

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Argument Diagramming: Annotation and Evaluation

Daniela Sá



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Henrique Daniel de Avelar Lopes Cardoso

Second Supervisor: Rui Manuel Sousa Silva

July 17, 2019



# **Argument Diagramming: Annotation and Evaluation**

**Daniela Sá**

Mestrado Integrado em Engenharia Informática e Computação

July 17, 2019



# Abstract

Argumentation is an essential tool that we use to communicate as a species, and it is the process of deriving a conclusion from a set of premises and the logical steps employed in the process. Without argumentation there would be no decision making, so we can almost say that our society as it is would not be able to survive without it.

However, spotting an argument is not always straightforward and that is where *Argumentation Mining* comes in – it aims at the automatic extraction of arguments from natural language text using machine learning techniques. Argumentation mining faces many issues, including the lack of annotated input data that can feed machine learning approaches, and the need to evaluate the reliability of such data.

In this thesis we present the current state of the ArgMine platform, which aims to the creation of annotated corpora. We will focus on managing annotation projects (by providing a platform with management options and that can make it easier to keep track of annotations and annotation projects) and on an annotator tool. This tool’s goal is to simultaneously provide more freedom to the annotation process (for example, by allowing users to edit the nodes’ textual content) and to provide more visual help to the interpretation of the annotations (for example, by allowing users to change the color or the nodes). Then, we will move on to presenting some evaluation metrics that can be used in the ArgMine platform to grade annotations. Finally, we will focus on metrics to analyze the reliability of a corpus – inter-annotator agreement metrics. Starting from an existing corpus, we apply some of the more widely used inter-annotator agreement metrics, and we explore the usage of grading metrics for this purpose.

With this thesis we contribute with an annotation platform that can be useful in a project-driven setting, thus encouraging the annotation of arguments and the building of annotated argumentative corpora – a valuable resource for Argumentation Mining. We also provide more comprehensive ways to evaluate corpora reliability.



# Resumo

Argumentação é uma ferramenta essencial que usamos para comunicar na nossa espécie. É o processo de chegar a uma conclusão desde um conjunto de premissas e os passos lógicos entre elas. Sem argumentação não existiria o processo de decisão, por isso quase podemos afirmar que a nossa sociedade atual não conseguiria sobreviver sem ela.

No entanto, identificar um argumento não é sempre óbvio e é aí que entra a Argumentation Mining. Esta visa a extração automática de argumentos de texto em linguagem natural usando técnicas de machine learning, o que é uma tarefa complexa considerando a ambiguidade da linguagem natural. Em adição a almejar uma tarefa difícil, Argumentation Mining também enfrenta vários problemas, começando com uma falta de *input* para o processo de machine learning e por formas de avaliar se este *input* é confiável.

Nesta tese apresentamos o estado atual da plataforma ArgMine, que tem por objetivo facilitar a criação de corpora anotadas, com foco em gestão de projetos de anotação (fornecendo uma plataforma com opções de gestão que pode facilitar a organização de anotações e de projetos de anotação) e de numa ferramenta de anotação. Esta ferramenta simultaneamente oferece alguma liberdade ao processo de anotação (por exemplo, ao permitir que o utilizador altere o conteúdo textual dos nós) e ajuda visualmente na interpretação de anotações (por exemplo, ao permitir que o utilizador altere a cor dos nós na anotação). Iremos depois prosseguir com a apresentação de algumas métricas de avaliação que podem ser usadas na plataforma ArgMine para avaliar anotações. Finalmente, vamos focar-nos em métricas para analisar a fiabilidade de um corpus – métricas de *inter-annotator agreement*. Começando com um corpus já existente, aplicamos algumas das mais usadas métricas de *inter-annotator agreement*, e exploramos o uso de métricas de avaliação para este propósito.

Com esta tese contribuimos uma plataforma de anotação que pode ser útil numa situação focada em projeto, assim incentivando a anotação de argumentos e a construção de corpora argumentativa anotada – um recurso valioso em *Argumentation Mining*. Também providenciamos maneiras mais compreensivas de avaliar a fiabilidade das corpora.





# Acknowledgements

A thank you to Henrique Lopes Cardoso, Rui Sousa Silva and Gil Filipe da Rocha for helping and guiding me with this document and research.

Daniela Sá



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context . . . . .	2
1.2	Motivation . . . . .	2
1.3	Problem Statement . . . . .	4
1.4	Research Goals . . . . .	6
1.5	Dissertation Structure . . . . .	6
<b>2</b>	<b>State of the Art</b>	<b>9</b>
2.1	Introduction . . . . .	9
2.2	Fundamental Concepts . . . . .	10
2.3	Annotation and Creation of Corpora . . . . .	12
2.4	Argumentation Mining: Process . . . . .	13
2.5	Inter-Annotator Agreement . . . . .	14
2.6	Argumentation Mining: Recent Developments . . . . .	19
2.7	Annotation Tools . . . . .	22
2.8	Conclusions . . . . .	26
<b>3</b>	<b>The ArgMine Platform</b>	<b>29</b>
3.1	Functionalities . . . . .	29
3.2	Architecture . . . . .	32
3.3	Interaction with the Platform . . . . .	36
3.3.1	The Participant User . . . . .	38
3.3.2	The Manager User . . . . .	40
3.3.3	The ArgMine Annotator Tool . . . . .	49
<b>4</b>	<b>Automatic Evaluation of Annotations</b>	<b>55</b>
4.1	Problem and Motivation . . . . .	55
4.2	New Approach to Node Matching . . . . .	57
4.3	F1-Based Metric . . . . .	59
4.3.1	Explanation . . . . .	59
4.3.2	Shortcomings . . . . .	60
4.4	Relation-aware-F1-based Metric . . . . .	60
4.4.1	Explanation . . . . .	60
4.4.2	Example . . . . .	63
4.4.3	Shortcomings and Possible Improvements . . . . .	64
4.5	UCP Porto Metric . . . . .	65
4.5.1	Explanation . . . . .	66
4.5.2	Example . . . . .	73

## CONTENTS

4.5.3	Shortcomings and Possible Improvements . . . . .	78
4.6	Comparing the Evaluation Metrics . . . . .	78
<b>5</b>	<b>Inter-Annotator Agreement</b>	<b>81</b>
5.1	Problem and Motivation . . . . .	82
5.2	Kappa . . . . .	83
5.3	Krippendorff's Alpha . . . . .	84
5.4	F1-Based Metric . . . . .	85
5.5	UCP Porto Metric as IAA . . . . .	85
5.6	Relation-Aware-F1-Based Metric as IAA . . . . .	85
5.7	IAA Metrics Application . . . . .	86
5.7.1	The ArgMine Corpus . . . . .	86
5.7.2	Results . . . . .	86
<b>6</b>	<b>Conclusion</b>	<b>89</b>
6.1	New Insights . . . . .	90
6.2	Future Work . . . . .	91
	<b>References</b>	<b>93</b>

# List of Figures

2.1	The basic components of argument diagramming [Fre91] . . . . .	12
2.2	An annotation of dialogue in the OVA+ annotator, adapted from [REE14] . . . . .	23
2.3	An annotation in the NOMAD annotator, adapted from [Pet14] . . . . .	24
2.4	An annotation in the AVIZE annotator, adapted from [LGBR18] . . . . .	25
3.1	The ArgMine Platform high-level architecture . . . . .	32
3.2	The ArgMine database . . . . .	33
3.3	The ArgMine Argument Evaluation server architecture . . . . .	35
3.4	The ArgMine Platform log in page with some credentials filled in . . . . .	37
3.5	The settings page . . . . .	37
3.6	The participant documents list . . . . .	38
3.7	Participants can add documents to a project. Notice that there is no place for them to add this document to a collection . . . . .	39
3.8	Participants can add documents from collections . . . . .	39
3.9	Participants can add documents in bulk to the project, using a properly formatted .csv file . . . . .	40
3.10	The manager project list . . . . .	41
3.11	A project's settings page, accessible only to its active managers . . . . .	41
3.12	A project's document list, as seen by an active manager . . . . .	42
3.13	Managers can add documents to the project . . . . .	44
3.14	A project's participant tab, as seen by an active manager . . . . .	44
3.15	Managers can add participants to the project . . . . .	45
3.16	Managers can add participants in bulk to the project, using a properly formatted .csv file . . . . .	46
3.17	The list that appears in the page on figure 3.16, showing the the participants to be added. The feedback described in the previous paragraph is visible on the left-hand side . . . . .	46
3.18	The managers tab, as seen by an active manager . . . . .	47
3.19	Managers can see a list of annotations of a certain document, and grade some stages . . . . .	48
3.20	Managers can see a list of annotations that a certain participant has saved, and grade some stages . . . . .	49
3.21	The ArgMine annotator with an annotation . . . . .	50
3.22	The annotator tool's "Help" pop-up . . . . .	51
3.23	The ArgMine annotator with an annotation with the two types of edited nodes. These are marked by stripes in their borders . . . . .	52
3.24	The ArgMine annotator with an annotation with two nodes with the same text . . . . .	52

## LIST OF FIGURES

4.1	The ArgMine annotator showing a node that could have been annotated in two different contexts . . . . .	56
4.2	The two annotations that will be graded in this example. We may consider the right-hand side annotation as the gold standard annotation . . . . .	63
4.3	The annotation used as gold standard annotation for this example . . . . .	73
4.4	The annotation used as target annotation for this example . . . . .	74

# List of Tables

4.1	The thresholds used in this example. The lighter shaded rows show the thresholds used for testing the text matching percentages, while the darker shaded rows show the thresholds used used for testing the range matching percentages . . . . .	64
4.2	An example grading scale to use in this level. The "Grading Scale Grade"s will be the "gradeNodeMatching" in Algorithm 4.2 . . . . .	67
4.3	The weights used in the UCP Porto metric for the sake of this example. The darker rows show the level's total weight in the grade, while the white rows show the weights used to grade within the levels . . . . .	74
4.4	The grading scale used in this example. The thresholds are shown in darker shade and the grades that are used up to that threshold are shown in white. For example, an annotations with 85% level 1 score will be graded . . . . .	75
4.5	The matched nodes table for this example. Target node T4 does not figure in it, as it matched no node on the gold standard annotation. It will be ignored after this level	75
4.6	The level 3 cumulative score table after the first step . . . . .	77
4.7	The level 3 cumulative score table after the second step . . . . .	77
4.8	The level 3 cumulative score table after the third step . . . . .	77
5.1	Table showing the possible distribution of tokens across two annotations made by two annotators: A1 and A2 . . . . .	84
5.2	Results of IAA metrics on the ArgMine corpus . . . . .	87

## LIST OF TABLES



# Abbreviations

AI	Artificial Intelligence
IAA	Inter-annotator Agreement
UCP Porto	Universidade Católica Portuguesa Porto
LIACC	Laboratório de Inteligência Artificial e Ciência de Computadores (Artificial Intelligence and Computer Science Lab)



# Chapter 1

## Introduction

Argumentation is one of the pillars of human communication. It consists of deriving a conclusion (or claim) given a certain set of information (or premises), using logical steps to connect the information. Since some kind of argumentation is present in every decision that we make, and every time we try to get our point across and share our opinions, we can say that it is almost impossible to live a day without engaging in argumentation. But more complex argumentation is also very common in our lives, whether we are the ones engaging with it or the ones who have to live with the results of this engagement. For example, everyday, doctors must consider the information about their patients and arrive at a logical conclusion about what might be wrong with those patients while the patients must also deal with the soundness of this process. As a consequence of its undeniable influence on our lives, argumentation has been a subject of study for centuries.

