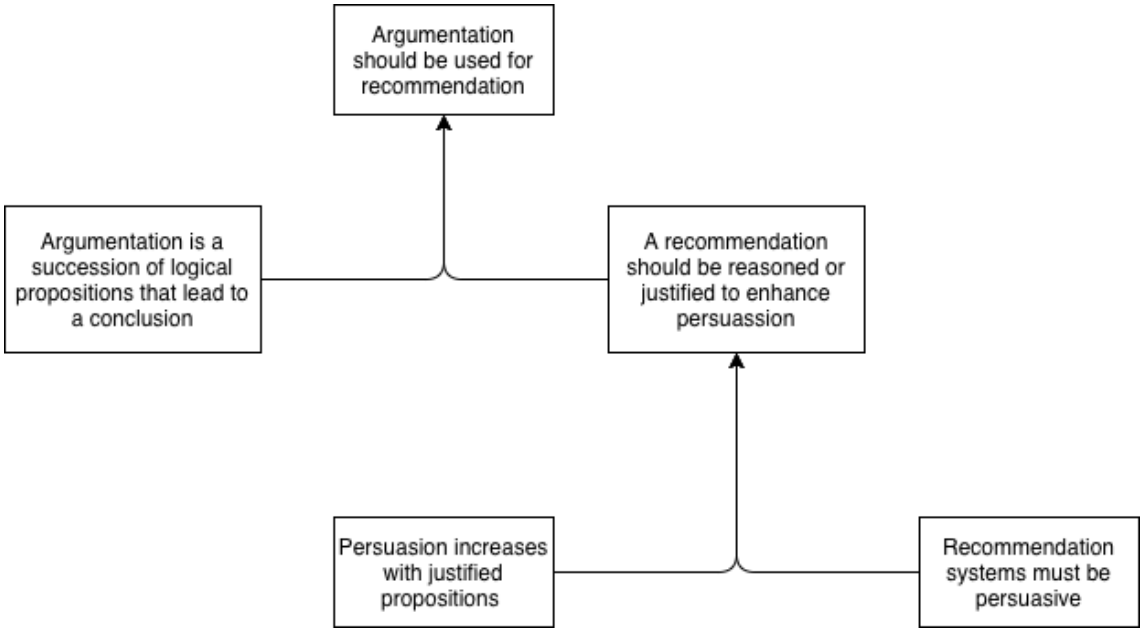

Figure 2.1: Example of an Argumentation Diagram

Argumentation schemes [38] are stereotyped patterns of human reasoning commonly used in argumentation. Some examples of these schemes can be *arguments from consequences* (i.e. arguments should be used to persuade, since an argument provides a reasoned explanation of a topic), *arguments from expert opinion* (i.e. the campaign advisor states that arguments should be used to persuade), *arguments from popular opinion* (i.e. it is well known that the use of arguments can make it easier to persuade), etc. Each scheme has a set of critical questions that can be made (e.g. are there other consequences that should be taken into account?, can you really trust the expert?, etc.). Those critical questions represent standard ways to doubt about an argument acceptability and therefore, determine the strength of an argument. Argumentation schemes can be very useful to perform the evaluation task since the main weaknesses of the most important schemes have been widely studied.

It is also important to define existing relationships between arguments. A way to attack an argument is to ask a concrete critical question that may create doubt about the acceptability of an argument. In order to graphically represent argument relationships, [16] proposes an argumentation graph $G_a = (A, R)$ where arguments are the set of nodes $A = \{ \alpha \in \{1, \ldots, |A|\}\}$ where each α is an argument, and edges are the relationships $R = \{(i, j) \in A \times A\}$, where argument i attacks argument j. A relationship between two different arguments can be seen as a support relationship or attack relationship. A support relationship between two different arguments happens when both arguments have the same claim. An attack relationship between two arguments happens when a critical question is asked. An example of this graph can be observed in Figure 2.2.

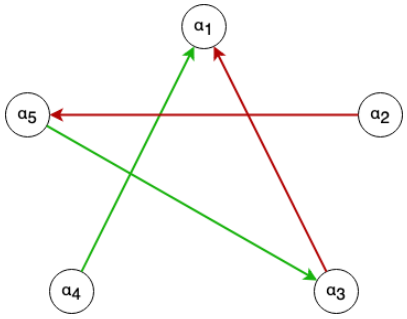

Figure 2.2: Argument Graph

On this example, α_4 supports α_1 and α_5 supports α_3 since those arguments are answering some critical questions of α_1 and α_3. On the other hand, α_3 attacks α_1 and α_2 attacks α_5 because in this case, they are asking some critical question of α_1 and α_5.

Having the most important theoretical concepts of argumentation defined, the next part focuses on describing the main approaches existing in the literature to deal with an argumentation environment: *Abstract Argumentation Systems* and *Structured Arguments*.

2.1.2. Abstract Argumentation Systems

An abstract argumentation system or framework [16] is essentially a tuple <A, R> where A is a set of arguments and R is the set of binary relationships between different arguments. On Dung's proposal, a relationship is called *attack relation* since there is only this type of relation between arguments.

An important element in Abstract Argumentation Systems are the semantics. In [4] an *argumentation semantics* is defined as the formal definition of a method ruling the argument evaluation process. In other words, given an argumentation framework, the semantics can be seen as the rules that will define the output (*acceptability*) of the final evaluation of the arguments. It is important to emphasise that for the same argumentation framework, two different semantics can derive in a different *acceptable* set of arguments. There exist two main styles of argumentation semantics in the literature: *extension-based* and *labelling-based*.

Extension-based semantics propose to derive a set of extensions from an argumentation framework following an specified method. Basically, an extension E can be seen as a subset of A from a given framework <A, R>. The main property of E is that all arguments contained on it are collectively acceptable, meaning that are compatible between them. Regarding this type of semantics, in [16] four different types were proposed: *complete*, *grounded*, *stable* and *preferred*. But new proposals have appeared in the literature such as *semi-stable*, *stage* or *ideal* semantics. These new semantics usually were created to get over some limitation or to improve any undesired behaviour of the former semantics.

On the other hand, labelling-based semantics propose a way to derive a set of labels from an argumentation framework. A labelling L is an assignment of a determined label from the set to each argument existing in A. Therefore, when working with labelling-based semantics, apart from the argumentation framework it is also compulsory to define the set of labels and the labelling function L.

Having explained the structure and the semantics of the basic argumentation frameworks, we will now focus on analysing different properties and variations of this basic framework existing in the literature. Concretely, we will focus on value-based and bipolar argumentation frameworks since they can be seen as the basis of our work.

Value-Based Abstract Argumentation systems

Sometimes it is not possible to have a unique acceptable argument (or a set of arguments with the same claim). It is very common to have different rationally valid positions or perspectives regarding to a determined discussion. For example, let's assume Alice and Bob are in a theme park deciding which roller coaster will they ride next. Alice offers the argument "let's go to roller coaster A because there is no queue, so we will ride earlier" for what Bob replied "we should go to roller coaster B since it is more intense". We can observe in this example how both arguments are acceptable, and therefore neither Alice or Bob can say the other is wrong. So that the final conclusion will be either A or B regarding an agreement based on the personal preferences of both Alice and Bob.

Proposed in [6, 7], value-based argumentation framework (VAFs) are an abstract argumentation approach to deal with problems like the one from the example above. As we can now imagine, to be able to persuade a determined audience in a value based argumentation case, it will depend on the audience preferences to determine which arguments will be more efficient. In order to handle with this, argument from VAFs usually have a related value and different arguments may promote different values. Formally, a Value Based Argumentation Framework is defined as an extension of the Abstract Argumentation Framework as follows.

Definition 1 (Value-Based Argumentation Framework)**.** A VAF is defined as a tuple <A,R,V,val,P> where A is the set of arguments, R is the set of binary relationships on A, V is a non-empty set of values, val is a function that maps elements from A to elements of V and P is the set of possible audiences.

A determined argument $\alpha_i \in A$ is related to a concrete value $v \in V$ if by accepting α we are promoting v, the value given by $val(\alpha)$. Therefore, for the example above, any argument that considers wait time will have a higher value for Alice (audience 1). And any argument that takes into account the roller coaster intensity will have a higher value for Bob (audience 2).

Bipolar Abstract Argumentation Systems

All the frameworks presented in this review of the state of the art have one common thing. There is only one type of relationship between arguments, the *attack* relationship. For some domains, having only *attack* relationships is not enough. Regarding preferences in VAFs, in the study made in [8] the conclusion is reached stating that two kind of preferences can exist, *positive* preferences representing what a user wants and *negative* preferences representing what the user rejects. Another approach where bipolarity can become a very interesting alternative and very related to this work objectives is decision making. In [3, 15] it is proposed that, when making a decision, usually two types of information are taken into account, arguments in favour and arguments against that concrete decision. With some of the main applications of the bipolar systems explained, an abstract bipolar framework is formally defined as follows.

Definition 2 (Bipolar Argumentation Framework)**.** A BAF is defined as a tuple <A, R^+, R^-> where A is the set of arguments, R^+ is the set of binary relationships on A named *support* and R^- is the set of binary relationships on A named *attack*.

With the definition of BAFs it is possible to represent both relationships of *attack* and *support* between arguments. Let's retake the example of the last subsection where a decision has to be done regarding which roller coaster ride next. Alice now wants to decide whether go to roller coaster A or not. To help answering this question, there will be two

main subsets of arguments, arguments in favour of riding and arguments against riding roller coaster A. With support relationships it is now possible to group all the arguments with the same orientation (in favour or against), and with attack relationships confront opposite arguments. Therefore, let's assume Bob wants to ride the roller coaster B. All the arguments uttered by Bob in order to persuade Alice to go to roller coaster B can be grouped with support relationships ($R_B^+ \subset R^+$). On the other hand, all the arguments uttered by Alice to persuade Bob to go to roller coaster A can be also grouped with support relationships ($R_A^+ \subset R^+$). Finally, both sets can be related between them with attack relationships (R^-) since the claim of each set is the opposite.

Quantitative Bipolar Argumentation Framework

Based on both VAFs and BAFs, the quantitative bipolar argumentation framework [5] is a bipolar framework where values are also taken into account.

Definition 3 (Quantitative Bipolar Argumentation Framework)**.** A QBAF is defined as a tuple <A, R^+, R^-, τ> where A is the set of arguments, R^+ is the set of *supports*, R^- is the set of *attacks* and τ is a score function that for any $\alpha_i \in A$, $\tau(\alpha_i)$ is the score of the argument α_i.

With a QBAF it is possible to work both with two types of relationships and values assigned to each argument. Retaking again the running example in this section, now with this framework it is possible to combine both last examples. Let's consider the situation where Alice must choose whether go to roller coaster A or not. With the use of a QBAF, all arguments in favour and all arguments against will be grouped with support relationships. There will be also attack relationships between arguments of both groups. But, each argument either in favour or against will have a value based on Alice preferences (wait time). It is possible that now an argument against going to roller coaster A because there is a roller coaster C where there is even less queue gets a higher score, since Alice preferred to ride the lesser wait time roller coaster.

2.1.3. Structured Arguments

Now we will focus on describing the shape of the own argument as an entity. The definition of an argument from a classical logic point of view consists in a tuple <Φ, α> where Φ is called the support of the argument, being all the logical formulae that proves α, the claim of the argument. Starting from this definition it is also possible to define how will attacking arguments or counterarguments be. We can consider a counterargument for an argument <Φ, α>, another argument <Ψ, β> whose claim β invalidates the support Φ. Two types of attacks are defined in the literature, the *undercuts* and the *rebuttals*. An argument a_1 = <Φ, α> is undercutted by a_2 = <Ψ, β> if and only if the claim of a_2, β is in contradiction with the support of a_1, Φ (e.g. a_1: "We should ride roller coaster A since it's the one with lesser queue.", a_2: "Roller coaster A has a lot of wait time since very few people ride it at the same time."). On the other hand, we say an argument a_2 = <Ψ, β> rebuts another argument a_1 = <Φ, α> if and only if both claims are contradicting each other so that $\beta \rightarrow \neg\alpha$ and $\alpha \rightarrow \neg\beta$ (e.g. a_1: "We should go to roller coaster A because there is no queue.", a_2: "We should not go to roller coaster A since it is boring.").

Having presented the basic concept of structured arguments, we will now briefly explain the Assumption-Based Argumentation, an argumentation approach based on structured arguments.

Assumption-Based Argumentation

The main peculiarity of Assumption-Based Argumentation (ABA) systems is that arguments are defined from deductions by the use of logical inference rules, and supported by assumptions. This approach was proposed in [9] as an extension of Dung's Abstract framework [16]. Thus, an Assumption-Based Framework is defined as follows,

Definition 4 (Assumption-Based Argumentation Framework)**.** An ABA Framework is a tuple <L, R, A, $\bar{}$> where (L, R) is the deductive system being L the language and R the inference rules defining each framework. On the other hand A is a subset from the language L whose elements are the assumptions. Finally, $\bar{}$ is the mapping from A to L.

