



José Manuel Perdigão Venâncio

BSc in Computer Science

A Tool for Online Debates

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering

Adviser: João Leite, Associate Professor,
Faculdade de Ciências e Tecnologia,
NOVA University of Lisbon

Co-advisers: Teresa Romão, Assistant Professor,
Faculdade de Ciências e Tecnologia,
NOVA University of Lisbon

João Leitão, Assistant Professor,
Faculdade de Ciências e Tecnologia,
NOVA University of Lisbon

Examination Committee

Chairperson: Prof. Doctor Ana Maria Diniz Moreira

Rapporteur: Prof. Doctor Paulo Miguel Torres Duarte Quaresma

Member: Prof. Doctor João Alexandre Carvalho Pinheiro Leite



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

February, 2017

A Tool for Online Debates

Copyright © José Manuel Perdigão Venâncio, Faculty of Sciences and Technology, NOVA University of Lisbon.

The Faculty of Sciences and Technology and the NOVA University of Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

To all my family, friends and colleagues.

ACKNOWLEDGEMENTS

I would like to firstly thank Prof. Dr. João Leite for all the support and knowledge given during the elaboration of this dissertation. I would also like to thank Prof. Dr. Teresa Romão and Prof. Dr. João Leitão for all the availability and knowledge that helped to accomplish my goals.

I would also like to thank all the participants of the tests, without them this work would not be possible.

At last (but not at least), I would like to thank my family and friends, especially my parents that gave me the possibility to study, and helped me throughout all my academic years.

ABSTRACT

Nowadays, internet is one of the most used vehicles to connect people. There are social networks such as Facebook, Twiter and Youtube, that allow the users to interact among them. Despite the existence of these social networks, the debates that arise in these platforms are often chaotic and unstructured.

The problem of unstructured or even chaotic debates was approached by Leite and Martins, they have studied some of the existing solutions on the web to tackle this problem, and proposed a model of Social Abstract Argumentation. This model defines a Social Abstract Argumentation Framework, based on arguments, attacks, and votes on arguments. The model was later extended to also consider votes on attacks.

The extension of Social Abstract Argumentation has many advantages that cannot be found on the existent systems over the internet. A tool based on this model has some important properties, such as, creating arguments and attacks without knowing any formal rules, reusing arguments, participate using only votes, and consult the debate outcome at any moment. All of these properties can be a major contribution to online debating.

In this work we have developed a web-application, based on the extension of Social Abstract Argumentation, as a proof of concept to validate this model.

The evaluation and validation of this tool is divided in three separate parts: Interface Usability, Semantics, and Algorithm performance. The results of this evaluation, provided clues to further improvements of the semantics and interface, and allowed us to validate the model of Social Abstract Argumentation.

Keywords: Online Debate, Abstract Argumentation Framework, Ranking-based Semantics for Abstract Argumentation, Social Abstract Argumentation

RESUMO

A internet é um dos principais veículos para ligar as pessoas. Existem redes sociais como o Facebook, o Twitter e o Youtube, que possibilitam a interação entre utilizadores. Apesar da existência destas redes sociais, os debates que surgem nestas plataformas são frequentemente caóticos e destrutturados.

O problema dos debates destrutturados ou até mesmo caóticos, foi abordado por Leite e Martins, que estudaram algumas das soluções existentes na web para combater este problema, e propuseram um modelo de Argumentação Abstrata Social. Este modelo define uma Framework de Argumentação Abstrata Social, tendo por base argumentos, ataques e votos em argumentos. Mais tarde, este modelo foi estendido de forma a considerar votos nos ataques.

A extensão da Argumentação Abstrata Social tem várias vantagens sobre alguns dos sistemas existentes na internet. Uma ferramenta baseada neste modelo possui algumas propriedades importantes, como a criação argumentos e ataques sem ter de conhecer quaisquer regras formais, a reutilização de argumentos, a participação utilizando apenas votos, e a consulta do resultado do debate em qualquer momento. Todas estas propriedades podem ser uma importante contribuição para o debate online.

Neste trabalho, foi desenvolvida uma aplicação web baseada na extensão da Argumentação Abstrata Social, como prova de conceito para a validação deste modelo.

A avaliação e validação desta ferramenta está dividida em três partes distintas: Usabilidade da Interface, Semântica e Performance do Algoritmo. Os resultados desta avaliação, facultaram-nos pistas para melhorias futuras da semântica e da interface, permitindo-nos, desta forma, validar o modelo da Argumentação Abstrata Social.

Palavras-chave: Debate Online, Framework de Argumentação Abstracta, Semânticas baseadas em ranking para Argumentação Abstracta, Argumentação Abstracta Social

CONTENTS

List of Figures	xv
List of Tables	xvii
Listings	xix
1 Introduction	1
1.1 Motivation and Context	1
1.2 Objectives	5
1.3 Contributions and Document Structure	6
2 State of the Art	7
2.1 Semantics for Abstract Argumentation	7
2.1.1 Dung’s Abstract Argumentation Framework	7
2.1.2 Ranking-based Semantics for Abstract Argumentation	9
2.1.3 Discussion	17
2.2 Existing argumentation/debate applications	18
2.2.1 Abstract Argumentation Systems	18
2.2.2 Structured Argumentation Systems	20
2.2.3 Discussion	21
3 Proposed Solution	25
3.1 Proposal	25
3.2 Project Requirements	25
3.2.1 Functional Requirements	26
3.2.2 Non-functional Requirements	26
3.3 System Architecture	26
3.3.1 Presentation Layer.	26
3.3.2 Requests to the Application Layer	32
3.3.3 Application Layer	33
3.3.4 Storage/Data Layer	38
3.3.5 Security	40
3.4 Summary	42

4	Evaluation and Results	43
4.1	Interface Usability Evaluation and Results	43
4.1.1	Test setup	43
4.1.2	Results	46
4.1.3	Discussion and Analysis	48
4.2	Semantics Evaluation and Results	50
4.2.1	Test Setup	51
4.2.2	Results	53
4.2.3	Discussion and Analysis	58
4.3	Algorithm Performance Evaluation and Results	62
4.3.1	Test Setup	62
4.3.2	Results	63
4.3.3	Discussion and Analysis	64
4.4	Summary	66
5	Conclusion	67
5.1	Future Work	68
	Bibliography	71
A	Usability Evaluation	75
B	Semantics Evaluation	85

LIST OF FIGURES

1.1	Social Abstract Argumentation Framework example	4
2.1	Graph Representation of a Social Abstract Argumentation Framework	8
2.2	Consider.it	19
2.3	CreateDebate	19
2.4	Debategraph interface	20
2.5	MindMeister interface	20
2.6	Structured Argumentation Systems: Agora-net	21
3.1	Three-tier architecture	27
3.2	User Interface Overview	28
3.3	Argument example	29
3.4	Relation between colors and acceptance	30
3.5	Attack example	31
3.6	Filters example	31
3.7	Top menu	32
3.8	Right click pop up menu	32
3.9	Adapter scheme	37
3.10	Neo4J Database Schema	39
3.11	Login procedure	41
4.1	Add argument button	49
4.2	Question 16	55
4.3	Question 18	56
4.4	Cycle of attacks between an argument A and B with the same votes on arguments but different number of votes on attacks	61
4.5	Performance of the algorithm as function of the size and density of the graphs	64
4.6	Algorithm performance before (blue) and after modifications (orange)	65
5.1	Comparison between the scores before and after the scores start at 50 (i.e. 10 positive votes and 10 negative votes)	69

LIST OF TABLES

2.1	The properties that take into account the Argument Strength and Attack Strength are marked with ✓ in the respective column, otherwise they are marked with ✗. (<i>nn</i> - the attack strength must not be null).	11
2.2	Results for SAA Extended	13
2.3	Discussion-based semantics	15
2.4	Burden-based semantics	15
2.5	Ranking Results Comparison	17
4.1	Summary of task easiness questionnaire results. The highest scores are highlighted.	46
4.2	Statements that composed the general evaluation part of the questionnaire.	48
4.3	Summary general evaluation questionnaire results. The highest scores are highlighted.	48
4.4	Results to the questions 7 and 8 of the questionnaire.	54
4.5	Results to the question 11 of the questionnaire.	54
4.6	Results to the question 9: Reasons to positive votes	54
4.7	Results to the question 10: Reasons to negative votes	54
4.8	Results to questions 12 and 13 and comparison with the scores assigned by the application. V^+ - Positive Votes, V^- - Negative Votes, VH - Very High, H - High, M - Medium, L - Low, VL - Very Low	55
4.9	Results to questions 14 and 15 and comparison with the scores assigned by the application. V^+ - Positive Votes, V^- - Negative Votes, VH - Very High, H - High, M - Medium, L - Low, VL - Very Low	56
4.10	Results to questions 22 to 25. A^+ - Positive Votes of argument A, A^- - Negative Votes of argument A, B^+ - Positive Votes of argument B, B^- - Negative Votes of argument B	57
4.11	Algorithm Performance Evaluation: Test Subsets	63
4.12	Average runtime improvement for graphs	65
5.1	Scores for the arguments on Figure 5.1	69

LISTINGS

2.1	Pseudocode of the ISS algorithm	14
3.1	Example of an Ajax request - Connecting two arguments	33
3.2	Example of parsing the results of a request - Get all nodes from a graph	33
3.3	Example of JSON returned by graph_id	34
3.4	Method: getAllnodesFromGraph(int id)	37
3.5	Adapted ISS algorithm Java implementation	38
3.6	Secure password generation	41

INTRODUCTION

1.1 Motivation and Context

The word *debate* is often used to define a discussion between two or more individuals. Every time two or more people gather around they tend to start a debate. However, in some countries, such as the United States of America, debate can be considered to be a sport and, as any other sports, it has its own rules. Depending on these rules, the debate can be classified in many different categories, from Judicial Debate or Parliamentary Debate ruled by the court of law and the parliamentary procedure, respectively, to Academic Debate that usually has its own rules depending on the institution or contest where it takes place [20]. All of these categories of debates have a common denominator, they all have intrinsic rules for the debate. There is also another category designated as Nonformal Debate. The Nonformal Debate is the only form of debate that is conducted without formal rules found in the categories mentioned before [20]. Although this category does not rely on formal rules, it is probably the most used form of debate. This is the kind of debate that one can have in a bar with their friends when arguing about the best beer in the table.

Since this form of debate is so simple, it also emerges naturally in society. Nonformal debate occurs in every kind of media platforms, such as television and radio, and also over the Web. Most of the newspapers websites provide a comment area associated to each article, where the users can express their opinion about the subject. The same model can be seen in social networks (e.g Facebook, Twitter and Youtube) where a user can freely comment on every post of the network. Despite of the freedom given to the users, these kinds of models usually lead to debates that easily deviate from the main topic, considering that not all of the users are serious and some of them just want to express their emotions, despite the main topic or main arguments. Additionally, in the platforms

mentioned before, the way the debate is presented is often based on a timeline where, at the top, the most recent comments appear.

The problem of unstructured or even chaotic debates was approached by Leite and Martins, in [23]. In this article, the authors envisioned a self-managing online debate system where both experts and non expert users can participate on a debate. Furthermore, this system would be able to autonomously maintain a formal outcome to debates. In order to produce this outcome, the system would assign a strength to each argument based on the votes and the attacks between arguments.

For this purpose, they have firstly studied some of the existing solutions on the web to tackle this problem and proposed a model of Social Abstract Argumentation, that we will explain later on this chapter. The authors found some websites where the users can participate in structured debates. Some of these systems were designed to a more serious debate, to ground a thesis, or to share different points of view and ideas. The websites, createdebate.com and agora.gatech.edu, are some of the examples that we can find over the web. In some of these platforms the debate may have some formal rules that are enforced, and mechanisms to restrain the user participation in the debate (e.g. the user may participate but has to formally define every relation between all the parts of the argument). There are also other debate tools, that are less formal, where a user can add new arguments to one or both sides of the thesis being debated, as well as support or dispute other arguments.

Despite the merits of these platforms, the authors considered that these websites have some characteristics that may limit their adoption in the Social Web, namely [17, 23]:

1. In some of these tools, only two antagonistic users can engage in a debate, others can vote for the winning side, but not on concrete arguments.
2. The debate structure is sometimes very rigid, with a pre-fixed number of rounds and strict debate rules not known by most users.
3. Only a few of these tools provide facilities to reuse arguments (e.g. consider.it [12]) and debates.
4. Most of the times the debate stops short of reasoning with the debate data and votes/opinions, yielding to simplistic and naïve outcomes.

In order to tackle this problem and contribute to richer interactions in the form of debates over the Social Web, the authors introduced the model of Social Abstract Argumentation. Abstract Argumentation Frameworks defined by Dung in [15], are defined by a set of abstract arguments, and a set of attacks between arguments. An abstract argument does not have any internal structure or specific interpretation. Thus, abstract arguments can provide great flexibility when specifying arguments. However, this flexibility may lead to statements that are not structured arguments. Social Abstract Argumentation

benefits for these flexibility on the arguments specification, while providing tools to evaluate these arguments, through social support (voting) on the arguments. Moreover the extension of Social Abstract Argumentation, introduced votes on attacks that can be used to evaluate their logical foundation, and also to avoid that senseless attacks have impact on the debate outcome.

The graph in figure 1.1 illustrates an example of a Social Abstract Argumentation Framework. In this graph, each node represents an argument and each edge represents an attack between arguments, both of them have the values of positive and negative votes associated to them.

The main topic of the debate illustrated in figure 1.1, is the benefits of using e-cigarettes over regular cigarettes ¹. Each node stands for an argument and has its respective votes, each edge stands for an attack relation and also has its respective votes. For example, the argument f attacks argument a , since the argument a states that the use of e-cigarettes can be safer than the use of regular cigarettes, and the argument f points out a study about people that got seriously hurt with the batteries explosions of e-cigarettes. Considering this specific case, it is clear that argument a safety was related to the smoke and not the technology. Therefore, the attack between f and a , has more negative votes than positive votes. This means that, although there are many users that agree with the argument f (57 positive votes and 12 negative votes), a great part of the users do not agree with the attack between f and a . Consequently, the impact of argument f on the acceptance of a is reduced, due to the social support of the attack. In conclusion, votes on attacks can be an important factor to avoid that senseless attacks have a great impact in the debate outcome, that without votes on attacks would not be achievable.

A tool based on the extension of Social Abstract Argumentation , as it was described in [17], would have properties that can be a major contribution to the online debate in the social web.

- The debate is not restricted to only two sides, a tool based on this model would allow the users to create an argument at any time, without choosing a side.
- An argument does not have to follow specific formal rules. This will allow any user (independently of their knowledge about argumentation) to participate in the debate.
- The arguments can be reused. For instance, recalling the debate in Figure 1.1, if a new argument stating that: *"E-cigarettes can help someone to stop smoking, but they only work for a limited time period"* was created, argument b could be used to attack this new argument.
- The users that are not interested in proposing new arguments can also be an active and important part of the debate, voting on arguments and attacks.

¹The content of the arguments were retrieved and adapted from the ARG-tech Argument Mining datasets [3]. The votes on both arguments and attacks were introduced manually.

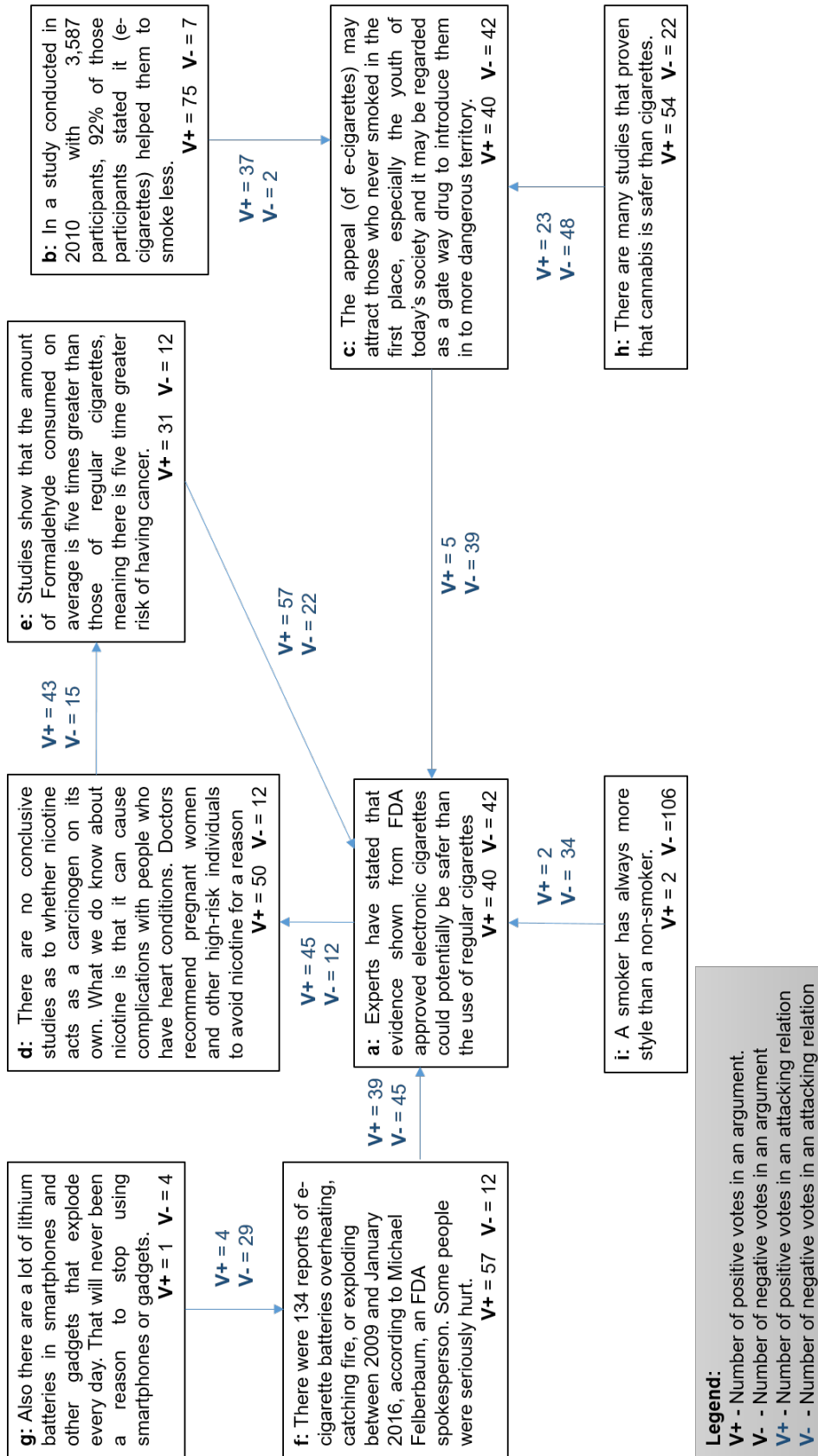


Figure 1.1: Social Abstract Argumentation Framework example

- Since every argument has its acceptance value, the actual outcome of a debate can be consulted at any moment.
- Since every attack has its acceptance value, senseless attacks will not have a great impact on the debate outcome if their acceptance is low.

In order to develop a tool based on Social Abstract Argumentation, an efficient algorithm for Social Abstract Argumentation (without votes on attacks) was proposed in [13]. This algorithm was tested for complex debates with thousands of arguments, and resulted on runtimes that make it suitable for an online debate tool.

Although there is a lot of work already done concerning Social Abstract Argumentation, this model has not ever been tested in the real world. Since this model was intended to tackle a problem in the social web, there is a necessity of validating and test a tool that implements Social Abstract Argumentation, that can be used by real users. In this dissertation we propose to develop a tool that will implement and validate the extension of Social Abstract Argumentation (with votes on attacks).

1.2 Objectives

The main goal of the work discussed in this document is to develop a web-application that will allow the users to debate. Furthermore, this web-application will be used as a proof of concept in order to validate Social Abstract Argumentation. There are some important properties that our web-application must have, that were already defined in [17, 23]:

- The users must be able to participate in several different ways (i.e. create an argument, create an attack, vote positively and negatively both on arguments and attacks).
- The user must receive the appropriate feedback. A user should easily assess the strength of each argument and attack.

In order to, develop this kind of application we must certify that it follows some unavoidable principles in a distributed system:

Scalability. The system must be able to deal with an increasing number of users and arguments and maintain its performance.

Security. The system must be able to keep the integrity and confidentiality of the data.

Performance. The system should run the valuation algorithms in a small amount of time, therefore both the algorithms and the system have to be optimized.

Also we need to provide the user a system that is both simple and complete:

Learnability. The user should be able to learn how to use the system fast. [29]

Efficiency. An experienced user should be able to achieve a high level of productivity. A user should be able to complete a task in the appropriate period of time, and without much effort.[29]

Memorability. The system should be easy to remember after a short or long period without using it.[29]

Satisfaction. The system should be pleasant to use.[29]

1.3 Contributions and Document Structure

Abstract Argumentation is one of the core studies in Artificial Intelligence. Since 1995 a lot of work has been done in this field of study.

With the work that was conducted in the context of this MSc dissertation, we built and evaluated a prototype that can serve as a proof of concept for Social Abstract Argumentation and also a tool to showcase this platform and test its acceptance among the users.

The remainder of this document is structured as follows:

Chapter 2 - State of the Art. This chapter is divided into two different sections. In the first section we will describe different models to approach our problem. In the second section we will discuss the different categories of the existing applications.

Chapter 3 - Proposed Solution. In this chapter we describe our solution, starting by explaining the project requirements and then discussing the system architecture, focusing on the implementation and the options that we made along the development process.

Chapter 4 - Evaluation and Results. In this chapter we describe the methods use to evaluate our system and present the results. Lastly the results are discussed and analyzed.

Chapter 5 - Conclusion. In this chapter we present our conclusions that were taken from the elaboration of this dissertation. Lastly we discuss the future work.

STATE OF THE ART

In this chapter we will start by describing the existing ranking-based semantics for Abstract Argumentation. Afterwards, we will discuss the existing categories of applications that may address our problem and compare them to Social Abstract Argumentation.

2.1 Semantics for Abstract Argumentation

Due to the necessities that were pointed out in the previous chapter, we are going to explore Semantics for Abstract Argumentation to tackle the problems specified before.

It is important to note that throughout this dissertation the definition of argument will be as follows: An argument is only a comment, a piece of information, an image, a video or any other content. The argument does not have to be structured nor meaningful, any kind of intervention can be considered an argument.

In this section, we will start by explaining what is an Abstract Argumentation Framework — using the model proposed by Dung back in 1995 [15] — and also *ranking-based argumentation semantics*. Afterwards we will explore some other relevant Abstract Argumentation semantics and their properties.

2.1.1 Dung’s Abstract Argumentation Framework

All the definitions throughout this section were first proposed and described in [15].

Definition 2.1. An Argumentation Framework is a pair

$$AF = \langle A, R \rangle,$$

where A is a set of arguments and R is a binary relation in A . The relation $R \subseteq A \times A$ defines an attack within A .