At first, it was approached by Aristotle from the perspective of Logics as far back as the 4th century B.C. Then, in mid-twentieth century, with Toulmin's [Tou58] and Perelman and Olbrechts-Tyteca's [OT69] works, among others, new areas of Argumentation Theory started to develop. The focus of these areas was on studying arguments that people use on a daily basis, rather than perfectly formal arguments. Eventually, with the rise of technology and Artificial Intelligence (AI), there have been attempts to use some of the ideas developed in Argumentation Theory to automatically extract arguments from text. This led to the appearance of a new field of AI called Argumentation Mining.

Argumentation Mining is a fairly new area of research, but lately it has received more and more attention. It aims at the automatic detection of arguments from natural language text using machine learning techniques, an end goal which is still not in sight for the near future. This is mainly because, to automatically extract a complete argument from natural text, there is a myriad of factors to be considered, such as identifying entities, identifying references to those entities within the text, identifying if and what parts of the text belong to an argument, checking if there are several arguments in the same text, checking whether a certain part of the text is in favour

or against another part, adding contextual information that may not be explicit, adding common sense information that may not be explicit either, along with many more sub-tasks. All this is to be done while dealing with the ambiguity of natural language, which can divide even experts on the matter.

For the same reason, it also becomes difficult to evaluate the results obtained by researchers in this area. If even experts have difficulties in agreeing on exactly what argument is in a text, meaning that there is no set method of extracting arguments manually, it is going to be even more difficult to be able to automatically judge if the right argument was extracted from a text. The usual approach to this issue includes gathering a large number of human annotations and comparing them either to each other or to an annotation considered to be correct – the gold standard annotation. There have been efforts to create and develop agreement metrics to achieve this, which is another task that is far from trivial. At the core of the difficulties to compare and evaluate the quality of annotations, again, lies the ambiguity of human language, but also the different thought processes of the different human annotators. This is because there are multiple ways of annotating the same argument.

Despite the obstacles that Argumentation Mining faces, every small advancement is worthwhile in the pursuit of the final goal of being able to extract arguments from natural language text.

### **1.1 Context**

This thesis is nested under the ArgMine Framework project, developed by a team at the Artificial Intelligence and Computer Science Lab (LIACC) at FEUP, which aims to facilitate firstly the creation of an annotated corpus of argumentative texts in Portuguese and then the experimentation of different models and features related with the machine learning process [Roc16]. This platform has been running for a few years, and has gathered hundreds of annotations on news articles in Portuguese. Recently, the opportunity has arisen to work with Universidade Católica Portuguesa (UCP) and giving the ArgMine Platform a new use.

Since there is a class in UCP where students are evaluated on their annotations of argumentative texts, the ArgMine Platform can provide a place for students to do these annotations and for professors to manage their classes and the students' annotations. This way, it is easier for teachers, who can have all the classes' annotations in the same place where they can be easily managed, organized and browsed. There are also other features that will benefit the teachers at UCP, such as a more uniform visual representation of the annotations, or the automatic grading of submitted annotations.

### **1.2 Motivation**

Picking out an argument from a piece of natural language text and turning it into a structured and labelled argument diagram brings advantages beyond only scholarly settings. Firstly, we must

## Introduction

bear in mind that the manual argument extraction process can be an arduous task even for humans, and it requires not only years of training, but also a substantial amount of time and research. So, one of the main motivations for Argumentation Mining is the time it can save, to then be applied in more productive ways.

So, we can foresee several application scenarios that motivate the development of Argumentation Mining in different fields, namely:

- **In Medicine** — the possible applications of Argumentation Mining extend from easier teaching and learning to better diagnosing, and even better understanding web forum drug reviews. For example, if doctors could automatically see the arguments and counterarguments that lead to a certain diagnosis, this would facilitate comparisons between cases, which in turn would make diagnosing easier and more accurate;
- **In Education** — argument diagrams can be useful not only in the study of logic and argumentation, but in almost all other areas of knowledge, because they can provide the students with structured ways to look at the study materials, and even bring to their attention previously unnoticed details. On the other side of education, Argumentation Mining can help teachers better understand their students and even facilitate exam ratings, and teacher-student communication;
- **In Law** — lawyers and paralegals dedicate a significant amount of time to cross-checking case files, the arguments in each trial and their outcome [CV18], to be used as precedent for ongoing cases or for educational purposes. If Argumentation Mining was successfully applied here, the whole justice system would become more efficient, because court cases would not take as long, and more just, because it would be easy to identify similar cases with wildly different outcomes;
- **In Politics** — Argumentation Mining has the potential to completely revolutionize the current dynamics between politicians and voters. By being able to represent the arguments figured in each speech, thus hurdling over what has become the notorious complexity of politician's speeches, the voters would have access to much clearer information. This is a way to enhance the engagement between both parties, since more people would become politically aware. This exposition of political arguments would also compel politicians to build better speeches and journalists to better hold them to their word. All in all, more politically aware citizens, accountable politicians and sharper journalists would result in a more transparent political system;
- **In Web-based content processing** — Argumentation Mining could also bring about serious changes that would impact the internet forever. The areas of application can be anything between checking Wikipedia articles for biased arguments, making sense of online product reviews, or even further analysing public opinion from tweets or Facebook statuses that are relevant for a certain issue;

- **In Multi-agent Systems** — the use of complex arguments could be useful in improving inter-agent communication and understanding, thus making them able to solve problems much more swiftly and probably more efficiently [XYLL14]. The benefits of better Multi-agent Systems are manifold, and range from video game enhancements to safer air traffic control.

Many of the applications of Argumentation Mining are unpredictable at this point, as it is still a recent field of AI. However, considering the possible applications we can imagine, it has a lot of potential and has been the target of more and more interest over recent years.

We worked on this thesis in an effort to contribute to Argumentation Mining. Namely, we wanted to create a place that would make people more prone to annotate argumentative corpora on their computers rather than on paper. This place, which turned out to be an online platform, can also make it much easier for users to annotate documents and keep track of their annotations. So, if a perfect all-encompassing platform was available to annotate documents, maybe people would default to annotating online. This, in turn, would result in more available corpora to use in the machine learning process.

Of course such corpora would not be usable without a measure of its reliability. Since the methods of measuring this reliability are also hampering the growth of Argumentation Mining, and as another way of making people more involved in the annotating documents, we also wanted to provide a way of automatically evaluating annotations. On one hand, this helps Argumentation Mining in that it can offer a different and more comprehensive way of evaluating the reliability of a corpus. On the other hand, having instant feedback is a feature that may encourage more people to annotate documents online and it can, again, lead to the creation of more corpora for machine learning purposes.

### 1.3 Problem Statement

Just like with most attempts at Argumentation Mining, the first problem we encounter is related to obtaining quality input data.

Argumentation Mining requires reliable corpora of annotated argumentative documents, and in Portuguese there are few resources of this kind. This is partly because creating good annotated corpora is an arduous job. First of all, reliable annotations require trained and experienced annotators. Secondly, there must be availability on their part to annotate the great number of texts that are necessary for Argumentation Mining. All this takes a disheartening amount of time and effort.

Furthermore, it is not sufficient to simply annotate a corpus. Even if this task is done by annotators with strong experience and background, there will still be different annotations for the same text. Annotators can annotate in distinct ways, for example choosing to divide text strings into two argumentative discourse units (ADUs) or keep it as one, as noted in [WMGA14], or deciding whether to include punctuation in the ADUs. On the other hand, if the annotators are not experienced, a whole different set of concerns arises in addition to these. Even if a lot of

attention is dedicated to their training and to making explicit annotation guidelines, there is still a high chance that the new annotators will not be able to recognize some of the more implicit parts of the argument, or that they will wrongly annotate some of it.

So, before we can deem that a corpus is ready to be used in machine learning techniques, we must determine its reliability. However, this is another challenging task because the most common and generally accepted way to do this is by calculating the inter-annotator agreement (IAA). This approach is the target of much contention, as mentioned in Chapter 2, since there are reports of inconsistencies and flaws of the existing metrics. Still, with no well-established solution to address this issue, most researchers use Krippendorff's  $\alpha$  or the F-measure to find the reliability of their corpora [HEKG14].

Another problem worth tackling is generating a gold standard annotation based on the human annotations. This generated gold standard annotation should represent the correct annotation of a certain text using the human annotations even if they are wrong, which would make the whole evaluation of the corpus reliability easier and more trustworthy. It would also make it faster to run agreement metrics on corpora.

All in all, there are two issues at hand:

- **Obtain high-quality annotations** — in sufficient number to later apply machine learning techniques. We must employ some visualization techniques in the ArgMine Platform to increase the chances of getting high-quality annotations;
- **Evaluate the reliability of the annotations** — by using the existing IAA metrics or proposing a new one.

To address these questions, we lined up the following goals, which were worked on during the dissertation:

### **Obtain high-quality annotations**

For this, we improved the ArgMine Platform by strengthening its project managing side, as well as its annotator tool. As far as the managing of projects in the ArgMine Platform is concerned, it should be noted that these projects are meant to be annotation projects. The result of this is presented in Chapter 3.