It is now possible to observe the importance of the inference rules for each ABA framework to be defined. Depending on the domain and the language, it is a very important task to consistently define those inference rules in order to be able to infer correctly all the arguments in a specific context. We can also observe how this framework does not have an *Argument* element as such. In ABA frameworks arguments have also a defined structure. An argument is a set of premises from the language that lead to a specific conclusion. Therefore, an argument has a claim inferred by a set of assumptions (support).

Apart from the structure it is also interesting to understand how do attacks and acceptability work on these frameworks. In ABA, an attack between arguments happens when they have contrary conclusions so, for example if argument α_1 states "*buy product A*" and α_2 states "*buy product B*" there will be an attack relationship between both arguments. On the other hand, the concept of acceptability is similar to the result of evaluation for a given semantics. An argument will be acceptable if it can be defended from counterarguments and stay coherent. Then, a set of arguments is acceptable if it does not attack itself and it attacks every other argument attacking it.

2.1.4. Complexity in Argumentation Solving

Once the argumentation graph is built from a specific framework, it comes the moment to define the acceptable and the defeated sets of arguments. Since one of the main applications of artificial argumentation is human interaction, it is not nonsense to assume that obtaining those argument sets must be achieved in a short period of time enough to keep the human involved in the dialogue. Therefore, computational complexity in argumentation solving is a first order concern when implementing an argumentation system.

In order to measure the complexity of this problem we will use the classical computational complexity classes (i.e. P (polynomial), NP (non-deterministic polynomial), etc.). In [18] a set of decision problems over argumentation frameworks are defined in order to measure the complexity. Concretely, *Verification* (VER), *Credulous Acceptance* (CA), *Sceptical Acceptance* (SA), *Existence* (EX) and *Non-emptiness* (NE). In our case we will focus on *Sceptical Acceptance* (SA) as this decision problem states for a given framework <A, R> and an argument $\alpha \in A$, to find if it is member of all *acceptable* extensions of the considered framework. That is equivalent to ask if the argument is valid in a specific instance.

As analysed in [17], to determine if an argument is *sceptical acceptable* under *preferred semantics* is a problem beyond P and NP complexity classes, that means it is considered highly unfeasible. Concretely, it is considered a $\prod_2^p$-complete problem meaning that, with the use of deterministic algorithms under the classical paradigms of computation, solving this problem would have a super-exponential worst case running time. That is even worse than NP problems whose usually are exponential worst case bounded.

Therefore, we must be very careful with the complexity of our solver algorithm when defining our argumentation system. Since our system is focused on the direct interaction

with humans, we must be able to compute the *sceptical acceptable* (SA) set of arguments in a non significant lapse of time.

Taking all the theoretical concepts explained in this chapter into account, a new framework will be defined in Chapter 4. Our approach is mainly based in the QBAF and its properties. Later on, in Chapter 5, the implementation and integration process of our theoretical framework in a real OSN will be explained.

2.2 Privacy Management in Online Social Networks

In this section, we will focus on the privacy management state of the art works. We will give a glimpse over different approaches, but focusing on the argumentative approach. We will finally make a comparison between all the different methods explained in this section.

Nowadays privacy management is a very important concern. In the domain of online social networks there have always been issues when trying to choose the correct privacy configuration. The following six relevant privacy management methods have been identified in the literature:

- The classical Facebook[1] privacy management system, which allow users to report any publication made by another user that may infringe someones privacy preferences. This privacy management system initially only takes into account the criterion of the publisher. It is only once the publication is made when the co-owners can take part into the privacy preferences matter.
- Primma-Viewer [40] is a collaborative privacy management tool where each user appearing in the publication can manage its privacy configuration. Initially, the privacy policy is specified by the owner of the publication. Other users can be invited to edit the policy previously defined. Therefore, always with the owner supervision, it is possible to modify the initial privacy configuration.
- FaceBlock [28], a project for managing user privacy in photos. Each user can define privacy rules with the use of a Semantic Web Rule Language. Then, the system uses a reasoner to find out if any of those rules is triggered. If this happens, a notification is sent with that user privacy preferences. If a user does not want to appear, FaceBlock distorts its face before making the publication.
- CoPE [34] is another collaborative privacy management system where users can define privacy policies for each publication. Each publication co-owner can define its own privacy preferences and finally the result is decided with a votation. Therefore, the privacy policy taken into account for a specific publication will be the most voted policy.
- PriNego [25] is an agent based negotiation protocol to settle differences in the privacy preferences of different users. In this protocol, each agent represents a user and it is responsible for keeping track of its privacy constraints. Each agent it is also responsible of making deals with other agents. In order to do this, the content owner agent creates a post request before making a publication, each agent can reject a specific request by giving a rejection reason. Once the creator of the request has received all the rejections, it can modify the privacy configuration and do another iteration. The negotiation finishes when all agents agree with the content owner or a maximum number of iterations (specified by the owner) is reached.

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

- PriArg [22] is an agent based framework argumentative approach for privacy management. Each agent has a specific ontology with the social network information, the relationships and the content being published. With all this information, each agent can make post requests to publish some content or evaluate other agents post requests. Agents can also communicate between them to acquire any missing information. Each agent is able to generate arguments from this set of information and the final decision is made by an ABA framework.

It is possible to observe some similar properties between some of these methods. Both Facebook and Primma-Viewer provide a *posteriori* solution to a privacy conflict for a given publication. FaceBlock shows a very important limitation, since it is only designed to work with photos. On the other hand, CoPE shows an interesting vote based approach, but this methods lacks of any type of customisation nor *intelligence*, in fact is the own user who must specify the privacy preferences. This fact can be a real problem since all members of publications have to vote before publishing the content. Finally, PriNego can be seen as a preliminary version of PriArg since both approaches are quite similar and proposed by the same people. PriNego shows an interesting agent system for choosing the privacy configuration of a publication. However, it also shows very important problems for a system intended to work in such a sensitive domain. In PriNego the negotiation about privacy preferences can finish when an agreement between all parties is found, that is the desirable case. But it also finishes when a maximum number of negotiation rounds are carried out, and that number is defined by the owner of the publication. This can be seen as a critical flaw in privacy terms, since the owner can set that number to a very small number of rounds and therefore, finish the negotiation when and how the owner agent obtains the maximum utility regarding its preferences. PriArg seems to have this issue solved since the final decision is made based on the number and the quality of the arguments in favor or against. Nevertheless, PriArg also shows an important problem while performing the argumentation between agents. In PriArg, the privacy configurations are not kept hidden. In fact, this information is sometimes used as arguments.

Having reviewed and analysed all of the most important methods found in the literature, it comes out another important problem. All of the methods analysed are focused on MPPC, but never consider the self violation of privacy preferences. At this point, it is interesting to explain the subtle nuance between privacy preferences and privacy configuration. In an OSN, the privacy configuration is defined by a user following its privacy preferences, therefore the privacy configuration can be seen as the tangible part of all this privacy concern. On the other hand, privacy preferences are intrinsic to the user, and those are usually based on users personality, education, awareness, etc. So it is important to be aware and to have a strong knowledge of the dangers of privacy violations in order to minimise the amount of undesired actions in the network. One may think that nobody is going to self violate its own privacy preferences, but sometimes, depending on multiple factors (e.g. emotional state, sentiments, etc.), people don't take into account the same criteria on decision making. In addition, as we will explain later on, it is also interesting to handle all these situations in an educational domain, since OSN users may not have a complete perspective to the dangers of privacy issues in OSNs yet. Also linked with the educational domain, there is also another important lack in the methods explained: we need an argumentation system that is able to provide the user a human readable reasoned explanation of what's wrong with the publication (in the case there is any privacy issue) in order to learn from it.

	Automation	Concealment	Protection	Reasoning	Genericity
Facebook	✗	✓	✗	✗	✗
Primma-Viewer	✗	✓	✗	✗	✗
FaceBlock	✓	✓	✓	✗	✗
CoPE	✗	✓	✓	✗	✗
PriNego	✓	✓	✓	✗	✗
PriArg	✓	✗	✓	✗	✗

Table 2.1: Comparative of different privacy management systems

2.2.1. Comparative of Existing Privacy Management Systems

In [25, 22] six concepts are presented in order to compare different privacy management systems: **automation**, **concealment**, persuasion, external consultation, fairness and **protection**. As we can observe in Table 2.1, the comparative between methods done in this work uses some of those concepts proposed in previous work but some new properties are also introduced, since it may be interesting for the understanding of our work.

Automation refers to the capacity of the system of working without human intervention. Only FaceBlock, PriNego and PriArg are automatic systems. With a previous user configuration these three systems are capable to handle with the privacy management by their own way. Other systems like Facebook, Primma-Viewer or CoPE require of user intervention for each privacy conflict detected.

Concealment is the property that determines if a system reveals the hidden configuration of a user to other users. Since we are working with privacy issues, we consider the own privacy configuration as an important feature to keep hidden from other users in order to preserve users privacy. Most of the systems have this property, in any of Facebook, Prima-Viewer, FaceBlock, CoPE or PriNego, privacy preferences are hidden from other users. But PriArg, reveals users privacy configuration in order to justify the arguments generated. Argumentation is an important positive step in the development of a privacy management system, however, we need an argumentative system that not endangers the most basic privacy feature, the user preferences.

Protection refers to the capacity of a method when dealing with privacy violations. A privacy violation can be solved before or after the publication has been done. It is a very important issue to deal with a violation before the content has been published. Therefore, we desire that any privacy management system has this property. Both Facebook and Primma-Viewer do not have this property. These two systems provide users with the options to deal with a privacy violation once the content has been published and therefore, the damage has already been done. FaceBlock, CoPE, PriNego and PriArg deal with any potential privacy violation before publishing the content so, with these systems there will always be less violations than with systems without protection property.

Reasoning is the property that determines if the user is given a reasoned explanation of which is the best decision and why. None of the systems gives an explanation to the user. Therefore, there is no system with reasoning property, a very important property for the specific domain of this work. As we have mentioned before, it is not only desirable to deal with privacy violations, but also to be able to give users some kind of feedback in order to reduce the number of future privacy violations.

Genericity is the property for determining whether a method is able to deal with any kind of situation or is focused on a specific type of conflict. This is another relevant property that is not fulfilled by any of the systems analysed in this work. It is desirable to have a flexible system capable to deal with privacy violations either in a party or individual

context. Most of the systems currently available are only focused on the controversial MPPC, but they ignore the existing self privacy violations.

We have analysed many privacy management systems in this section, some of them useful for some specific domains. However, we have defined the five desirable properties that our system should have, and none of the already existing systems was complete regarding the desired properties. In the next chapter we will specify the requirements our system will have in order to settle the bases of this whole work.

CHAPTER 3
Requirements Specification

3.1 Introduction

3.1.1. Purpose

The requirements specification carried out in this work has been done following the IEEE830 standard. The main purpose of this chapter is to provide a detailed compilation of all the system functions and design constraints to be taken into account in order to accomplish all the objectives of this project. There will also be an analysis over the expected user characteristics and a review over all the constraints and dependencies of our system.

3.1.2. Scope

An argumentation system is going to be designed and developed in this work. Our system must be able to automatically extract the information needed from the social network, to generate user profiles based on their personality and the data available on the network, and it also must be able to capture when a new interaction is going to be performed and generate arguments based on the context. The system will determine the acceptable set of arguments and, in the case that user interaction is required, the system must also be able to carry out a persuasive direct interaction between the social network and the human user. The final goal of this work is to completely integrate the argumentation system in a real social network and be able to validate and evaluate the behaviour of it in a real environment.

3.1.3. Overview

In order to totally accomplish the objectives of this work, it is interesting to define a range of requirements. These requirements must guarantee the correct behaviour of our system: data extraction and processing, argument generation and solving, human-network interaction, etc. Therefore, this chapter has been structured as follows. Section 3.2 contains a general description of the work in order to contextualise our objectives. In Section 3.3 all the requirements and constraints of our system are listed and defined. In this work, we will focus on the functionality of the system since we consider it as the key feature to completely achieve our objectives.

3.2 Overall Description

3.2.1. Work Perspective

The argumentation system proposed and implemented in this work is framed in the *Agentes Inteligentes para Asesorar en Privacidad en Redes Sociales* (AI4PRI) Spanish project TIN2017-89156-R funded by the *Ministerio de Ciencia, Innovación y Universidades*. In fact, one of the project's objective (T03) is "To develop an argumentation framework that allows influencing on the behavior of users through the exchange of arguments, based on the preferences and values associated with the user's privacy.". Therefore, the system developed in this work will be completely functional in the PESEDIA environment, the social network that frames AI4PRI project. The system will be implemented in Python3 and deployed as a service accepting requests from PESEDIA.