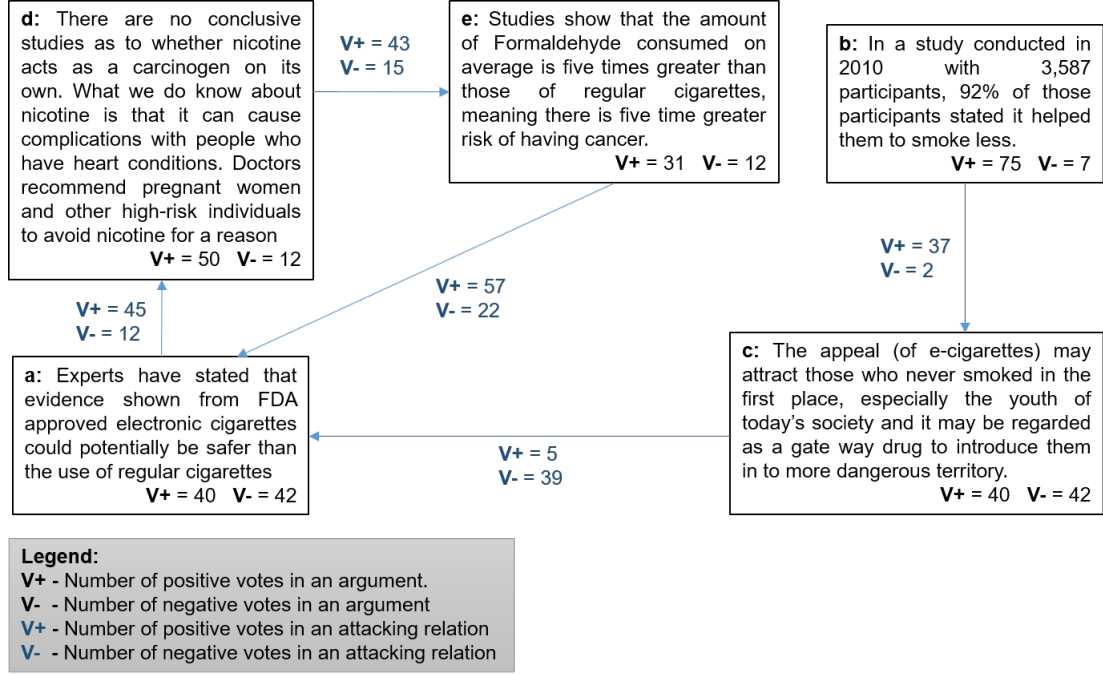


Figure 2.1: Graph Representation of a Social Abstract Argumentation Framework

An Abstract Argumentation Framework can be represented by a directed graph where the nodes stand for arguments and the edges represent attacks. If some argument b attacks an argument c , where $b, c \in A$ and $(b, c) \in R$, we will have an edge directed from b to c (Figure 2.1). The AF represented in figure 2.1 (ignoring the votes) is defined as follows: $AF_{2.1} = \langle \{a, b, c, d, e\}, \{(a, d), (c, a), (d, e), (e, a), (b, c)\} \rangle$

Definition 2.2. A set S of arguments is said to be conflict-free if:

$$\forall a, b \in S : \neg \exists (a, b) \in R$$

Definition 2.3.

1. An argument $a \in A$ is acceptable to (defended by [4]) a set S if and only if:

$$\forall b \in A : (b, a) \in R \Rightarrow \exists x \in S : (x, b) \in R$$

2. A conflict-free set of arguments S is admissible if and only if:

$$\forall a \in S : a \text{ is acceptable with respect to } S$$

Definition 2.4. A preferred extension of an argumentation framework AF is the maximal admissible set (with respect to set inclusion) of AF .

The notion of preferred extension is used to define the semantics of an argumentation framework.

Definition 2.5. Let S be a conflict-free set of arguments. S is called *stable extension* if and only if S attacks every argument that does not belong to S (i.e. $\forall a \in A \setminus S, \exists b \in S : (b, a) \in R$)

Definition 2.6. A set S of arguments is called a *complete extension* if and only if each argument, acceptable with respect to S , belongs to S .

Definition 2.7. Two Argumentation Frameworks $AF_1 = \langle A_1, R_1 \rangle$ and $AF_2 = \langle A_2, R_2 \rangle$ are isomorphic if and only if there is a bijective mapping $m : A_1 \rightarrow A_2$ such that:

$$(a, b) \in R_1 \Leftrightarrow (m(a), m(b)) \in R_2$$

2.1.2 Ranking-based Semantics for Abstract Argumentation

In this section a set of Ranking-based semantics for Abstract Argumentation will be compared. The ranking-based semantics classify each argument with a large range of levels of acceptability [7], producing a unique ranking between the arguments. As we imagine a platform meant for Social Abstract Argumentation, we envision a debate as a graph where nodes represent arguments and edges represent attacks. Both, edges and nodes, have negative and positive votes associated (see figure 2.1). The strength or acceptance of an argument will be defined by the relations and the votes only. The strength of an attack will be solely defined by its votes.

First, we will need to define some properties, used in [2, 7, 23], in order to compare the different semantics. Since this properties were not defined to a debate as we envision, we will also mention what it is needed to be changed in order to use each property. Throughout this section we will define the direct attackers of a as $R^-(a)$ and the defenders (i.e. an attacker of an attacker of a) as $R^+(a)$.

Abstraction. *The ranking on A should be defined on the basis of attacks between arguments.* This is a relevant property, however we must add that the ranking depends not on the content of the argument but on its attacks and votes.

Independence. *The ranking between two arguments $a, b \in A$ should be independent of any other argument $c \in A$, if c is not attacked/attacks neither a nor b .*

Void Precedence. *An argument that has not been attacked is ranked strictly higher than one which was attacked.* This is a relevant property, however it does not consider the number of votes of an argument. So it only stands if the strength of both arguments is the same and if at least one of the attacks' strength is not null.

Self-Contradiction. *An argument that attacks itself is ranked strictly lower than any other argument that does not attack itself.* This property can only be verified if the strength of all the attacks is the same, as well as the proportion of positive and negative votes on the arguments.

Cardinality Precedence. *The greater the number of direct attackers of an argument, the weaker the level of acceptability it has.* We need to reformulate this property, in order to apply it to our concrete use case. If we have two arguments a, b where a is being attacked by n arguments and b by m , and $m > n$, a has a better level of acceptability than b , if and only if the strength of the attack relations and the attackers is equal for a and b , we also have to consider that both arguments, a and b have the same proportion of positive and negative votes.

Quality Precedence. *The greater the acceptability of an direct attacker of an argument, the weaker the level of acceptability of the argument.* This property does not consider the strength of the attack relation.

Counter-Transitivity. *If the direct attackers of a are at least as numerous and acceptable as those of b , then b is at least as acceptable as a .* This property does not consider the strength of the attack relation, therefore it is only relevant if we consider that every attack in the example has the same strength.

Strict Counter-Transitivity. *If the direct attackers of a are strictly more numerous or acceptable than those of b , then b is strictly more acceptable than a .* This property does not consider the strength of the attack relation, therefore it is only relevant if we consider that every attack in the example have the same strength.

Defend Precedence *For two arguments $a, b \in A$ with the same number of direct attackers, if a is defended and b non-defended, then a will have a higher ranking than b .* This property does not consider the strength of the attack relation, therefore it is only relevant if we consider that every attack in the example have the same strength.

Distributed-Defend Precedence *The best defense is when each defender attacks a distinct attacker.* This property is relevant if we consider that all the attack relations strengths are not null.

Partial. *All pairs of arguments can be compared.* This property is relevant since, in order to make a ranking it is mandatory that every two arguments can be directly compared, even if they are not directly related. The fact that two arguments can be directly compared does not mean that they cannot be equal (i.e. with the same acceptance).

Non-attacked equivalence. *All the non-attacked arguments must have the same rank.* This is a relevant property because, this property must be fulfilled, in order to keep a meaningful feedback to the user.

Other properties:

Properties related to adding/increasing an attack/defense branch. *An attack branch on a is a branch of arguments attacking its sibling where a is the leaf of the branch and the number of arguments is odd. A defense branch on a only differs to the attack branch on the total number of arguments, in this case we have an odd number of arguments in the branch. These properties are not relevant to the solution since they do not take into account the weight of the attack (defined by votes).*

Table 2.1 sums up the aspects that are not considered in the properties presented previously, that are relevant to our problem.

	Argument Strength	Attack Strength
Abstraction	✗	✓
Independence	✓	✓
Void Precedence	✗	✓ ⁿⁿ
Self-Contradiction	✗	✗
Cardinality Precedence	✗	✗
Quality Precedence	✓	✗
Counter-Transitivity	✓	✗
Strict Counter-Transitivity	✓	✗
Defend Precedence	✗	✗
Distributed-Defend Precedence	✓	✓ ⁿⁿ
Partial	✓	✓
Non-attacked equivalence	✗	✓

Table 2.1: The properties that take into account the Argument Strength and Attack Strength are marked with ✓ in the respective column, otherwise they are marked with ✗. (*nn* - the attack strength must not be null).

A system, as it was envisioned in [23], must have some of the properties presented previously, however adapted to our specific problem. Therefore our system must guarantee the property of **Abstraction**, **Partial**, and **Independence**, considering that the property of abstraction was redefined to consider the votes on the arguments. The properties of **Void Precedence**, **Cardinality Precedence** and **Defend** and **Distributed-Defend Precedence** only make sense for arguments and attacks with the same strength and non null. Only in these concrete cases these properties can be verified. The properties **Quality Precedence**, **Counter-transitivity**, and **Strict Counter-Transitivity** can only be verified when the strength of all the attacks is the same and different from 0. The property **Self-Contradiction** cannot be considered since they do not take into account the arguments' votes and the attacks' strength.

To sum up, the system semantics should guarantee that:

- The ranking between the arguments is defined by the attacks, votes on the arguments, and votes on attacks (adaptation of **Abstraction**).

- The ranking between two arguments, a and b , is independent of any other argument that does not attack or is attacked by a or b (**Independence**).
- All the arguments with the same number of positive and negative votes that are not attacked, must have the same rank (adaptation of **Non-attacked equivalence**).
- All pairs of arguments can be compared. (**Partial**).
- If two arguments, a and b , have the same strength and a attacks b , the attacker (a), must be ranked strictly higher than the attacked (b), if the attack's strength is not null (adapted from **Void Precedence**).
- If an argument a attacks an argument b and the attack strength is null, the acceptability of argument b must not change.
- If an argument a attacks an argument b and the strength of argument a is null the acceptability of argument b must not change.

2.1.2.1 Social Abstract Argumentation Framework

The Social Abstract Argumentation Framework[13, 17, 23] is an extension of Dung's AAF since it adds votes to the previous definition. The Social Abstract Argumentation Framework was first proposed considering only votes on the arguments. Later an extended version of the Social Abstract Argumentation Framework introduced the votes on attacks. The following definitions are related to the extension to the Social Abstract Argumentation Framework.

Definition 2.8. Let AF be a Social Abstract Argumentation Framework where $AF = \langle A, R, V_A, V_R \rangle$. $V : A \rightarrow \mathbb{N} \times \mathbb{N}$, is a total function mapping each argument (V_A) or attack (V_R) to its number of positive and negative votes.

Definition 2.9. A social abstract argumentation semantic framework is a 6-tuple $\langle L, \tau, \wedge_A, \wedge_R, \vee, \neg \rangle$, where:

- L is a totally ordered set with top and bottom elements \top, \perp , containing all possible valuation for an argument.
- $\wedge_A, \wedge_R : L \times L \rightarrow L$, are two binary algebraic operations to restrict strengths to given values (where A stands for argument strengths, and R to attack strengths).
- $\vee : L \times L \rightarrow L$, is a binary algebraic operation on argument valuations used to combine or aggregate valuations and strengths.
- $\neg : L \rightarrow L$, is a unary algebraic operation for computing a restricting value corresponding to a given valuation or strength.

- $\tau : \mathbb{N} \times \mathbb{N} \rightarrow L$, is a function that aggregates positive and negative votes into a social support value.

Definition 2.10. Let F be a social abstract argumentation framework and S a semantic framework, where $S = \langle L, \tau, \wedge, \vee, \neg \rangle$. A total mapping $M : A \rightarrow L$ a social model of F under semantics S , or S -model of F , if:

$$M(x) = \tau(V_A(x)) \wedge_A \neg \bigvee_{x_i \in R^-(x)} (\tau(V_R(x_i, x)) \wedge_R M(x_i))$$
¹

Definition 2.11. Let $S_\epsilon = \langle [0, 1], \wedge, \vee, \neg, \tau_\epsilon \rangle$ be a semantic framework where $x, y \in [0, 1]$ and:

- $x \wedge y = x \cdot y$
- $\vee(x, y) = 1 - (1 - x)(1 - y)$
- $\neg x = 1 - x$
- $\tau_\epsilon(a) = \frac{V^+(a)}{V^+(a) + V^-(a) + \epsilon}$, with $\epsilon > 0$, and similarly for attacks.

This semantics consider the votes on the arguments to compute the strength of an argument as well as the votes on the attack relations. If one argument $a \in A$ has a high valuation, and it attacks $b \in A$, we can not guarantee that b will have a low valuation, since the votes on the relation $(a, b) \in R$ may decrease the strength of the attack to nearly 0, thus this attack will have almost no influence on the valuation of b .

After using these semantics, to calculate the score of each argument in the graph represented in the figure 2.1, we got the following values for SAA Extended — with votes on attack relations. The SAA Extended introduces votes on attacks, therefore an attack with no votes will have no influence on the attacked argument. The results are shown in table 2.2.

	a	b	c	d	e
SAA Extended	0.342254	0.914634	0.064522	0.588548	0.406361

Table 2.2: Results for SAA Extended

It is important to note that, if the strength of an attack or argument was 0 (e.g. *positive votes* = 0), this would change the ranking order completely, since an argument with 0 votes would have its strength equal to 0, and the attack would also have its strength equal to 0, thus not having any influence on the attacked argument.

Algorithm for Social Abstract Argumentation.

An efficient implementation of an algorithm for Social Abstract Argumentation (without votes on attacks), was introduced in [13]. We will present the ISS (Iterative Successive

¹ $\vee\{x_1, x_2, \dots, x_n\} \triangleq ((x_1 \vee x_2) \vee \dots \vee x_n)$. $M(x)$ is referred as the social strength, or value, of x in M .

```

1 while (max >= precision) {
2
3     max = 0;
4
5     for (argument in Arguments) {
6
7         newx = argument.tau;
8
9         for (parent_argument in argument.parents)
10            newx *= 1 - parent_argument.x;
11
12        max = max(max, abs(argument.x - newx));
13        argument.x = newx;
14    }

```

Listing 2.1: Pseudocode of the ISS algorithm

Substitution) algorithm briefly. The algorithm is based on successive substitutions, and uses the following iteration rule:

$$x_i^{(k+1)} = \tau_i \prod_{j < i, j \in A_i} (1 - x_j^{(k+1)}) \prod_{j \geq i, j \in A_i} (1 - x_j^{(k)})$$

Let us recall that τ_i only depends on the argument's positive and negative votes and A_i are the set of arguments that attack i . The value of x^k when $k = 0$ is the initial guess, the iteration process only stops when the stopping criterion² is attained (i.e. when ISS converges to a solution $x^* \in]0, 1[^n$).

The listing 2.1 presents the pseudo code for an implementation of the algorithm. Note that precision is the value used in the stopping criterion, and max is the maximum difference between strengths of two iterations, i and $i+1$. The initial guess of every argument was set to τ .

It is important to mention that this algorithm was already tested in [13], and can be adapted to include votes on attacks.

2.1.2.2 Categoriser

The categoriser [5] is a function that assigns for each argument a value, given the value of its direct attackers. $Cat : A \rightarrow]0, 1]$ is defined as follows:

$$Cat(a) = \begin{cases} 1, & \text{if } R_1^-(a) = \emptyset \\ \frac{1}{1 + \sum_{c \in R_1^-(a)} Cat(c)}, & \text{otherwise} \end{cases}$$

These semantics takes into account only the direct attackers of an argument to compute its rank. Furthermore, the votes on arguments and attacks are not considered by these semantics. Therefore if we had an argument with no votes, or attack with no votes, that would not make any difference on the ranking calculated by this function.

²If there is at least one variable x_m converging to x_m^* then the algorithms converge to a solution $x^* \in]0, 1[^n$.

If we apply this function to the example in the previous subsection (Figure 2.1) we get: $Cat(a) \approx 0.477$, $Cat(b) = 1$, $Cat(c) = 0.5$, $Cat(d) \approx 0.677$, $Cat(e) \approx 0.596$.

2.1.2.3 Discussion-based semantics

Discussion-based semantics [2] are based on a linear discussion. A linear discussion is a sequence of arguments such that each argument attacks the argument preceding it in the sequence. We define $Dis_i(a)$ as follows:

$$Dis_i(a) = \begin{cases} -|R_i^+(a)|, & \text{if } i \text{ is odd} \\ |R_i^-(a)|, & \text{if } i \text{ is even} \end{cases}$$

In these semantics the number of attacks to an argument is more important than the strength of the arguments that are attacking. When applying this semantics to the AF represented in figure 2.1, we get the results reported on table 2.3 from where we get the following ranking, that is ordered lexicographically: $b \succ_{DBS} d \succ_{DBS} e \succ_{DBS} c \succ_{DBS} a$.

step	a	b	c	d	e
1	2	0	1	1	1
2	-2	0	0	-2	-1

Table 2.3: Discussion-based semantics

2.1.2.4 Burden-based semantics

The Burden-based semantics [2] are based on Burden numbers. The Burden number of an argument a is defined based on the burden numbers of its direct attackers. We define $Bur_i(a)$ as follows:

$$Bur_i(a) = \begin{cases} 1, & \text{if } i = 0 \\ 1 + \sum_{b \in R_1^-(a)} \frac{1}{Bur_{i-1}(b)}, & \text{otherwise} \end{cases}$$

As on Discussion-based semantics the number of attacks to an argument is more important than the strength of the arguments that are attacking it. When applying this semantics to the AF represented in Figure 2.1, we get the results reported on table 2.4 from where we get the following ranking: $b \succ_{BBS} d \succ_{BBS} e \succ_{BBS} c \succ_{BBS} a$.

step	a	b	c	d	e
1	3	1	2	2	2
2	2	1	2	1.33	1.5
3	2.17	1	2	1.5	1.75

Table 2.4: Burden-based semantics

2.1.2.5 Valuation with tuples

The valuation with tuples[11] takes into account all the ancestor branches of an argument stored in tupled values. Let $a, b \in A$, we consider the number of attack branches on a and b , as well as the defense branches. Afterwards we compare those numbers to define whether $a <_{VT} b$ or $b <_{VT} a$.

Note that if the AF has cycles, some tuples can be infinite, to solve this, the algorithm transforms cycles into infinite acyclic graphs to calculate them. This is not a trivial algorithm to calculate, and as there was no implemented version of the algorithm, we will not show the results for this algorithm and we will only consider its main properties.

2.1.2.6 Matt & Toni

Matt and Toni [25] compute the strength of an argument based on a game. This game confronts two players, a proponent and an opponent of a given argument, where the strategies of a player are sets of arguments. Let S_P be the strategies for the proponent and S_O for the opponent, $S_O, S_P \subseteq A$.

Definition 2.12. The degree of acceptability of P with respect to O is given by

$$\phi(P, O) = \frac{1}{2}[1 + f(\text{N. of attacks from P to O}) - f(\text{N. of attacks from O to P})]$$

where $f(n) = \frac{1}{1+n}$

Definition 2.13. The rewards of P, denoted by $r_F(P, O)$, are defined by:

$$r_F(P, O) = \begin{cases} 0, & \text{iff } \exists a, b \in P, (a, b) \in R \\ 1, & \text{if N. of attacks from O to P} = 0 \\ \phi(P, O), & \text{otherwise} \end{cases}$$

Definition 2.14. For each argument $a \in A$ the proponent's expected payoff is given by:

$$E(a, p, q) = \sum_{j=1}^n \sum_{i=1}^m p_i q_j r_{i,j}$$

Where p and q are probability distributions with $p = (p_1, p_2, \dots, p_m)$ and $q = (q_1, q_2, \dots, q_m)$.

Lastly the value of the game is denoted by:

$$s(a) = \max_p \min_q E(a, p, q)$$

In Matt & Toni the strength of the attackers has more impact on the valuation of the arguments than the number of attackers. Note that this is not a trivial algorithm to calculate, and as there was no implemented version of the algorithm, we will not show the results for this algorithm and we will only consider its main properties.

2.1.3 Discussion

After analyzing all these semantics, we can reach the following conclusions. Table 2.5 shows the resulting ranking for some of the semantics examined before. When we look at the table 2.5, it is clear that the votes (on arguments and attacks) have a significant impact on the argument ranking, even for a small example with only five arguments, SAA Extended produces a different result from all the other semantics.

	1	2	3	4	5
SAA Extended	b	d	e	a	c
Categoriser	b	d	e	c	a
DBS	b	d	e	c	a
BBS	b	d	e	c	a

Table 2.5: Ranking Results Comparison

All the discussed semantics, except from **Social Abstract Argumentation Framework** (section 2.1.2.1), do not consider votes on arguments and votes on attacks. However, these semantics could be reformulated in order to take into account votes on arguments and attacks. The **Categoriser** could be adapted to have votes on both arguments and attacks, in order to do that the Categoriser function should be changed in the following way:

$$Cat(a) = \begin{cases} \tau(a), & \text{if } R_1^-(a) = \emptyset \\ \frac{\tau(a)}{\tau(a) + \sum_{c \in R_1^-(a)} \tau(c, a) \times Cat(c)}, & \text{otherwise} \end{cases}$$

Where $\tau(a)$ is the same function defined in the section 2.1.2.1.

Although these semantics can be easily adapted in order to consider both argument strength and attack strength, this solution is not experimented or studied as Social Abstract Argumentation Framework.

The **Discussion-based semantics** are difficult to adapt in order to take into account the votes on arguments and attacks, since these semantics rely on the number of the sequence of attacks. On the contrary **Burden-based semantics** can be adapted in order to consider votes on arguments. To achieve that the Burden-based semantics function must be adapted in the following way:

$$Bur_i(a) = \begin{cases} \tau(a), & \text{if } i = 0 \\ \tau(a) + \sum_{b \in R_1^-(a)} \frac{\tau(a)}{\tau(b, a) \times Bur_{i-1}(b)}, & \text{otherwise} \end{cases}$$

Matt & Toni and **Validation Tuples** semantics rely on more complex algorithms where it is difficult to introduce a change to consider the votes on arguments and votes on attacks, without having to reformulate the whole algorithm.

To sum up, although some of the semantics can be easily adapted to consider votes on arguments and attacks, there are no evidence that these semantics would guarantee

the properties that our solution requires, and since the **Social Abstract Argumentation Framework** already considers votes on arguments and votes on attacks, and also proposes an efficient algorithm [13] to compute the argument and attack strengths, these semantics seem to be the most appropriate to address our problem.

2.2 Existing argumentation/debate applications

There are already a lot of debate systems in the web that we will analyse throughout this section. In order to do that, we will divide the different systems in classes and discuss the varied features of each one.

In this section we will consider two main types of existing applications. First, we will see the systems that are based on Abstract Argumentation, although we will use a broader definition of Abstract Argumentation, where the only property is that an argument is not divided into parts. Afterwards we will study the systems based on structured argumentation³.

2.2.1 Abstract Argumentation Systems

In this section we are going to present systems that allow users to have a debate based on abstract argumentation.

2.2.1.1 Pros and Cons

In this class of solutions, only one thesis is presented to the user and then users are given the possibility to discuss the pros and the cons of this thesis. In most models, the user can contribute to both sides of the thesis. The systems [consider.it](#) [12] and [createdebate.com](#) [30] are two good examples of this kind of systems. Consider.it lets the user add an argument for or against the main thesis. It also lets the user express his/her opinion by asking how much does he/she supports one or other side of the thesis. A user can also reuse existing arguments to express his/her point of view (similar to vote on an argument, see figure 2.2).

CreateDebate also lets the user add arguments to both sides of the thesis, although it differs from the previous system because the user can dispute, clarify, or support any argument as well as upvote or downvote each one of them (see figure 2.3).

Clearly the main purpose of this type of applications is to freely discuss a single thesis, without having to be concerned with whether or not some argument is valid, since the judgement is left to the users of the system.

³An argument is said to be structured if the premises and claim are explicit, and the relationship between them is formally defined[6].