These changes will help us reach the goal of obtaining high-quality annotations because a more attractive platform and a more pleasant annotating experience will drive users to use the platform more and more. In the end, we will be able to use those annotations for purposes of Argumentation Mining.

### **Evaluate the reliability of the annotations**

To evaluate the reliability of a corpus, as explained in Chapter 5, we not only run some of the usual IAA metrics but we also propose an approach to use the evaluation metrics we implemented as IAA metrics.

This can help with evaluating corpora reliability because evaluation metrics can take into account the relations between the annotation nodes rather than just the text in the nodes. Furthermore, it would be easier to pinpoint what aspect of a given annotated corpus yielded less agreement (node text, node role, or relations, as exposed in Chapter 4) and further develop said corpus or the annotation guidelines that resulted in the corpus.

### 1.4 Research Goals

With these problems in mind, we aimed to answer the following research questions:

1. What are the properties of annotations of arguments that are more relevant to calculate agreement and how can we use them to create comprehensive evaluation metrics?
2. How can we determine the reliability of an annotated corpus with arguments and which are the appropriate evaluation metrics?

The process of finding answers to these questions entailed the improvement of an existing platform, the ArgMine Platform that will be introduced in Chapter 3, and its annotator tool. In addition, to improve the platform and motivate its use, we implemented annotation grading metrics that are flexible to match different grading styles and priorities. Finally, addressing the fourth question, we implemented IAA metrics and adapted the evaluation metrics to be used as IAA metrics. Then, we applied all of them to an existing corpus. In short, this thesis provides the following contributions:

- A platform focused on the management of annotation projects;
- An annotator tool that makes it easier to annotate texts and keep track of complex annotations;
- Annotation evaluation metrics which take a global look at annotations and so yield grades that encompass all aspects of these annotations, answering the first research question;
- A showcase of some of the most widely used inter-annotation metrics for argumentative corpora, as well as a comparison between their results and the results of the proposed evaluation metrics used as IAA on a corpus, answering the second research question.

### 1.5 Dissertation Structure

After this introductory chapter, this dissertation has 5 more chapters. In Chapter 2, there will be a description of the state of the art in Argumentation Mining, as well as in IAA techniques. In Chapter 3, there will be an exposition of the ArgMine platform, including its annotator tool. Then, Chapter 4 will include a description of the newly implemented annotation evaluation metrics, followed by Chapter 5 where we implement these same evaluation metrics as IAA metrics,



## Introduction

along some of the most widespread IAA metrics, and run them on a corpus of annotations. Lastly, Chapter 6 summarizes this dissertation and proposes some future work to be developed.

## Introduction

## Chapter 2

# State of the Art

This chapter introduces Argumentation Mining. Firstly, there will be an introduction on how this field emerged, followed by some definitions of fundamental concepts that are necessary to understand the following sections. Then, because this dissertation will focus on inter-annotator agreement (IAA), a detailed explanation of the measures that are currently used for this purpose and an overview of IAA in recent research will be provided. Finally, we survey the recent developments on Argumentation Mining.

### 2.1 Introduction

The study of argumentation can be traced as far back as ancient Greece, when Aristotle began the study of Logic [MM11]. Since then many different disciplines have stemmed from logic, including Argumentation Theory. Nowadays, with the rise of technology in every single field of human intervention, it is only a matter of time before there are automated ways to apply Argumentation Theory. This falls in the field of Artificial Intelligence, one of the current front-runners of the technology world which is expanding in many fields, including Argumentation Theory.

This particular application of AI, called Argumentation Mining, aims to extract arguments from natural text into structured diagrams representing each part of the argument and the connections between them, which is anything but an easy task [MM11]. Since natural text is highly variable, ambiguous, subjective and context dependent, Argumentation Mining has a lot of obstacles to overcome. So, it has only seen progress in recent years, with the development of text mining techniques and of new ways of approaching natural language.

Nevertheless, Argumentation Mining has a lot of potential in any context which deals with written natural language, because argumentation is one of the staples of human communication and we are confronted with it on a daily basis.

## 2.2 Fundamental Concepts

Since the establishment of Argumentation Theory there have been several definitions for the many concepts that comprise this area of study, and the ones presented here are simply meant to provide a comprehensive basis that will allow for the understanding of the approaches adopted in this dissertation, based on [BH08]:

- **Argument** — A set comprised of a claim, one or more premises, and the series of reasoning steps that connect the premises in a way that successfully proves the claim;
- **Claim** — the conclusion that the argument aims to prove;
- **Premise** — the information that is assumed to be true in the process of proving the claim;
- **Counterargument** — given two arguments,  $A_1$  and  $A_2$ ,  $A_2$  is a counterargument of  $A_1$  if it claims the negation of one of  $A_1$ 's premises or of  $A_1$ 's claim;
- **Argumentation** — the process of constructing, comparing, evaluating, and judging arguments.

To illustrate the previous concepts, we can look at the following example:

*Premise: "Zoos cause a lot of suffering to animals"*

*Premise: "The benefits that zoos provide can be obtained through more ethical methods"*

*Claim: "Zoos should be banned"*

*Argument: "Zoos cause a lot of suffering to animals and the benefits that zoos provide can be obtained through more ethical methods. So, zoos should be banned."*

*Counterargument: "This is untrue because zoos do a lot of important work to improve animal well-being, as well as play a critical role in the conservation of endangered species. They would not be able to do this if they were not able to fund themselves, so zoos should not be banned"*

It is worthwhile to note that an argument may be true or false despite its logical validity. So, an argument achieves its goal of proving the claim if the logical steps between premises are sound and if the premises are undeniably true.

As one of the first attempts at looking at an argument in a more organized manner, Whately [Wha41] proposed argumentation schemes in the 1800s – arguments were composed of only premises and a conclusion. Much later, in 1958, Toulmin introduced a more complex view of arguments that included the concepts of Data (factual information to prove the claim), Claim (the stance that is meant to be proved with the argument), Warrants (a possible hypothetical statement to illustrate the logical steps between the data and claim), Qualifiers (classifying the level of certainty that the speaker has in the argument), Rebuttals (a counterargument), and Backings (that support the

warrants). However, for the most part, these concepts are still too complex to be successfully integrated with Argumentation Mining efforts.

The further development of Argumentation Theory also brought some diagramming concepts that make analysing arguments much simpler:

- **Argumentation Diagram** — can be thought of as blueprints mapping the actual words of the argument to the corresponding theoretical sections, “revealing the internal structure of a single argumentation” [VEG09]. The basic components of an argument diagram will be presented below;
- **Argumentation Scheme** — the internal structure, or "form" of an argument [Wal13].

An argumentation diagram may include the basic components in Figure 2.1 [Fre91]. These include, from left to right:

1. **basic argument** – when a single premise is used to support a claim in an argument;
2. **linked argument** – when two or more premises are used in tandem to support the argument claim and are meant to be considered together. In this diagram in Figure 2.1, claim 1 could not be supported with only premise 2 or only premise 3. In addition, in this situation the connection between the premises and the claim can also be called "linked". So, we can say that nodes 2 and 3 are connected to node 1 by a linked connection (or linked relation);
3. **convergent argument** – when two premises support the claim independently, not only is the argument labeled as "convergent", but so are the connections between the premises and the claim. So, we can say that nodes 2 and 1 of this argument diagram are connected by a convergent connection (or convergent relation);
4. **serial argument** – when the premise that supports the claim has another premise backing it, as seen on the left-hand side diagram in the second row of Figure 2.1;
5. **example argument** – when the claim is supported by providing examples, as seen on the right-hand side diagram in the second row of Figure 2.1;

Finally, another important concept is that of the gold standard annotation. In an annotated corpus, there can be one gold standard annotation per document, and it is the annotation that is considered to be the correct representation of the argument in the text [WAMP14].

The gold standard annotation is something very hard to define for any given text, because different people will have different opinions not only on the argument at hand but also on how to annotate it. On top of this, it must also be done and chosen by humans. These aspects add biases to the gold standard annotations of a corpus, even if they are considered to be the correct ones.

## State of the Art

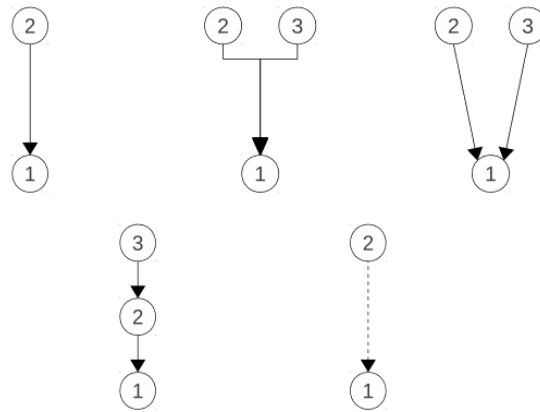


Figure 2.1: The basic components of argument diagramming [Fre91]

### 2.3 Annotation and Creation of Corpora

Before even beginning the Argumentation Mining process described in the next section, there is a need for annotated argumentative corpora. These are collections of argumentative documents and the arguments that people have extracted from them in the form of annotations, and they are an essential part of Argumentation Mining because they are used as input data for the machine learning process.

For that reason, the characteristics of each corpus become a very important aspect. Researchers must be aware of what type and genre of text they choose for their corpora, as well as the language of the documents, which will typically be the language of the annotations. Other important factors in the creation of corpora include the argumentation theory that will determine the building blocks of the annotations, and the annotating experience of the annotators [MI09].

Given the limitations that a corpus' language brings to its possible usages, it is important that there is an effort to create corpora for each language. Some annotated argumentative corpora can be found in [Bra00], [BEF<sup>+</sup>06], [Bud11], amongst others. In Portuguese specifically, there is only one corpus available, presented in [Roc16], and this thesis is also part of the same project which resulted in that corpus.

The first concern in creating an argumentative corpus is writing the annotation guidelines that the annotators will follow when annotating the documents. A globally accepted set of guidelines does not exist as of yet, so each research team must establish the one they will use for that project. The guidelines will include instructions, such as whether to annotate final punctuation, that will influence the argumentative discourse units (ADUs) that will be annotated.

ADUs are the minimal units of argument, and there is no standard that specifies what boundaries they should have. So ADUs are not necessarily words, but whichever textual unit (word, whole sentence, part of a sentence, etc.) is considered to be the most promising in terms of yielding significant information, given the annotation task at hand. For example, if we consider a "Named-entity Recognition" task (as described later in this chapter), where annotators will most

likely annotate single words. While if we consider the more complex task of argument annotation, for example, the annotated ADUs will most likely be argumentative phrases.