PESEDIA [10] is an educational social network developed with Elgg and using PHP. A plugin for the network will also be implemented in order to capture events and make requests to our argumentation system properly. Therefore, our plugin will follow all the Elgg constraints and requirements in order to be able to totally integrate our argumentation system in the PESEDIA environment.

3.2.2. System Functions

The argumentation system carried out in this work must be able to assist human users in privacy management when a privacy violation is detected. Therefore, the system must accomplish the following functionalities:

- Receive requests from the OSN plugin.
- Detect privacy violations before the content is published.
- Extract the required information from the social network.
- Automatically generate user profiles based on the actions made in the OSN and personality surveys.
- Generate internal arguments in favour or against sharing some content based on the information retrieved from the network.
- Score the arguments based on the value preferences of the user profiles generated.
- Internally decide whether some content should or not be published taking into account the scores obtained by the arguments.
- Translate internal arguments to human readable template based arguments.
- Save statistics about behavioural changes in OSN users.
- Send results to the OSN plugin.

In addition to these functions defined for the argumentative system, the plugin developed to integrate the system within the OSN must also implement the following functions:

- Send requests from the OSN to the argumentation system.

- Make specific queries to retrieve the required information from the OSN database.
- Receive results from the argumentation system.
- Display the arguments to the OSN users in order to give a reasoned explanation about the privacy issues.

3.2.3. User Characteristics

The argumentation system is a user assisting tool for privacy management. Therefore, the target user is a non-specific knowledge user. Since this work is framed in an educational project, the main users of the system will be 13-14 year old users with elementary school knowledge. Even though these will be the main users of this system, it is not limited to any type of specific user/knowledge so, any social network user is a potential user of our system, regardless of the age and/or the knowledge.

3.2.4. Constraints and Dependencies

The system will be implemented in python and deployed as a web service. In order to correctly run our system, we identify the following constraints and dependencies:

- The installation of Python3 is required.
- Numpy and Flask python libraries must be installed.
- The system must run on a server or a stable machine connected to the OSN.
- Some specific ports must be opened in order to correctly communicate with the social network plugin.

All these dependencies are defined regarding the argumentation system service by its own, but a plugin must also be implemented. The plugin has to be integrated into the allready developed OSN called PESEDIA, therefore some other constraints and dependencies show up regarding the plugin being integrated in the OSN:

- The machine holding the OSN must have Ubuntu 16+ operative system.
- Apache server, MySQL 5.7+ and PHP 7+ are required.
- Elgg 2.3.10 must be installed.

Considering all these constraints and dependencies defined, the next section provides a detailed list of specific requirements defined in order to fully meet our objectives.

3.3 Specific Requirements

In this section, the compilation of all the requirements of our system has been listed. The section has been divided into two different blocks. The first block contains the list of all the functions that our system must be able to carry out. The second block consists of a list of all the requirements and constraints regarding the design of the system.

3.3.1. Functions

Identifier	F01
Name	Privacy violations detection
Description	Automatic detection of privacy violations.
Input	The textual content of the publication and user profiles involved
Output	Yes/No

Table 3.1: Function F01

Identifier	F02
Name	Privacy analysis
Description	Calculation of user's concern regarding privacy
Input	Privacy preferences
Output	Value in range [0,1] being 1 the most private configuration and 0 the less private configuration settings

Table 3.2: Function F02

Identifier	F03
Name	Trust measurement
Description	Computation of the trust value between two different users.
Input	Direct trust rating values and user profiles
Output	Trust value in range [1,0] being 1 the highest trust and 0 the lowest trust between 2 users. Two different values are computed, since trust is not a symmetric value.

Table 3.3: Function F03

Identifier	F04
Name	PRS computation
Description	Obtain the propagation risk metric for a given publication and a specific OSN topology.
Input	Publisher user id and OSN user relationships
Output	PRS value in range [0,1] being 1 the maximum risk and 0 if there is no risk

Table 3.4: Function F04

Identifier	F05
Name	Content analysis
Description	Analysis of the content of some specific publication.
Input	The textual content of the publication
Output	Vector with the parameters defining each type of content appearing in the publication

Table 3.5: Function F05

Identifier	F06
Name	Feature extraction
Description	Adapts and processes all the features previously extracted in order to make them usable by the system
Input	F02, F03, F04, F05 outputs
Output	Dictionary structured data with requested features

Table 3.6: Function F06

Identifier	F07
Name	User profiling
Description	Automatic generation of user profiles based on actions in the OSN and personality surveys
Input	Survey results
Output	Vector with the parameters defining an specific user profile (big 5 values)

Table 3.7: Function F07

Identifier	F08
Name	Internal argument generation
Description	Automatic generation of internal arguments from the features extracted from the network
Input	Feature dictionary
Output	List of 3 element tuples that define each internal argument

Table 3.8: Function F08

Identifier	F09
Name	Argument scoring
Description	Assign each argument a customised score depending on each user profile involved
Input	User profile and internal argument
Output	Score value for the argument

Table 3.9: Function F09

Identifier	F10
Name	Argumentation solving
Description	Deciding whether some content should be published or not
Input	List of scored internal arguments
Output	Yes/Not

Table 3.10: Function F10

Identifier	F11
Name	Argument translation
Description	Generate human readable arguments from the internal arguments (3 element tuple)
Input	Internal argument
Output	Template generated argument

Table 3.11: Function F11

Identifier	F12
Name	Persuasive statistics saver
Description	Save statistics about the behaviour changes occurred in the network in order to evaluate the persuasiveness of our system. The number of times that the argumentation system has interacted with human users, which type of arguments have been involved in users persuasion and the order of the displayed arguments.
Input	Directory
Output	File with statistics recorded (Times persuaded/not persuaded, type and order of the arguments used by the system)

Table 3.12: Function F12

3.3.2. Design Constraints

Identifier	DC01
Name	Time efficiency
Description	Since all the process will happen whenever a user tries to publish some content, the computation must be time efficient in order to don't make the user wait

Table 3.13: Design Constraint DC01

Identifier	DC02
Name	Automation
Description	The system must operate automatically, the user must not be involved in any systems task except from the direct user interaction phase

Table 3.14: Design Constraint DC02

Identifier	DC03
Name	Concealment
Description	Any user privacy preference or configuration must never be revealed by the system

Table 3.15: Design Constraint DC03

Identifier	DC04
Name	Protection
Description	The system must be able to prevent any type of privacy violation detected in the social network and persuade the user with arguments

Table 3.16: Design Constraint DC04

Identifier	DC05
Name	Reasoning
Description	The system must be able to provide reasoned explanations of the violation detected to the OSN user

Table 3.17: Design Constraint DC05

Identifier	DC06
Name	Genericity
Description	The system must be able to take part in any type of interaction in the OSN domain (not only MPPC)

Table 3.18: Design Constraint DC06

Identifier	DC07
Name	Persuasive
Description	The template based arguments generated by the system must show good persuasive performance

Table 3.19: Design Constraint DC07

Having defined all the requirements of our system we have been able to set the bases of this work. In the next chapter the we will define the argumentation system, the formalisation of our framework, the designed architecture and the human-computer interaction.

CHAPTER 4
Argumentation System

In this chapter, we define our argumentation system as an educational tool to preserve the privacy of the users of an online social network. To this end, we assume that the system operates in a OSN that includes the common features of these networks (e.g. user information and preferences, friends, groups, privacy configuration) and that allows users to perform common social actions (e.g. posting a comment, sharing a photo) [35].

Taking [32] as the starting point to define our argumentation system, the chapter is organised as follows: in Section 4.1, the framework designed in this work is formally defined, in Section 4.2, the architecture of the system is depicted. Finally, in Section 4.3, the chosen way to carry out the user-network direct interaction is explained.

4.1 Framework Formalisation

Our Argumentation Framework is based on Quantitative Bipolar Argumentation Frameworks (QBAFs) [5] previously defined.

Definition 5 (Argumentation Framework for Online Social Networks)**.** We define an argumentation framework for online social networks as a tuple $AFOSN = <A, R, P, \tau_p>$ where: A is a set of n arguments $[\alpha_0, \ldots, \alpha_n]$; R is the attack relation on A such as A$\times$A $\rightarrow$ R; P is the list of e profiles involved in an argumentation process $[p_0, \ldots, p_e]$; and τ_p is a function $A \times P \rightarrow [0, \ldots, 1]$ that determines the score of an argument α for a given profile p.

In our framework, each individual argument $\alpha = (\beta, T, D)$ is defined by three parameters. β is the *claim* or bias of the argument. It is represented as a binary variable that determines whether an argument acts in favour or against performing an action in the social network. T is the label of the argument, which represents the four different types of arguments that can be generated by our argumentation system: Privacy, Trust, Risk and Content arguments. Finally, D is the *support* of the argument. This parameter consists of a value representing all the information gathered from the social network in order to infer the *claim* of a determined type of argument, and hence depends on the type of argument.

Each relationship $r = (\alpha_i, \alpha_j)$ represents an attack from α_i towards α_j. As proposed in [27] and [29], a *rebuttal* attack happens when an argument invalidates other argument's *claim* (e.g. α_1 = (-1, T_1, D_1) *rebuts* α_2 = (+1, T_2, D_2) and vice versa). On the other hand, an *undercut* attack is carried out when an argument's *claim* invalidates other argument's *support*. The internal argumentation process performed by our system to generate arguments for a particular action in the OSN only allows *rebuttal* attacks.

Regarding the user profile, we define $p = (\nu,\rho,M)$ as the combination of the preference values ν, the personality ρ and a list of miscellaneous information M.

The preference values ν is a vector containing the preferences that each user has towards a determined value that the arguments of our system may promote [6]. We propose the following preferences based on [33] to be considered in our system: *Privacy/Popularity*, *Closeness/Openness*, *Flexibility/Intransigence* and *Content Sensitivity*. The first three bipolar preferences $P/\overline{P}$, are defined as a value v in the [0,1] range, being v the value assigned to P and (1 - v) to $\overline{P}$. Therefore, a user profile with *Privacy/Popularity* = 0.2 would have a 0.2 preference for Privacy and (1 - 0.2) preference for Popularity. The *Content Sensitivity* preference is defined as a value v in the range [0, 1] being 1 the maximum concern about the content sensitivity and 0 if the user does not really care about this preference. This value is calculated as the average of the 6 different types of content considered in this work and explained in subsection 4.2.1.

The personality of a user profile ρ is a 5 dimension vector that models the personality of a determined user based on the five parameters proposed in [31]. The personality dimensions taken into account are the *Openness*, *Conscientiousness*, *Extraversion*, *Agreeableness* and *Neuroticism*. Here we assume that this information is available since users of the network have undertaken a personality test or else, that these dimensions can be automatically determined by the activities of the user in the social network [21].

The last part of the user definition (M) is a set of general information extracted from a specific user profile such as the age, location, likes, etc.

Finally, we define the scoring function τ_p as the function that takes an argument α and a profile p as input and determines the value of the argument in the context of a specific user profile. In order to obtain this score, function τ_p is defined as,

$$\tau_p(\alpha, p) = \alpha_\beta \cdot \alpha_D \cdot p_{v_i} \tag{4.1}$$

Thus, the score of an argument for a determined user profile is basically the product of the *claim* α_β (in favour or against) of that argument, the *support* value α_D of the argument, and the preference value p_{v_i} that promotes the argument.

Definition 6 (Defeat)**.** An argument $\alpha_i \in A$ defeats another argument $\alpha_j \in A$ in a context determined by a user profile p iff $(\alpha_i, \alpha_j) \in R \wedge |\tau_p(\alpha_i, p)| > |\tau_p(\alpha_j, p)|$.

Then, we can define $defeat_p(\alpha_i, \alpha_j)$ if there exists a relation between both arguments and the score of the argument α_i is higher than the score of the argument α_j. It means that the argument α_i is promoting a value that is preferred by the user that receives the argument.

Definition 7 (Acceptability)**.** An argument $\alpha_i \in A$ is acceptable in a context determined by a user profile p iff $\forall\ \alpha_j \in A \wedge defeat_p(\alpha_j, \alpha_i) \rightarrow \exists \alpha_k \in A \wedge defeat_p(\alpha_k, \alpha_j)$.

In other words, we consider that an argument is acceptable if there are not other undefeated arguments attacking it.

4.2 System Architecture

An argumentative process is defined by the achievement of four main tasks: identification, analysis, evaluation and invention [39]. Our system consists of four different modules designed to perform all these essential tasks (see Figure 4.1).