2.2. EXISTING ARGUMENTATION/DEBATE APPLICATIONS

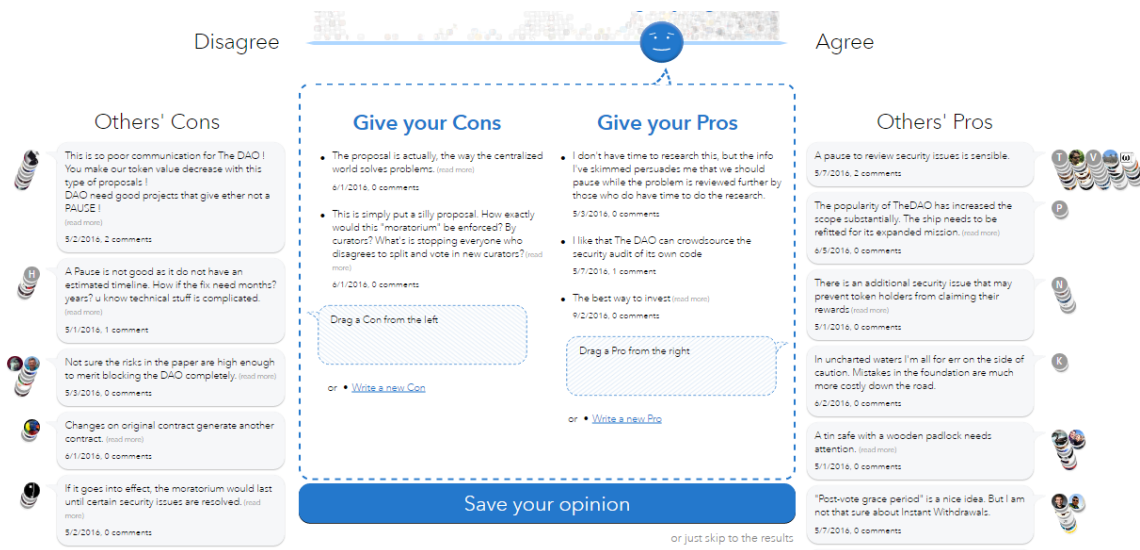


Figure 2.2: Consider.it

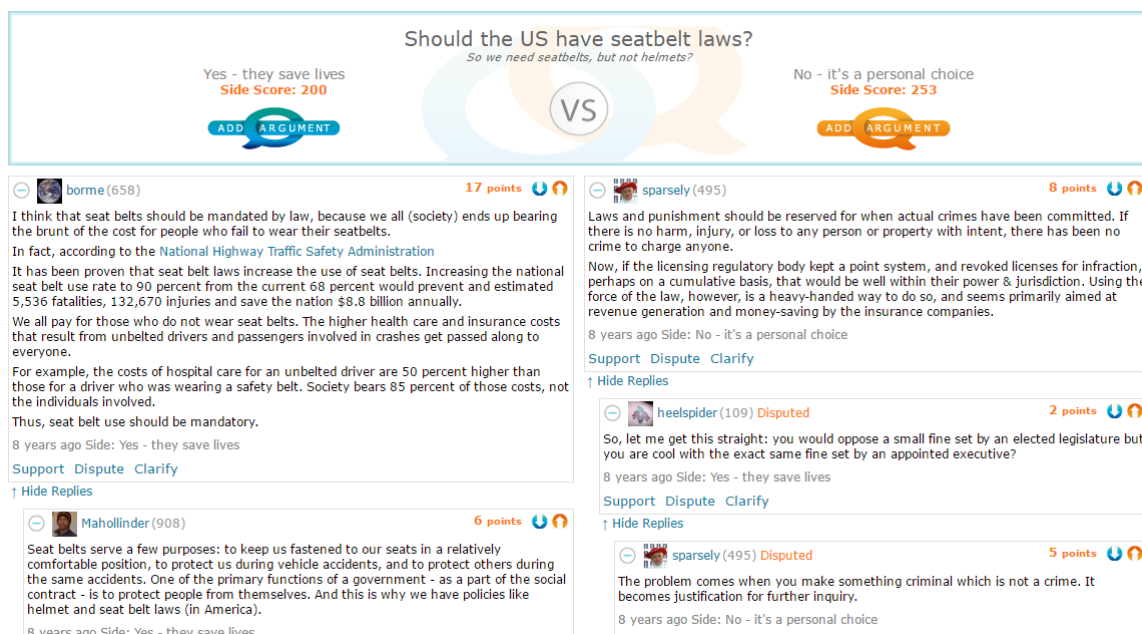


Figure 2.3: CreateDebate

2.2.1.2 Mind Map Systems

Another way to organize arguments is to use a mind map, these applications are not argumentation systems. In this kind of applications the user starts with a main idea and decompose the idea into smaller ones. In this section, we are going to examine two different systems that are based in this concept, debategraph.org[14] and mindmeister.com[26].

Debategraph starts with a main idea in the center node of the graph and the subideas all around the center node connected to that main idea. Everytime a user clicks on a

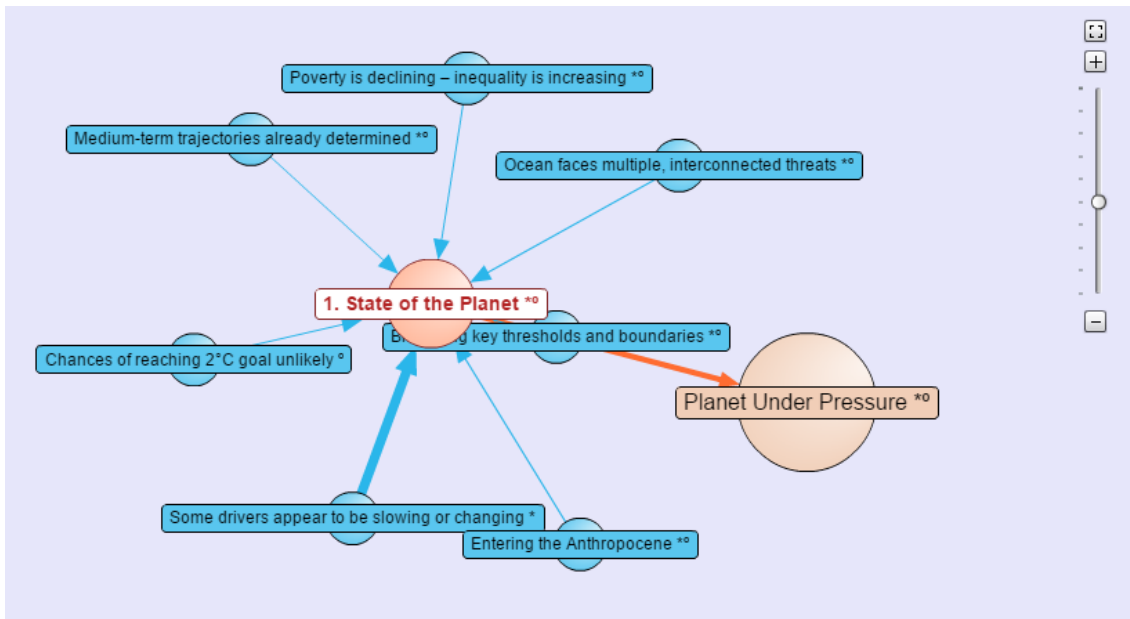


Figure 2.4: Debategraph interface

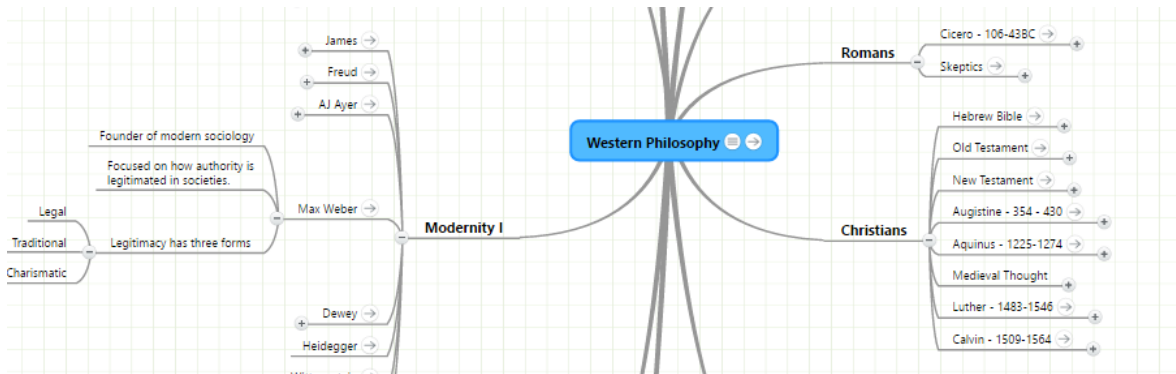


Figure 2.5: MindMeister interface

node, that node goes to the center and the other nodes disappear, appearing new nodes all around the center node, representing the subideas of the current selected node (see figure 2.4).

MindMeister is a mind map. In the center we get the main idea and as we get further from the center we start to get that idea decomposed into smaller branches (see figure 2.5).

These kind of systems are used to expose and navigate all the smaller aspects of a main thesis, this is useful to organize a discussion or even to study some subject without losing the overall view of the main subject.

2.2.2 Structured Argumentation Systems

In this section we will focus on structured argumentation systems. In Structured Argumentation systems, each argument is divided in different parts. For example, we can have

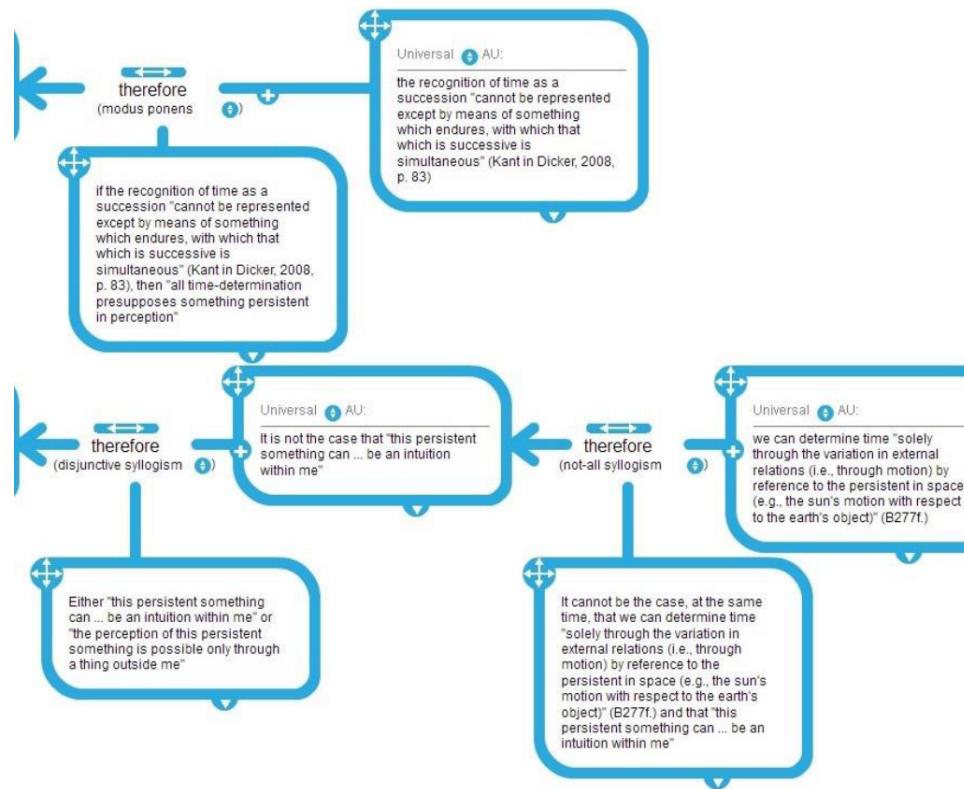


Figure 2.6: Structured Argumentation Systems: Agora-net

some premises and infer some claim, all the parts together build an argument. Since the relations between every part of the argument are explicitly defined, the role of each part is also clearly defined.

Let us look at the example of agora.gatech.edu[1], where the arguments are represented as nodes and the relations are represented as edges (figure 2.6). The role of each node is directly associated with the relation that it has with other nodes. If the relation between a node a and a node b is "modus ponens" we can conclude that a is a premise to conclude b .

To sum up, these kind of systems are used to structure and strengthen a thesis rather than to provide a open debate to the users.

2.2.3 Discussion

After analyzing all these systems we note that, although there are some systems that meet some of the requirements of Social Abstract Argumentation, none of the explored tools meets all the requirements referred in the previous chapter. The applications that are based on abstract argumentation could be adapted to Social Abstract Argumentation semantics more easily. However, the applications that are based on structured argumentation, also have some features that can be adapted to our solution.

From the two applications that we examined, that were based on abstract argumentation, consider.it does not allow the users to vote on the arguments and also it is

restricted to only two different thesis. It introduces a different way for the user to express their opinion about the whole debate, instead of the argument voting that is considered in Social Abstract Argumentation.

The application createdebate.com allows the users to vote positively and negatively on each argument, and also to dispute, support and clarify an argument. This application is also limited to a fixed number of thesis. The option to dispute an argument could be similar to the idea of attack in Social Abstract Argumentation, although the idea may be similar, the behaviour is different from what we expected in an implementation of the Social Abstract Argumentation. In createdebate.com the element that disputes another argument works as a reply (that can also be upvoted and downvoted, disputed, clarified and supported), therefore it can only dispute (attack) that argument and it does not appear in the arguments of any other thesis.

The interface of both applications is presented in the form of threads, with one thread for each thesis. If we try to adapt this kind of display to the Social Abstract Argumentation it may be limiting since there are no restrictions to the number of thesis that a debate can have. Some options such as the idea of support and the possibility to clarify an argument can possibly be considered to an adaptation of the Social Abstract Argumentation.

The applications based on structured argumentation focus more on the relations between arguments, narrowing the user's freedom to argue. However, both systems interfaces display the arguments in a more suitable way, considering the many attack relations that can be established in an implementation of the Social Abstract Argumentation.

The system *agora-net* represents the distinct parts of the arguments as nodes, and the relations between the arguments as edges. This graph display can be easily adapted to Social Abstract Argumentation. If we consider each argument as a node and each attack as an edge, we can have a suitable presentation of the debate. Despite this, the system does not allow much user interaction, and the relations between the different argument parts are limited. This make this tool suitable to structured argumentation, but not suitable for abstract argumentation.

Mind maps are used to decompose and organize ideas into smaller parts, they do not let more than one user participate, leading to an informative hierarchical graph. Although these systems do not have many of the properties that we aim, there are some points that can be considered to our solution. The possibility of focusing on smaller parts of the graphs, and also the ability to filter some of the nodes in order to see only the relevant nodes for the user, are some of the features that can be considered in our solution.

In this work we focused on some of the systems that exist and that may be representative of the classes defined in this section. However, there are a lot of other systems that can also be grouped in the different classes that we defined previously.

To conclude, we want to create a system that has different goals or properties from the solutions discussed before, although it might also share some of them.

The system should provide an easy way to interact, allowing the user to participate and give his/her opinion, via new arguments, voting, adding attacks between arguments,

and also by voting on attacks.

PROPOSED SOLUTION

In this chapter we describe the proposed solution, project requirements, and the system architecture in which our prototype is based.

3.1 Proposal

In this dissertation we propose to develop an Online Debate Tool that allow us to validate the Social Abstract Argumentation model proposed in [17, 23]. To achieve this, we developed a prototype of an Online Debate Tool that allows the users to participate in a debate. This tool can be divided in three main components that must be well integrated among each other.

The presentation layer is the topmost level of the application. This layer manages the user interface and also sends requests to the application layer. The application layer receives the requests from the presentation layer, compute the results and also sends requests to the storage layer, that stores all the data needed to maintain the application state.

In order to adequately capture the notion, underlying the Social Abstract Argumentation model, our solution must enable users to execute specific actions, such as the possibility to create arguments, vote on arguments, create attacks between arguments, and vote on attacks. All these interactions must be intuitive to the final user, therefore our system must also be visually appealing and easy to use.

3.2 Project Requirements

In this section we will define the project requirements, divided in functional and non-functional requirements.

3.2.1 Functional Requirements

1. The user must be able to add a new argument.
2. The user must be able to add a new attack relation between two arguments.
3. The user must be able to vote positively or negatively on an attack or on an argument.
4. The debate must be presented in the form of a graph.
5. The user must be able to log in.
6. The user must be able to see which arguments and relations are already voted by himself.
7. The user must be able to see the debate ranking.

3.2.2 Non-functional Requirements

The system must provide adequate performance to enable an adequate interaction between users and the system, and guarantee that the users receive the appropriate feedback. The results of the operators that modify the state of the debate should be visible to the user with a low latency envelope.

3.3 System Architecture

Our framework architecture is based on the Three-tier model firstly proposed by Eckerson in 1995 [16] (Figure 3.1). The system is divided in presentation layer, application layer, and storage layer. In this section we will describe each tier and the interactions between them.

3.3.1 Presentation Layer.

The Presentation Layer is the topmost level of the application. In our solution the presentation layer includes the User Interface and also the requests to the application layer. This layer also deals with all the information that comes from the application layer.

3.3.1.1 User Interface

The user interface was developed in *HTML5* and *Javascript* and also relies on the library *Vis.js*[10] to enable the visualization of the graph that represents the ongoing debate. *Vis.js*

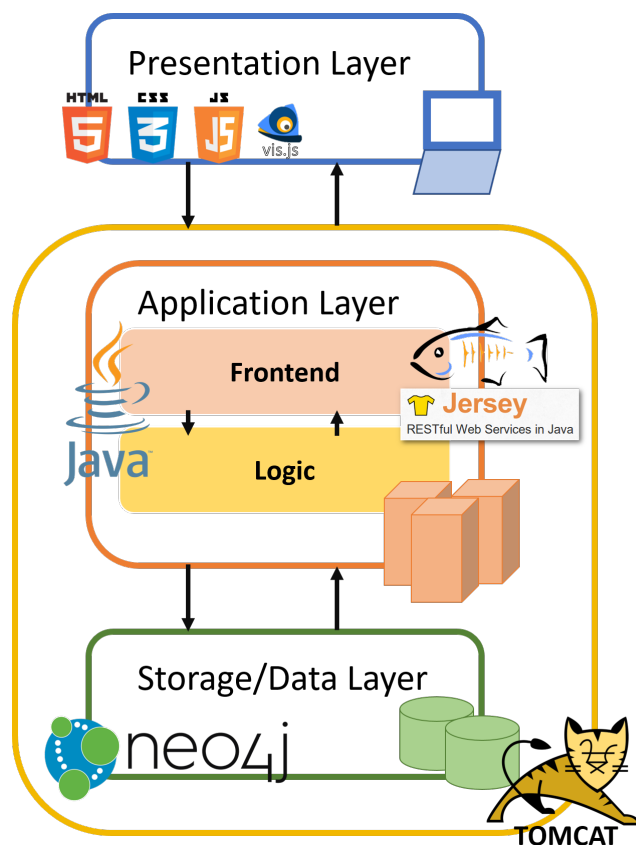


Figure 3.1: Three-tier architecture

library allows the customization of the graph elements and also uses physics inspired algorithms to position the elements on the canvas. Figure 3.2 presents the overview of the user interface, each component will be described further in this section.

There are four main elements that compose the debate: arguments, votes on arguments, attacks, and votes on attacks.

The arguments are represented using a node object of *Vis.js*. In order to represent an argument, the node object from *Vis.js* has to be used by setting some specific configurations. Firstly, the shape of the node had to be changed to a box, and also other properties had to be changed such as border radius, label and title in the pursuance of achieving a more appealing design to the final user. The node color also changes depending on the strength of each argument.

To represent an attack, the arrow object of *Vis.js* was used. Some configurations were also adjusted such as the arrow title and width to make the arrow more appealing to the final user, and also more appropriate to its use in our system. The arrow width varies from values slightly above 0 to 20. The function that determines the width value is the following:

$$width_a = strength_a * 20$$

Where a is the attack and the strength value varies from 0 to 1 according to [23]. If an



Figure 3.2: User Interface Overview

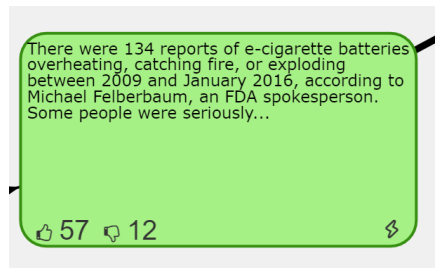


Figure 3.3: Argument example

attack has 0 width, *Vis.js* library does not make the arrow disappear, instead the arrow width will be the minimum width available in the *vis.js* library.

There were some other libraries that were candidates to be chosen to represent the debates in the form of graphs such as *D3.js* [8]. This library allows to create directed graphs that can be customized thorough a set of mechanisms made available by the library.

Both libraries had the limitation of not allowing to introduce custom *html* in the nodes. This limitation prevented us from further customizing the node and led us to having to place an *html* object on top of the node every time the user interacts with it. Since the limitation is the same for both of the libraries the choice of using *Vis.js* was only a matter of convenience to the development of the tool, in particular because *Vis.js* provides a clear documentation and a large set of examples that allow a fast learning of the library.

Argument. Each argument is represented through a node. The text inside the node is the argument overview and there are also three buttons, that the user is able to interact with. The argument can be previewed on the bottom right of the page and it can also be opened using double click or using the right click pop up menu and then choosing "View argument". When the mouse is over an argument, a pop up box with information about the argument (i.e. positive and negative votes, strength), is displayed by the user interface. Note that the strength of an argument is represented by a number from 0 to 100, instead of 0 to 1 used by the algorithm.

The *thumbs up* and the *thumbs down* buttons allow the user to vote positively or negatively on the argument. There is also a third button with a *bolt*, which is used to attack (create a directed edge towards) another argument or the argument itself (see Figure 3.3).

Each argument can have 10 different background colors forming a gradient between red, yellow and green. The relation between the colors and the strength of each argument is represented in the Figure 3.4.

The red color was chosen for the arguments with the smaller strength since red is commonly associated to "*unaccepted*" or "*wrong*". The green color is associated with the strongest arguments since it is often associated with "*right*" and "*acceptance*". Therefore yellow is in the middle being a more neutral color representing the arguments that are

not weak nor strong.

There are two possible ways to create an argument: (1) double clicking on an empty space on the canvas or (2) using the right click to display a pop up menu followed by the click on the option "New Argument", and then fill the argument input box and click on the create button.

There are three possible ways to vote on an argument: (1) using the buttons on the bottom left corner of the argument representation in the graph; (2) using the buttons on the bottom left corner of the argument preview on the right side of the interface; (3) using the buttons on the right bottom corner of the opened argument view.

Attack. The attack relation between two arguments is represented through directed edges that vary their width depending on the strength (i.e. relation between positive and negative votes) of the attack.

The user may create an attack in two different ways: (1) clicking on the bolt symbol on the bottom corner of the argument (Figure 3.3) and then clicking on the target argument; (2) right clicking on the source argument of the attack and then clicking on the target argument. To vote on an attack it is necessary to click on the edge that represents the attack, which makes a pop-up box appear with 2 buttons, *thumbs up* and *thumbs down*, that enable the user to either vote positively or negatively respectively (see Figure 3.5). Note that the user can vote only once on each attack.

Moving and zoom. At the bottom right corner of the screen the user can find zoom buttons that let him/her to zoom in or zoom out the canvas. This allows the user to focus on a small part of the debate or to have an overall view of the debate. At the left bottom corner, there is an arrow pad that allows the user to move along the canvas, in order to inspect different parts of the debate (see Figure 3.2).

It is also possible to navigate along the debate by using the mouse. To move along the debate a user can click and drag in an empty space of the canvas. To zoom in and zoom out the user can use the mouse scroll wheel.