It follows that an ADU may vary in its boundaries from annotation to annotation. This happens especially when the task requires annotating bigger ADUs, as different annotators may have different ways of annotating [WMGA14]. For example, if two annotators annotated the same sentence but one of them also annotated the connection word "because" (which is typically left out), their ADU boundaries will differ by 1 word.

Once the argumentative texts, the annotation guidelines, and the annotators are available, the annotation of the corpus can begin. However, the annotation of the corpus alone is not enough for it to be usable as input data for machine learning.

To finally use the corpus for machine learning purposes, researchers must first guarantee that it is reliable and reproducible, which is where the inter-annotator agreement (IAA) comes in. IAA will be further explained in this chapter, but the fact that good corpora entails high inter-annotator agreement influences the creation of corpora in the following ways:

- Need for clear and detailed annotation guidelines – so that the annotators will annotate the same argument the same way and the IAA result will be high;
- Need for more than two annotators – so that the IAA result will be more meaningful, there should be a way to break a tie between two annotator with radically different argument annotations.

## 2.4 Argumentation Mining: Process

In Argumentation Mining, all the previously mentioned concepts will come into play as we attempt to extract arguments from natural language text. This process is essentially done in four steps [PS15b]:

- **Segmentation** — dividing the text into argumentative discourse units (ADUs), the minimal units of argument. There is no standard that specifies the boundaries of ADUs. They could be a whole sentence, part of a sentence, two parts of a sentence combined, etc.;
- **Classification** — determine what the identified ADUs are in the argument. Under different argumentation models, this could entail more or less complexity. In the case of the premise-claim model, an ADU can only be classified as a premise or a claim. In more complex models like Toulmin's, an ADU can fall into the categories of claim, data, warrant, backing, qualifier, or rebuttal;
- **Relation Identification** — relate the ADUs to each other, either directly or indirectly, thus constructing the explicit scheme of the argument;

- **Completion** — adding to this scheme all the parts of the argument which may not be explicitly stated in the text, such as context or information that is assumed to be commonly known. This leads to a complete representation of the argument.

## 2.5 Inter-Annotator Agreement

To carry out the Argumentation Mining process, there is a need for argumentative texts with their arguments duly annotated to use as input. Unfortunately, annotation is a costly operation. Firstly, because the most credible argument annotations would ideally be performed by trained and experienced annotators, who may not be available to annotate the extensive amount of texts necessary for machine learning techniques.

Secondly, because there are not any standard annotation guidelines, so many research teams end up having to create new ones to fit their datasets.

However, we cannot consider that an annotated corpus is perfectly reliable after the annotators have finished their task, because even experienced annotators trained with excellent guidelines may annotate arguments differently. For example, in [WMGA14], for the case of argument annotation, the authors find that there are mainly two types of annotators. Those who, upon annotating a sentence with two or more premises, considered it as only one ADU (the "lumpers"), and those who, faced with the same task, divided it into several ADUs (the "splitters"). Some of these issues can be solved by comprehensive annotation guidelines. For example by telling annotators never to include the final punctuation of a sentence in the ADUs. Still, there will always be some inherent differences in the way people annotate arguments.

To deal with this and find how much we can trust a certain annotated corpus, it is customary to calculate the inter-annotator agreement between the annotators of the corpus on each document. By doing this, we make the assumption that if a lot of people agree that an annotation should represent a given text, then that annotation is probably the correct one. Nonetheless, calculating the IAA between annotations is not a trivial matter. There is contention on exactly how to do this, on how to interpret the results of the current metrics and even on whether the available metrics are reliable. For example, in [WMGA14], it is found that inter-annotator agreement methods can provide inconsistent results when they are applied to annotations including fuzzy boundaries and multiple coders. We will go into more detail about the most common IAA metrics and their shortcomings later in this chapter.

Despite the drawbacks, and in lack of a perfectly uniform solution for the evaluation problem, a few metrics have been proposed to fulfill the need for a method to compare annotations, mostly coming from statistics and "inter-rater agreement" metrics from the last half of the twentieth century. Among these are [AP08]:

- **Kappa ( $\kappa$ ) measures** — such as Cohen's  $\kappa$ , Fleiss'  $\kappa$ , Scott's  $\pi$ ,  $S$  (introduced in [BAG54]), and others. They are based on comparing the obtained agreement ( $A_0$ ) with the expected agreement ( $A_e$ ) between annotators, the expected agreement being the result of annotators



answering randomly. These measures use the following ratio to calculate the agreement found between annotators:

$$\frac{A_0 - A_e}{1 - A_e},$$

where  $A_0 - A_e$  represents the agreement beyond the accidental agreement expected if the annotations were random, and  $1 - A_e$  represents how much actual agreement can be expected. Then, what fundamentally differentiates the kappa measures is how they calculate the expected random agreement.

In  $S$ , proposed by Bennett et al. for two coders (in this case, annotators) in [BAG54], it is considered that if all annotations were done randomly it would result in one uniform distribution. This can be a problem because, if all the categories are equally likely, the score can be raised by adding categories that would never be used.

In  $\pi$ , proposed by Scott for two coders (in this case, annotators) in [Sco55], it is considered that if all annotations were done randomly, the result could be represented by one non-uniform distribution. This distribution is calculated by taking the frequency that all annotators classified a unit into a certain category and dividing it by the total number of items classified by all annotators. Because it is more in-line with the real tendencies of the annotators, the expected agreement used in  $\pi$  will be higher than that used in  $S$ . In turn, this means that  $\pi$  will always result in less agreement than  $S$ .

In  $\kappa$ , developed by Cohen for two coders (in this case, annotators) as an extension of Scott's  $\pi$  [All19], it is considered that if all annotations were done randomly, all of them would result in different distributions, one for each annotator. This distribution is calculated by taking the frequency that each annotator classified a unit into a certain category and dividing it by the total number of items classified by that annotator, joining this probability for all annotators, and then summing the joint probabilities across all the categories. Cohen's  $\kappa$  takes each annotator's idiosyncrasies individually, rather than lumping them all into an hypothetical annotator with all the flaws and biases of both annotators. So,  $\kappa$  will always yield less expected agreement than  $\pi$  and consequently the inter-rater agreement calculated with  $\pi$  will always be lesser or equal to that which is calculated with  $\kappa$ .

In multi- $\pi$ , proposed by Fleiss as a generalization of Scott's  $\pi$  for more than two coders, the agreement is calculated with pairwise classification agreement rather than item (in this case, ADU) classification agreement. The agreement for each ADU is calculated using the following formula:

$$agr_i = \frac{1}{\binom{c}{2}} \sum_{k \in K} \binom{n_{ik}}{2},$$

where  $i$  is the ADU to be classified,  $c$  is the number of annotators,  $K$  is the set of possible categories, and  $n_{ik}$  is the number of times that the ADU  $i$  was classified into the category  $k$ . This means that we take the summation, for all categories, of the number of pairs of annotators who agree that the ADU should be in a certain category and divide that sum by the total number of possible pairs of annotators. As for the expected agreement, it is

also calculated with pairs of agreement rather than individual ADU classification. As this measure is a generalisation of Scott’s  $\pi$ , Fleiss also used the combination of all annotators’ resulting distributions to calculate the expected agreement. In this case, the final value for the expected agreement is computed as the joint probability of all annotators obtaining the generalised distribution of ADU classifications.

In Multi- $\kappa$ , proposed by Davies and Fleiss as a generalization of Cohen’s  $\kappa$  for more than two coders, the observed agreement and the expected agreement are also approached in a pairwise manner. However, as in Cohen’s  $\kappa$ , when in Multi- $\kappa$  the expected agreement is calculated as the joint probability of each annotator obtaining their own random distribution of ADU classifications.

However, these metrics are not without flaws. For example, Cohen’s  $\kappa$  will be undefined if two annotators completely agree in one of the categories, and 0 if one of the annotator classifies all ADUs into the same category [Xie]. Furthermore, the adverse effects of prevalence (the distribution of the data throughout the categories) and bias (the discrepancies between each annotator classifications) is also well documented, for instance in [BBC93], [BCMS99], and [Gwe02]. In general, it is reported that a higher prevalence index (the difference between the probability of the categories) results in a lower  $\kappa$  while a higher bias index (the difference of the proportions of a category for both annotators) will result in a higher  $\kappa$ . Similar problems with Scott’s  $\pi$  have also been identified [G<sup>+</sup>02]. For these reasons, it is advised to measure and present prevalence and bias data alongside with the final result of one of these  $\kappa$  metrics. Another limitation of these metrics is that they are only applicable when there is a set of categories into which the preset ADUs can fall into. An example of this type of annotation is classifying words into parts of speech (verb, noun, adverb, etc), since the ADUs and the categories they can be put into are well defined. Clearly, this takes a lot of flexibility out of the annotation process.

We can see the kappa measures being used in [PS15a], [ZHHL17], [MSK<sup>+</sup>18], amongst others.

- **Krippendorff’s  $\alpha$**  — presented by Krippendorff in 1980 [Kri80], is based on expected disagreement between annotators. The formula can be written in function of the expected agreement as follows [Gwe11]:

$$\alpha = \frac{P_\alpha - P_e}{1 - P_e},$$

where  $P_\alpha$  is the weighted percent observed agreement and  $P_e$  is the weighted percent random agreement, calculated on a pairwise basis and based on the severity of the disagreement. Like the case of calculating the agreement in Scott’s  $\pi$ , the randomness here is considered to be independent from any individual annotator [AVL14]. So, for  $P_\alpha$  and  $P_e$  to be calculated, there is a need to classify the severity of disagreement between categories, a notion that is completely absent from the  $\kappa$  measures. The severity of disagreement will be set by weights

given to each pair of categories, ranging from 0 (full disagreement) to 1 (full agreement), and Krippendorff has suggested different "metric differences" to employ based on the type of data to be analysed. These include nominal data, ordinal data, ratio data, interval data, among others [Gwe11]. To illustrate the importance of different severity of disagreement, we can use the following example: we can assume that an annotator who classifies an ADU as "data" agrees to a higher degree with an annotator who classified the same ADU as a "warrant" rather than with an annotator who classified it as a "rebuttal" of the argument. Another thing to consider when calculating Krippendorff's  $\alpha$  is that there might be ADUs that are classified by only one annotator, and some that aren't classified at all. These are ignored in the formula.