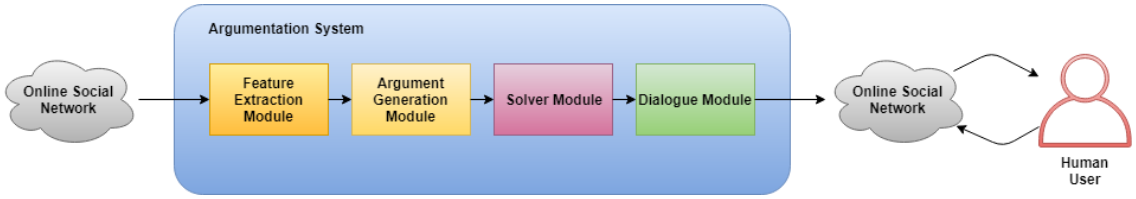

Figure 4.1: Architecture of the argumentation system.

4.2.1. Feature Extraction Module

This module is in charge of processing all the relevant information for our framework directly extracted from the OSN. The information obtained by this module will be used to model user profiles and to get the parameters that will define an argument in our system.

The main purpose of modelling user profiles is to be able to learn which arguments are more persuasive to what type of user. Since one of our goals is to maximise the persuasion of our system, by defining a set of user profiles it is possible to do a generalisation of that problem depending on the user's activity in the social network. The information extracted to define the profiles of users is obtained basically from three sources: the users profile preferences, the personality analysis and the miscellaneous information.

The **privacy configuration** of each user is the first source from where, it is possible to define a privacy vector that characterises each user. Usually, a social network user defines some privacy parameters when registering. Those parameters usually are the profile visibility (e.g. public, friends, private, etc.) and the default target of own publications (e.g. public, friends, group, private, etc.). With the use of the OSN this information can be updated in order to accurately model the privacy preferences of the users.

For the **personality analysis**, as pointed out before, the users' personality can be obtained either with a survey or by analysing users activity and interactions, considering the big five model [31].

Finally, when creating a profile in a social network some personal information is added by the user (e.g. the age, the location, the likes, etc.). All this data can also be extracted to generate the user profiles for our system. Concretely, the **miscellaneous information** can be useful to determine the differences between two different user profiles.

Apart from the users profile information, the feature extraction module also obtains all the parameters required to generate arguments in our system from the OSN data. Those parameters are mainly divided into three types: the trust, the Privacy Risk Score and the content features.

Trust is commonly understood as a way to measure the strength of a tie in a social network. In order to formally define the trust metric we can assume a directed graph $G_t = (N, E)$ that represents the topology of the OSN. Let $N = \{i \in \{1, \ldots, |N|\}\}$ be the set of nodes where i represents a user, and $E = \{(i, j) \in N \times N\}$ be the set of edges representing an existing relationship from i towards j. Therefore, the trust value $t_{i,j}$ indicates the strength of the tie that links user i with user j. It is important to emphasise that, since one of the main properties of the trust is the bidirectionality, the value of $t_{i,j}$ may differ from $t_{j,i}$.

Another important parameter that the system extracts from the network usage is the **Privacy Risk Score** (PRS). Proposed in [2], PRS is an alternative way to measure the reachability of a determined user in the social network. Therefore, the PRS provides information of the risk of sharing a determined information to non-desired users.

To calculate the PRS, the number of stages through where the shared information passes is defined as T. This variable represents the maximum deepness that a post can reach starting from the original user. It is also important to define a $T \times N$ reachability matrix γ_i for each user i of the total OSN's population N. This matrix is built in order to register the number of posts spreaded by a determined user i in a concrete stage t and have been received by other users.

The matrix $\gamma_{i_{t,j}}$ is used to refer to the value from the γ matrix of a user i in a determined stage t respect to another user j. That value can be found in the t row and the j column of the γ_i matrix, being the number of messages sent by i and reached by j in a determined stage t. On the other hand, the notation $L_{a_i}(l)$ means the set of users that can be found in the l deepness level starting from the user i.

Therefore, given a stage t of the flowing process, at a deepness level l and starting from the i user we define the Equation 4.2 as the average of the users that on this deepness level can read the post in the stage t.

$$p(i,t,l) = \frac{\sum_{j \epsilon L_i(l)} \gamma_{i_{t,j}}}{\gamma_{i_{t,i}}} \tag{4.2}$$

With the value obtained from the latter equation, it is possible to get the PRS value for a user i in a level l as proposed in Equation 4.3. This value can be interpreted as the percentage of users that can read the post made by i at any stage.

$$PRS(i,l) = \frac{1}{T} \sum_{t=1}^{T} \left(\frac{p(i,t,l)}{|L_i(l)|} \right) \tag{4.3}$$

Finally, by combining this two equations it is possible to obtain the PRS value from all the OSN population. This computation is defined in Equation 4.4 for any i user as the percentage of users that will be able to read the post made by that user at any stage.

$$PRS(i) = \frac{1}{T} \sum_{t=1}^{T} \left(\frac{\sum_{j \epsilon N} \gamma_{i_{t,j}}}{\gamma_{i_{t,i}} |N|} \right) \tag{4.4}$$

This measure can be really useful combined with the user profiles and their privacy configurations in order to prevent making any information public for more than the expected users.

The third parameter used in our model to build arguments is the result of analysing the own content of the publication. The **content features** are extracted from an analysis of the text, we have defined the following classes of sensitive information considering the ones proposed in [11]:

- **Location**. Information that reveals the location of any user involved in the interaction.
- **Medical**. Information that reveals the medical condition of any user involved in the interaction.
- **Drug**. Information that reveals the use of any kind of drugs/alcohol.
- **Personal**. Information that reveals any kind of personal information. From the sexual orientation or the job, to more identifiable information as the credit card number, the address or the birth date of any user involved in the interaction.

- **Family/Association**. Information that reveals the family members or their associations.
- **Offensive**. Information that may harass or offend other users.

The feature extraction module must determine whether a text feature contains sensitive information or not, and classify the sensitive information in any of those classes.

4.2.2. Argument Generation Module

The argument generation module processes all the information gathered by the Feature Extraction Module in order to create abstract arguments following the guidelines of the argumentation framework. This module generates four different types of arguments based on the type of information extracted from the OSN:

- **Privacy Arguments**. This class of argument emphasises on privacy vulnerabilities. The argumentation system is able to create privacy arguments with the data obtained from the user profile modelling. The argument is generated by computing the distance between the privacy configuration of the user and the privacy configuration of the publication. If the distance does not surpass a predefined threshold, the argument will have a positive bias. On the other hand, if it surpasses the threshold the bias of the argument will be negative. Let us assume for example, a user profile with a very restrictive privacy configuration. If that user tries to make a publication containing personal information and sharing it with all the network, the argumentation system will create a set of arguments regarding the privacy incoherence between the user profile and the action being done. Therefore, the main feature to generate privacy arguments is the privacy configuration vector of each user.
- **Trust Arguments**. Since the purpose of an OSN is to interact with other users, it is a very common situation when a user involves other users with its actions. An effective way to handle those privacy conflicts is to generate trust arguments. An argument of trust contains all the information extracted from the social network relative to the strength of the ties between users. These arguments are generated from the trust and the reputation computed with the information from the feature extraction module. Concretely, if trust from users involved in the publication towards the user making the publication is not enough (i.e. does not surpass an established threshold), an argument of trust against performing the action will be generated.
- **Risk Arguments**. When making a publication, it is impossible for the author to estimate how many users will be able to reach the information being published. Risk arguments are generated in order to warn the user about the risk of the publication being read by any undesired user of the network. The main feature used to generate arguments of this type is the PRS, since the own metric is a risk indicator. Having a high risk value will make the system generate an argument of risk against making the publication.
- **Content Arguments**. Content arguments are generated from the data obtained by the content features analyser. Therefore, there can be as many content arguments as classes of content defined before. In addition, depending on the user personality and privacy configuration some types of content arguments may be more or less persuasive. Let us suppose that there is a specific user who usually shares medical content on the OSN. To warn that user of the risks of sharing medical content may

have no sense. But, now we will assume that same user shares a post containing personal information. The system will detect the risk of sharing personal information and a higher score will also be given to this specific content argument than to the medical one.

The output of this module is an argumentation graph $G_a = (A, R)$ where arguments are the set of nodes $A = \{ \alpha \in \{1, \ldots, |A|\}\}$ where each α is an argument, and edges are the relationships $R = \{(i, j) \in A \times A\}$, where argument i attacks argument j. Therefore, once the set of arguments is generated, the module also creates the relationships between arguments.

In our argumentation framework, a relationship between arguments is defined by the attack relation. To determine if there exist an attack relationship between two different arguments, the parameter *bias* is used. An argument positively biased and an argument negatively biased are both attacking each other by definition.

4.2.3. Solver Module

The solver module of our argumentation system performs the task of evaluating the argumentation graph. To solve our argumentation graph we apply the function τ_p to the set of arguments generated, and the profiles involved in the argumentation process. Finally all the scores are added and the system checks if the result is positive or negative to decide the set of *acceptable* arguments.

$$sol = \sum_{i=1}^{|A|} \tau_p(\alpha_i, p) \tag{4.5}$$

where A is the set of arguments, α is a specific argument, p is the user profile creator of the content being evaluated and τ_p is the score function defined before. If the result is positive, there are no reasons to persuade the user on modifying his/her action. On the other hand, if the result is negative, the system keeps all the negative biased arguments and sorts them by their score in order to try to persuade the user to modify his/her action.

4.2.4. Dialogue Module

The purpose of this module is to handle the communication between the argumentation system and the human user. This module receives the set of A' *acceptable* arguments and an argumentation strategy π_a. We define an argumentation strategy as the policy that an argumentation agent adopts when facing an *opponent* (either human or agent). In our system, we have implemented this policy as a specific order to show arguments to the user. The dialogue module uses each argument $\alpha_n \in A$ following the strategy π_a in order to persuade the user to modify his/her action. In our case, the arguments are used following a decreasing order, from higher to lower score. However, the definition of specific argumentation strategies remains opened as future work. In the next section there is a more detailed review about our system behaviour when facing the user interaction phase.

4.3 User-network Interaction

Once the *acceptable* set of arguments is defined, only in the case where the arguments are against doing an action, an interaction between the social network and the user must

be started. On this section we will define how does the interaction process between the social network and the human user works.

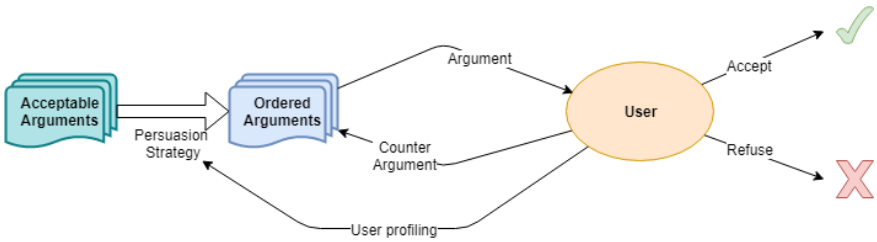

Figure 4.2: Structure of interaction process

A scheme of this process is depicted in Figure 4.2. As it can be observed, the interaction is divided in three main steps. The first one is to apply the persuasion strategy selected and to order the set of arguments in order to maximise the persuasion. This problem can be approached either with reinforcement learning techniques, as proposed in [20] and [26] or by matching argument schemes with persuasive properties as proposed in [37]. In this work, we have followed the basic approach of taking the order of the score of the arguments obtained for each user profile involved in the process. Implementing and testing more complex approaches remains future work.

The second step of the interaction process consists on the own argument exchange with the user. The system will use each argument of the ordered set one by one and the user will have the possibility of asking the system for more arguments. Finally, the last step is completely in the hands of the user. Once the user made up his mind (persuaded or not), the user must accept or refuse. By accepting, the user will stop from making the publication in the OSN. The privacy violation can be fixed by modifying the initial publication in order to respect the users privacy preferences, or in the worst case by giving up of doing that action. On the other hand, if the user refuses to take the arguments into account, the publication will be done and a potential privacy violation with it.

The direct interaction with the human user is very important and the system should be able to persuade any kind of user independently of their profile. Once the system has computed all the scores for all the existing arguments, the arguments against making the publication (the ones used to persuade the human user) are ordered from the highest score to the lowest score. Since the user profile is involved in score calculus, we can consider that each strategy will be different for each user unless two users share a profile with the same features. When all the arguments are ordered, the system generates the template based readable arguments as we explain in the next subsection.

4.3.1. Template Based Arguments

From the persuasive viewpoint, it is very important how the information is displayed to the user. Once all arguments are ordered and ready to be used, they must be *translated* into a human readable shape. Therefore, we have chosen, as a first approach, to generate arguments from a template. Based on the values that define each internal argument (3 element tuples) we will generate one or another argument with some specific shape as depicted in Table 4.1. Since the environment for which the system is being developed is an educational Spanish platform, the arguments showed below have been written in Spanish.