Strength	Node Color
0.0 - 0.1	Red
0.1 - 0.2	Orange-Red
0.2 - 0.3	Orange
0.3 - 0.4	Light Orange
0.4 - 0.5	Yellow
0.5 - 0.6	Light Yellow
0.6 - 0.7	Light Green
0.7 - 0.8	Green
0.8 - 0.9	Dark Green
0.9 - 1.0	Dark Green

Figure 3.4: Relation between colors and acceptance

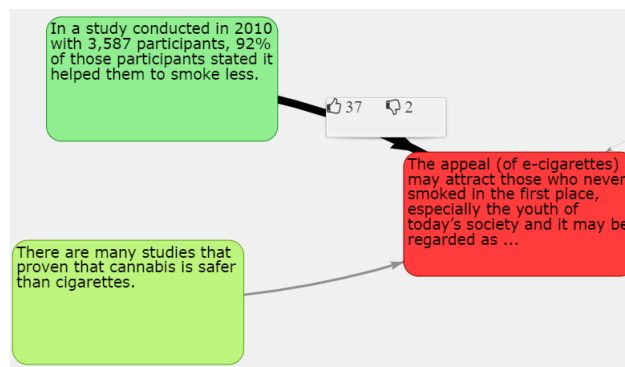


Figure 3.5: Attack example

Debate Ranking. It is possible to consult the Debate Ranking by clicking on the *Debate Ranking* tab on the top menu or using the right click pop up menu and choose the option "Debate Ranking". The debate ranking shows all the arguments in decreasing order taking into account the argument strength.

Filters. The filters appear on the right side of the screen. The arguments can be filtered by their color. In order to make arguments with a specific color disappear, one must click on the desired color. The arguments with that specific color will only disappear if they are not connected with any of the other existing and displayed arguments (see figure 3.6).

Top Menu. The top menu of the application has three different options: Login, De-

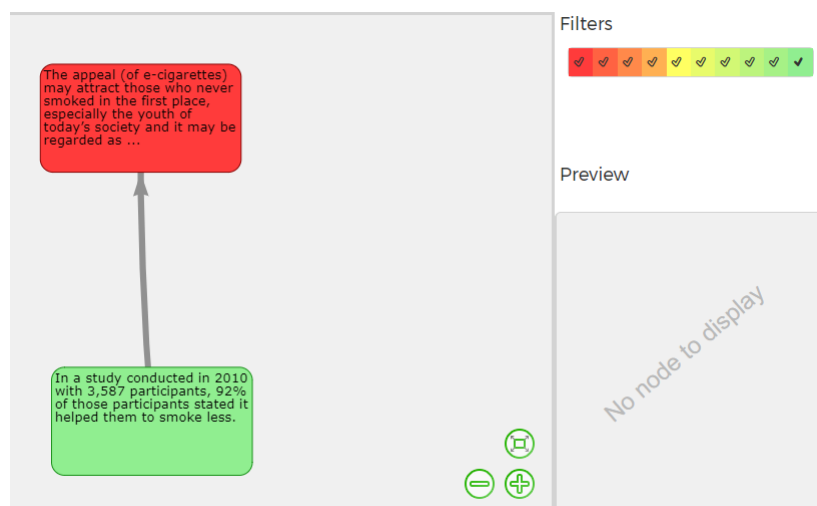


Figure 3.6: Filters example

bate Ranking, and Register (see Figure 3.7). When a user clicks on the login button a modal that allows the user to login appears requesting the user credentials (i.e. username and password).

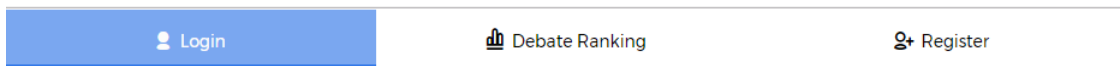


Figure 3.7: Top menu

The register modal allows the user to register on the system, a modal appears where the user must fill the username, email and password fields. The email is validated in this form and also the passwords must match and be at least 6 characters long.

Right Click Pop Up Menu. When a user clicks with the right button of the mouse on the canvas a pop up menu is shown with the following options: "New Argument", "Start Attack", "View Argument", and "Debate Ranking" (see Figure 3.8). If the user right clicks on an empty canvas space only the options to create a new argument and to consult the debate ranking are enabled. If the user right clicks above an argument the options "Start Attack", "View Argument" and "Debate Ranking" also become enabled, while the "New Argument" option is disabled.

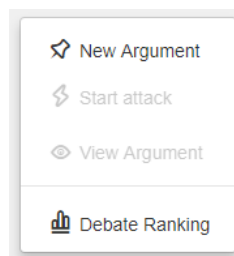


Figure 3.8: Right click pop up menu

3.3.2 Requests to the Application Layer

The presentation layer makes the needed requests to the application layer and also deals with all the data that comes from the application layer. The REST requests use the jQuery library [18] and are made using AJAX (*Asynchronous Javascript and XML*). The Listing 3.1 provides an example of a request made to the application layer. This request occurs when a user creates an attack between two nodes s (source) and t (target).

After the request is made the application layer returns a JSON formatted message to the Presentation Layer. This message is then parsed by the presentation layer and used to display the proper information to the user. Listing 3.2 shows an example of the parsing of the request that returns all the nodes of a specific graph (i.e. *graph/graph_id*).

In the Listing 3.2 the list of nodes of the requested graph is iterated. For each argument, a node object from *Vis.js* is created (line 3 to 15) and the dataset of the nodes is updated and consequently the nodes are updated in the canvas (line 16).

```

1
2   $.ajax({
3     'async': true,
4     'type' : "POST",
5     'global': false,
6     "headers": {
7       "authorization": "Bearer_" + token},
8     'url': "http://localhost:8080/helloworld/rest/connectnodes/g/"
9     + graphId + "/s/" + s + "/t/" + t + "/" + token,
10    'dataType': "json",
11    'success': function (data) {
12      json = data;
13    }
14  });

```

Listing 3.1: Example of an Ajax request - Connecting two arguments

```

1   for(i = 0; i < json.nodes.length; i++) {
2     var debnode = json.nodes[i];
3     node = {id: debnode.id,
4             title: "Score: " +
5             (Math.round(debnode.x * 10000) / 100) +
6             "/100" + "<p>Positive: " + numberRed(debnode.positiveVotes) +
7             "</b>" + "Negative: " + numberRed(debnode.negativeVotes) + "</b></p>",
8             label: debnode.text, arg: debnode.arg, score: debnode.x,
9             userId: debnode.userId,
10            votePos: debnode.positiveVotes, voteNeg: debnode.negativeVotes,
11            text: debnode.text,
12            color: {background: defineBackground(debnode.x),
13                  border: defineBorder(debnode.x)},
14            chosen: {label: false, node: changeColor(debnode.x)},
15            widthConstraint: {maximum: 600, minimum: 600},
16            hidden: false, shapeProperties: {borderDashes: false}, };
17     nodes.update(node);
18   }

```

Listing 3.2: Example of parsing the results of a request - Get all nodes from a graph

3.3.3 Application Layer

The application layer deals with all the logic behind the tool (e.g. semantics algorithm) and it also retrieves the data from the storage layer. It is divided into two sublayers, Frontend and Logic. The Frontend receives and forwards all the requests that come from the presentation layer and also sends the expected responses to the presentation layer. The logic sublayer deals with all the computations and requests to the storage layer.

The application layer was implemented as a web service running on a Tomcat Server [19] that allows requests and it also runs all the algorithm computations and logic.

3.3.3.1 Frontend

This sublayer of the application layer provides a REST API to connect with the Presentation Layer. Frontend organizes and filters the information that comes both from the algorithm and the storage layer providing the necessary methods to the Presentation Layer:

```
1 {
2   id:2,
3   nodes:[
4     {
5       id:0,
6       x:0,
7       text:"Argument_1",
8       arg: "",
9       userId:"user1",
10      positiveVotes:0,
11      negativeVotes:0
12    },
13    {
14      id:1,
15      x:0,
16      text:"Argument_2",
17      arg: "",
18      userId:"user2",
19      positiveVotes:0,
20      negativeVotes:0
21    },
22    {
23      id:2,
24      x:0,
25      text:"Argument_3",
26      arg: "",
27      userId:"user3",
28      positiveVotes:0,
29      negativeVotes:0
30    }
31  ],
32  nodesSize:3
33 }
```

Listing 3.3: Example of JSON returned by graph_id

graph/graph_id. This method returns all the arguments in a debate graph with a specific id. An example of an answer returned by this method is presented listing 3.3. Listing 3.3 shows all the nodes for the graph with the $id = 2$ and the respective nodes list of the graph. Each node has several properties: id, strength (represented by x), argument text preview (represented by text), argument extension (represented by arg), userId, positiveVotes, and negativeVotes.

attacks/graph_id. This method returns all the attacks that exist in a debate graph with a specific id.

debatenode/node_id. This method returns all the information about an argument/debate node with a specific id. This information includes: id, node strength, node text, node argument extension, user id of the creator, number of positive votes and number of negative votes.

addnode/node_id. This method receives a JSON with all the information needed to create a new debate node (i.e. an argument in a debate).

deletenode/nodeId/node_id/token/token. This method receives an id of an argument and deletes it. This action can only be successfully performed if the user with the provided token is the owner of the argument (i.e. the creator of the argument), and if the argument has no votes. The argument can only be deleted when it has no votes since we do not want to interfere on the debate, and an argument with any vote has impact in the debate.

connectnodes/g/graph_id/s/source/t/target. This method creates a new attack between two debate nodes (i.e. two arguments in a debate). The attack starts in the node with the id *source* and ends in the node with the id *target*.

votepos/g/graph_id/node/node_id/token/token. This method adds a positive vote to a node with a specific id made by the user with the provided token. In the case the user has already voted positively on the argument, the existing vote is removed. Otherwise, if the previous vote of the user is negative, a positive vote is added and a negative vote is subtracted from the argument.

voteneg/g/graph_id/node/node_id/token/token. This method adds a negative vote to a node with a specific id made by the user with the provided token. In the case the user has already voted negatively on the argument, the existing vote is removed. Otherwise, if the previous vote of the user is positive, a negative vote is added and a positive vote is subtracted from the argument.

voteposE/g/graph_id/attack/attack_id/token/token. This method adds a positive vote to an attack with a specific id made by the user with the provided token. In the case the user has already voted positively on the attack the previous vote on that attack is removed. Otherwise, if the previous vote of the user is negative, a positive vote is added and a negative vote is subtracted from the argument.

votenegE/g/graph_id/attack/attack_id/token/token. This method adds a negative vote to an attack with a specific id made by the user with the provided token. In the case the user has already voted negatively on the attack the previous vote on that attack is removed. Otherwise, if the previous vote of the user is positive, a negative vote is added and a positive vote is subtracted from the argument.

createuser. This method receives a JSON message with all the information needed to create a new user. If there is no user with the same username a new user will be created.

The information provided is: username, email and password.

existsuser. This method receives a JSON message with a username and a password and returns a JSON message with a boolean that will be true if both the username and password exist and match, and false otherwise. The JSON message also provides a temporary token needed to execute most of the requests and an expiration date.

getuserbytoken/token This method returns all the information of the user that currently owns the specified token. This method allows the user to close the page and still be logged in if he returns within one hour from the last time he logged in to the application. The token is stored in the browser's local storage.

getuservotes/token/g/graph_id This method returns all the votes (on arguments and attacks) that the user with a specified token owns.

The API was developed using Jersey [22] and Tomcat [19]. Jersey is a library that provides a framework for developing RESTful web services in Java. The Jersey framework allows to add filters to the Web Application (e.g. every time a user makes a request to the web application a function will be called automatically by the Jersey runtime).

Authentication Filter. In order to guarantee the user authentication an Authentication Filter was added to the Web Application. The authentication filter intercepts the user requests and verifies if the token that is present in the request header matches any valid token in the database. If it does, the request proceeds, otherwise, an exception is thrown and the request is cancelled.

3.3.3.2 Logic

The Logic sublayer computes the algorithm and makes requests to the storage layer. In order to implement all the requests to the storage layer, it was used the adapter design pattern¹[21], which consists in a class that permits to retrieve all the necessary information from the database.

Adapter to the database. The adapter is used as an abstraction to the client. The client makes a specific request, this request is received by the adapter that transforms it into a query. The query is forwarded to the Neo4J Driver that sends it to the database and receives the response records. After these records are received, the adapter parses and organizes the records. Finally the client receives the answer (see Figure 3.9). The Singleton²

¹The adapter design pattern purpose is to "Convert the interface of a class into another interface clients expect" [21]

²The Singleton design pattern guarantees that there is only one object of the desired class.


```

1 public List<DebateNode> getAllnodesFromGraph(int id){
2     List<DebateNode> list = new ArrayList<>();
3     StatementResult r = session.run(
4         "MATCH_(g:Graph)-[r:BELONGS_TO]->(n:DebateNode)"
5         + " WHERE_(g.gid=_"+id+" )"
6         + " RETURN _n.nid_as_id, _n.posVotes_as_posVotes, _n.negVotes_as_negVotes"
7         + ", _n.x_as_x, _n.text_as_text, _n.userId_as_userId");
8     while(r.hasNext()){
9         Record record = r.next();
10        int nid = record.get("id").asInt();
11        int posVotes = record.get("posVotes").asInt();
12        int negVotes = record.get("negVotes").asInt();
13        double x = record.get("x").asDouble();
14        String text = record.get("text").asString();
15        String userId = record.get("userId").asString();
16        DebateNode n = new DebateNode(nid, posVotes, negVotes, x, id, text, userId);
17        list.add(0, n);
18    }
19    return list;
20 }

```

Listing 3.4: Method: getAllnodesFromGraph(int id)

design pattern was also used to guarantee that there is only one instance of the Neo4J Adapter in the system.

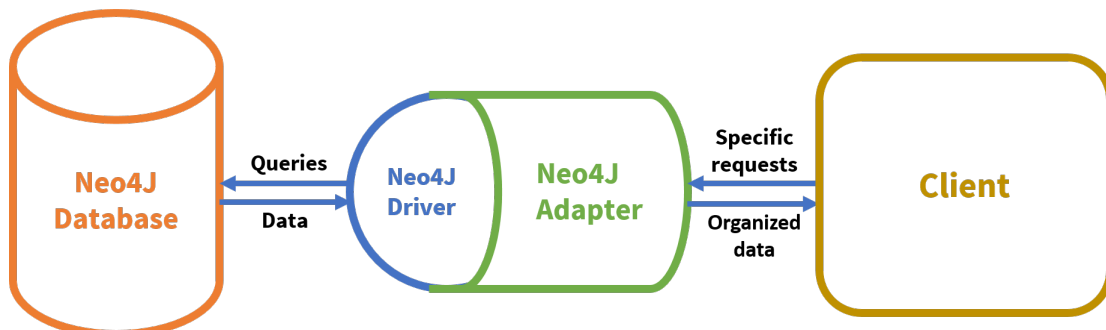


Figure 3.9: Adapter scheme

An example that can illustrate this process is when the client asks all the nodes from a specific graph (*graph/graph_id*). The method *getAllnodesFromGraph(int id)* is called (listing 3.4):

This method sends a query to the driver using the driver session (line 3 to 7 - Listing 3.4) and converts the answer into objects (*DebateNode*) that are in a format understood by the presentation layer.

Algorithm. The algorithm that was explained in section 2.1.2.1 was adapted to allow votes on attacks and implemented in Java — previously there was only an implementation in C++. This was done to allow an easy integration of the algorithm with the database library and also Jersey. The ISS algorithm iteration rule was adapted in order to consider

```

1 double prec = 1e-12;
2 while(max >= prec){
3     max = 0;
4     for(int i=0; i < scc.length; i++){
5         DebateNode pNode = g.nodesMap().get(scc[i]);
6         double newx = pNode.getA();
7         for(Pair attacker: pNode.getParents(g)){
8             newx *= 1 - attacker.getNode().getX() * attacker.getValue();
9         }
10        max = Math.max(max, Math.abs(pNode.getX() - newx));
11        pNode.setX(newx);
12    }

```

Listing 3.5: Adapted ISS algorithm Java implementation

the strengths of the attacks:

$$x_i^{(k+1)} = \tau_i \prod_{j < i, j \in A_i} (1 - x_j^{(k+1)} \cdot \tau_{j,i}) \prod_{j \geq i, j \in A_i} (1 - x_j^{(k)} \cdot \tau_{j,i})$$

To develop the re-implementation of the algorithm, several classes and functions were created. The classes *DebateNode*, *Graph*, *Pair* and *SAAF* were created.

The class *DebateNode* provides all the functions to access the necessary fields to compute the strength of the debate node: positive votes, negative votes and attackers list with the respective positive and negative votes.

The class *Graph* contains a list of all the *DebateNodes* that compose that graph and also the graph id.

The class *Pair* contains a node (attacker) and the respective positive and negative votes.

The class *SAAF* (Social Abstract Argumentation Framework) has all the functions that are part of the semantics algorithm.

Listing 3.5 presents the code of the implementation of the adapted version of the ISS algorithm in Java.

Executor thread to run the algorithm. To run the algorithm without compromising the response times of the API the algorithm runs in a separate thread. This separate thread is an *Executor* class that is running in background for each debate and recalculates the scores of the arguments within a time interval of 3s. The API returns the JSON answer to the client whilst the executor thread runs on background and updates the arguments' values in the database.

3.3.4 Storage/Data Layer

This layer stores all the data that defines the state that it is needed to maintain the application, such as users, arguments, debates, and relation data (i.e. attacks, votes, and to which graph an argument belongs).

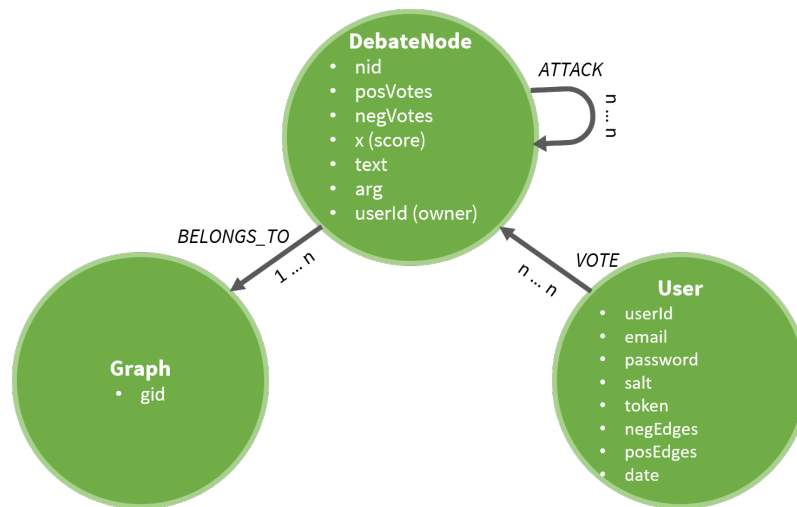


Figure 3.10: Neo4J Database Schema

This layer uses `neo4j`[28], a Graph Database that is well adapted to our problem. In the `neo4j` every entity is a node and every relation is an edge. This abstraction allows to establish relations between entities in a more intuitive way. Furthermore, `neo4j` is optimized to graph queries with highly connected data, allowing to run queries, over a graph with a considerable number of attacks, potentially faster.

Other Graph databases such as `HypergraphDB`[27] were considered throughout the development of this tool, however the performance to run a query, the size of the stored data, and the documentation led us to choose the `neo4j` database instead.

We only need 3 different entities and 3 distinct types of relationships — as we can see in figure 3.10 — to store all the required information to run our tool.

Entities:

User. The `User` entity stores all the needed information about each user and also information about votes on edges. The field `userId` stores the username of the user. The field `email` contains the email that the user provided in the registering. The field `password` stores the encrypted password of the user.

The field `salt` stores the random salt generated in the registering, and it is used to encrypt the password in the application layer. The field `token` stores the token generated every time a user logs in, and the field `date` stores the expiry date of that token. The fields `posEdges` and `negEdges` store the edges that the user has voted, positively and negatively respectively.

DebateNode. The `DebateNode` entity contains all the information related to each argument.

The `DebateNode` entity has several fields that are described next. The field `nid` stores

the id of the argument. The fields *posVotes* and *negVotes* contain the positive and negative votes of the argument, respectively. The field *x* stores the computed strength of the argument. The fields *text* and *arg* contain the argument content. The field *userId* stores the owner username.

Graph. The *Graph* entity contains the information about the graph. The only field in the *Graph* entity is *gid* that stores the graph id.

Relationships:

BELONGS_TO. This relation links the *DebateNode* entity to *Graph* entity. This means that each *DebateNode* belongs to one graph.

VOTE. This relation goes from the *User* entity towards the *DebateNode* entity. Each user can vote only once on each *DebateNode*. The *VOTE* relation has only the field *type-OfVote* which can be either positive or negative.

ATTACKS. This relation goes from the *DebateNode* entity to the *DebateNode* entity. Each *DebateNode* is able to attack every other *DebateNode* in the same graph including itself. The *ATTACKS* relation also contains two fields to store the positive votes and negative votes on the attack.

3.3.5 Security

This application provides secure connections (SSL) to the two methods where a password is sent to the server: *createUser* and *existUser*. Most of the other methods require an user token to run — only the method to get all the graph nodes and the method to get all the attacks do not require a token to be executed.

Every time a user logs in a new token is generated and stored in the database. When the user tries to perform an action a token is sent to the server. The server will try to match this token with all the existing tokens in the database, if it finds a match the user may proceed with the action.

The token that is sent to the user is stored on the web browser local storage. This will let the user reload the page and keep the session. However, this stored token also has an expiration date of one hour after its generation.

The user registering requires a valid *username* (i.e. an username that is not used yet), an *email* and a *password*. When a user does his registers itself all these three fields are sent to the application layer. The application layer stores the user and the email in plain text, and encrypts the password without having access to its plain text. A random salt string is also generated and stored along with the user information.

In the login procedure only the username and the password are sent to the application layer. The application layer receive the username and get the respective salt string. Afterwards, this salt string is used to encrypt the password and the password is matched against the user password. If it matches, a true boolean is returned from the storage layer and a token is generated and sent both to the presentation layer and storage layer. Figure 3.11 illustrates the login procedure.

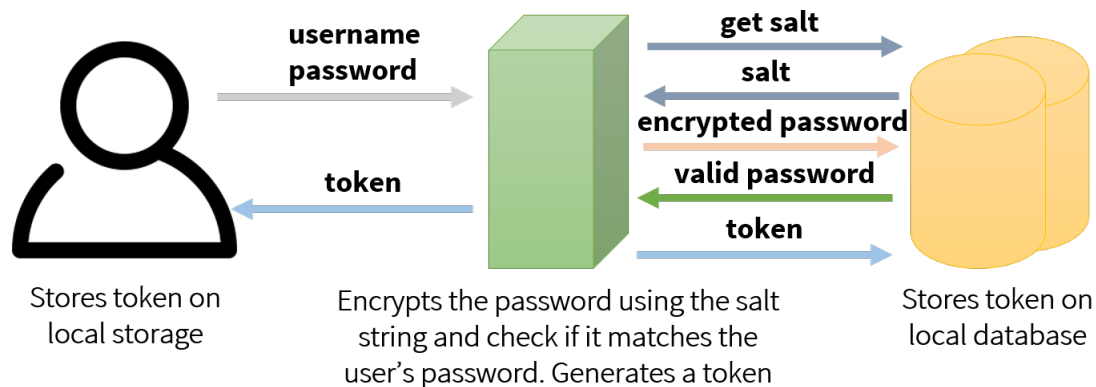


Figure 3.11: Login procedure

Password encryption. The password encryption is granted by the SHA512 algorithm. Before the encryption of the password a random generated salt string with 18 characters is added to the beginning of the password. This salt is stored along with the user information to allow the encryption of the password again. This assures that two passwords

```

1 private String get_SHA_512_SecurePassword(String passwordToHash, String
  salt)
2     throws UnsupportedOperationException{
3     String generatedPassword = null;
4     try {
5         MessageDigest md = MessageDigest.getInstance("SHA-512");
6         md.update(salt.getBytes("UTF-8"));
7         byte[] bytes = md.digest(passwordToHash.getBytes("UTF-8"));
8         StringBuilder sb = new StringBuilder();
9         for(int i=0; i< bytes.length ;i++){
10            sb.append(Integer.toString((bytes[i] & 0xff) +
11                0x100, 16).substring(1));
12        }
13        generatedPassword = sb.toString();
14    }
15    catch (NoSuchAlgorithmException e){
16        e.printStackTrace();
17    }
18    return generatedPassword;
19 }

```

Listing 3.6: Secure password generation

with the same plain text will result in two different encrypted passwords and also that the encryption of a password from a specific user will result in the same outcome since the salt used is always the same (see Listing 3.6).