Krippendorff's  $\alpha$  brings advantages relative to the  $\kappa$  measures, namely the possibility to adapt to any number of annotators and allowing them to choose which ADUs to annotate, as well as being capable of handling missing data [AVL14]. So, it is a promising measure for calculating the inter-annotator agreement between argument annotations.

However, like all the other existing measures, Krippendorff's can yield some questionable results. Various shortcomings of  $\alpha$  are reported in [ZLD13], for example its unignorable dependency on sample size.  $\alpha$  tends to favour smaller sample sizes and punish larger ones based on the assumption that if the sample size is larger, there must be more random agreement to be accounted for. In penalizing agreement when the sample size is larger, it may happen that completely random annotations will result in higher  $\alpha$  values than true annotations of a larger sample [ZLD13].

- **F-Measure** — also referred to as F1 score or F score, requires a gold standard annotation against which to compare other annotations. It is based on precision and recall. The precision,  $P$ , is the number of ADUs which matched gold standard ADUs (true positives,  $tp$ ) divided by the total number of annotated ADUs (true positives and false positives,  $fp$ ):

$$P = \frac{tp}{tp + fp}$$

The recall,  $R$ , is the number of annotated ADUs which matched gold standard ADUs divided by the total number of gold standard ADUs:

$$R = \frac{tp}{tp + fn}$$

Finally, the  $F$  score is the harmonic average of the  $P$  and  $R$  values, and it results on a scale of 0 to 1:

$$F = \frac{2}{\frac{P+R}{PR}}$$

An advantage of using the F-Measure is not having to know the number of negative cases, which in the case of evaluating agreement between argument annotations is advantageous because in a text, the number of unmarked ADUs and the number of ADUs that are not

related to the argument is difficult to define [HR05]. However, there are also reported shortcomings of the F-Measure. First of all, being the harmonic average of precision and recall, some detail is lost if we just look at the overall F-Measure value. That is, by looking at an F-score we do not know whether precision is lower than recall or vice-versa. Another serious issue with the F-Measure is the influence of bias. That is, if one category is much more common than another, an annotator who blindly chooses this category every time can still perform better than an honest annotator who makes some mistakes [Pow15]. Typically, this measure is used to evaluate classification systems' performance, but it has been used to indicate the reliability of corpora, as was done in [Bra00], [WMGA14] and [LGP18], perhaps because of how easy it is to calculate and interpret [LGP18].

IAA metrics can be used to evaluate the agreement on various kinds of annotations, and naturally the most appropriate metric depends on the type of annotation. For example, the  $\kappa$  measures are only suitable for when there are strict categories into which the annotators are classifying the ADUs [AP08]. For this reason, they are quite helpful in calculating agreement when, for example, the annotations classify words into the categories "country" and "person". When the complexity of the annotations increases, there is a need for more flexibility in the selected IAA metric. For these cases, which include the type of whole argument annotations which we will be working with, Krippendorff's  $\alpha$  and the F-Measure are the most appropriate ones because they adapt to whatever ADUs are chosen for the annotation.

Out of these and many more proposed measures, the one that seems to prevail in recent research for evaluating the outcome of classification systems, after being popularised in information extraction competitions [Pow15], is the F-measure. On the other hand, the most common measures for evaluating the reliability of an annotated corpus seem to be Cohen's  $\kappa$  and its generalization by Fleiss, the Multi- $\kappa$ , as well as Krippendorff's  $\alpha$ .

More recently, the authors in [DLBR16a] have developed a technique to integrate the scoring of several parts of the annotations and combining them in a final score. It is called Combined Argument Similarity Score (CASS), and it aims to bring uniformity to the comparison of the results of different research teams and thus catalyse communication in the field.

In [DLBR16a], the authors proposed that the score of each annotation should have three components:

- Segmentation ( $S$ ) – the component that scores the agreement on the ADUs on each annotation. As the components are allowed to be measured using whatever technique, this score may focus more on ADU boundaries, the number of tokens, or other ADU-related scores. In [DLBR16a], the authors suggest three techniques to use in this step that will not result in big penalization for small differences in segmentation;
- Propositional content relations ( $P$ ) – the component that scores agreement on the relations between segments. In [DLBR16a], the authors begin by using the Levenshtein distance

[Lev66] to match segments between annotation, without forgetting to account for each segment's words positions in the original text. Then, they check if the connections between matching segments also match in type (support/attack);

- Dialogical content relations ( $D$ ) – the component that scores agreement on the dialogical relations in the annotation. These types of relation differ from the relations in the component above in that they concern the dialogue and intentions of the speakers during the argument. They are considered separately so as not to penalise doubly because of the segmentation component, and the Levenshtein distance is calculated here again for the dialogical content nodes. These nodes are then matched between annotations. Finally, to score this component, the authors use two calculations to take into account the type of node that corresponds to the relation and the origin and destination nodes of the relation;

After scoring all these components, they are finally aggregated using the CASS technique. For this, there are two formulas:

$$M = \frac{\sum P + \sum D}{n}$$

$$CASS = 2 \times \frac{M \times S}{M + S}$$

The top equation ( $M$ ) is the arithmetic mean of all the propositional content scores and the dialogical content scores. For that we sum all the  $P$  scores and all the  $D$  scores and divide them by the total amount of scores ( $n$ ). The bottom and final equation, is the one that provides the CASS score of the two annotations.

The CASS technique is very promising in terms of allowing a great deal of flexibility when choosing which agreement metrics or techniques to use to score each of the three components. It is also "largely independent of annotation scheme" [DLBR16b], which includes the AIF format. However, when proposing this technique, the authors had in mind annotations with dialogical data, and it is unclear how annotations with no such data would fare when evaluated using this technique.

## 2.6 Argumentation Mining: Recent Developments

With Argumentation Mining being a recent field with many unanswered questions and warring definitions across the literature, it is still making use of generic algorithms in pursuit of its first milestones. So, at this point, the main success factor is the features which are fed to the algorithms [LT15]. This is a problem that emerges at the very first stages of Argumentation Mining, which is an indication of the precariousness of the area.

### Segmentation and Classification Steps

As these are the first steps in the Argumentation Mining process, most of the research efforts have been concentrated in this step. In the engineering of features to represent the target texts, the most common approaches include Bag-of-Words, Part-of-Speech Tagging, Named-Entity Recognition, amongst others [MRS10]. An approach like Bag of Words can be considered very naïve [LT15], since it cannot account for similarity between words and words that are more likely to follow one another, as it simply takes into account the frequency of each word in a text. However, approaches like this are still widely used because of their simplicity and a lack of any general method with a better performance. Furthermore, a common tagging format used across Argumentation Mining is BIO tagging, where tokens which do not belong to a chunk (a chunk can be, for example, an argumentative phrase) will be tagged as O, whereas tokens initiating or continuing a chunk will be tagged as B and I, respectively.

There have been efforts to overcome the simplicity of the traditional feature engineering methods. For example, [LBH<sup>+</sup>14] use the WordNet, a lexical database that groups together words with similar meanings, to be able to incorporate the detection of synonyms in the texts. In 2015, Habernal and Gurevych managed to outperform the previous argumentation unit classification regarding “cross-validation, cross-domain and cross-validation evaluation scenarios” [HG15]. They consider each sentence as a token, and use sequence labelling based on Support Vector Machines with the Hidden Markov Model to classify them. With this, they successfully employ a semi-supervised method that comes closer to eliminating the need for manual text annotations. In their research they manage to obtain F1 scores of 0.30 to 0.40, which are comparable to other less complex approaches.

Some other innovative methods that have been attempted recently in the segmentation and classification steps include Multi-Task Learning (MTL). This recent research has found, contrary to what was previously thought, that MTL performs well when dealing with little training data and even with semantic tasks in addition to syntactic tasks [SED<sup>+</sup>18]. [SED<sup>+</sup>18] found that by using semantic tasks as auxiliary tasks, they could obtain better results than when using Single-Task Learning (STL). These results were visible both on the segmentation step (BIO tagging), and on the classification step, and with datasets as small as 21,000 tokens.

This MLT was later incorporated into Graph Convolutional Networks (GCN) in [Mor19] to tackle the aforementioned first two steps, segmentation and classification. Here, the authors “employ a bi-directional LSTM (BiLSTM) and conditional random field (CRF) layers, namely, a BLC” [Mor19] and report promising results regarding the use of syntactic GCNs, even with small sets of data. However, their data were not highly imbalanced, so the performance of their method in such a case is still undetermined. Other researchers have also tried to make use of neural networks in these steps, an effort which likely started with [CWB<sup>+</sup>11], where the researchers presented a versatile unified neural network architecture. For example, in [LBG<sup>+</sup>18] a weak signal is used to train Deep Neural Networks (DNN). The quality of the obtained results is comparable to those obtained by supervised methods, but this approach entails the advantages of not needing a homogeneous data source and of not being domain-specific [LBG<sup>+</sup>18]. So much so that they used a

Wikipedia dump in their work, albeit pre-processed before use.

### Relation Identification and Completion Steps

The last two steps are, understandably, the most challenging ones - especially since there is no bullet-proof way to deal with the first two steps, which should be used here. Nonetheless, there have been recent developments in solving these problems.

For example, in [CSH18] a method is proposed to identify uncertainty in scientific texts of a given discipline, as a means to try to account for truthfulness and consensus across said discipline. The research team proposes a framework that makes use of principal component analysis to group uncertainty words from the corpora into dimensions representing the type of uncertainty they provide (misleading, dispute, sceptical, amongst others) [CSH18]. Then, the authors employed Word2Vec, a well-established way to map words according to their semantic properties based on neural networks, to expand their model with new cue words. After this, they submitted the new cue words to two human evaluators and, using words that both agreed were uncertainty cues, they ran several different machine learning algorithms to evaluate them. The algorithm that performed best in identifying the uncertainty keywords was the Recurrent Neural Network, which, despite taking longer to be trained, has been gaining popularity in the machine learning community. This is a case of research that attempts to incorporate context into Argumentation Mining, and it showed promising results [CSH18].

Other attempts at dealing with context can be found in [NL16], where the researchers get encouraging results of F1 scores rounding 0.65 for corpus of student essays. Here, Nguyen and Litman use the concept of a “context window” surrounding a certain source where they look for possible argument components, as well as the concept of “topic-context” to deal with certain words which should be globally defined in the text.