It is very important to remark that none of the templates used reveal any type of other users' confidential information (e.g. privacy configurations, trust values, etc.) since that could be a critical issue in our system. In fact, if we analyse each one of the templates

Type of Argument	Argument Generated
Privacy	*'La publicación va a ser leida por... ('nadie.', 'tus amigos.', 'una colección de amigos.', 'todo el mundo.')'*
	The publication is going to be read by... (no one., your friends., a collection of friends., all the users.)
Trust	*'Alguna de las personas que mencionas podría molestarse.'*
	Some of the people you mention might get upset.
Risk	*'Tu publicación podrá ser leída por personas desconocidas.'*
	Your publication may be read by unknown people.
Location	*'Puedes estar dando información de dónde estas o dónde vas.'*
	You can be revealing information about where you are or where you're going.
Medical	*'Puedes estar publicando información médica privada.'*
	You may be publishing private medical information.
Drugs	*'La gente podría pensar que consumes drogas/alcohol.'*
	People might think you're on drugs/alcohol.
Personal	*'Podrías estar publicando datos personales sensibles.'*
	You could be publishing sensitive personal data.
Relatives	*'Podrías estar haciendo pública información relacionada con familiares o amigos.'*
	You could be making public information related to family or friends.
Insults	*'Tu publicación podría ofender a las personas que la lean.'*
	Your publication might offend the people who read it.

Table 4.1: Template based arguments generated by our system.

we can observe a common feature, all the arguments used can be seen as reasoned soft recommendations since none of them are aggressive towards the human user to be persuaded. With this, we attempt to work with a friendly environment and don't make the user feel self-conscious.

4.3.2. User Responses

Finally, when a user receives an argument it is also very important to define the set of actions available as response to that argument. When a privacy violation is detected and therefore, an argumentation process is started, the user will have the first argument displayed in the interface. The options available for this argument will be to agree or to refuse it. If the user agrees with the argument, the user will have the chance to modify the publication or, in a worst case, to decide not to publish that content. On the other hand, if the user wishes to have more reasons, the second argument will be displayed and so on. Finally, if the user disagrees with the argument, it is also possible to publish the content without regarding the information provided by our argumentation system even though that action can start a privacy dispute in the OSN. It is important to guarantee that the users can make their decisions freely.

With this approach, when a privacy violation is detected, all users will have at least to read the first argument before making the publication. Then the user can decide to publish the content without reading more arguments or to disagree with the argument but keep reading the following available arguments. The idea is to be able to measure the effectiveness of our system approach by analysing how many users take into account the arguments provided and displayed by the system.

In this chapter, the argumentation system has been proposed and a fine-grained description of the system behaviour has been provided. This proposal has been published at the 19th Workshop on Computational Models of Natural Argument (19th CMNA) [32]. In the next chapter, we will explain the implementation process carried out to make our

argumentation system completely functional and the process followed in order to integrate our system with the PESEDIA social network.

CHAPTER 5
Implementation and Integration

5.1 Introduction

The main purpose of this chapter is to describe the implementation process carried out and the integration of the argumentation system into PESEDIA, an educational OSN.

This chapter is structured as follows. In Section 5.2 all the technologies used to carry out this work are presented. In Section 5.3 the implementation process of the argumentation system is depicted and in Section 5.4 the deployment process of the system is explained. In Section 5.5 we explain the development of the plugin that integrates the argumentation system with the social network. Finally, in Section 5.6 the communication between the system and the network, and the *dataflow* between both of them is explained.

5.2 Technologies

To fulfil the development of both argumentation system and plugin, several technologies have been used. The decision of using the following technologies has been made taking into account the context of this work, but also trying to be as straightforward as possible. Each of the technologies used is briefly described below.

- **Python**. This programming language is an interpreted high-level, dynamic typed and multiplatform programming language. One of its main characteristics is its readability due to the use of the own indentation as one of its syntax rules. Even though Python is an scripting language, it also supports object oriented programming, functional programming or procedural programming, among others. This language has also a huge collection of libraries and packages with many varied functions that make easier to start working on new projects. Python has been chosen as the main language to implement the argumentation system engine.
- **Numpy**. This is the main Python package for scientific calculus. NumPy provides us the main tools and technologies to work with huge matrices and to process data easily. It is mainly used for efficient vector operations in the argumentation system.
- **Flask**. This package makes possible to build web services. Flask is a web framework developed in Python that does not require any additional library. With Flask, we have deployed our argumentation system as a web service in order to make possible the communication with the social network and the exchange of messages.
- **Apache**. This technology provides the necessary tools to build HTTP servers. Apache is an open source project that makes possible to develop and maintain HTTP servers

in modern operating systems. With Apache, it has been possible to do the communication between the argumentation system and PESEDIA safely and complying with all the restrictions that have been found in the process.

- **Elgg** is an open-source social networking engine. It provides a framework to build any type of social network from scratch. The PESEDIA network has been developed using this framework. To integrate our argumentation system with this network, an Elgg plugin has been implemented.
- **PHP**. This programming language, originally designed for web development, is the main way to build and customise an Elgg social app. In this work, PHP is used to implement the Elgg plugin required to integrate the argumentation system with the OSN.
- **JavaScript** is an interpreted high-level programming language with many purposes, commonly used in the development of web applications. This multi-paradigm language supports event-driven, functional or imperative programming styles. In the context of this work, we have used JavaScript in the plugin development. Specifically, to add all the new functions to the OSN.
- **Git**. In order to have a proper version control of the project, Git has been used. The implementation of both system and plugin has been carried out by one person, despite this, the use of a version control system can be very useful to have a controlled register of the modifications made to the project.

As it can be appreciated above, we can clearly distinguish two main groups from the technologies depicted. The technologies used to implement the argumentation system (Python, NumPy and Flask) and the technologies used to develop the social network plugin (Elgg, PHP and JavaScript). The next sections give all the details about the process carried out to implement and integrate the argumentation system and how these technologies have been used for each purpose.

5.3 Argumentation System Implementation

The argumentation system has been completely implemented using Python as the programming language. In order to have a proper organisation and to make easier to understand the project for possible future modifications, the code has been structured with the Python modules described below and graphically depicted in Figure 5.1.

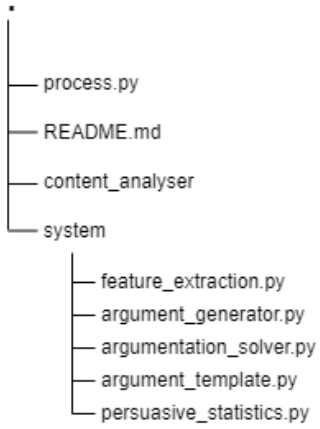

Figure 5.1: Argumentation system structure

where *process.py* is the web service file that makes the required calls to our system files.

5.3.1. Feature Extraction

The *feature_extraction.py* module provides all the required functions to obtain all the information from the social network. Specifically the functions to retrieve the post privacy configuration and content features, the author privacy preferences, the privacy risk score (PRS) and the content values defining authors content preferences.

In order to capture the content features of the publication, a *content analyser* is used. The content analyser integrated in the argumentation system has been developed by other researchers of the group as another part of the same project (TIN2017-89156-R). This content analyser makes possible to detect any of the 6 types of content defined in this work. Therefore, a 6 element vector is obtained indicating which type of content has been detected.

On the other hand, the values of author content preferences are also managed by this module. Initially, every user has the same value (1) towards each type of content. As posts are generated, the value regarding each type of content must be adjusted as depicted in Equation 5.1

$$value(t) = max(1 - \frac{n(t)}{total}, \epsilon) \tag{5.1}$$

where n is the number of publications containing some specific type t and $total$ is the total amount of publications made by the user. We decided to model the content preferences with this function since it is a way to have a proportional adjustment of each value regarding the total amount of publications made by the user. It is also important to emphasise that, with only the initial publications, the value regarding some specific type of content can be dropped to the minimum without having much information. This may happen if the user focuses all the initial publications on the same topic. To correctly handle this situation, we decided to smooth the initial decrements with a minimum ϵ higher than zero.

5.3.2. Argument Generator & Argumentation Solver

The implementation of both *argument_generator.py* and *argumentation_solver.py* modules has been quite straightforward regarding the definitions provided in the subsection 4.2.2 and in the subsection 4.2.3. The generator receives the data gathered by the feature extraction module and generates all the *computational* arguments, being each argument a list of three elements (*claim*, *type* and *score*). The argumentation solver receives the list of all arguments, aggregates all the scores and depending on the result of the aggregation determines if there has been a significant privacy violation (negative aggregation) or not (positive aggregation).

5.3.3. Argument Template

Assuming there has been a significant privacy violation regarding some specific publication, the main purpose of the *argument_template.py* module is to translate our *computational* arguments into a human readable shape following the criteria described in the last chapter. Therefore, this Pyhton module receives as input the set of acceptable arguments

given by the solver and, depending on the type of the argument, returns the pre-defined text template associated to that specific type.

5.3.4. Persuasive Statistics

Finally, once the direct interaction with the human user finishes, the *persuasive_statistics.py* module gathers and stores all the useful information from the argumentation process in order to be able to evaluate and improve our argumentation system in newer versions. Therefore, the data stored is the number of times that a user gets persuaded when starting an argumentation process, the arguments required to persuade the user and its order, and the number of times that the system fails to persuade a user. With all this information it is possible to observe not only the effectiveness of our system but also the effectiveness of each type of argument defined in our framework.

At this point it is important to remark that all the data gathered by this module is completely anonymous. In addition to all the data described before, an ID is also saved in order to have a reference to match data from different sources. Once the usage of the network finishes, it is not possible to reach any real names or personal data with this ID. As stated before, the only purpose of this ID is to be able to link data gathered by different modules in a coherent way.

Before registering to the PESEDIA network, all the data that is going to be gathered and used is clearly enumerated and the user must agree with it. Therefore, the data processed and stored by this module complies with all restrictions stated in the Spanish *"Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales."* (LOPD-GDD).

5.4 Argumentation System Deployment

In order to manage the communication of the argumentation system with PESEDIA, the system has been deployed as a web service. First of all, the system has been deployed using Flask, the Python library. It is also important to remark that the web service has been deployed in the same server as PESEDIA. Additionally, due to the nature of PESEDIA, our web service must be able to handle HTTPS requests, since all the data is encrypted following the HTTPS protocol. Therefore, all the required certificates are loaded and used by the web service.

On the other hand, since the web service is deployed in the same server as PESEDIA, we developed an Apache configuration file. This file has been developed as a proxy, to redirect the requests from the url in which the OSN is located to the port being listened by our web service. This step is very important to correctly send and receive requests, since due to the limitations of the UPV, it was not possible to easily open the desired ports of the server. It is also important to remark that the Apache configuration file developed also complies with the HTTPS protocol. Therefore, all the communication steps are secure, and all the data transferred has been encrypted following the HTTPS protocol.

5.5 Network Integration

The most important features of the implementation process of the argumentation system have been explained. However, the system must be integrated into a social network in order to take advantage of it. As it has already been mentioned in this work, the network

chosen to integrate the argumentation system is PESEDIA, an educational OSN. Since one of the main objectives of the network is to educate teenagers on privacy management and risks of OSNs, the use of the argumentation system developed in this work is ideal. With it, the users can have a reasoned feedback on their actions in the social network and justified recommendations on whether share or not some specific content. To make easier to understand the context of this section, a brief description of PESEDIA has been provided.

5.5.1. PESEDIA: An Educational OSN

PESEDIA [10] is a pedagogical OSN for educational and research purpose. Some of these purposes are the development of behaviour change methods that can improve privacy management from the user point of view, or the implementation of new functions to assist users with content management. Therefore, PESEDIA is an ideal context to carry out this work.

The social network was implemented using Elgg [14], an open source engine designed to build and create social environments. The resulting social network is quite similar to other social environments (e.g. Facebook). Figure 5.2 shows the architecture of PESEDIA. As it can be observed, PESEDIA architecture is divided into two main layers: the *Platform Layer* and the *User Layer*. The *Platform Layer* is the core of the architecture, it contains the *Social Network Services* that provide the social network basic functions, and the *Storage System*, which allows to store all the data generated with the usage of the OSN. On the other hand, the *User Layer* has the purpose of managing the data associated to each user profile. This data can be divided into three main categories, user information (e.g. profile items, publications), user contacts and user privacy settings (e.g. privacy policies, audience selectors).