3.4 Summary

In this chapter we have presented our solution. Our solution is divided in three different layers: Presentation Layer, Application Layer and Storage Layer. The Presentation Layer is divided in the User Interface and the requests to the Application Layer. The Application Layer deals with all the logic of the system, receives the requests from the Presentation Layer, and also sends the requests to the Storage Layer. Finally, the Storage Layer keeps the system state and is consulted by the Application Layer.

In the next chapter we will describe how we have evaluated our application and the results will be presented and discussed.

EVALUATION AND RESULTS

In this chapter we describe the tests that were conducted in order to evaluate our solution. We also present the results of those tests and we discuss them. The evaluation of the system is divided in three separate parts: Usability, Semantics, and Algorithm Performance.

4.1 Interface Usability Evaluation and Results

In this section we describe how we evaluated the interface usability. We start by describing the tests that were conducted and after we present and discuss the obtained results.

4.1.1 Test setup

To evaluate the interface, we did usability tests with specific tasks to cover all the interactions with the user interface. All the tests were done under supervision of the development team. Notes of the participants behaviour and doubts, were taken along the process. Before the tests some pilot tests were conducted with two voluntary participants in order to prepare the real tests and correct minor bugs and problems with the questionnaire and tasks.

Participants. The usability tests were conducted with 16 voluntary participants (13 male and 3 female) aged between 18-57 with an average age of 24.75 (standard deviation of 8.9). A large majority of the participants (81.3%) use social networks on a daily basis. The participants have different academic degrees: 4 have completed a master degree, 4 have completed high school and the other 8 participants have completed a bachelor degree.

Methodology. Before the tests each participant was given a briefing on the application.

The briefing was focused on the purpose of the application and not on the functioning, so as not to bias the participant's behavior during the tasks.

During the test, the users performed the following tasks:

- **Task 1: Create an argument**
The user must create an argument with the specific text: "Cigarettes are dangerous for your health. It does not matter if they are electronic or not."
- **Task 2: Vote on an argument.**
The user must vote positively on a specific argument, and afterward he/she must vote negatively on another specific argument.
- **Task 3.1: Create two attacks.**
The user must use the argument created on the first task and create two attacks between that argument and two other specific arguments.
- **Task 3.2: Vote on an attack.**
The user must vote on the second attack that he/she created in the Task 3.1.
- **Task 4: Delete an argument.**
The user must create an argument and vote positively on that argument. Afterwards the user deletes the argument. In order to do that the user must remove his positive vote on the argument.
- **Task 5: Consult the debate ranking.**
The user must open and consult the debate ranking.

After completing each task the user had to answer some questions in order to understand the difficulty of the task and the way he/she completed the task. The first question of every task is about the task easiness (i.e. "The task was easy to complete."). The participants answered using a likert scale, where they can choose from a scale of 1 to 5. Choosing 1 means that they strongly disagree with the statement, while choosing 5 means that they strongly agree with the statement. Also the last question of each task lets the users add suggestions and comments related to that task.

Questionnaire. The questionnaire is divided into 5 different tasks that test possible interactions with the system. These tasks were the following:

Task 1. Questions 5 to 8 of the questionnaire are related to Task 1. Question 6 asks the user how he/she created the argument. Question 7 asks the user about the easiness to understand how to create an argument.

Task 2. Questions 9 to 15 are related to Task 2 which is subdivided in Task 2.1 and

Task 2.2. Questions 10 and 13 were used to check if the task was completed successfully asking the score of the argument, after the user completed the task. Questions 11 and 14 aim to understand how did the user voted on the arguments.

Task 3.1. Questions 16 to 20 of the questionnaire are related to Task 3.1. Question 17 aims to understand if the user considers that the interface provided the proper feedback. Question 18 aim to understand how did the user created the attack. Lastly, questions 19 and 20, ask the users about the number of votes of the attack after its creation and also the score of the attacked argument after the attack. These two questions are used to check if the user successfully completed the task.

Task 3.2. Questions 21 and 23 are related to Task 3.2. Question 22 asks the score of an attacked argument after the completion of the task. This question is used to check if the user has completed the task successfully.

Task 4. Questions 24 to 26 of the questionnaire are related to Task 4. Question 24 asks the user if he completed the task successfully. Question 25 asks about the easiness of the task.

Task 5. Questions 27 to 30 of the questionnaire are related to Task 5. Question 28 is used to check if the user completed the task successfully, asking the user to copy the text of the argument placed in second in the ranking. Question 29 aim to understand how did the user opened the debate ranking.

Visualization and Navigation. In this part of the questionnaire (Questions 31 to 34) we asked the users the following questions:

- **Q31. Which color is associated to the strongest arguments?**

This question aimed to understand if the user could distinguish the strongest arguments from the weakest arguments.

- **Q32. It was easy to move (right, left, up, down) along the debate.**

In this question the users were asked on how much they agree with this sentence, using the same likert scale mentioned before.

- **Q33. It was easy to zoom in and zoom out.**

In this question the users were asked on how much they agree with this sentence, using the same likert scale mentioned before.

- **Q34. What were the main obstacles when performing these tasks?**

In this question the user could alert to any problem they found when performing tasks related to the visualization and navigation.

We also inquired the users about the system usability as a whole (Questions 35 to 38). The full usability test questionnaire is available in the Appendix A.

4.1.2 Results

In this section we will present the results to the questionnaire described previously. The table 4.1 shows the results of the first question of each task (questions 5, 9, 16, 21, and 27).

	Strongly Disagree (1)	Disagree (2)	Neutral (3)	Agree (4)	Strongly Agree (5)	AVG	SD
T1	12.5%	31.25%	12.5%	31.25%	12.5%	3	1.32
T2	0%	0%	0%	12.5%	87.5%	4.88	0.34
T3.1	0%	0%	6.25%	12.5%	81.25%	4.75	0.45
T3.2	0%	6.25%	25%	37.5%	31.25%	3.94	0.93
T4	0%	18.75%	62.5%	18.75%	0%	3	0.63
T5	0%	0%	0%	0%	100%	5	0

Table 4.1: Summary of task easiness questionnaire results. The highest scores are highlighted.

Task 1: Create an argument. In the questionnaire section about the first task (Questions 5 to 8), we asked the users to express their opinion about the statement "Understanding how to create an argument was easy." (Question 7) using the same scale from 1 to 5 used before, and how did they created the argument. The results reveal that the participants disagree with the statement (the average is 2.5 with a standard deviation of 0.97).

To the question about how the users have created the argument (Question 6), 15 users used the *right click pop up menu* and only 1 of the participants have used the *double click*. There were also 2 users that suggested the creation of a button to add a new argument.

Task 2: Vote on an argument. Since there are several ways to vote positively or negatively on an argument, the users were asked how did they vote on the argument (Question 11 and 14). This question was asked twice because the users had to vote positively and negatively on different arguments. The aggregate values revealed that the participants used the *buttons in the argument preview* 14 times, the *graph view* 17 times and the *view with the argument opened* only once.

Task 3.1: Create an attack. On the questions related to the task 3.1 (Questions 16 to 20), the participants were asked to express their opinion about the following statement "The system gave clear evidence of the success of the task." (Question 17). The answers to this question reveal that the majority (14 participants) strongly agree (5) with the statement and the other 2 participant agree (the average is 4,875 and the standard deviation is 0.34).

The participants were also asked how did they created the attacks (Question 18). The answers to this question showed that 3 of the participants used the bolt symbol in the argument and the remaining 13 participants used the right click pop up menu to start the attack.

Task 3.2: Vote on an attack. The task 3.2 of the questionnaire (Questions 21 to 23) aimed to understand if the user could perform a vote on an attack. The notes taken during the task reveal that some of the participants had difficulties to click on the attack and make the pop up to vote appear.

Task 4: Delete an argument. In the questions related to the task 4 of the questionnaire (Questions 24 to 26), the results revealed that every participant was able to complete the task, however some suggestions were made by the participants during the process. Some of the users referred that the message shown to inform the users the conditions to delete an argument is small and insufficient to understand why they can not delete the argument.

Task 5: Consult the debate ranking. Question 29 asked the participants how did they opened the debate ranking. The answers revealed that 2 participants used the *right click pop up menu* and the remaining 14 used the *button on the top menu* of the application.

The questions 10, 13, 22, and 24 were introduced to verify if the tasks were completed in the correct way. These tasks asked the user to enter the value of a particular argument after the completion of the task.

Visualization and Navigation. The first question was about the color relation with the strength of the arguments (i.e. Which color is associated to the strongest arguments?). The participants could choose between 4 different colors (Green, Yellow, Red and Blue) and the results showed that 15 users associated the green color with the strongest arguments and only 1 participant thought the strongest arguments were red. Furthermore we observed that the majority of the users had to check the argument strengths to answer this question. Three participants also commented that for them, the red color would be more intuitive to represent the strongest arguments.

In the other two questions the users were asked to comment two statements with the same likert scale that was used previously. With the statement "It was easy to move (right, left, up, down) along the debate." All the participants strongly agreed. The other statement asked about the difficulty to zoom in and zoom out ("It was easy to zoom in and zoom out."). There were 15 participants answered that they strongly agree and only one who answered he/she agrees (the average is 4.94 and the standard deviation is 0.25).

General questions. This part of the questionnaire was based on an adaptation of the

System Usability Scale (SUS) — proposed by John Brooke in 1996 [9] — and comprises the statements on Table 4.2.

Statements
Statement 1. It is easy to learn how to use the system.
Statement 2. I found the system unnecessarily complex.
Statement 3. I thought the system was easy to use.
Statement 4. I think that I would need the support of a technical person to be able to use this system

Table 4.2: Statements that composed the general evaluation part of the questionnaire.

Table 4.3 presents the percentage of users for each option of the scale for every statement. As shown in Table 4.3 most of the users agree that it is easy to learn how to use the system and disagree that the system is unnecessarily complex. Furthermore most of the participants agree that the system is easy to use and disagree that they would need the support of a technical person to be able to use the system.

4.1.3 Discussion and Analysis

In this section we will analyze and discuss all the tasks and try to identify possible usability problems and improvements. Firstly, we will focus on the different tasks and features tested by the participants. When we analyze Table 4.1 three of the tasks (T1, T3.2, and T4) have an average below 4 (Agree). Consequently we will examine in detail the participants evaluation of these 3 distinct tasks.

Task 1: Create an argument. The first question of task 1 (question 5) had an average score of 3, however participants voted on every option of the likert scale concerning the task difficulty — 5 voted on Disagree and 5 on agree, each of the other option had 2 voters. Furthermore, the question that asked about the ease of understanding on how to perform the task had an average score of 2.5 emphasizing that is difficult to understand how to create an argument and hence it may be also difficult to complete the task.

There were two possible solutions that were pointed out by the users: Add a button to create an argument and warn the users about the possibility to use the right click and

	Strongly Disagree (1)	Disagree (2)	Neutral (3)	Agree (4)	Strongly Agree (5)	AVG	SD
S1.	0%	0%	12.5%	75%	12.5%	4	0.55
S2.	6.25%	68.75%	25%	0%	0%	2.21	0.58
S3.	0%	0%	6.25%	81.25%	12.5%	4.07	0.47
S4.	25%	75%	0%	0%	0%	1.79	0.43

Table 4.3: Summary general evaluation questionnaire results. The highest scores are highlighted.



Figure 4.1: Add argument button

the double click.

The users do not have any introduction on how to use the system, therefore it is difficult for most of the user to guess how to create an argument since there are no instructions, no explicit buttons or visual indications on how to do it.

To overcome this issue we decided to add a new button on the right side of the screen that will allow the users to create a new argument (Figure 4.1). In addition a textual indication about the possibility to use the right click was also added to the box that appears when the mouse is over an argument.

Task 3.2: Vote on an attack. This task had an average score of 3.94, 5 votes had a score under 4. These results suggest that although the majority of the users could complete the task without much difficulty, the task is not intuitive to some of the users. Some participants also referred that it was difficult to select an attack when the arrow was thinner (i.e. lower strength).

In order to reduce the task difficulty, a textual indication explaining that the attack must be clicked in order to vote on it, was added to the box that appears when the mouse is over an attack (arrow).

Task 4: Delete an argument. This task had an average score of 3 revealing that deleting an argument it is not an easy task to perform. An argument can only be deleted if it has 0 votes (positive and negative) and if it is owned by the current user. This means that we only want to allow users that created an argument unintentionally to delete the argument.

The task asked the user to create an argument, vote positively on that argument and delete the argument. The delete button was disabled while the positive vote had not been removed. A message with the conditions to delete an argument is shown as a *tooltip* when the user overs the delete button, however the participants commented that they were not able to see the message and that a pop up message should appear to make the conditions to delete perfectly clear for the user.

To overcome the presented issue we decided to add a pop up modal presenting the necessary conditions to delete the argument. This modal also has a confirmation button in order to assure that the user read these conditions.

The tasks *T2*, *T3.1*, and *T5* had average scores above 4, nonetheless there are also some observations and improvements that can be made in some of the related features. In task *T3.1* the participants were asked to create an attack between two arguments, although the results suggest that the task was easy to complete (average score = 4.75) users commented that the modal that appears after the start of the attack should have a confirmation button and only disappear after the user clicked that button.

The tasks *T2* and *T5* have averages of 4.88 and 5 respectively, with only two votes with the score of 4 on the task *T2*, the remaining votes had the score of 5. Consequently we considered that the features related to these tasks do not need to be improved.

Visualization and navigation. One of the users considered that the strongest arguments were represented with the red color. Three users also mentioned that the color red, is associated with strength and the reason they have chosen green is related to the textual indication of the argument's strength.

Despite these observations, there was only one user that did not associate the green color with the strongest arguments and considering that only a minor part of the users did not found this association intuitive (between the green color and the strongest arguments), we decided to keep the system as it is. The fact that the acceptance of the arguments is displayed as their strength, may have deceived the users, and make them associate the red color with the strongest arguments.

The participants found it easy to move and zoom along the debate graph, using the buttons that exist for this purpose and also the mouse (dragging and scrolling).

General Evaluation. The results of the questions 35 to 38 may indicate that there are still some improvements that must be made to the interface, however it is already acceptable.

To sum up, the system usability is acceptable as a whole although there are some critical issues that were solved in order to allow the user to perform all the possible interactions. The features that were improved are the following:

- Add an argument.
- Vote on an attack.
- Delete an argument.

Some improvements related to the interface design and also other minor issues may also be considered in the future.

4.2 Semantics Evaluation and Results

In this section we are trying to understand the users' vision about several aspects of the semantics, such as the definition of argument and attack, the definition of positive and

negative votes, the definition of argument and attack strength, and the influence of the attacks and votes on the strength of an argument.

4.2.1 Test Setup

To evaluate the semantics we conducted some tests with one group of 15 voluntary participants (none of the users of this group has participated on the interface usability tests). The application was set up on a Ubuntu Server 16.04.3 [24] running a Tomcat8[19] and neo4j[28] community edition. A username and a password was provided to each user. The username was not related to the participants' name so that it would not be possible to relate each user to the person itself.

The users were given the system with 3 arguments and 2 attacks already created and also a main subject to start a debate. The main subject of the debate was euthanasia and assisted suicide. The participants used the system for a four days period, creating debates with a relevant number of arguments and attacks. After these tests, all the participants answered a questionnaire (see Appendix B) where we tried to learn and understand more about the following questions:

1. What is an argument for the users?

With this question we are trying to understand the definition of argument for a user. An hypothesis we may consider that an argument will be a comment about the subject. However some more debate-experienced users may define an argument in the formal way.

In order to understand the definition of argument for the users we introduced 3 different questions (Questions 7, 8, and 11). Questions 7 and 8 ask about the user's and the other users' conception of an argument, respectively. Question 11 presents several statements and asks the user to select the ones that he/she consider to be arguments.

2. What is the reason behind positive/negative votes on the arguments?

In this question we are trying to understand what means a positive vote and a negative vote to most of the users. The user may vote positively or negatively due to several different reasons:

- a) The argument is/is not well-structured.
- b) The argument is structured, well formed, and the user agrees/does not agree with its premises.
- c) The argument is structured, and the user agrees/does not agree with its conclusion.
- d) The argument is a comment that the user agrees/does not agree with.
- e) The argument is a comment that the user finds funny/disrespectful.

There could also be other reasons that may influence the user to provide a positive or a negative vote that we were not aware of before the tests.

In order to better understand the meaning of votes on arguments 2 questions (Questions 9 and 10) were introduced on the questionnaire. These questions go straight to the point, asking the users' reasons to vote positively and negatively on an argument. The options presented to the users are the reasons listed above. Moreover, the users were able to add other different reason from the ones that were presented.

3. What is the meaning of attacking other argument? What is the difference between an attack and a negative vote on an argument?

With these questions we are trying to understand in which situations an user creates an attack between two arguments and also the difference between an attack and a negative vote. In order to understand this, questions 16, 17, and 19 were introduced to the questionnaire.

Question 16 presents two different arguments and asks the user what attacks he/she would create between the two arguments. Questions 17 and 19 address the question related with the difference between a negative vote and an attack, asking the users what they would do if they found an argument (B) that they do not agree with or found disrespectful. The options available are: (1) Create an argument and use it to attack the argument B; (2) Vote negatively on argument B; (3) None of the above.

4. When do the users vote positively/negatively on an attack?

In this question we are trying to understand when do the users vote positively/negatively on an attack. Question 18 was introduced in the questionnaire in order to answer to this specific question. Although, some of the questions related to the argument strength can also give clues to answer this question.

Question 18 presents two different arguments to the user, with an attack between them, and asks the user how and if he/she would vote on that specific attack.

5. Do the users understand and agree with the strength of the arguments and how they change? Do the users perceive how the attacks influence the arguments' strength?

With this question we are trying to understand the users' view about the score that is assigned to each argument, taking into account the votes on arguments and attacks, and also the attacks between the arguments. In order to answer this, questions 12 to 15 and 20 to 31 were introduced in this questionnaire. Questions 12 and 13 present the users 7 unattacked arguments and their respective positive and negative votes, and ask the users to assign a score to each argument — within a range¹— and to

¹The different options are the following: Very low (0 to 20), Low (20 to 40), Medium (40 to 60), High (60 to 80), and Very High (80 to 100)

arrange the arguments in the descending order, respectively. Questions 14 and 15 are similar to the previous questions but present only 3 arguments that attack each other.

Questions 20, 21, 30, and 31 focus on the influence of an attack and its dependence on the source argument score. Questions 22 to 29 target the influence of the number of votes on the arguments' strength/score.

There was also an initial question (Question 5 of the questionnaire) that asked the users' how well the scores/strengths assigned by the application reflected the overall view of the debating group. The users answered using a likert scale, where they can choose from a scale of 1 to 5, choosing one means that they strongly disagree with the statement and 5 means that they strongly agree.

The full questionnaire is available in the Appendix B.

4.2.2 Results

The tests were conducted with 15 voluntary participants (11 female and 4 male) aged between 18-29 with an average age of 22.07 (with a standard deviation of 3.1). A large majority of the participants (93.3%) use social networks on a daily basis. The participants have different academic degrees: 6 have completed a master degree, 6 have completed high school and the other 3 participants have completed a bachelor degree. These tests resulted in a debate with 19 arguments, 58 attacks, and 115 votes. The resulting graph of the debate can be seen in Figure 3.2.

Question 5 that asked the users about their overall view of the scores/strengths assigned by the application had the following results:

- The majority of the participants (9) agrees (4) with the scores/strengths assigned by our application.
- 3 participants strongly agree (5) with the scores/strengths assigned by our application.
- 3 participants are neutral (3) about the scores/strengths assigned by our application.

Therefore this results in an average of 4 with a standard deviation of approximately 0.65.

Questions 7 and 8 that were focused on the definition of argument to the users. We got the results that are presented in Table 4.4, where the percentage of users that chose each option is displayed (note that the participants could chose both options).

Table 4.5 presents the results to the question 11 of the questionnaire.

This table shows that only the structured arguments with true premises were considered arguments to the users. Note that the third sentence is a structured argument where the conclusion is implicit.

What is an argument...		
	To you	To the other users
<i>A structured assertion (i.e. with premises and conclusion)</i>	86.7%	80.0%
<i>An unstructured comment</i>	13.3%	33.3%

Table 4.4: Results to the questions 7 and 8 of the questionnaire.

Which of the following sentences do you consider to be arguments?		
S1. <i>"I don't agree!"</i>		0%
S2. <i>"Animals are born in the wild, and they are supposed to be living in the wild too. Therefore, you should not lock any kind of animal in your home!!"</i>		93.3%
S3. <i>"There are many types of animals and some of them are already adapted to live inside."</i>		66.7%
S4. <i>"Snails are slow, and I like snails. So snails should live inside."</i>		0%
S5. <i>"I like turtles!"</i>		0%

Table 4.5: Results to the question 11 of the questionnaire.

Questions 9 and 10 are intended to understand the reasons why users vote positively and negatively on an argument. Tables 4.6 and 4.7 present the results to this two questions.

When you voted positively on an argument what did you want to express?		
<i>"The argument is well structured, independently of whether I agree with the conclusion."</i>		20%
<i>"The argument is structured, well formed, and I agree with its premises."</i>		66.7%
<i>"The argument is structured and I agree with its conclusion, independently of everything else."</i>		46.7%
<i>"The argument is a comment that I agree with."</i>		46.7%
<i>"The argument is a comment that I find funny"</i>		0%

Table 4.6: Results to the question 9: Reasons to positive votes

When you voted negatively on an argument what did you want to express?		
<i>"The argument is not well structured."</i>		40%
<i>"The argument is well formed and I do not agree with its premises."</i>		60%
<i>"The argument is structured and I do not agree with the conclusion."</i>		46.7%
<i>"The argument is a comment that I do not agree with."</i>		53.3%
<i>"The argument is a comment that I find offensive/disrespectful."</i>		26.7%

Table 4.7: Results to the question 10: Reasons to negative votes

Questions 16, 17 and 19 are focused on the meaning of an attack and the difference between an attack and a negative vote on an argument.

Question 16 presented the users the situation in Figure 4.2(a) and ask them how would they create an attack. The results are presented in Figure 4.2(b).