More research including context can be found in [HJ17], where, skipping the segmentation and classification steps, relations of agreement or disagreement are found across different texts with the same topic. Hou and Jochim aimed to use these relations to find the stance in a given argumentative text. The logic behind this is that the stance must be something that the arguments will agree with and that the counterarguments will disagree with. For this they employ Markov Logic Networks, which with their testing dataset resulted in F1 scores of about 0.65. The authors claim that this F1 score is a good result, considering that the arguments they used for training the algorithm came from different texts, and so they might not have included the usual tell-tale connectors such as “however”, “and”, etc.

### End-to-end Approaches

Lastly, there have been attempts to perform end-to-end parsing of argument structures using neural networks, mostly since Miwa and Bansal presented their work in 2016 [MB16] [ZZF17]. In [SG17], the authors “encode the argument components using an IOB-tagset [RM99] and consider an entire essay as a single sequence. As a learner [they] use a CRF (Lafferty McCallum, and Pereira 2001) with the averaged perceptron training method [Col02]. Because a CRF considers

contextual information, the model is particularly suited for sequence labeling tasks [GLPK14].” [SG17]. However, this approach does not get to the “completion” step, so it is not capable of identifying missing premises, or incorporating context into the resulting structures.

Another recent development also using neural networks can be found in [ZZF17], where the authors use a globally optimized neural network rather than the more commonly used local classification, thus mitigating the threat of label bias. They manage to deal with context with long short-term memory (LSTM) features [ZZF17]. LSTM units are used for Recurrent Neural Networks. In this work, the innovative factor is how they see the output. Instead of using hidden vectors to represent words, they also use sequence representations. The authors then use the extracted base features into non-linear neural layers. Again, this approach is not capable of fully completing the argument, but it shows significant improvement over the statistical models usually employed in this field.

### Developments in Portuguese

As Argumentation Mining is a language sensitive research area and this thesis project was developed while working with text in Portuguese, it is relevant to go over some of the recent developments in Argumentation Mining in Portuguese.

To the best of our knowledge, LIACC has been pioneering in the field of Argumentation Mining in Portuguese. The work has been focused on:

- ADU boundary detection – as exposed in [Roc16], the authors use supervised machine learning techniques to identify ADUs in a Portuguese language corpus;
- Argumentative sentence detection – in [RLC17], the authors use supervised machine learning techniques in a relation-based approach with the objective of identifying inference relations between premise and conclusion in argumentative texts. In [Roc16], an approach to identifying argumentative sentences is also presented, using supervised and semi-supervised learning algorithms;
- Argumentative Relation Identification – in [RSLCG18] the authors tackle the task of identifying relations between ADUs using cross-language learning techniques, given the lack of annotated resources in Portuguese;
- Argumentation Mining framework – LIACC has been continuously developing and improving a set of tools meant to facilitate and partially automate Argumentation Mining research, as introduced in [RLCT16] and further explored in [Roc16]. This framework will be mentioned and further detailed in this dissertation.

## **2.7 Annotation Tools**

Over time, some annotation tools have been developed to facilitate the annotation process and the visualization of annotations. These include:



- **OVA+** — a project developed by ARG-tech in response to the introduction of the AIF file format. so, the main point of OVA+ is to visually represent arguments and allow for them to be shared and reused easily [REE14]. In this updated version of the previously called OVA, the annotator is geared towards the annotation of dialogue, as can be seen from Figure 2.2.

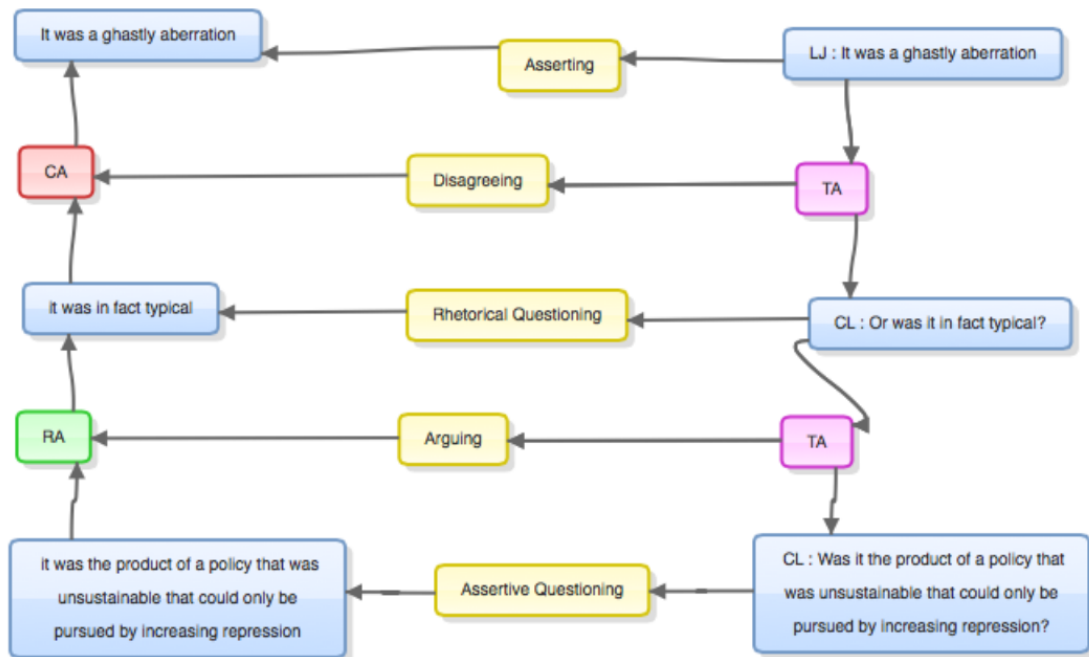


Figure 2.2: An annotation of dialogue in the OVA+ annotator, adapted from [REE14]

Everyone can upload a text, annotate it however they wish and then save the annotation (either as an image file, json file, or as an AIF file). It is even possible to not upload any text and create the nodes with the users own text. As we can see in Figure 2.2, all the nodes are rectangular and the relation nodes show different colors depending on their type. The text nodes can be created by selecting text from a panel that is not visible in Figure 2.2, or by double-clicking the annotation panel.

On the current version of the OVA annotator, the text nodes may also change color depending on what kind of relation nodes they are connected to. All the nodes can be changed by double-clicking them. If a user double-clicks the text nodes, they can alter the text, and if they double-click the relation nodes, they can alter their type. This annotator has a great potential for adapting to each user's needs, as there is a great number of different types of "edges" (relation nodes) that can be used to relate two nodes. On the conflict edges alone there are 38 different possible labels that the user can choose from, ranging from "Conflict from Virtue/Goodwill", to "Ad Hominem" to a few translations of the word "Conflict".

However, this ability to adapt and be specific to each user's needs also brings some drawbacks. Mainly that new users can be overwhelmed when taking a first crack at the annotator

## State of the Art

and become discouraged, and that the sheer amount of options can significantly slow down the process of annotating even simple arguments;

- **NOMAD** — a collaborative annotation tool that allows annotators to communicate amongst themselves during the annotation process [Pet14].

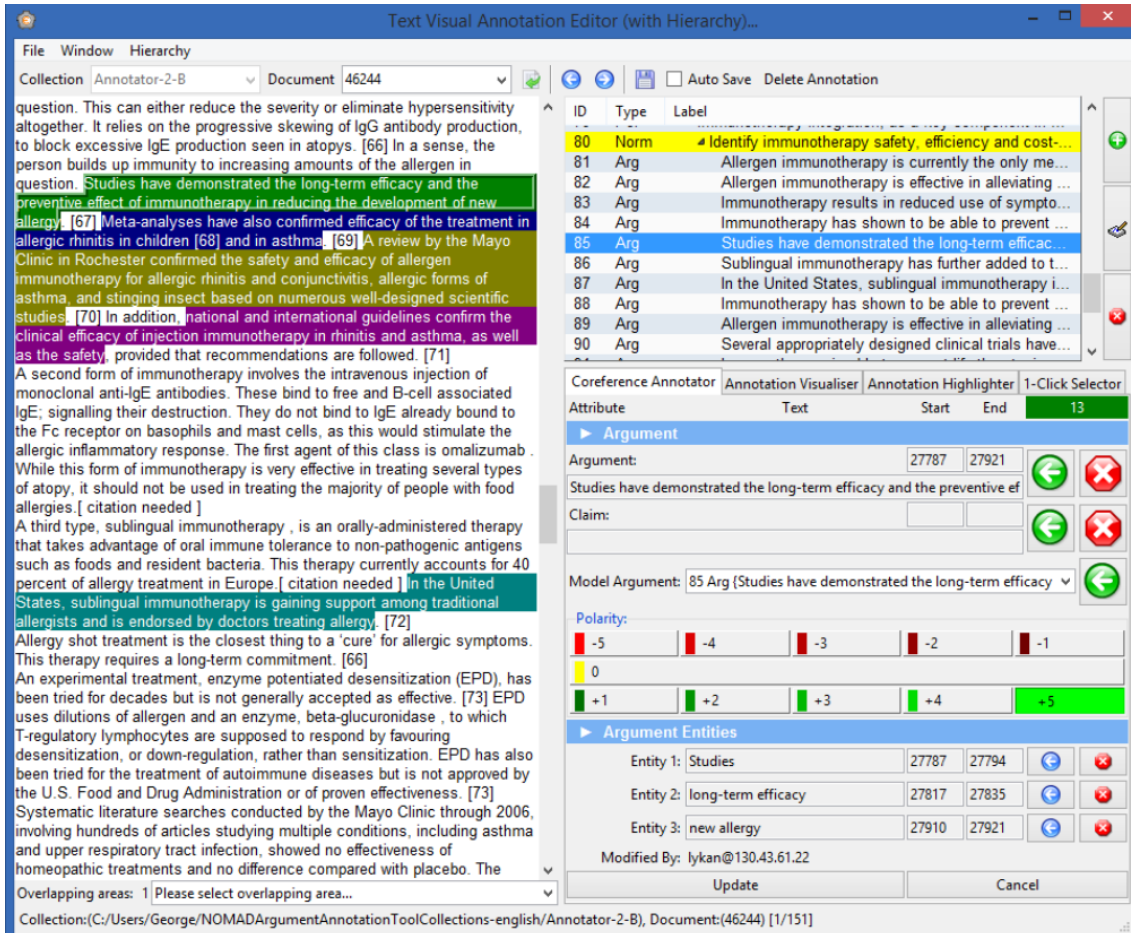


Figure 2.3: An annotation in the NOMAD annotator, adapted from [Pet14]

This is not an online annotator, but a single executable which requires no installation. Naturally, it needs internet access to function properly, as it will send notifications to other users in the annotation group. It focuses on matching arguments with preset argument models, and if the existing model does not fit the user's needs, they must create a new model. The default model used by the tool is the NOMAD model, consisting of four levels: domain, policy, norm and argument.