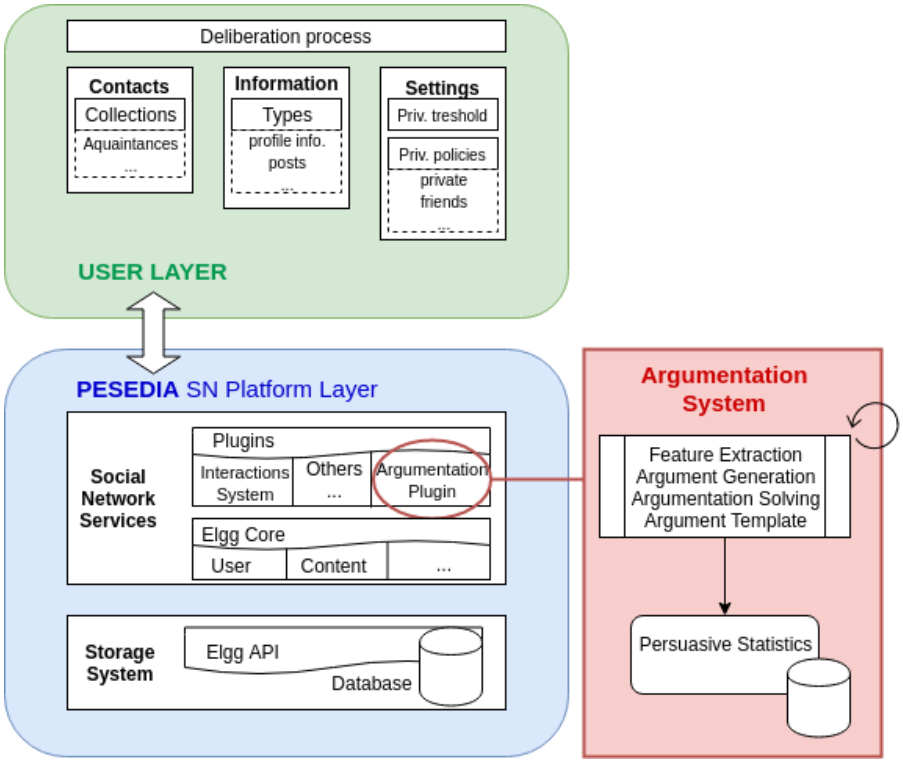

Figure 5.2: PESEDIA architecture diagram

5.5.2. Argumentation Plugin

The main way to modify, extend and add functions to an Elgg network is by the development of plugins. Therefore, in order to integrate our argumentation system in PESEDIA we have implemented an Elgg plugin that communicates with the system and makes possible the direct human-network interaction. Every plugin must follow some specific structure in order to be correctly loaded by the OSN. The structure of the plugin developed in this work can be seen in Figure 5.3.

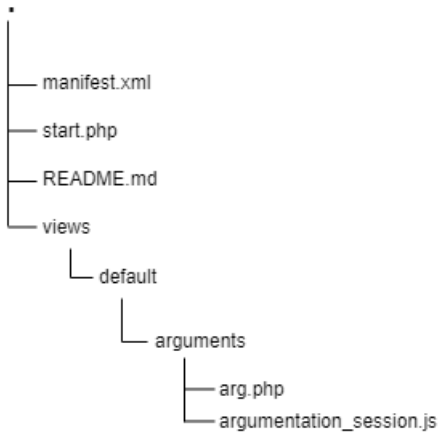

Figure 5.3: Plugin structure

The files located in the root directory *manifest.xml* and *start.php* are the ones that make possible to load our plugin and define its behaviour:

- ***manifest.xml*** contains the description of the plugin with the following mandatory fields: *id*, *name*, *author*, *version*, *description* and *requires*. These parameters will be displayed in the OSN configuration site and it is the main way to identify and be able to distinguish between the different plugins available.
- ***start.php*** contains all the initial calls made by the plugin. In our case, this file will extend the main view with the code located in */views/default/arguments*.

Thus, the main view will be modified with the new functions defined in both files *arg.php* and *argumentation_session.js*. These files make possible to display the arguments processed by the argumentation system to the user and to have some interaction with him/her.

- ***arg.php*** is the file that defines the extension of the main view. The main purpose of this file is to load *argumentation_session.js* and to gather the required information for the argumentation system. The information gathered by this file is the user ID, the prs and the user privacy preferences.
- ***argumentation_session.js*** is the code in charge of adapting the network interface to display arguments and making requests to the argumentation system when a user wants to share some content. It also gathers the lacking data required to generate arguments: the content of the publication and its privacy configuration.

As it can be appreciated, there must be communication between the argumentation system and the network plugin in order to have the system properly working. The purpose of the next section is to explain that communication process and give a complete overview of the system operation.

5.6 System Operation

When designing all the components of this work, one of the most important constraints to take into account was to have a fast and smooth communication between the argumentation system and the Elgg plugin. Since the system is intended to be used in a social network, it would make no sense to make the user wait for the results of the system. Therefore, the user experience must be similar to the use of the network without having the argumentation system integrated. The interaction process between OSN and argumentation system is graphically depicted in Figure 5.4.

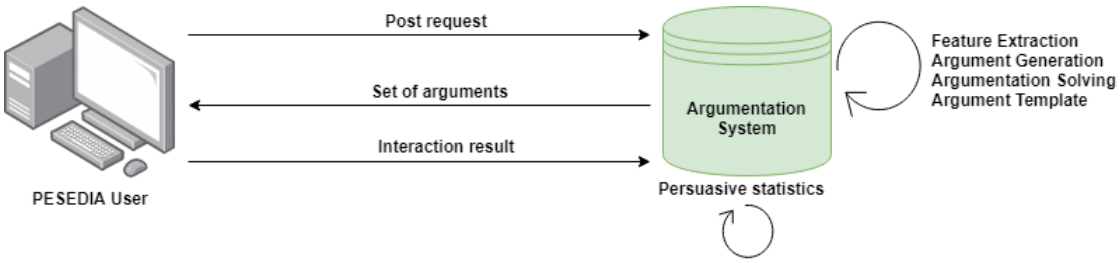

Figure 5.4: Diagram of the communication between the network and the argumentation system

The argumentation system web server will always be listening for requests. Then, every argumentation process will always be initialised by the plugin. For every user connected to the OSN, when trying to publish some content to the network (e.g. Figure 5.5), a request with all the gathered data is sent to the argumentation system web server. Once the list of readable arguments is generated, a response to the OSN with this information is sent. When receiving the response, the OSN displays the arguments to the user with a pop-up as the one shown in the Figure 5.6.

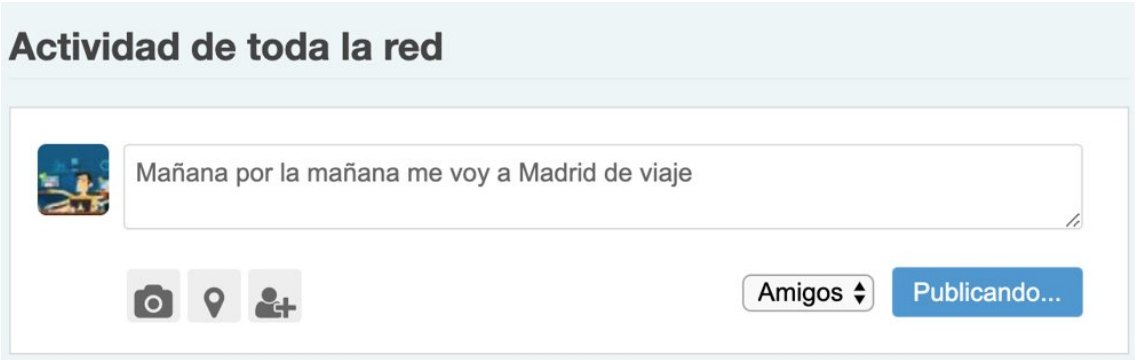

Figure 5.5: Example of post triggering an argument in the PESEDIA network

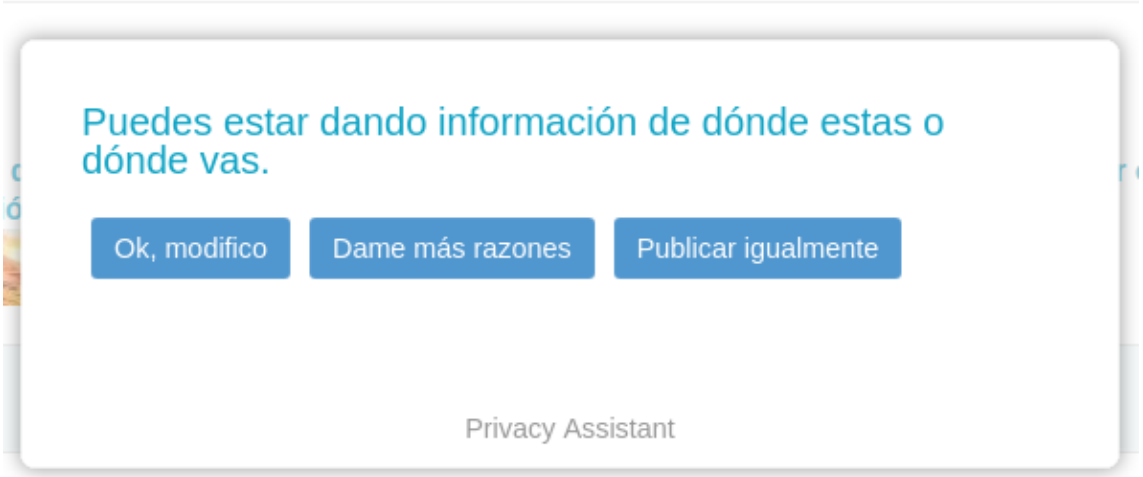

Figure 5.6: Pop-up displayed to the user with a content argument of location (*You can be revealing information about where you are or where you're going.*)

As illustrated in the figures, several options are given to the user following the user-network interaction protocol defined in Section 4.3. In the case of detecting a privacy violation, a list of arguments is generated. Therefore, the user can *accept* the argument by choosing to modify the content of the publication, *reject* the argument by publishing the content without taking into account the system arguments, and *request* another argument in the case of not being yet sure of what to do.

The interaction with the user finishes when either he/she modifies the content or publishes it without doing any change. Then, depending on the result of the interaction, another message is sent to the web server containing all the data regarding the interaction (e.g. the number of arguments needed to persuade the user, the type of the persuading argument). Finally, the argumentation system processes and stores all these data in order to be able to analyse it later on.

CHAPTER 6
System Evaluation

In this chapter, we present the different tests and evaluations made to validate the argumentation system proposed and its implementation. The chapter is divided into three main sections. Section 6.1 details the process carried out to analyse the performance of the argumentation system web service by varying the amount of simultaneous requests. In Section 6.2 it is explained all the functional validation performed in order to guarantee the correct behaviour of our system in multiple situations. Finally, Section 6.3 explains the persuasive evaluation defined to be carried out once having gathered all the usage results.

6.1 Argumentation System Stress Test

The purpose of the argumentation system stress test is to find out the maximum feasible load that our system can handle from a computational viewpoint. It is important to define the feasibility concept in order to understand the analysis performed in this section. The social network where the system is going to be integrated is an educational social network (PESEDIA) that will have a total population of no more than 200 users. In addition, the activity in the social network will be divided into batches. There will never be all the users registered in the network connected at the same time. There will be 3 different batches of 75 users each one. In order to carry out this stress test we have used JMeter[1], an open source application designed for testing Web Applications.

Taking the previous specifications into account, we performed different stress tests, each test with a different amount of requests at the same time. Each petition contains the same features. The chosen features trigger the feature extraction module so the system is forced to generate both positive and negative arguments. Figure 6.1 represents the amount of requests processed per second by our system. The red line represents an ideal throughput where all the requests received are processed in a second. It is possible to observe that only with 5, 10 and 15 requests at the same time we can approximate that ideal situation. When the system receives more than 20 requests at a time, it is possible to appreciate that the number of requests per second drops and gets stabilised around 12-13 pet/sec. Even though this may seem an inconvenient, it is also possible to observe that there is no significant difference between receiving 20 or 75 requests at a time. As we are making worst case assumptions for the stress test, it is possible to conclude that the system will handle smoothly all the requests received.