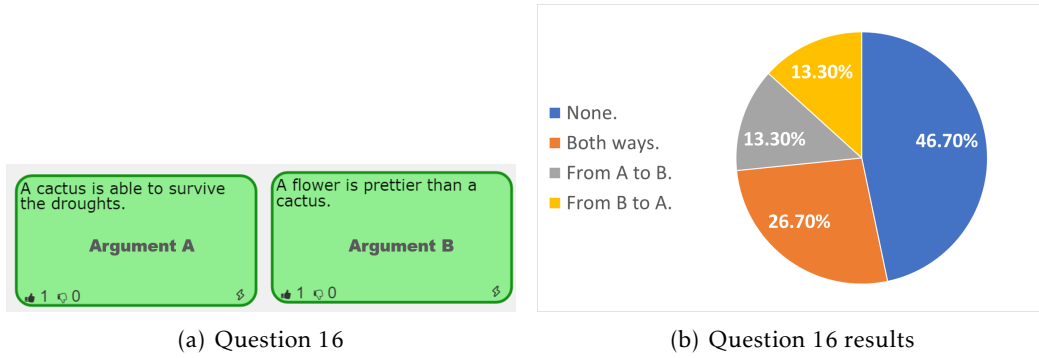


Figure 4.2: Question 16

	V^+	V^-	Mode Score (Users)	Actual Score (Application)	Mode Order (Users)	Actual Order (Application)
A	22	2	VH (80 to 100)	VH (80 to 100)	#1	#2
B	1	0	L (20 to 40) H (60 to 80)	VH (80 to 100)	#2	#1
C	2	2	M (40 to 60)	M (40 to 60)	#3	#3
D	1	23	VL (0 to 20)	VL (0 to 20)	#6	#5
E	0	2	VL (0 to 20)	VL (0 to 20)	#5	#7
F	42	43	M (40 to 60)	M (40 to 60)	#4	#4
G	1	43	VL (0 to 20)	VL (0 to 20)	#7	#6

Table 4.8: Results to questions 12 and 13 and comparison with the scores assigned by the application. V^+ - Positive Votes, V^- - Negative Votes, VH - Very High, H - High, M - Medium, L - Low, VL - Very Low

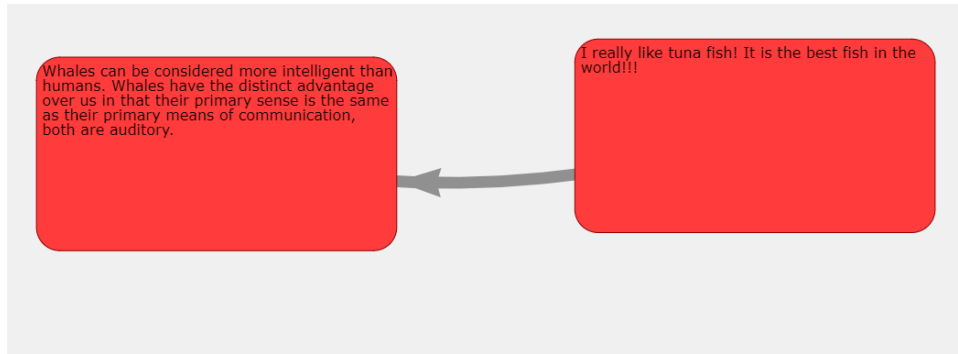
In the question 17, users revealed that, when they do not agree with an argument or find it disrespectful, almost every user would vote negatively on that argument (86.7%). There were also 10 users (66.7%) that said that they would create an argument and attack the argument they do not agree with or find disrespectful. Lastly, only one user (6.7%) considered that he/she would not perform any of the actions above.

In the question 19, 13 users said that they would not create an argument saying "I do not agree", in the case they do not agree with an argument or find the argument disrespectful. Only 2 users (13.3%) chose the opposite option.

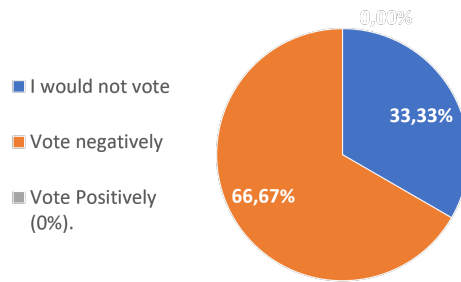
Question 18 was intended to understand when the users voted positively or negatively on an attack. This question displayed the situation in Figure 4.3(a) and had the following results (Figure 4.3(b)).

Questions 12 and 13 focus on the users' understanding of the scores/strengths assigned by the semantics to each argument, without taking attacks into account. Table 4.8 summarizes the results of both questions, comparing them to the actual score/strength calculated by our application.

Questions 14 and 15, are similar to questions 12 and 13 but the debate has only three



(a) Question 18



(b) Question 18 results

Figure 4.3: Question 18

	V+	V-	Mode Score (Users)	Actual Score (Application)	Mode Order (Users)	Actual Order (Application)
A	30	27	M (40 to 60)	M (40 to 60)	#1	#1
B	1	0	M (40 to 60)	M (40 to 60)	#2	#2
C	2	22	VL (0 to 20)	VL (0 to 20)	#3	#3

Table 4.9: Results to questions 14 and 15 and comparison with the scores assigned by the application. V^+ - Positive Votes, V^- - Negative Votes, VH - Very High, H - High, M - Medium, L - Low, VL - Very Low

arguments and two attacks (i.e. A attacks B and B attacks C). Table 4.9 summarizes the results of questions 14 and 15, comparing them to the actual score/strength calculated by our application.

Questions 20 to 31 focus on the users' valuation on the strengths/scores assigned to each argument. Questions 22 to 29 are related with the influence of the votes on the arguments' strengths/scores. Questions 22 to 25 present the users two arguments with distinct values of positive and negative votes. Table 4.10 compiles all the results to those questions.

Questions 26 to 29 also present the users two arguments with distinct positive and negative votes, but the score/strength assigned to each argument is showed to the participants. The participants are then asked to say if they agree or not with the score assigned

Which one should be the strongest (Q. 22 & 24)/weakest (Q. 23 & 25) argument?							
	A ⁺	A ⁻	B ⁺	B ⁻	A	B	None
Q. 22	100	0	50	0	80%	0%	20%
Q. 23	0	100	0	50	66.7%	20%	13.3%
Q. 24	5	1	124	121	33.3%	53.3%	13.3%
Q. 25	13	19	234	240	6.7%	66.7%	26.7%

Table 4.10: Results to questions 22 to 25. A⁺ - Positive Votes of argument A, A⁻ - Negative Votes of argument A, B⁺ - Positive Votes of argument B, B⁻ - Negative Votes of argument B

by our application and to explain why (questions 27 and 29).

In the questions 26 and 27 argument A had 0 positive votes and 1 negative vote and a score of 0, while argument B had 1 positive vote and 50 negative votes and a score of 1.96. The majority of the users (60%), agreed with the score assigned by the application, explaining that, since B has one positive vote while A has 0 positive votes, B should have a slightly higher score/strength than A. The other 40% of the participants did not agree with the score assigned by our application, arguing that if there are 50 users voting negatively on argument B and only one voting negatively on A, B should be weaker than A, even if it has 1 positive vote.

In the questions 28 and 29 argument A had 1 positive votes and 0 negative votes and a score of 100, while argument B had 50 positive votes and 1 negative vote and a score of 98.04. Almost every participant (93.3%) did not agree with the score assigned by the application, the users mentioned that, since B has a greater number of positive votes it should be stronger than A, and that one negative vote should not make it weaker than A. There was only one participant that agreed with the score assigned by the application. That user mentioned that an argument that has no negative votes should be stronger than an argument with negative votes, even if it is only one.

Finally, questions 20, 21, 30, and 31 are related with the influence of the attacks on the arguments' score. In the question 20 and 21 a situation is presented to the user with two arguments A and B, where A has no votes and B only has one positive vote. After A attacks B, and the attack has 1 positive vote, the user is asked what will it happen to score of B: (1) Decrease; (2) Maintains; (3) Increase.

The majority of the participants (60%), considered that the score of B would not change after the attack. The participants explained that, since argument A has a score of 0 it should not have any impact on B. There were also 26.7% of the participants considering that the score of B would decrease. The only reason pointed out by the participants was that when an argument is attacked its score should decrease. Lastly, only 13.3% of the participants considered that the score of B should increase, pointing out that, if the attack is strong and the argument A is weak, the score of B should increase.

Questions 30 and 31 also presented two arguments, A and B, where A has 1 positive votes and 0 negative votes, and B has 50 positive votes and 0 negative votes, A attacks

B with an attack with 1 positive vote. The score assigned by our application to each argument is presented to the users — A: 100, B: 0. The users are asked if they agree, and why they agree with those scores. There was no users agreeing with the scores/strengths assigned by our application, the participants pointed out that A has only one vote, therefore it should not decrease the score of B to 0 since it has 50 votes. There was also one user that mentioned that when an the argument A attacks argument B, the number of positive votes of A should be transformed to negative votes on B.

4.2.3 Discussion and Analysis

In this section we will discuss and analyze all the results and try to identify problems and possible improvements that can be made in the semantics.

Firstly, we will start by analyzing and discuss the questions related to the definition of argument. The results to questions 7, 8 and 11 — presented on Tables 4.4 and 4.5 — revealed that most of the users consider that only structured arguments can be considered arguments, although there will be some users that will consider an "*unstructured comment*" also an argument. Furthermore, question 11 shows that structured arguments that are not well formed, are also not considered arguments. We can conclude that, although the users can freely create their arguments, they understood an argument should be well structured and well formed in order to be relevant in a debate.

Questions 9 and 10 focus on the reasons behind a positive and a negative vote on an argument. When we analyze Table 4.6 we reach the following conclusions. The majority of the users vote positively on an argument, when that argument is well formed, structured and they agree with the premises. There were also nearly half of the participants (46.7%) that vote on an argument when they agree with the conclusion, whether the argument is structured or unstructured. Given these points, the most important factors are the users' agreement with the premises and the conclusion on an argument.

Now we will analyze the results related to the meaning of attacking other arguments and the difference between an attack and a negative vote. The results related to the difference between an attack and a negative vote reveal that:

- The majority of the participants (86.7%) would vote negatively on an argument that they do not agree with or find disrespectful.
- The majority of the participants (66.7%) would create an argument to attack an argument that they do not agree with or find disrespectful.

Although 66.7% of the participants said they would create an argument and attack the argument they do not agree with or find disrespectful, they would only do this if they have a meaningful argument to add to the debate — as it was showed by the results of question 17.

Question 16 presented the users two arguments that should not attack each other. However argument A presents a fact, while argument B presents an opinion. The results

to this question were dispersed (Figure 4.2(b)). It seems that part (46.7%) of the participants would not create any attack, and also a smaller part of the participants (26.7%) would create an attack in both ways. One half of the remaining participants (13.3%) would create an attack from A to B, and the other half from B to A (13.3%). The majority of the users would not create any attack or create an attack in both ways. This seems to happen because the different views of what is a debate to the users. On the one hand, if the user considers that the debate is a battle of opinions where it does not matter if the opinions are facts or not, then they chose to attack both ways. On the other hand, if the user embraces the meaning of a debate where the arguments are based on facts and only attack other arguments if they contradict or invalidate them, then they would not create any attack.

When we analyze the results to question 18 (Figure 4.3(b)), that is related with the reasons behind the votes on attacks, we can conclude that, the participants seem to understand the meaning of an attack and the impact of a vote on an attack. The majority of the participants (66.67%) would vote negatively on the attack. This makes sense since the argument that is attacking is not even related to the attacked argument. The remaining participants (33.33%) chose not to vote.

Lastly, we will analyze the results related to the arguments' score/strength and the influence of the attacks on the arguments score/strength. Table 4.8 shows the results to questions 12 and 13 of the questionnaire. If we analyze this table it is possible to understand that the score that most of the participants suggested for each argument, match the score assigned by the application semantics, for every argument except B. The fact that argument B has only one positive vote (and 0 negative votes) seems to have led the participants to doubt about the strength that argument B should have. This indecision was revealed also by the results since there were 2 participants that thought that the strength should be very low, 3 that thought it should be low, 2 that thought it should be medium, 3 that thought it should be high, and 1 that thought it should be very high. We can infer some reasons for this dispersion of results in the case of argument B:

- Since argument B has only one vote some users, may have considered that its score should not be too high.
- The total number of votes should influence the arguments' score. In an example where are arguments with 42 positive votes a user may consider that an argument with only 1 positive vote should have a low score.
- Since argument B has only positive votes its score should be very high.

Considering the order of the arguments, we notice that there are several arguments (i.e. A, B, D, E, and G) that were not placed in the same order that our semantics produced. Arguments A and B are misplaced (i.e. argument A should be on the 2 position and argument B on the first), this reinforces that the total number of votes should be taken into account when the score is calculated. Also arguments D, E and G are misplaced

between them. Argument E has the lowest score in our application since it has no positive votes, and 2 negative votes, arguments D and G both have one positive vote, but a higher number of negative votes — D has 23 negative votes, and G has 43. Again this seems to support that the participants think that the total number of votes should be taken into consideration when the score of each argument is calculated.

The results to the questions 22 to 25 are indicators that can help to identify the influence that the total number of votes should/should not have in the score of an argument. Table 4.10 presents the results to these questions, and it is possible to identify that in the questions 22 and 23 the majority of the users chose the arguments that have the greater number of votes, positively or negatively. In fact, in our application, both arguments have the same score, but, as we noticed before, the total number of votes is a parameter that the participants take into account when comparing arguments.

In the questions 24 and 25, the arguments A and B have the same difference between positive and negative votes (i.e 4 in the question 24 and -6 in question 25). Although they have the same difference between positive and negative votes the total number of votes is notably higher in argument B. Therefore, the participants considered that argument B was the strongest in question 24 and the weakest in question 25. In question 25 argument B had an higher number of positive votes than in argument A, and in question 26 it had a higher number of negative votes than in argument A. Once again, the participants considered the total number of votes as an important factor when calculating the score of an argument.

Questions 26 to 29 are also focused on the importance of the total number of votes when the score of the arguments is calculated. In the questions 26 and 27, 60% of the participants agreed with the score assigned by our application. According to the comments the participants provided it seems that, although the participants consider that the total number of votes should be taken into account to calculate the score of an argument, the absence of positive votes should also be an important factor.

Questions 28 and 29 emphasize the importance of the total number of votes to the participants. Every participant except one (93.3%) did not agree with the score assigned by the application. The users also considered that the total number of votes should be taken into consideration (*"Argument A should not be stronger, or as strong as B, because only one person voted (positively) for A, and 50 people voted (positively) for B."*).

Questions 14 and 15, focus on the influence of the attacks on the arguments' score. As we can see in Table 4.9, both the results, the scores and the order, match with the scores produced by the application. It seems that the users understand and agree with the way the attacks influence the arguments' score. However, later in this section we will present some examples where the influence of the attack does not produce the "expected" result.

In the questions 20 and 21 the majority of the participants (60%) considered that the attacked argument score would not change after the attack. This reveals that the users agree that, when an argument with a score equal to 0 attacks another argument, the score of the attacked argument should not change.

Questions 30 and 31 presented two arguments, A and B, both with only positive votes and no negative votes. Argument A has only one positive vote while argument B has 50 positive votes. There is also an attack from A to B, with 1 positive vote and 0 negative votes. The scores assigned by our application are displayed along with the arguments. The fact that none of the participants agreed with the score, and the reasons they pointed out can provide clues to how they think the attacks and the votes should influence the arguments' score:

- The total number of votes should have influence on the score of A and, consequently the score of B should not decrease to 0.
- If the number of positive votes of A was converted to negative votes on B (as suggested by one of the participants) the score of B would decrease only a little, and the score of argument A would still be the highest.

Question 5 and 6 revealed that the majority of the users agree that the strength of the arguments reflect the overall view of the debating group. In the question 6 some users mentioned that as the strengths of the arguments were changing over the course of the days you can see the debate evolving and the most relevant arguments standing out. However there are some situations that were identified throughout this discussion where the behaviour is not what the participants expected. Moreover, we also identified another situation that we will describe next.

Consider the situation represented in Figure 4.4, where we have two arguments, A and B, both arguments have a score of 100 (e.g. 1 positive vote and 0 negative votes). The attack from A to B has a score of 100 and the attack from B to A has a score of 50.

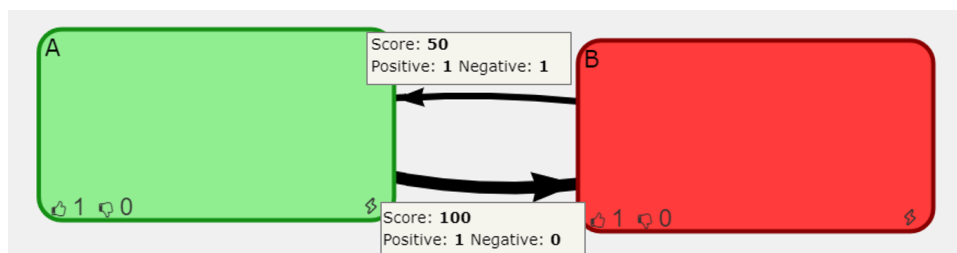


Figure 4.4: Cycle of attacks between an argument A and B with the same votes on arguments but different number of votes on attacks

When we look at this situation we expect that argument A will be stronger than argument B, however we do not expect that its score is 100, and we also do not expect that the score of argument B is 0. An argument should only have the maximum strength when everyone agrees with it. Although argument A only has positive votes, there are at least one user (e.g. 1 positive vote and one negative vote in the attack from B to A) that thinks that the argument B attacks argument A. Therefore there is at least 1 user that does not agree with argument A, hence it should not have the maximum score and consequently B should not have the minimum.

Our semantics use the strength of the attackers to influence directly the strength of the attacked arguments (see Section 2.1.2.1). The strength of the attack that goes from A to B is two times the strength of the attack that goes from B to A. Consequently, the values change drastically from iteration to iteration and only converge when A is 100 and B is 0. It is also important to mention that this only happens when both of the arguments have the maximum strength (100). If the arguments have lower strengths (e.g. 50) the semantics do not converge to the maximum (100) and minimum (0) values. For instance, if we have the same situation — but the strengths of the arguments A and B is 50 instead of 100 — after the attacks, the strength of A will be 43 and the strength of B will be 29.

To sum up, we have reached some important conclusions that may help to improve the semantics and the application:

- The users generally understood what is an argument, an attack, votes on arguments and votes on attacks.
- The users understood that they can express their disagreement using negative votes and also attacking other arguments.
- The strength of the arguments should change more smoothly.

4.3 Algorithm Performance Evaluation and Results

In this section we will explain the methods that we used to evaluate the algorithm performance. Afterwards we present the results of the evaluation of the algorithm performance.

4.3.1 Test Setup

The methods used to evaluate the algorithm performance are similar to the ones used by Correia et al. [13] to test the algorithm implemented in C++.

All algorithms were implemented in Java and compiled with Java 1.8. The tests were performed on an Intel Core i7 CPU @ 2.3GHz, running Windows 10. To test the algorithm performance we registered the times that the algorithm took to run, varying the number of nodes and the graph density. The graph density is defined in the following way:

$$D = \frac{2 \times \text{Number of Attacks}}{\text{Number of Arguments} \times (\text{Number of Arguments} - 1)}$$

The algorithm was tested with 9 different subsets. Each subset consists of 1000 graphs generated randomly, size and density are uniformly distributed between the bounds defined for each subset. The initial weight of each argument and attack was also generated by an uniform random distribution, and the guess for the argument strength was set to the argument initial weight. The tolerance was set to 10^{-12} . Table 4.11 presents the boundaries of each generated subset.

To evaluate the algorithm runtime after user interactions we generated 9 subsets with the same generator used before but with a different seed. The algorithm ran for

	Minimum Number of Nodes	Maximum Number of Nodes	Minimum Density	Maximum Density
Subset 1	10	100	0	0.01
Subset 2	10	100	0.01	0.1
Subset 3	10	100	0.1	1
Subset 4	100	1000	0	0.01
Subset 5	100	1000	0.01	0.1
Subset 6	100	1000	0.1	1
Subset 7	1000	10000	0	0.01
Subset 8	1000	10000	0.01	0.1
Subset 9	1000	10000	0.1	1

Table 4.11: Algorithm Performance Evaluation: Test Subsets

all the generated graphs. Afterwards we introduced 5 to 14 modifications to the graph (the number of modifications is randomly uniformed distributed) and ran the algorithm again.

These tests goal is to evaluate the algorithm performance in a more realistic scenario where the debate is always changing and the new strengths are being calculated with the previous strengths (i.e. the initial guess is set to the strength calculated previously).

4.3.2 Results

Our tests resulted in the graphs represented in Figure 4.5. The graph on the Figure 4.5(b) shows that for debates with one hundred arguments or less, the algorithm runs instantaneously.

For debates with a number of arguments between 100 and 1000 (figure 4.5(d)) the algorithm runtime is always below 0.5 seconds. For debate graphs with more than 1000 arguments the runtime of the algorithm increases considerably reaching runtimes of 3 seconds for graphs with ten thousand arguments and density near 0.1 (see Figure 4.5(f)).

For debates with densities below 0.01 the longest runtime is near 0.5 seconds for graphs with approximately ten thousand arguments (see Figure 4.5(a)). Graphs with ten thousand arguments with densities between 0.01 and 0.1 (see Figure 4.5(c)), reach runtimes of 3 seconds.

The longest runtime, of 64 seconds, is for a complete graph with nearly ten thousand arguments and density close to 1.0^2 (see Figures 4.5(e) and 4.5(f)).

The evaluation of the algorithm behaviour after a few modifications are introduced in the debate resulted in the graphs represented in Figure 4.6.

The graphs are divided by classes of number of modifications, number of nodes and density. It is possible to observe that the difference from the times before and after the modifications is bigger when the number of modifications in lower (see Figure 4.6(a)). For

²A graph with a density of 1.0 is a complete graph where every node is connected to all the other nodes and also itself. This means that a graph with 10000 arguments will have 100 millions of attacks.

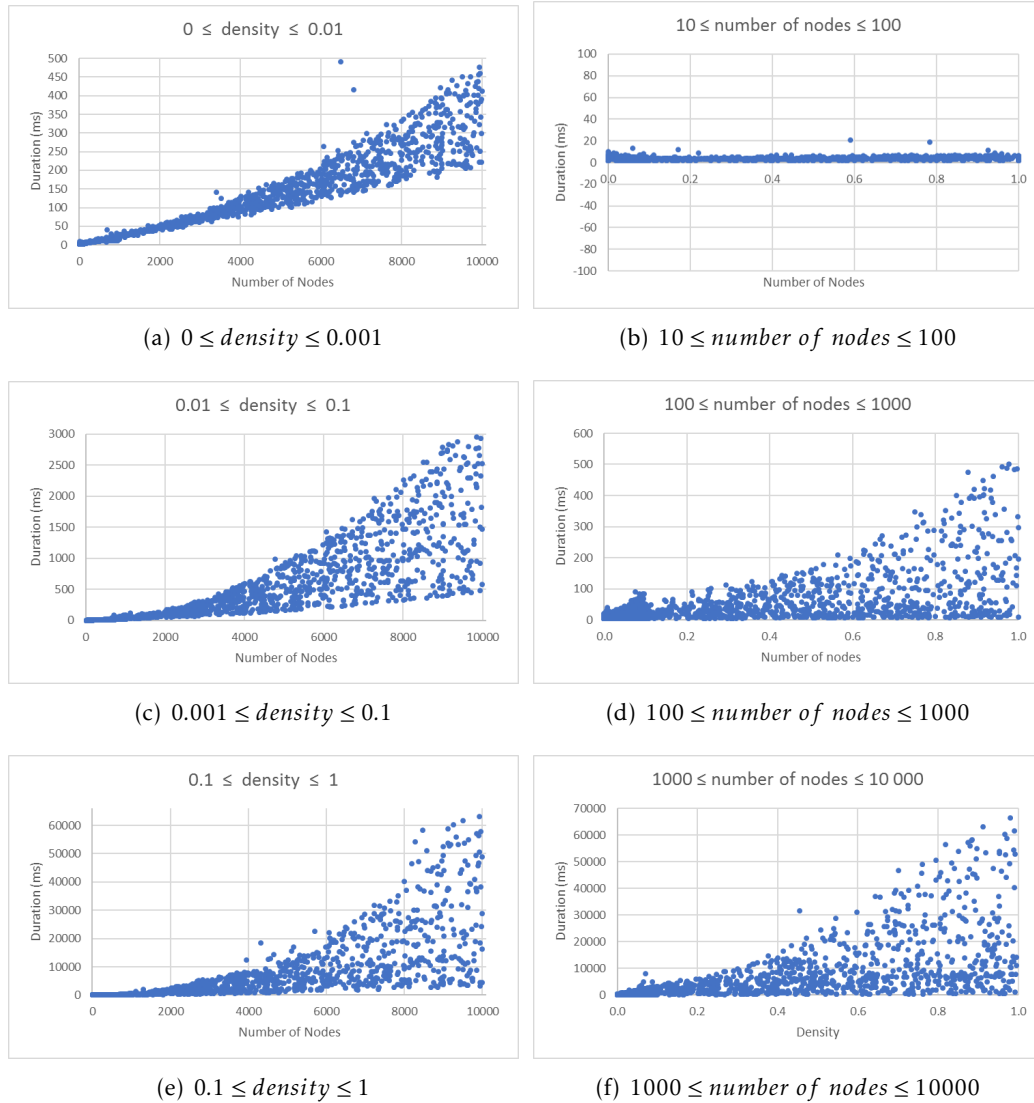


Figure 4.5: Performance of the algorithm as function of the size and density of the graphs

a higher number of modifications this difference is smaller and also there are more cases of higher duration as we can see in Figure 4.6(c).