The annotation process has four steps:

1. Select a part of the argument from the text;
2. Choose what part of the model the selected argument belongs to. The instantiated model in the example from figure 2.3 can be seen in the top-right square. There are

## State of the Art

many portions of text annotated as "arguments" under a portion of text annotated as "norm";

3. Define the polarity of the argument, according to the strength with which it supports or attacks the policy or norm it is under;
4. Input the entities related to the selected argument, as seen in the example from figure 2.3.

A web version of this tool, called CLARIN\_EL, was introduced in [KPK16]. It presents mostly the same characteristics and annotation process.

All in all, the clear definition of models can help guide users through a text, and lead to better annotations and more inter-annotator agreement. However, the downside to this kind of annotator is that the creation of the argument graph is not very intuitive, as the user is not seeing it in front of them. This can seriously hamper the speed and rate of use of this application;

- **AVIZE** — a recently proposed annotation platform targeted at helping to construct argument schemes in the domain of international politics. For this, AVIZE also provides preset argument schemes to guide the user through the annotation process [LGBR18]

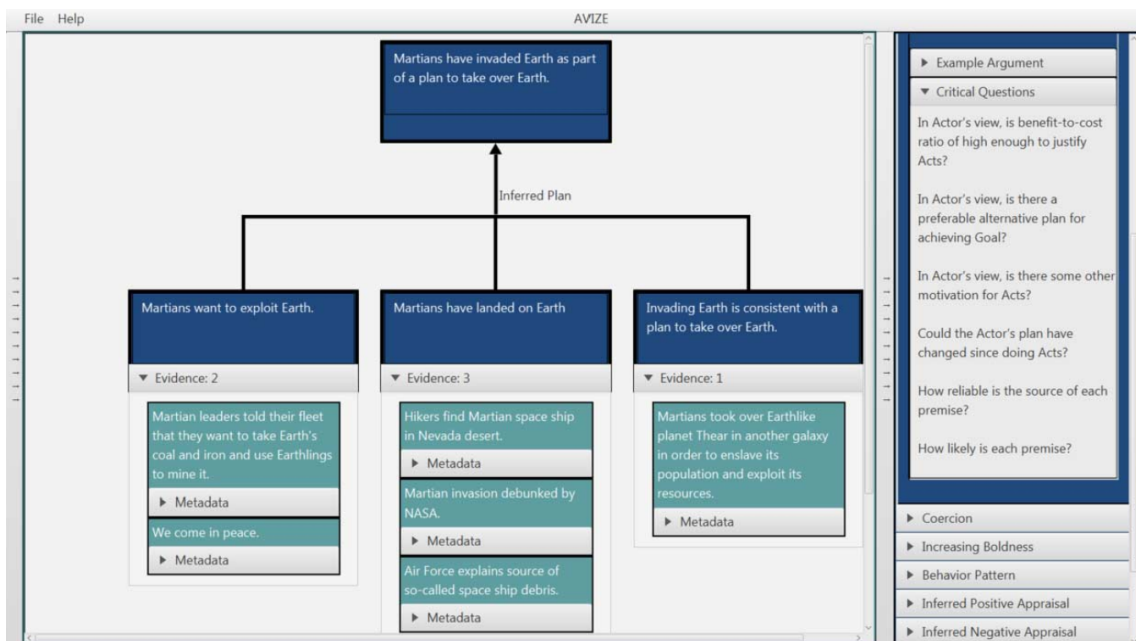


Figure 2.4: An annotation in the AVIZE annotator, adapted from [LGBR18]

In this annotator, the annotation process is as follows:

- Drag the desired portions of argumentative text into the annotation area (the panel on the left-hand side);

- Select an argument scheme from the existing ones on the panel on the right-hand side. These schemes can be dragged onto the left panel, and they will automatically morph from lists into graphs;
- Drag each previously selected portion of text to its match in the argument scheme boxes. If all the boxes are filled, or if a certain text does not match any boxes, it is possible to add more boxes (evidence-based proposition boxes are recommended [LGBR18]) or dragging more templates onto the left panel;
- Annotate relations of support or attack to connect the argument boxes.

In this annotator it is also possible to comment on nodes and express the likelihood or believability of certain premises or conclusions, which may be very useful for the understanding of arguments if it does not clutter the argument graph. It may also be good for the inexperienced annotator to be able to select existing schemes, but this is something of a drawback as well, since the combination of schemes and personalizable boxes may be too complex.

## 2.8 Conclusions

Argumentation Mining is a growing field with many potential applications and benefits. But it is also surrounded by a lot of instability regarding even basic concepts like which argumentation model to use, what annotation guidelines to follow, or which metric to employ when comparing results of different research teams.

While researchers started by mining legal arguments in strictly or at least very well-structured argumentative texts, the trend has been to expand beyond those. Nowadays we can find research which uses corpora of less structured or unstructured text (like online comments, threads, etc., such as used in [PC19]), even if it still has to be manually annotated as a means of measuring the success of the results.

Another shift that Argumentation Mining research has undergone is from statistical or supervised learning approaches to semi- or unsupervised approaches (e.g. [GNP16]), some more recent ones employing neural networks and deep learning. Since this way of tackling the problem has been showing good results, there has been further investment on behalf of more research teams.

As it is still in its infancy, there are a lot of directions on which Argumentation Mining research could focus on.

One of the big issues is the lack of a unifying set of guidelines for annotating corpora. Without this, it becomes harder to rely on metrics and compare results across the field. However, the fact that a unifying set of annotation guidelines have not been created yet only speaks to the difficulty of the task, and it seems difficult that a solution should arise in the near future.

As hinted by current research, the most promising courses of action for research in this area seem to be using neural networks and multi-task learning [CV18]. These approaches, in addition to offering above average results, seem to do so under precarious conditions, such as small

## State of the Art

datasets. So, they have the potential of leaping over some of Argumentation Mining's most relevant problems.

Finally, as the extraction of arguments from text migrates to less structured text, and eventually to completely unstructured text, it seems only natural that sentiment analysis should become a gradually bigger part of Argumentation Mining. Only then will we be able to complete the fourth step in Argumentation Mining process - adding the parts of the argument which may not be explicitly stated and that may indicate bias, irony, and other indicators that would further indicate the true intentions of the argument - and extract complete and accurate arguments from text.

## State of the Art

## Chapter 3

# The ArgMine Platform

The ArgMine Platform is part of the ArgMine Framework, which aims to bring together the process of annotating argumentative corpora and the semi-automated process of experimenting with different features pertaining to the different steps of the Argumentation Mining process [Roc16]. The ArgMine Platform has two main modules: the Machine Learning module, related to the natural language processing and machine learning process, and the Corpus Creation module, related to the creation of an annotated argumentative corpora [Roc16]. The ArgMine Platform, belongs to the Corpus Creation module.

As a result of a case study proposed by Universidade Católica Portuguesa Porto<sup>1</sup> (UCP Porto), the ArgMine Platform was changed to accommodate for another type of use. Universidade Católica, as it can also be referred to, holds a class where around 300 students learn about argumentation theory. For this class students annotate arguments, and the professors have to go through all the annotations and grade them individually. For the case study, we improving several aspects of the original ArgMine Platform, in an effort to tailored for the needs of the UCP Porto class.

In this chapter, we will present an in-depth description of the ArgMine Platform after improvements that were made during this thesis project.

### 3.1 Functionalities

We wanted a platform that could be used in the widest range of situations possible. But we decided that it should be grounded in managing annotation projects and suitable for a teaching setting as well, as we saw the most potential in that area. So, some of the most significant functionalities of the ArgMine Platform are:

- **Participant and Manager accounts** – The same email/password combination may be used to login to two different kinds of accounts: a Participant account or a Manager account.

---

<sup>1</sup><https://www.porto.ucp.pt/en>

## The ArgMine Platform

We kept these two roles separate in order not to create confusion in the platform about which projects a user joined as a participant or as a manager. The two roles differ in that the manager user can create and edit project, whereas the participant user can only, if the manager allows it, annotate and/or add documents. For the manager, editing projects means everything from changing the project name, to changing annotation settings that will be later mentioned in this section, to adding and removing documents, adding and removing participants, adding and removing other managers, etc.;

- **Project management capabilities** – It is possible to create projects and add participants and documents to these projects;
- **Active and inactive projects** – to be able to add participants, documents and managers, as well as set up the whole project before allowing participants to add annotations, the project is created as inactive by default. In inactive state, the participants are not able to do anything other than view annotations and documents. The manager can switch the state of the project from inactive to active at any time;
- **Allow adding several project managers in a project** – since the platform is suitable for a teaching setting, it would be natural that in this setting several professors would need access to the same project. So, a project may have any number of managers as long as this number is greater than 1. Furthermore, there are two types of managers: view-only managers and active managers. The latter type of manager is indistinguishable from the project creator, and once the manager becomes an active manager, they can even remove the privileges or delete the manager who originally created the project. Besides that, the difference between active managers and view-only managers lies in the fact that view-only managers cannot alter anything about a project.;
- **Create projects with stages** – in the class management setting that we envisioned for the platform, the same document may need to be annotated several times with different complexity levels, or different goals in mind (such as experimenting, correcting, etc). The stages are meant to represent these slightly different uses of the same document. The project managers can alter the stage and the number of maximum stages, and the participants are only able to annotate the current stage;
- **Automatically grade annotations** – it was perhaps the biggest point of interest when it came to our goal of having a platform that could be useful academically. We wanted the system to provide a way to automatically obtain the grades of each annotation. This instant feedback has the potential to motivate the students to correct and improve their annotations on their own. In the project settings, the manager may choose both the evaluating metric and whether the project participants are able to see their grades;
- **A visual annotator** – the ArgMine annotator includes several features that make it a great way to understand argument annotations. Users can choose the color of any text nodes, and