[1] https://jmeter.apache.org/

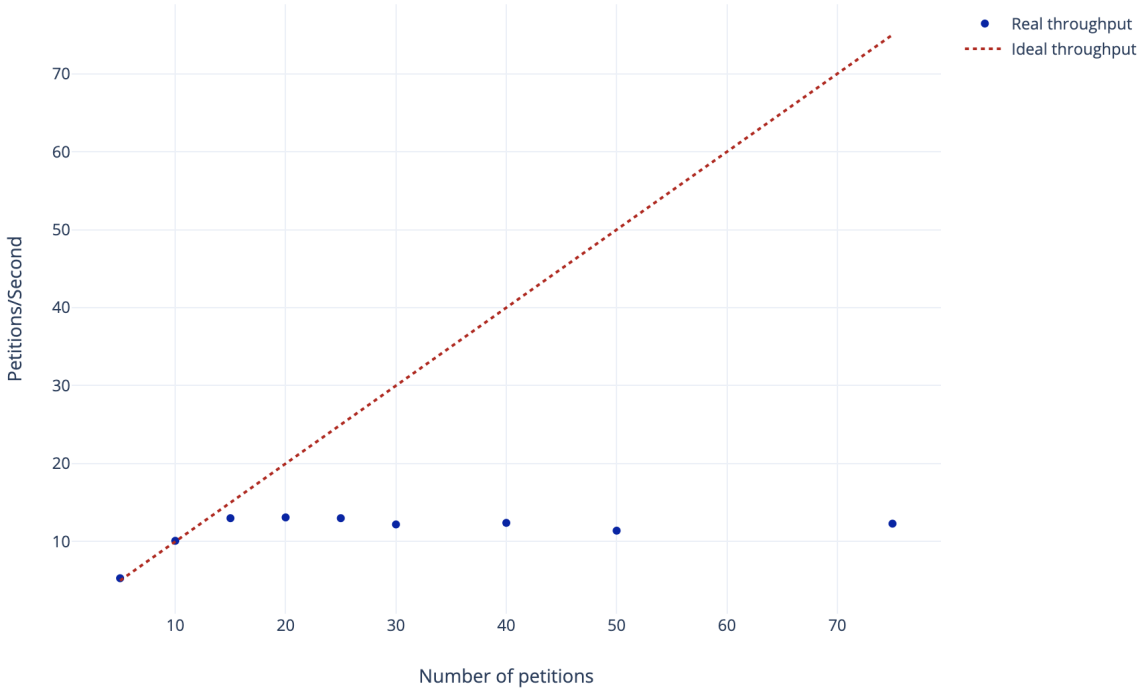

Figure 6.1: Requests per second handled by the argumentation system

Figure 6.2: Time required to solve N requests at a time

Another interesting graphical representation of the data retrieved from the stress tests can be observed in Figure 6.2. It graphically depicts the amount of time required to answer all the requests received at the same time. It is possible to appreciate how the time required to give an answer to 200 requests or less at a time increases moderately. However, with more than 200 requests, the time required to process all of them starts to increase exponentially. Anyway, focusing on the worst case that may happen when using

the network (75 requests at a time) the response time will never surpass 6 seconds. This is an acceptable wait time for our domain.

6.2 Functional Validation

Taking into account the nature of this work, it is very important to be sure that the system will behave as expected. In order to be able to evaluate the persuasiveness of the argumentation system proposed, we need to previously validate both generation and scoring of arguments of the own system.

The functional validation is divided into four main validation steps, one per each type of argument. For each validation step, we also defined different cases in order to test that the system is generating the expected arguments and scoring them following the strategy previously defined.

6.2.1. Validation of content arguments generation

The first validation step has been designed taking into account the content arguments generation. Since six different types of arguments have been defined in this work, in order to be able to see the behaviour of both argument generation and scoring, we decided to define a content value vector that represents the author where the values follow a decreasing relationship. Six different cases with this configuration have been designed. In each case a permutation over the content value preferences of the author is done. Additionally, we also considered the extreme cases where all the values are ones and zeroes as depicted in Table 6.1. For all the cases considered in this validation step, we assume that the same publication is going to be published. That publication contains all the possible types of content. Therefore, we expect the system to generate all the possible content arguments.

Taking into account the definitions provided in the Chapter 4, the arguments (*claim*, *type* and *support*) have been generated correctly considering the content values predefined. For instance, we can compare the data in the two first rows. It is possible to observe how, with a publication containing all the content types, when having the content values defined as input in the first row (1 to location, 0.9 to medical, 0.7 to drug, 0.5 to insults, 0.3 to relatives and 0.1 to personal) the system generates all the arguments scored with the result obtained by applying the score equation τ previously defined. When we apply a permutation on the values vector as in the second row respect to the first, the output behaves as expected by swapping the scores between the expected arguments (location with medical).

On the other hand, when looking at the extreme cases where all the values are the same either ones or zeroes, we can observe how the arguments are also generated and scored properly. Assuming the initial case when registering to an OSN (all values to one) all the arguments generated regarding the content features of the publication will have maximum score. If all the preference values are forced to be zero (this case will never happen in a real situation due to the smoothed decreasing of the conten values) none of the arguments will have any score different than zero.

For this first validation step, a second situation has been considered. In order to see that the system is correctly generating the arguments not only depending on the user content values but also depending on the types of content detected, we propose the following situations. As depicted in Table 6.2, different combinations have been defined. The first publication containing the three first types of content (location, medical and drug), the

Author: Content Values	Arguments (Score)
[1, 0.9, 0.7, 0.5, 0.3, 0.1]	[-1, 'Location', 1](-1.0), [-1, 'Medical', 1](-0.9), [-1, 'Drug', 1](-0.7), [-1, 'Insults', 1](-0.5), [-1, 'Relatives', 1](-0.3), [-1, 'Personal', 1](-0.1)
[0.9, 1, 0.7, 0.5, 0.3, 0.1]	[-1, 'Location', 1](-0.9), [-1, 'Medical', 1](-1.0), [-1, 'Drug', 1](-0.7), [-1, 'Insults', 1](-0.5), [-1, 'Relatives', 3](-0.3), [-1, 'Personal', 1](-0.1)
[0.9, 0.7, 1, 0.5, 0.3, 0.1]	[-1, 'Location', 1](-0.9), [-1, 'Medical', 1](-0.7), [-1, 'Drug', 1](-1.0), [-1, 'Insults', 1](-0.5), [-1, 'Relatives', 1](-0.3), [-1, 'Personal', 1](-0.1)
[0.9, 0.7, 0.5, 1, 0.3, 0.1]	[-1, 'Location', 1](-0.9), [-1, 'Medical', 1](-0.7), [-1, 'Drug', 1](-0.5), [-1, 'Insults', 1](-1.0), [-1, 'Relatives', 1](-0.3), [-1, 'Personal', 1](-0.1)
[0.9, 0.7, 0.5, 0.3, 1, 0.1]	[-1, 'Location', 1](-0.9), [-1, 'Medical', 1](-0.7), [-1, 'Drug', 1](-0.5), [-1, 'Insults', 1](-0.3), [-1, 'Relatives', 1](-1.0), [-1, 'Personal', 1](-0.1)
[0.9, 0.7, 0.5, 0.3, 1, 0.1]	[-1, 'Location', 1](-0.9), [-1, 'Medical', 1](-0.7), [-1, 'Drug', 1](-0.5), [-1, 'Insults', 1](-0.3), [-1, 'Relatives', 1](-0.1), [-1, 'Personal', 1](-1.0)
[1, 1, 1, 1, 1, 1]	[-1, 'Location', 1](-1.0), [-1, 'Medical', 1](-1.0), [-1, 'Drug', 1](-1.0), [-1, 'Insults', 1](-1.0), [-1, 'Relatives', 1](-1.0), [-1, 'Personal', 1](-1.0)
[0, 0, 0, 0, 0, 0]	[-1, 'Location', 1](0.0), [-1, 'Medical', 1](0.0), [-1, 'Drug', 1](0.0), [-1, 'Insults', 1](0.0), [-1, 'Relatives', 1](0.0), [-1, 'Personal', 1](0.0)

Table 6.1: Validation step 1 results (first part)

Publication: Content Detected	Arguments (Score)
[1, 1, 1, 0, 0, 0]	[-1, 'Location', 1](-1.0), [-1, 'Medical', 1](-1.0), [-1, 'Drug', 1](-1.0)
[0, 0, 0, 1, 1, 1]	[-1, 'Insults', 1](-1.0), [-1, 'Relatives', 1](-1.0), [-1, 'Personal', 1](-1.0)
[1, 0, 1, 0, 1, 0]	[-1, 'Location', 1](-1.0), [-1, 'Drug', 1](-1.0), [-1, 'Relatives', 1](-1.0)
[0, 1, 0, 1, 0, 1]	[-1, 'Medical', 1](-1.0), [-1, 'Insults', 1](-1.0), [-1, 'Personal', 1](-1.0)

Table 6.2: Validation step 1 results (second part)

second one containing the last three types of content (insults, relatives, personal) and the third and the last ones containing alternatively the different types of content. For this second situation, we consider the same author is making those publications. The author has all the content values settled to one.

Once again, it is possible to see how the correct arguments are generated. For this second situation, different arguments have been generated for each case. In every case the expected types of arguments have been generated with the maximum score each one. This is due to the author that has all the content values initialised to one.

6.2.2. Validation of privacy arguments generation

The purpose of the second validation step is to assure that the privacy arguments are generated correctly. Four different cases are defined in this step. In each case we assume that the user has a different privacy/popularity preference value: private (1), specific groups or collections of friends (0.75), friends (0.5) or public (0), towards his/her publications. We also considered different publications being shared. One publication configured to be shared with public settings (0), one publication being shared with friends only (0.5) and finally one publication shared in private settings (1). In Table 6.3 it is possible to observe the results obtained.

Author: Privacy Value	Publication: Privacy Setting	Arguments (Score)
1	0	[-1, 'Privacy', 1](-1.0)
0.75	0	[-1, 'Privacy', 0.75](-0.5625)
0.5	0	[-1, 'Privacy', 0.5](-0.25)
0	0	[+1, 'Privacy', 1](0.0)
1	0.5	[-1, 'Privacy', 0.5](-0.5)
0.75	0.5	[-1, 'Privacy', 0.25](-0.1875)
0.5	0.5	[+1, 'Privacy', 1](0.5)
0	0.5	[+1, 'Privacy', 1](0.0)
1	1	[+1, 'Privacy', 1](1.0)
0.75	1	[+1, 'Privacy', 1](0.75)
0.5	1	[+1, 'Privacy', 1](0.5)
0	1	[+1, 'Privacy', 1.0](0.0)

Table 6.3: Validation step 2 results

In all these situations we can observe that privacy arguments have been generated. It is also possible to appreciate how the arguments are scored. If the privacy of the publication is higher or equal than the user privacy/popularity preference, a positive argument

is generated. In any other case, a negative argument is generated and scored following the score function defined in this work. For example, the second row shows a situation in which the privacy configuration of the publication is lower than the privacy preference of the user. In this case, the support is computed as the distance between the value and the privacy setting (0.75). The score is the product of the support by the value (with the claim sign) as it can be observed (-0.5625). Therefore, both privacy arguments and their scores are generated as expected by the argumentation system.

6.2.3. Validation of risk arguments generation

The third validation step aims to check the generation of risk arguments. Five different cases have been defined for this validation step. Since the PRS is a user related parameter computed and collected directly from PESEDIA, a gradual variation over this value is taken into account. It is posible to appreciate the different configurations and results obtained for this test in Table 6.4.

Author: PRS	Arguments(Score)
0	[+1, 'Risk', 1](1.0)
0.1	[+1, 'Risk', 0.9](0.9)
0.4	[-1, 'Risk', 0.4](-0.4)
0.8	[-1, 'Risk', 0.8](-0.8)
1	[-1, 'Risk', 1](-1.0)

Table 6.4: Validation step 3 results

Here, it is important to remark that the score assigned to each argument is the same as its support. This is because we modeled all the users for this test having the maximum preference value (1) regarding its privacy configuration (the score is computed by multiplying the support by the privacy value). It is also interesting to observe how the first two cases generate a positive risk argument, when considering a positive argument, its support is the complementary to the PRS value up to one. This is due to the threshold defined. We consider any PRS over 0.2 a potential risk so, in order to minimise the *noise* in the argumentation solving process, if the PRS value is under 0.2 the argument generated is positive. Hence, the expected risk arguments are generated with their expected scores.

6.2.4. Validation of trust arguments generation

The last validation step tackles the trust arguments generation. Since two parameters are involved in the scoring process of trust arguments, we tested the system taking into account the values 1, 0.7, 0.4, 0 for the trust value and 1, 0.5, 0 for the closeness/openness preference value (value that represents the importance of trust involved interactions for a specific user). All possible combinations with these values have been tested. Therefore, twelve different cases that cover all the possibilities have been considered as represented in Table 6.5.

User tagged		Arguments (Score)
Trust towards author	Closeness/Openness value	
1	1	[+1, 'Trust', 1](1.0)
0.7	1	[+1, 'Trust', 0.7](0.7)
0.4	1	[-1, 'Trust', 0.6](-0.6)
0	1	[-1, 'Trust', 1](-1.0)
1	0.5	[+1, 'Trust', 1](0.5)
0.7	0.5	[+1, 'Trust', 0.7](0.35)
0.4	0.5	[-1, 'Trust', 0.6](-0.3)
0	0.5	[-1, 'Trust', 1](-0.5)
1	0	[+1, 'Trust', 1](0.0)
0.7	0	[+1, 'Trust', 0.7](0.0)
0.4	0	[-1, 'Trust', 0.6](0.0)
0	0	[-1, 'Trust', 1](0.0)

Table 6.5: Validation step 4 results

A couple interesting properties can be seen in this table. First of all, a trust threshold is defined. We consider that a tag in a publication may bother the user tagged if the trust between the tagged user towards the author (first column) is lower than 0.6. Therefore, in all the cases where the trust towards the author is higher than the threshold the arguments generated are positive. On the other hand, in all the cases where the trust value is lower than the threshold, a negative argument has been generated. Here it is important to focus on the score assigned to those arguments. Since the lowest trust is 0, we need to generate a negative argument with maximum score for that case depending also on the preference value of the user (e.g. if the tagged user does not care about being tagged in publications with other unknown people, the argument must not have a high score). In order to achieve that, as for the negative arguments we expect the lower the trust, the higher the score, the complementary value up to one is taken into account as support. Therefore, it is possible to observe how the system is generating the expected trust arguments with the correct scores following the properties previously defined.