It is possible to observe that in every graph of the Figure 4.6 the difference between the first iteration of the algorithm (before the modifications) and the second iteration of the algorithm, increases with the growth of the number of nodes and density.

Table 4.12 shows the average difference of runtimes where we introduced c modifications, and with different values of density (d) and different total number of nodes (n).

4.3.3 Discussion and Analysis

In order to evaluate the algorithm performance we need to check if the values of the algorithm running times are within the goal defined by Correia et al. [13]. The authors

4.3. ALGORITHM PERFORMANCE EVALUATION AND RESULTS

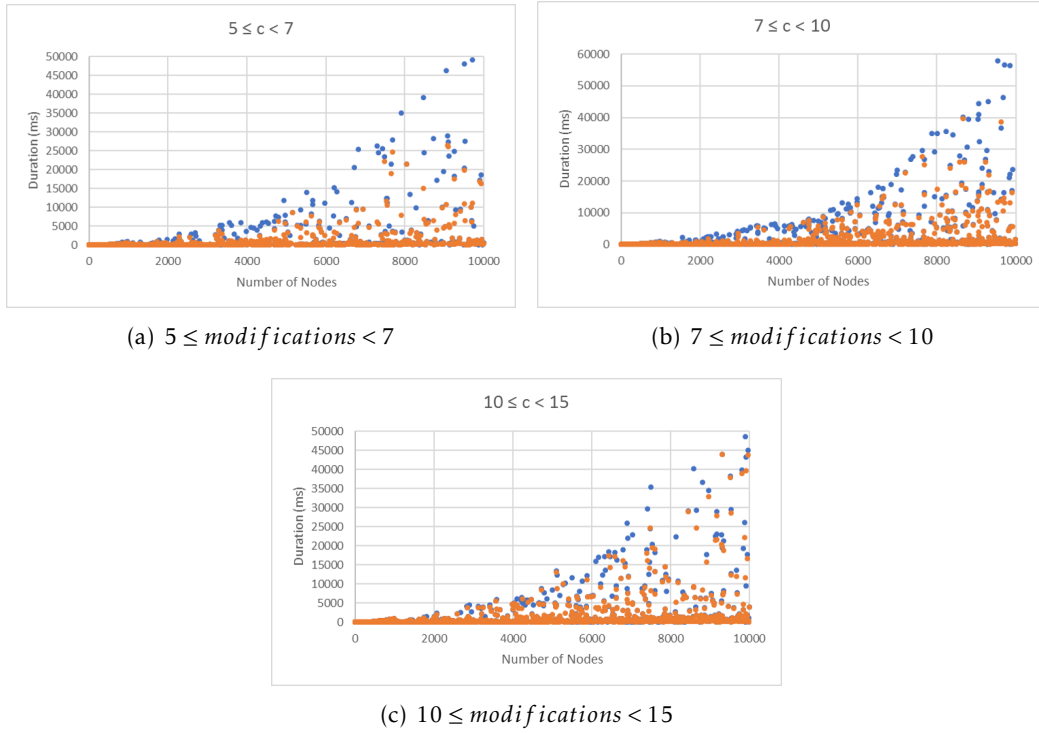


Figure 4.6: Algorithm performance before (blue) and after modifications (orange)

	Nodes/ Density	$10 \leq n < 100$	$100 \leq n < 1k$	$1k \leq n < 5k$	$5k \leq n < 10k$
$5 \leq c < 7$	$0 \leq d < 0.01$	0 ms	0 ms	-18 ms	83 ms
	$0.01 \leq d < 0.1$	0 ms	1 ms	12 ms	101 ms
	$0.1 \leq d < 0.5$	0 ms	25 ms	163 ms	1129 ms
	$0.5 \leq d < 1$	0 ms	126 ms	2456 ms	9911 ms
$7 \leq c < 9$	$0 \leq d < 0.01$	0 ms	0 ms	-20 ms	83 ms
	$0.01 \leq d < 0.1$	0 ms	0 ms	14 ms	97 ms
	$0.1 \leq d < 0.5$	0 ms	12 ms	225 ms	895 ms
	$0.5 \leq d < 1$	0 ms	84 ms	1444 ms	9822 ms
$10 \leq c < 15$	$0 \leq d < 0.01$	0 ms	0 ms	-20 ms	57 ms
	$0.01 \leq d < 0.1$	0 ms	0 ms	4 ms	25 ms
	$0.1 \leq d < 0.5$	0 ms	10 ms	74 ms	447 ms
	$0.5 \leq d < 1$	0 ms	62 ms	1041 ms	6201 ms

Table 4.12: Average runtime improvement for graphs

defined that for a graph with a density of 0.1 with 5000 nodes the algorithm must run under 1 second.

In our solution the average running time of the algorithm graphs with size between 4500 and 5500 and a density between 0.08 and 0.12 is 870 milliseconds (0.87 seconds), a graph with 5000 nodes with a density of 0.1 means that each node attacks on average 500 other nodes, this totals 2.5 millions of attacks in the graph.

When we analyze the graphs presented in Figure 4.5 it is possible to see that with the increase of the number of attacks and number of nodes the running time also grows, as expected. The graphs with higher densities also take much longer to run than a graph with density near 0.1.

The results of the tests, before and after, a small number of modification is introduced, reveal that the time runtimes after the modifications are introduced to the system, are significantly lower than the previous runtimes. We also observed that the improvement of the runtimes is as bigger as the graph density and size grows (see Figure 4.6).

The number of modifications between two iterations of the algorithm will depend on the number of active users on the debate. Moreover every time a user interacts with the system, the algorithm runs, this means that all the interactions that were made in a short time interval will be taken into account in that iteration. We can expect that the number of changes is small, proportional to the number of active users on a specific debate.

If we look at Table 4.12 it is possible to observe the times that the algorithm takes to converge improves from the first to the second iteration. These improvements are more visible for graphs with more than one thousand arguments and with high densities.

The results also shows that, as it was expected, the times are better when the number of modifications is smaller although this depends on the actions that were performed.

To conclude, the algorithm was successfully re-implemented and runs within the expected times. In real world scenarios the argument strengths will always be converged before any change is made, unless the graph is imported. This means that the times that we expect that the algorithm takes to run are similar to the ones that we represent in figure 4.6 in orange. Moreover, we expect that the number of attacks that come from each node does not exceed a fixed limit, consequently we expected that bigger graphs have smaller density values when compared to smaller graphs.

Although some similar tests were performed in [13], we could not compare the results since the tests were made for the version of the algorithm that does not consider attack strengths.

4.4 Summary

In this chapter we have presented the evaluation, results and discussion of the developed tool. This evaluation was divided in different parts: Semantics, Usability and Performance, reaching the following conclusions. In terms of semantics, although the participants agreed with the semantics as a whole, there are some important adaptations and improvements that can be made in the future. In terms of the system usability, however it is acceptable and easy to use, the system still needs some improvements in order to be more appealing and easy to use. In terms of performance, the tool has an adequate performance that does not compromise the system usability.

CONCLUSION

In this work we developed a tool that is based on the extended version of Social Abstract Argumentation [17]. This tool was used as a proof of concept to this model, allowing to evaluate the extended version of Social Abstract Argumentation.

We proposed an application where the debate is represented through a graph. The arguments are nodes in the graph, and the attacks are edges. Furthermore, both arguments and attacks have positive and negative votes.

The usability of the application was tested with a group of 16 voluntary participants and, although the interface was improved taking into account the identified issues, there is still space for further improvements.

The algorithm presented in [13] was re-implemented and adapted in order to also consider votes on attacks. The algorithm performance was evaluated using randomly uniform distributed test sets, and we concluded that the running time for a debate is within the values proposed in [13] (i.e. a graph with a density of 0.1, and 5000 nodes should run under 1 second). In addition, the algorithm was tested in a more realistic scenario. In this scenario, the strengths of a debate graph were calculated. After that, a small number of modifications was introduced to the debate, and the strengths of the arguments were recalculated. We verified that the runtimes after the modifications were introduced are, in most of the cases, significantly lower.

The semantics proposed in [17] were also evaluated using a group of 15 voluntary participants, that used the system for a four days period. After analyzing the results, we could identify specific situations where the users disagree with the assigned strengths, and also the reasons that make them disagree. The most prominent factor that make the participants disagree is the fact that the semantics do not consider the total number of votes on a debate, in order to calculate the arguments' strength.

There was also another issue identified during the tests that is related to cycles of

attacks between two arguments. This only happens when both of the arguments have a score of 100 and different scores on the attacks, however one of the attacks must have the maximum strength. This happens due to the definition of the semantics, that forces the argument that is attacked with a strength of 100 by an argument that also has the maximum strength to reduce drastically every iteration until it reaches 0.

Although we have identified these issues with the semantics they were generally approved and appreciated by the participants. There are some comments that reveal their recognition on this tool:

- *"(...) one thing we are not able to do in social networks, such as Facebook, is to see the scores in the comments on a publication. Therefore, the most pertinent comments are lost between many others that are not as good. This app allows us to debate a subject, see the overall view and realize if we agree with the majority or if we are a minority. As I see it, applying this to even broader groups and social networks would have a major impact on how we discuss subjects online."*
- *"This project is an excellent idea because it is possible to perceive the dominant ideas and through the connection between the arguments, it is possible to avoid the repetition of ideas. It also makes the debate very dynamic because the score of the arguments is changing over the course of days."*

5.1 Future Work

There are several aspects of the system that can be improved, starting with the system usability. One of the improvements that can be made is to allow the arguments to show a preview of the links, videos, images, etc.. This feature would allow the users to have a whole view of the debate with more relevant information.

The semantics can be adjusted in order to provide a more adequate strength to every argument. To achieve this we have to consider the two situations mentioned before, taking into account the total number of votes and also the issue related with cycles between two arguments.

One way that could be used to solve this, is starting all the arguments with a strength of 50 (e.g. every argument starts with 10 positive votes and 10 negative votes). This way, two arguments with only positive votes, where one of them has 1 positive vote and the other 100, would have significantly different score values (52 and 92 respectively). Also the problem concerning cycles between two attacks will also be reduced since that it would be necessary a high number of positive votes and no negative votes, in order to achieve a score near to 100. Figure 5.1 presents an example that considers the scenarios presented before, in the actual system (Figure 5.1(a)) and after introducing 10 positive votes and 10 negative votes to every argument (Figure 5.1(b)).

Table 5.1 presents the scores of the arguments before and after this modification to the system.

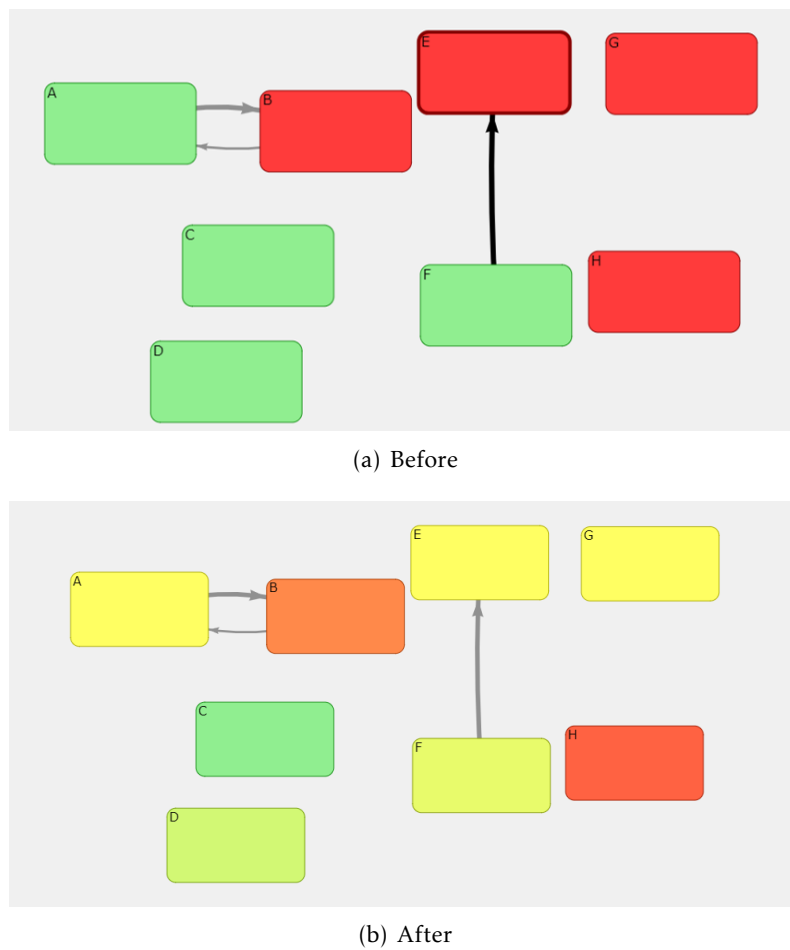


Figure 5.1: Comparison between the scores before and after the scores start at 50 (i.e. 10 positive votes and 10 negative votes)

	V^+	V^-	Score (Before)	Score (After)
A	1	0	100	45
B	1	0	0	29
C	100	0	100	92
D	10	0	100	67
E	1000	0	0	47
F	1	0	100	52
G	0	1	0	48
H	1	50	2	15

Table 5.1: Scores for the arguments on Figure 5.1

There are also other fields where this specific tool could be tested such as online newspapers, participatory democracy, decision making, brainstorming, and others that we may not be aware of. We think that this tool can be an asset in a vast range of applications in the real world.

BIBLIOGRAPHY

- [1] *Agora-net. Collaborative and web-based argument visualization software that can be used for free all over the globe.* URL: <http://agora.gatech.edu>.
- [2] L. Amgoud and J. Ben-Naim. “Ranking-Based Semantics for Argumentation Frameworks”. In: *Scalable Uncertainty Management - 7th International Conference, SUM 2013, Washington, DC, USA, September 16-18, 2013. Proceedings.* 2013, pp. 134–147. DOI: 10.1007/978-3-642-40381-1_11. URL: https://doi.org/10.1007/978-3-642-40381-1_11.
- [3] ARG-tech. Centre for Argument Technology. URL: <http://www.arg-tech.org/>.
- [4] P. Baroni and M. Giacomin. “Semantics of Abstract Argument Systems”. In: *Argumentation in Artificial Intelligence.* 2009, pp. 25–44. DOI: 10.1007/978-0-387-98197-0_2. URL: https://doi.org/10.1007/978-0-387-98197-0_2.
- [5] P. Besnard and A. Hunter. “A logic-based theory of deductive arguments”. In: *Artif. Intell.* 128.1-2 (2001), pp. 203–235. DOI: 10.1016/S0004-3702(01)00071-6. URL: [https://doi.org/10.1016/S0004-3702\(01\)00071-6](https://doi.org/10.1016/S0004-3702(01)00071-6).
- [6] P. Besnard, A. J. García, A. Hunter, S. Modgil, H. Prakken, G. R. Simari, and F. Toni. “Introduction to structured argumentation”. In: *Argument & Computation* 5.1 (2014), pp. 1–4. DOI: 10.1080/19462166.2013.869764. URL: <https://doi.org/10.1080/19462166.2013.869764>.
- [7] E. Bonzon, J. Delobelle, S. Konieczny, and N. Maudet. “A Comparative Study of Ranking-based Semantics for Abstract Argumentation”. In: *CoRR* abs/1602.01059 (2016). URL: <http://arxiv.org/abs/1602.01059>.
- [8] M. Bostock. *D3. Data-Driven Documents.* URL: <https://d3js.org/>.
- [9] J. Brooke et al. “SUS-A quick and dirty usability scale”. In: *Usability evaluation in industry* 189.194 (1996), pp. 4–7.
- [10] A. B.V. *vis.js. A dynamic, browser based visualization library.* 2016. URL: <http://visjs.org/>.

- [11] C. Cayrol and M. Lagasquie-Schiex. “On the Acceptability of Arguments in Bipolar Argumentation Frameworks”. In: *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 8th European Conference, ECSQARU 2005, Barcelona, Spain, July 6-8, 2005, Proceedings*. 2005, pp. 378–389. DOI: 10.1007/11518655_33. URL: https://doi.org/10.1007/11518655_33.
- [12] Consider.it. *Consider.it. Creates CIVIL, ORGANIZED, AND EFFICIENT ONLINE DIALOGUE by visually summarizing what your community thinks and why*. URL: <https://consider.it/>.
- [13] M. Correia, J. Cruz, and J. Leite. “On the Efficient Implementation of Social Abstract Argumentation”. In: *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*. 2014, pp. 225–230. DOI: 10.3233/978-1-61499-419-0-225. URL: <https://doi.org/10.3233/978-1-61499-419-0-225>.
- [14] DebateGraph. *To change the world you need to look at it in a different way*. URL: <http://debategraph.org>.
- [15] P. M. Dung. “On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games”. In: *Artif. Intell.* 77.2 (1995), pp. 321–358. DOI: 10.1016/0004-3702(94)00041-X. URL: [https://doi.org/10.1016/0004-3702\(94\)00041-X](https://doi.org/10.1016/0004-3702(94)00041-X).
- [16] W. W. Eckerson. “Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications”. In: *Open Information Systems 10*. Jan. 1995.
- [17] S. Egilmez, J. Martins, and J. Leite. “Extending Social Abstract Argumentation with Votes on Attacks”. In: *Theory and Applications of Formal Argumentation - Second International Workshop, TAFE 2013, Beijing, China, August 3-5, 2013, Revised Selected papers*. 2013, pp. 16–31. DOI: 10.1007/978-3-642-54373-9_2. URL: https://doi.org/10.1007/978-3-642-54373-9_2.
- [18] T. jQuery Foundation. *jQuery. Write less, do more*. URL: <https://jquery.com/>.
- [19] T. A. S. Foundation. *Apache Tomcat®*. URL: <https://tomcat.apache.org/>.
- [20] A. J. Freeley and D. L. Steinberg. *Argumentation and debate*. Cengage Learning, 2013.
- [21] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of*. 1994.
- [22] Jersey. *RESTful Web Services in Java*. 2016. URL: <https://jersey.github.io/>.
- [23] J. Leite and J. Martins. “Social Abstract Argumentation”. In: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. 2011, pp. 2287–2292. URL: <http://ijcai.org/Proceedings/11/Papers/381.pdf>.

- [24] C. Ltd. *Ubuntu Server*. URL: <https://www.ubuntu.com/server>.
- [25] P. Matt and F. Toni. "A Game-Theoretic Measure of Argument Strength for Abstract Argumentation". In: *Logics in Artificial Intelligence, 11th European Conference, JELIA 2008, Dresden, Germany, September 28 - October 1, 2008. Proceedings*. 2008, pp. 285–297. DOI: 10.1007/978-3-540-87803-2_24. URL: https://doi.org/10.1007/978-3-540-87803-2_24.
- [26] mindmeister. *Mindmeister. Collaborative mind mapping*. URL: <https://www.mindmeister.com/>.
- [27] Neo4j. *HyperGraphDB*. URL: <http://hypergraphdb.org/>.
- [28] Neo4j. *Neo4j*. URL: <https://neo4j.com/>.
- [29] J. Nielsen. *Usability engineering*. Elsevier, 1994.
- [30] TidyLifeInc. *CreateDebate - a social tool that democratizes the decision-making process through online debate*. URL: <http://www.createdebate.com/>.

A P P E N D I X



UsABILITY EVALUATION

In this appendix we present the questionnaires that were made during the user interface evaluation. The questionnaire was developed using Google Forms and took place under the supervision of the development team.

Debate Tool Thesis Evaluation

This form has the main purpose of evaluating the user interface of the debate system developed in a MSc Thesis context at FCT-UNL.

Your responses to the surveys are confidential. We value your privacy and will not share your information with anyone beyond the research team. Data will be averaged and reported in aggregate.

Look at the graph (debate) in the screen. The debate is about cigarettes and e-cigarettes and its health related issues. Try to understand the whole debate before you start performing the tasks.

***Required**



1. Gender: *

Mark only one oval.

- Male
- Female

2. Age: *

3. Academic degree: *

Mark only one oval.

- Highschool
- Bachelor
- Master
- Doctorate
- None of the above

4. How often do you use social networks (Facebook, Twitter, ...) ? *

Mark only one oval.

- Everyday.
- Several days a week.
- Once a week.
- Once a month.
- I do not use social networks.

Task 1 - Create an Argument

Create an argument with the following text: "Cigarettes are dangerous for your health. It does not matter if they are electronic or not. "

5. The task was easy to complete. *

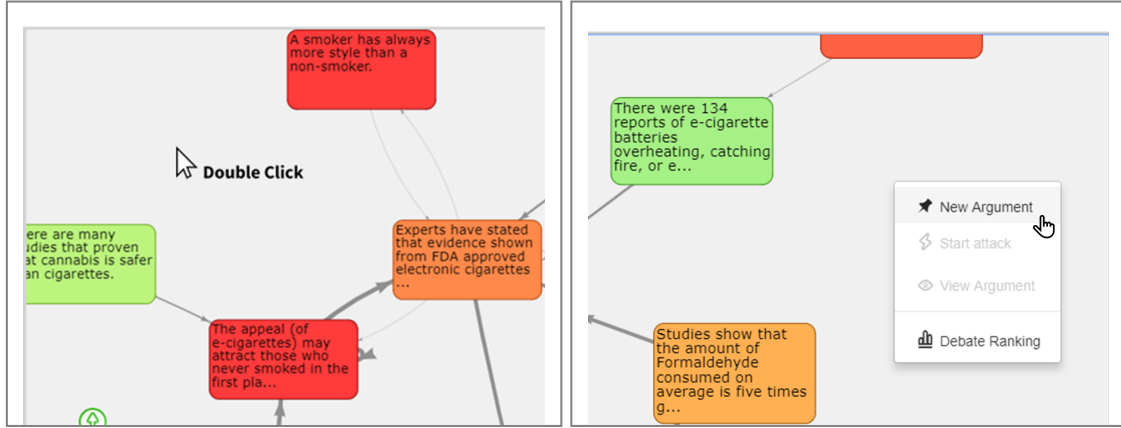
Mark only one oval.

1 2 3 4 5

Strongly disagree Strongly agree

6. How did you create the argument? *

Mark only one oval.



Double Click

Right click -> New Argument

7. Understanding how to create an argument was easy. *

Mark only one oval.

1 2 3 4 5

Strongly disagree Strongly agree

8. Suggestions and comments:

Task 2 - Vote

2.1 - Vote positively on the argument that says “There are no conclusive studies as to whether nicotine acts as a carcinogen on its own...”

9. The task was easy to complete. *

Mark only one oval.

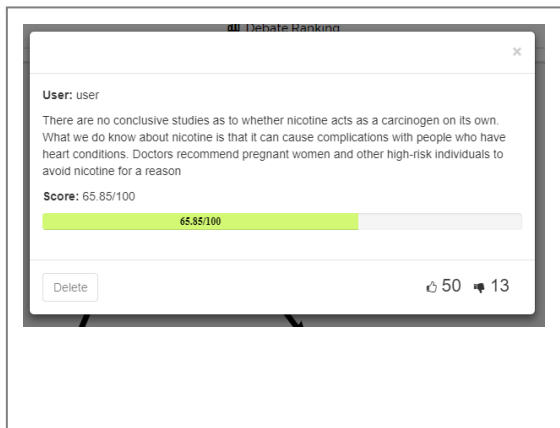
1 2 3 4 5

Strongly disagree Strongly agree

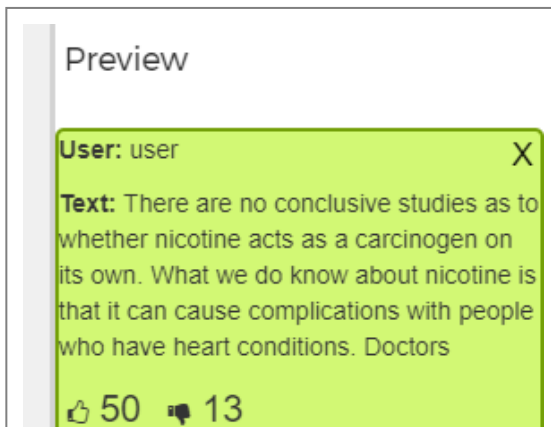
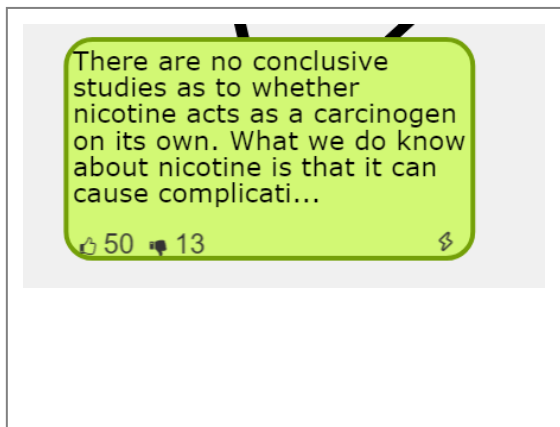
10. After your vote, what was the score of the argument? (2.1) *

11. How did you vote on the argument? (2.1) *

Mark only one oval.