6.3 Persuasive Evaluation

One of the main purposes of the system proposed and developed in this work is to be able to persuade human users on decision making. Therefore, it is important to measure the effectiveness of our system. In order to be able to evaluate how well does the system work with an specific user we define the Persuasion Ratio (PR) as depicted in Equation 6.1.

$$PR(u_i) = \frac{per(u_i)}{tot(u_i)} \tag{6.1}$$

Where $per(u_i)$ is the number of times a specific user u_i has been persuaded (i.e. when receiving a system argument, the user decided either to cancel or to modify the publication) and $tot(u_i)$ is the total amount of argumentation processes started with that user. It is also interesting to evaluate the system in a more global perspective. We define the Average Persuasion Ratio (APR) depicted in Equation 6.2 in order to perform a global evaluation taking into account the whole population of the network.

$$APR = \frac{\sum_i^{|U|} PR(u_i)}{|U|} \tag{6.2}$$

where U is the set of users of the OSN involved in, at least, one argumentation process. With this metric it is possible to know the success rate of our argumentation system on persuading human users. The system is currently being tested with human users. Therefore, we are currently processing these data to perform the evaluation defined in this section.

CHAPTER 7
Conclusions

It is possible to identify two different lines of work in this final master thesis, both research and development. Our research work resulted in a publication in one of the most relevant workshops of the computational argumentation research community (19th International Workshop on Computational Models of Natural Argument, CMNA[1]). Where the new argumentation framework and the structure of the system has been presented. Also an important development work has been carried out in order to implement all the ideas and concepts proposed in the paper.

In the first chapter, we defined all the objectives that must be completed along the process of carrying out this work. All those objectives have been achieved, as explained below.

A review over the most important concepts of computational argumentation and their implications in artificial intelligence has been performed (objective 1). The most important existing privacy management systems have also been compared and their main properties identified (objective 2). Therefore, a complete revision over the existing work on the main topics of this project has been performed. Once having acquired all the basic knowledge, it has been possible to start with our proposal.

All the requirements of this work have been identified, and a formal requirement specification has been performed (objective 3). With this specification, it has been possible to carry out this work with an organised planning. Having all the requirements identified, a new argumentation framework for dealing with potential privacy conflicts in online social networks has been proposed (objective 4). This framework has been designed in order to fulfil all of these requirements.

Finally, the implementation of the framework has been carried out. The argumentation system has also been implemented according to the requirements. After that, it has been integrated into a real OSN (objective 5). Once having the system working in a real context, it has been tested and validated in order to make sure that it will behave properly with human users activity (objective 6). A metric to evaluate the persuasiveness of the argumentation system has also been proposed (objective 7). However, it has not been possible to evaluate it yet since it is currently being used in an experiment with real users (July 2019) and the data has not already been processed.

To sum up, a completely functional argumentation system for assisting users with privacy management in OSNs has been proposed and implemented in this work covering all the objectives defined in Chapter 1. It is also possible to observe how many different fields of computer science (i.e. computational argumentation, artificial intelligence, programming, etc.) and other disciplines (i.e. persuasion, psychology, etc.) converge in

[1] http://www.cmna.info/CMNA19/

this work. The development of this system will make possible not only to assist human users with privacy management, but also to study and observe users behaviour when confronted of the arguments provided by our system.

CHAPTER 8
Future Work

This work provides a framework and a tool inside a real OSN that leaves many open doors to future work. After using the actual approach with human users, and based on the network usage, it will be possible to analyse user preferences towards some specific type of argument. The addition of new types of arguments can be also considered. It will be possible to observe the relevance of each feature taken into account, modify, or add features, in order to try to improve the performance of our approach. Finally, regarding network usage data it will also be possible to propose a new version of the system even more focused on an educational domain.

Nevertheless, other lines of research keep opened for future work. Argument detection is an interesting future line of work and research. Currently the system provides users with a close set of options that they can select as an answer. It would be a great improvement to have an open interaction with the human user. An important problem to tackle in this case is automatic argument detection [24, 23]. In order to interact in a coherent way with the human user, the system must be able to understand the answers given by the user. Since the system is giving arguments to the user, it is possible that the user answers the system with a counter argument. Therefore, the system must be able to detect the arguments written by the user and analyse the main components of an argument (i.e. claim, support, etc.).

Related to argument detection, and walking towards that open argumentation system, argument generation is another interesting line of work and research. Automatic argument generation is a state of the art research topic, many companies and researchers are working on it nowadays[1]. Therefore, it can be interesting to carry on the research started in this thesis by generating more complex arguments. First steps towards this challenge can be to consider the six persuasion prinicples identified by Cialdini [13] when generating arguments or to use different argumentation schemes. The final goal would be to build a system able to automatically detect and generate arguments for a specific context (e.g a chatbot).

Finally, a remaining field of research that keeps also open is the definition of persiuasion strategies. As stated before, once analysing the usage of the system it will be possible to extract relationships between types of argument and different user personalities. With this information, we will be able to propose a more complex strategy not only based on argument scores. There is a promising line of research that approaches the problem of defining dialogue strategies by the use of reinforcement learning [20, 26, 1]. This also paves the way for our future work in this line.

[1] https://www.research.ibm.com/artificial-intelligence/project-debater/

Bibliography

[1] Sultan Alahmari, Tommy Yuan, and Daniel Kudenko. Reinforcement learning for dialogue game based argumentation. In *Accepted of the 19th Workshop on Computational Models of Natural Argument (CMNA19)*, 2019.

[2] J Alemany, E del Val, J Alberola, and Ana García-Fornes. Estimation of privacy risk through centrality metrics. *Future Generation Computer Systems*, 82:63–76, 2018.

[3] Leila Amgoud, Jean-François Bonnefon, and Henri Prade. An argumentation-based approach to multiple criteria decision. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 269–280. Springer, 2005.

[4] Pietro Baroni and Massimiliano Giacomin. Semantics of abstract argument systems. In *Argumentation in artificial intelligence*, pages 25–44. Springer, 2009.

[5] Pietro Baroni, Antonio Rago, and Francesca Toni. How many properties do we need for gradual argumentation? AAAI, 2018.

[6] Trevor Bench-Capon. Value based argumentation frameworks. *arXiv preprint cs/0207059*, 2002.

[7] Trevor JM Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.

[8] Salem Benferhat, Didier Dubois, Souhila Kaci, and Henri Prade. Bipolar representation and fusion of preferences on the possibilistic logic framework. *KR*, 2:421–432, 2002.

[9] Andrei Bondarenko, Phan Minh Dung, Robert A Kowalski, and Francesca Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial intelligence*, 93(1-2):63–101, 1997.

[10] José Alemany Bordera. Pesedia. red social para concienciar en privacidad. 2016.

[11] Aylin Caliskan Islam, Jonathan Walsh, and Rachel Greenstadt. Privacy detective: Detecting private information and collective privacy behavior in a large social network. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 35–46. ACM, 2014.

[12] CarlosIván Chesñevar, Ana Gabriela Maguitman, and María Paula González. Empowering recommendation technologies through argumentation. In *Argumentation in artificial intelligence*, pages 403–422. Springer, 2009.

[13] Robert B Cialdini and Robert B Cialdini. *Influence: The psychology of persuasion*. Collins New York, 2007.

[14] Cash Costello. *Elgg 1.8 social networking*. Packt Publishing Ltd, 2012.

[15] Didier Dubois and Hélène Fargier. On the qualitative comparison of sets of positive and negative affects. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 305–316. Springer, 2005.

[16] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357, 1995.

[17] Paul E Dunne and Trevor JM Bench-Capon. Coherence in finite argument systems. *Artificial Intelligence*, 141(1-2):187–203, 2002.

[18] Paul E Dunne and Michael Wooldridge. Complexity of abstract argumentation. In *Argumentation in artificial intelligence*, pages 85–104. Springer, 2009.

[19] Ricard L Fogues, Pradeep Murukanniah, Jose M Such, Agustin Espinosa, Ana Garcia-Fornes, and Munindar Singh. Argumentation for multi-party privacy management. 2015.

[20] Kallirroi Georgila and David Traum. Reinforcement learning of argumentation dialogue policies in negotiation. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[21] Jennifer Golbeck, Cristina Robles, and Karen Turner. Predicting personality with social media. In *CHI'11 extended abstracts on human factors in computing systems*, pages 253–262. ACM, 2011.

[22] Nadin Kökciyan, Nefise Yaglikci, and Pinar Yolum. An argumentation approach for resolving privacy disputes in online social networks. *ACM Transactions on Internet Technology (TOIT)*, 17(3):27, 2017.

[23] Ran Levy, Ben Bogin, Shai Gretz, Ranit Aharonov, and Noam Slonim. Towards an argumentative content search engine using weak supervision. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2066–2081, 2018.

[24] Ran Levy, Shai Gretz, Benjamin Sznajder, Shay Hummel, Ranit Aharonov, and Noam Slonim. Unsupervised corpus–wide claim detection. In *Proceedings of the 4th Workshop on Argument Mining*, pages 79–84, 2017.

[25] Yavuz Mester, Nadin Kökciyan, and Pınar Yolum. Negotiating privacy constraints in online social networks. In *Advances in Social Computing and Multiagent Systems*, pages 112–129. Springer, 2015.

[26] Ariel Monteserin and Analía Amandi. A reinforcement learning approach to improve the argument selection effectiveness in argumentation-based negotiation. *Expert Systems with Applications*, 40(6):2182–2188, 2013.

[27] Donald Nute. Defeasible logic. In *International Conference on Applications of Prolog*, pages 151–169. Springer, 2001.

[28] Primal Pappachan, Roberto Yus, Prajit Kumar Das, Tim Finin, Eduardo Mena, Anupam Joshi, et al. A semantic context-aware privacy model for faceblock. In *Second International Workshop on Society, Privacy and the Semantic Web-Policy and Technology (PrivOn 2014), Riva del Garda (Italy)*, 2014.

[29] Henry Prakken and Giovanni Sartor. A dialectical model of assessing conflicting arguments in legal reasoning. In *Logical models of legal argumentation*, pages 175–211. Springer, 1996.

[30] Iyad Rahwan and Guillermo R Simari. *Argumentation in artificial intelligence*, volume 47. Springer, 2009.

[31] Sebastiaan Rothmann and Elize P Coetzer. The big five personality dimensions and job performance. *SA Journal of Industrial Psychology*, 29(1):68–74, 2003.

[32] Ramon Ruiz-Dolz, Stella Heras, José Alemany, and Ana García-Fornes. Towards an argumentation system for assisting users with privacy management in online social networks. In *Proceedings of the 19th Workshop on Computational Models of Natural Argument co-located with the 14th International Conference on Persuasive Technology, CMNA@PERSUASIVE 2019, Limassol, Cyprus, April 9, 2019.*, pages 17–28, 2019.

[33] Shalom H Schwartz. An overview of the schwartz theory of basic values. *Online readings in Psychology and Culture*, 2(1):11, 2012.

[34] Anna C Squicciarini, Heng Xu, and Xiaolong Zhang. Cope: Enabling collaborative privacy management in online social networks. *Journal of the American Society for Information Science and Technology*, 62(3):521–534, 2011.

[35] Kaveri Subrahmanyam, Stephanie M Reich, Natalia Waechter, and Guadalupe Espinoza. Online and offline social networks: Use of social networking sites by emerging adults. *Journal of applied developmental psychology*, 29(6):420–433, 2008.

[36] Jose M Such, Joel Porter, Sören Preibusch, and Adam Joinson. Photo privacy conflicts in social media: A large-scale empirical study. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 3821–3832. ACM, 2017.

[37] Rosemary J Thomas, Judith Masthoff, and Nir Oren. Is argumessage effective? a critical evaluation of the persuasive message generation system. In *International Conference on Persuasive Technology*, pages 87–99. Springer, 2019.

[38] D Walton. Argument schemes for presumptive reasoning. 1996, 1996.

[39] Douglas Walton. Argumentation theory: A very short introduction. In *Argumentation in artificial intelligence*, pages 1–22. Springer, 2009.

[40] Ryan Wishart, Domenico Corapi, Srdjan Marinovic, and Morris Sloman. Collaborative privacy policy authoring in a social networking context. In *2010 IEEE International Symposium on Policies for Distributed Systems and Networks*, pages 1–8. IEEE, 2010.