Open the argument -> Click on the thumbs up.



Thumbs up (graph view)

Thumbs up (argument preview)

2.2 - Vote negatively on the argument that says, "A smoker has always more style than anon-smoker".

12. The task was easy to complete. *

Mark only one oval.

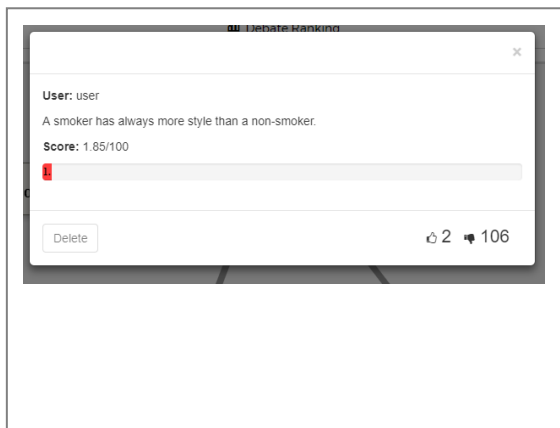
1 2 3 4 5

Strongly disagree Strongly agree

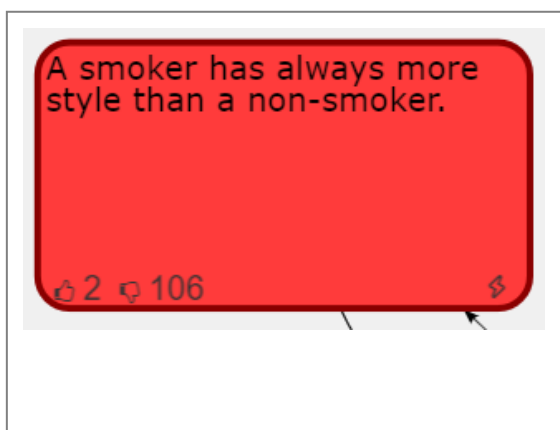
13. After your vote, what was the score of the argument? (2.2) *

14. How did you vote on the argument? (2.2) *

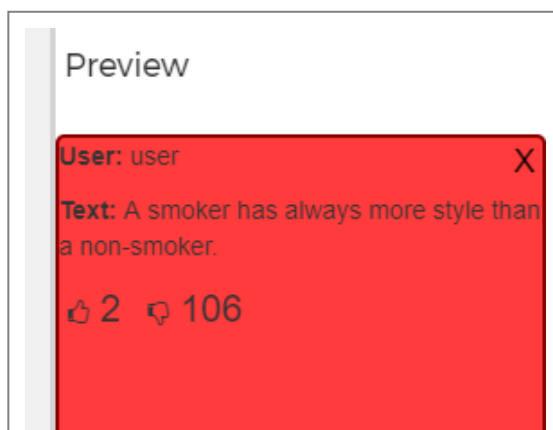
Mark only one oval.



Open the argument -> Click on the thumbs down.



Thumbs down (graph view)



Thumbs down (argument preview)

15. Suggestions and comments:

Task 3 - Attack other arguments

3.1 - Use the argument that you have created ("Cigarettes are dangerous for your health. It does not matter if they are electronic or not. "), to attack the arguments that states:

A: "Experts have stated that evidence shown from FDA approved electronic cigarettes could potentially be safer than the use of regular cigarettes" and B: "The appeal (of e-cigarettes) may attract those who never smoked in the first place, ...".

16. The task was easy to complete. *

Mark only one oval.

1 2 3 4 5

Strongly disagree Strongly agree

17. The system gave clear evidence of the success of the task. *

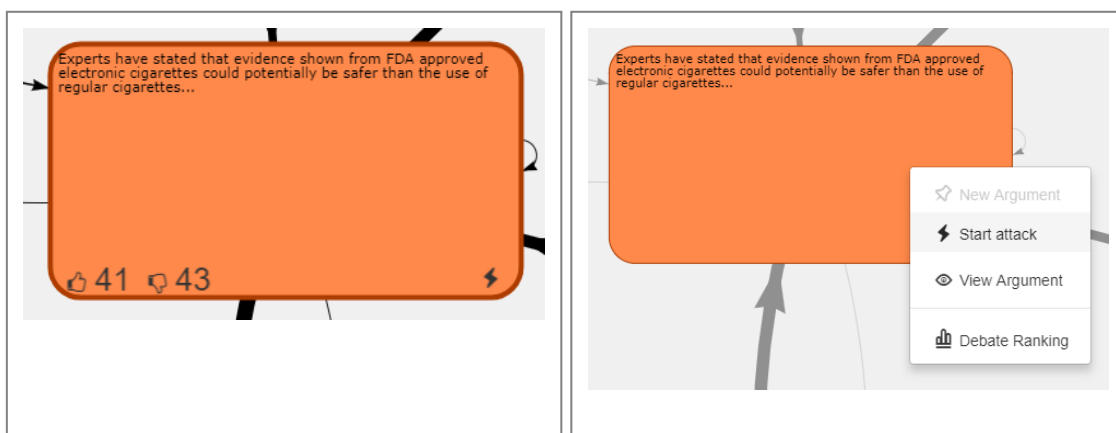
Mark only one oval.

1 2 3 4 5

Strongly disagree Strongly agree

18. How did you create the attacks? *

Mark only one oval.



Bolt symbol (graph view)

Right click -> Start attack

19. How many positive votes do attacks have after their creation? *

20. What was the score of the argument A after the completion of the task? *

3.2 - Change the vote of the second attack to negative.

21. The task was easy to complete. *

Mark only one oval.

1 2 3 4 5

Strongly disagree Strongly agree

22. What is the score of the argument B after the completion of the task? *

23. **Suggestions and comments:**

Task 4 - Create and delete an argument

Create an argument. Vote positively on that argument. Delete the argument that you have just created.

24. **Did you manage to complete the task successfully? ***

Mark only one oval.

Yes

No

25. **The task was easy to complete. ***

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly agree

26. **Suggestions and comments:**

Task 5 - View the debate ranking

Open the debate ranking.

27. **The task was easy to complete. ***

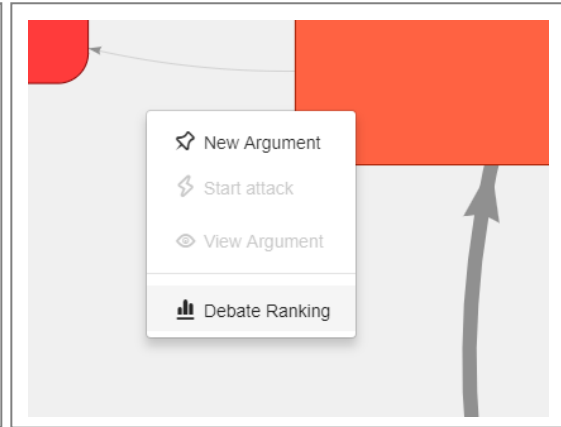
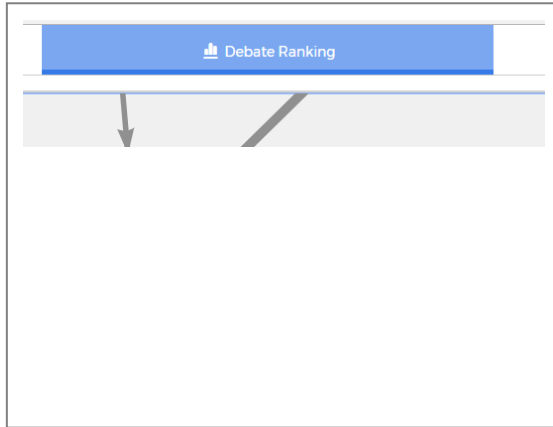
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly agree

28. **Copy the text of the argument that was in second place. ***

29. How did you open the debate ranking? *

Mark only one oval.



Top menu -> Debate Ranking

Right click -> debate ranking

30. Suggestions and comments:

General Questions

Visualization and Navigation

31. Which color is associated to the strongest arguments? *

Mark only one oval.

- Green
- Yellow
- Red
- Blue

32. It was easy to move (right, left, up, down) along the debate. *

Mark only one oval.

1	2	3	4	5	
Strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> Strongly agree

33. It was easy to zoom in and zoom out. *

Mark only one oval.

1	2	3	4	5	
Strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> Strongly agree

34. What were the main obstacles when performing these tasks?

Overall Evaluation**35. It is easy to learn how to use the system. ****Mark only one oval.*

	1	2	3	4	5	
Strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly agree

36. I found the system unnecessarily complex **Mark only one oval.*

	1	2	3	4	5	
Strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly agree

37. I thought the system was easy to use **Mark only one oval.*

	1	2	3	4	5	
Strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly agree

38. I think that I would need the support of a technical person to be able to use this system **Mark only one oval.*

	1	2	3	4	5	
Strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly agree

Powered by



APPENDIX



SEMANTICS EVALUATION

In this appendix we present the questionnaires that were made during the semantics evaluation. The questionnaire was developed using Google Forms.

Debate Tool Semantics Evaluation

This form has the main purpose of evaluating the semantics of the debate system developed in a MSc Thesis context at FCT-UNL.

Your responses to the surveys are confidential. We value your privacy and will not share your information with anyone beyond the research team. Data will be averaged and reported in aggregate.

*Required



1. Gender: *

Mark only one oval.

- Male
 Female

2. Age: *

3. Academic degree: *

Mark only one oval.

- Highschool
 Bachelor
 Master
 Doctorate
 None of the above

4. How often do you use social networks (Facebook, Twitter, ...) ? *

Mark only one oval.

- Everyday.
 Several days a week.
 Once a week.
 Once a month.
 I do not use social networks.

5. At the end of the debate do you think that the strength/score of the arguments reflects the overall view of the debating group?

Mark only one oval.

	1	2	3	4	5	
Strongly disagree.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly agree.

6. Comments

7. Considering the way you use the system, an argument is... *

Tick all that apply.

- A structured assertion (i.e. with permises and conclusion)
- An unstructured comment
- Other: _____

8. Consider the way other people use the system, an argument is... *

Tick all that apply.

- A structured assertion (i.e. with permises and conclusion)
- An unstructured comment
- Other: _____

9. When you voted positively on an argument, what did you want to express? *

Tick all that apply.

- The argument is well structured, independently of whether I agree with the conclusion.
- The argument is structured, well formed, and I agree with its premises.
- The argument is structured and I agree with its conclusion independently of everything else.
- The argument is a comment that I agree with
- The argument is a comment that I find funny.
- Other: _____

10. When you voted negatively on an argument, what did you want to express? *

Tick all that apply.

- The argument is not well structured.
- The argument is well formed and I do not agree with its premises.
- the argument is strucutred and I do not agree with the conclusion.
- The argument is a comment that I do not agree with.
- I find the argument offensive/disrespectful.
- Other: _____

11. Which of the following sentences do you consider to be arguments? *

Tick all that apply.

- "I don't agree!"
- "Animals are born in the wild, and they are supposed to be living in the wild too. Therefore, you should not lock any kind of animal in your home!!"
- "There are many types of animals and some of them are already adapted to live inside."
- "Snails are slow, and I like snails. So snails should live inside."
- "I like turtles!"

12. Consider the following arguments. Assuming they are not being attacked, how strong do you think they should be (i.e. their score) ? *

A 22 positive votes 2 negative votes 👍 22 👎 2	B 1 positive votes 0 negative votes 👍 1 👎 0	C 2 positive votes 2 negative votes 👍 2 👎 2
D 1 positive votes 23 negative votes 👍 1 👎 23	E 0 positive votes 2 negative votes 👍 0 👎 2	F 42 positive votes 43 negative votes 👍 42 👎 43
G 1 positive votes 43 negative votes 👍 1 👎 43		

Mark only one oval per row.

	Very low (0 to 20)	Low (20 to 40)	Medium (40 to 60)	High (60 to 80)	Very high (80 to 100)
A	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
B	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
E	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
F	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
G	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

13. Consider the following unattacked arguments. Arrange the arguments in descending order of strength/score (i.e. strongest arguments in the first positions). *

Your answer must be in the following way: A-B-C-D-E-F-G, where A is the strongest argument and G the weakest.

A

22 positive votes
2 negative votes

👍 22 👎 2

B

1 positive votes
0 negative votes

👍 1 👎 0

C

2 positive votes
2 negative votes

👍 2 👎 2

D

1 positive votes
23 negative votes

👍 1 👎 23

E

0 positive votes
2 negative votes

👍 0 👎 2

F

42 positive votes
43 negative votes

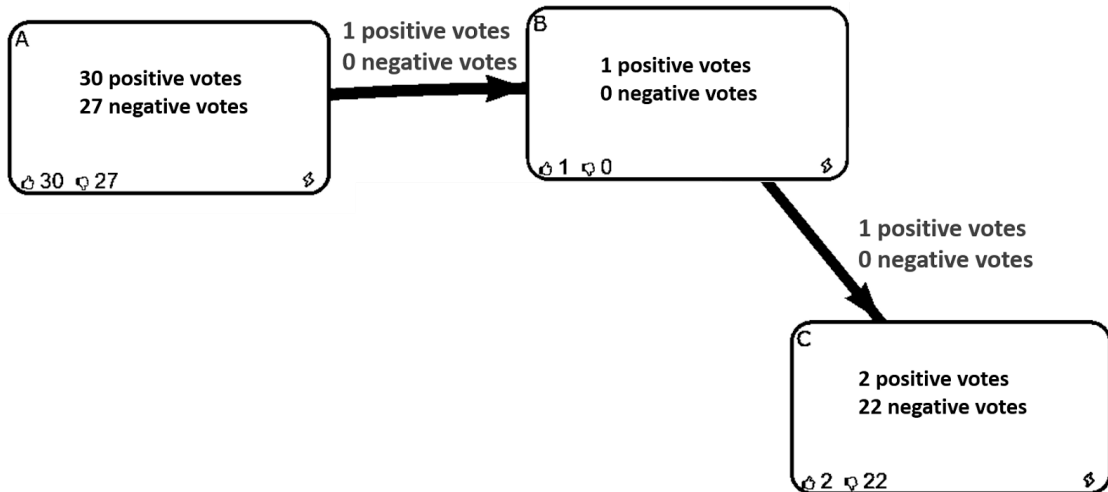
👍 42 👎 43

G

1 positive votes
43 negative votes

👍 1 👎 43

14. Consider the following debate. How strong do you think each argument should be (i.e. their score)? *

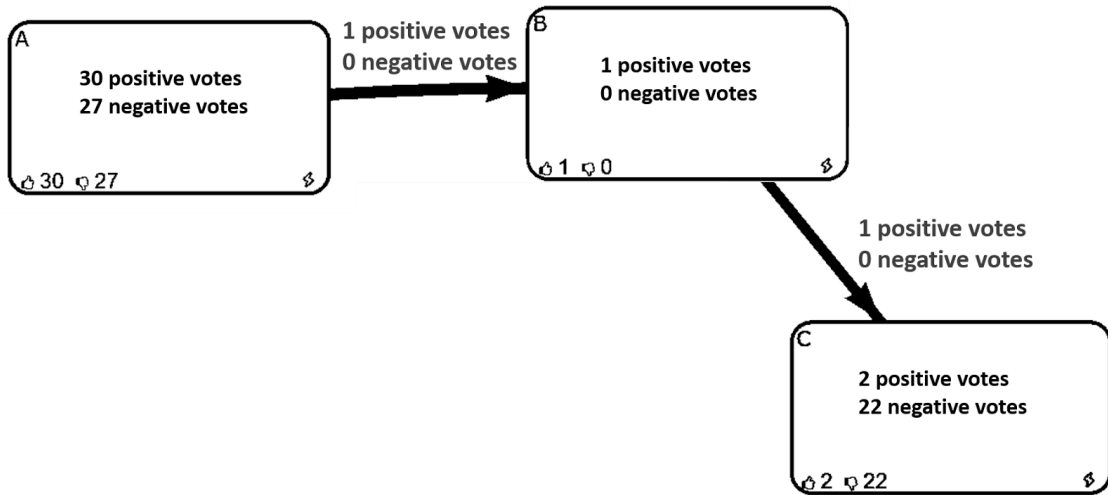


Mark only one oval per row.

	Very low (0 to 20)	Low (20 to 40)	Medium (40 to 60)	High (60 to 80)	Very high (80 to 100)
A	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
B	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

15. Consider the following debate graph. Arrange the arguments in descending order of strength/score (i.e. strongest arguments in the first positions). *

Your answer must be in the following way: A-B-C, where A is the strongest argument and C the weakest.



16. Consider the following situation. What attacks would you create between these two arguments?

A cactus is able to survive the droughts.

Argument A

👍 1 👎 0

A flower is prettier than a cactus.

Argument B

👍 1 👎 0

Mark only one oval.

A cactus is able to survive the droughts. → A flower is prettier than a cactus.

A flower is prettier than a cactus. → A cactus is able to survive the droughts.

From A to B.

From B to A.

A cactus is able to survive the droughts. ↔ A flower is prettier than a cactus.

A cactus is able to survive the droughts. A flower is prettier than a cactus.

Both ways.

None.

17. Imagine an argument (B) that you do not agree with and/or you find disrespectful. How would you express yourself using our tool in this specific situation? *

Tick all that apply.

- Create an argument and use it to attack the argument B
- Downvote argument B
- None of the above

18. How would you vote on the following attack? *

Mark only one oval.

- Positively
- Negatively
- I would not vote

19. Consider you voted negatively on a certain argument B. Would you create an Argument stating "I do not agree" and use it to attack B? *

Mark only one oval.

- Yes
- No

20. Consider the following situation. After the attack what is going to happen to the strength/score of argument B? *

Mark only one oval.

- Decrease
- Mantains
- Increase

21. Please explain why. *

22. Consider the following 2 arguments which one should be the strongest. *



A

B

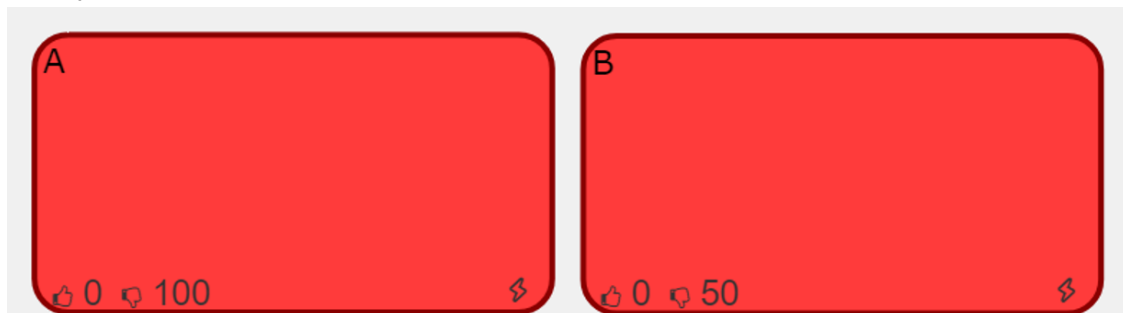
100 0

50 0

Mark only one oval.

- A
- B
- None (They should be equal)

23. Consider the following 2 arguments. Which one should be the weakest (i.e. with the lowest score) ?



A

B

0 100

0 50

Mark only one oval.

- A
- B
- None (They should be equal)

24. Consider the following 2 arguments. Which one should be the strongest (i.e. with the highest score)? *

<p>A</p> <p>👍 5 👎 1</p> <p style="text-align: right;">⚡</p>	<p>B</p> <p>👍 124 👎 121</p> <p style="text-align: right;">⚡</p>
---	---

Mark only one oval.

- A
- B
- None (They should be equal)

25. Consider the following 2 arguments which one should be the strongest (i.e. with the highest score) . *

<p>A</p> <p>👍 13 👎 19</p> <p style="text-align: right;">⚡</p>	<p>B</p> <p>👍 234 👎 240</p> <p style="text-align: right;">⚡</p>
---	---

Mark only one oval.

- A
- B
- None (They should be equal)

26. Consider the following situation. Do you agree with these strengths/scores? *

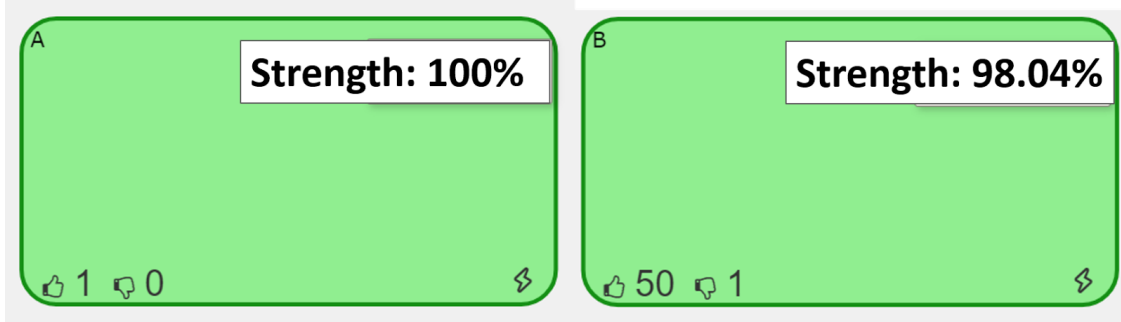
<p>A</p> <p style="text-align: center; background-color: white; color: black; padding: 5px;">Strength: 0%</p> <p>👍 0 👎 1</p> <p style="text-align: right;">⚡</p>	<p>B</p> <p style="text-align: center; background-color: white; color: black; padding: 5px;">Strength: 1.96%</p> <p>👍 1 👎 50</p> <p style="text-align: right;">⚡</p>
---	---

Mark only one oval.

- Yes
- No

27. Please explain why. *

28. Consider the following situation. Do you agree with these strengths/scores? *

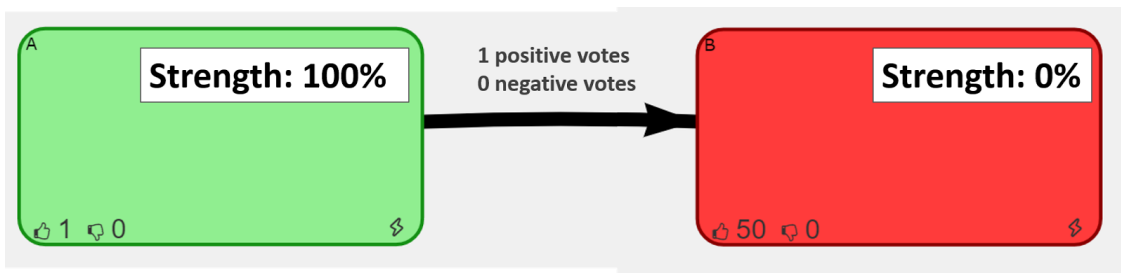


Mark only one oval.

- Yes
- No

29. Please explain why.

30. Consider the following situation. Do you agree with these strengths/scores? *



Mark only one oval.

- Yes
- No

31. Please explain why.