## The ArgMine Platform

these are always easily distinguishable from the relation nodes that connect text nodes. Because relation nodes have a hexagonal shape, whereas text nodes have a rectangular shape. Aside from this, the annotator helps the user keep track of which nodes have edited text in them, by showing a striped border instead of a solid one.;

- **Freedom of annotation** – keeping in mind that we want the ArgMine annotator tool to be usable in a wide range of contexts, it was very important factor to allow freedom of annotation. So, the annotator tool allows the user to edit the text of any text node, allows the user to create nodes without selecting any text from the document that is being annotated, and allows the user to create more than one text node by selecting the same portion of text;
- **Document collections** – meant to facilitate the creation of projects, it is possible for managers to create their own document collections. when adding documents to a project, the manager may also choose to add them to an existing or new collection. Afterwards, those documents can be added individually or by collection into different projects, saving the manager from having to manually input them again. Furthermore, it is a way for the manager to keep track of their documents, if they create different collections for different contexts, or different types of text;
- **Manage documents** – for successful project management, it is imperative for the manager to have a wide range of options regarding the documents in the project. So, in the ArgMine Platform, the manager has the possibility not only to add documents, but also to edit their title and body (if they have not been annotated yet), duplicate them (along with a gold standard annotation, if present), and to remove them from the project (which will remove the document from the database if it is not part of a collection and if it is not associated to another project);
- **Allow participants to add documents to projects** – upon the manager’s permission, participants are documents to projects. However, they cannot create or access document collections. This feature is very important for our goal of creating a platform that could be used in a variety of contexts, because by adding customization to the projects we also add some freedom for the user to make them into what they need.
- **Allow participants to view the document’s gold standard annotations** – also keeping the flexibility of the ArgMine Platform in mind, there is an option for the managers to allow the participants to see all the documents’ gold standard annotations;
- **Add participants and documents to a project using a .csv file** – using a duly formatted .csv file, managers can add many participants and documents to a project at once. Since adding participants individually can be very time consuming, this can be immensely helpful when the list of participants is long, or the manager must add the same students to more than one project;

## The ArgMine Platform

- **Add custom information about participants** – when adding participants via .csv files, the manager may add as many columns with information as they wish. Then, these columns will be displayed in the participants' table.
- **Filter and sort tables** – the users can swiftly navigate the documents', participants' and managers' tables and sort them any way they would like. In combination with document collections and the custom information that the managers can attach to each participant, this becomes especially useful because the managers can use these columns in the tables to sort them;

These usability changes and the case study from the professors at Universidade Católica made the platform into its current version. In the next section we will describe the ArgMine Platform and its usage, and then focus on the ArgMine annotator tool.

### 3.2 Architecture

The implementation of the features listed above resulted in the high-level architecture in Figure 3.1.

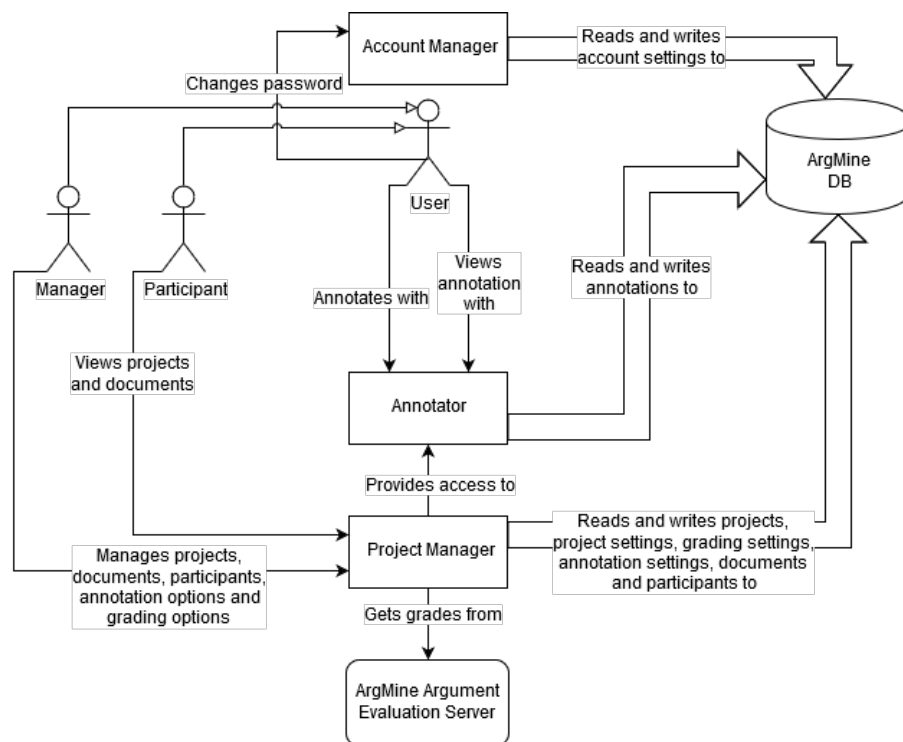


Figure 3.1: The ArgMine Platform high-level architecture

In Figure 3.1, the "Annotator" box represents the annotator tool detailed in Section 3.3.3, the "Project Manager" box represents all the pages accessible from the project list page introduced in Section 3.3, and the "Account Manager" box represents the page in Figure 3.5. In addition, the

## The ArgMine Platform

"ArgMine DB" box represents the ArgMine database that will be presented in this section and the "ArgMine Argument Evaluation Server" represent the server that stores the code for all the metrics detailed in Chapter 4 and Chapter 5.

As we can see in Figure 3.1, the two types of user (manager and participant) can perform the same actions when it comes to annotating documents and managing their account. However, they have different sets of actions available to them when it comes to managing projects. While both kinds of users can view annotations and annotate documents, as well as change their passwords, the participants are limited to viewing projects and documents. On the other hand, active managers have full authority to change anything about the project, as will be described later on in this chapter.

Not shown in Figure 3.1 is the difference between active managers and read-only managers, and the possibility that participant users can add documents to the project, since those are two specific cases that would clutter a more general diagram.

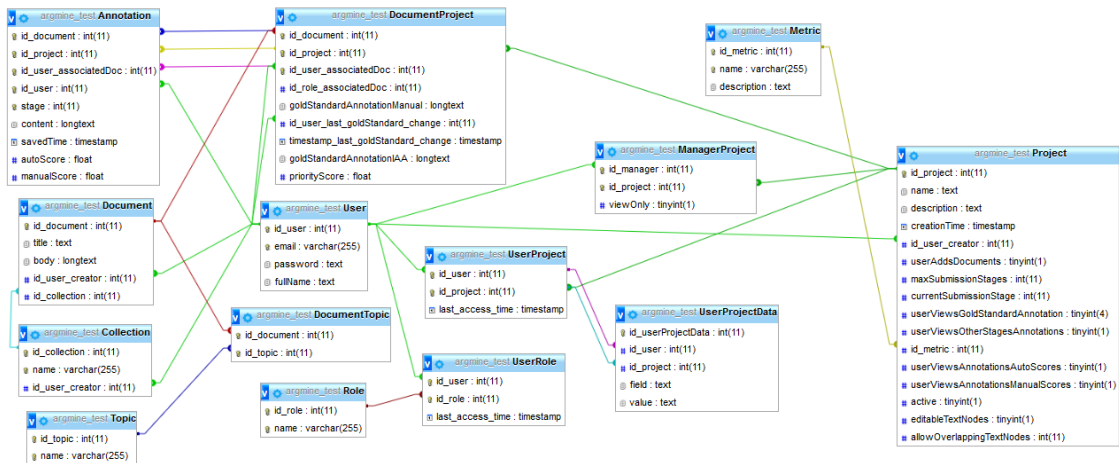


Figure 3.2: The ArgMine database

The ArgMine database, diagrammed in Figure 3.2, stores all the information related to the ArgMine Platform and it is managed using the MariaDB management system. It should be noted that the connections between tables in Figure 3.2, used to denote foreign keys, are colored purely for the purpose of better telling each connection apart when reading the diagram. Also, the symbols before the field names indicate the following:

- Key – the field belongs to the primary key of the table;
- Hashtag – the field is a number (such as "int" or "float");
- Piece of paper - the field is a text field;
- Calendar – the field is a date (such as "timestamp").

The database has the following tables:

## The ArgMine Platform

- **Project** – stores all the projects' information and their settings. The user permission settings are stored in columns "userAddsDocuments", "userViewsGoldStandardAnnotation", "userViewsOtherStagesAnnotations", "userViewsAnnotationsAutoScores" and "userViewsAnnotationsManualScores", while the annotation settings are in columns "editableTextNodes" and "allowOverlappingTextNodes" in table Project;
- **Metric** – stores information about each metric available to be used as an evaluation metric in the projects;
- **User** – used for storing the users' data, namely the login data;
- **UserProject** – a link between tables User and Project. If a user is linked to a project in this table, it means that that user is a participant in the project. The column "last\_access\_time" is updated every time a participant opens a certain project.. ;
- **ManagerProject** – a link between tables User and Project. If a user is linked to a project in this table, it means that that user is a manager in the project;
- **UserProjectData** – stores the extra information that the managers can add about the participants, as mentioned in Section 3.1;
- **UserRole** – stores the roles that a certain user may have in the platform. A user may be a manager, a participant, or both (in different projects). The column "last\_access\_time" is updated every time a user logs in;
- **Role** – stores the names of the roles, in case we might want to change them;
- **Document** – stores all the information about each document;
- **Collection** – stores the information about each collection of documents;
- **DocumentProject** – the link between tables Document and Project. If a document is linked to a project in this table, it has been added to said project. Columns "id\_user\_last\_goldStandard\_change" and "timestamp\_last\_goldStandard\_change" save the author and the timestamp of the last change to the gold standard annotation;
- **DocumentTopic** – the link between tables Document and Topic, to connect each document to its topic. This table is not in use yet, so no documents have topics yet;
- **Topic** – stores the topics of the documents, although this feature is not yet functional;
- **Annotation** – stores information about all the annotations saved in the platform's annotation tool.

With these changes, the database is able to answer to the needs of any actions that the users might perform using the Annotator or Project Manager in Figure 3.1. As to the evaluation server, it is described in more detail in Figure 3.3.

## The ArgMine Platform

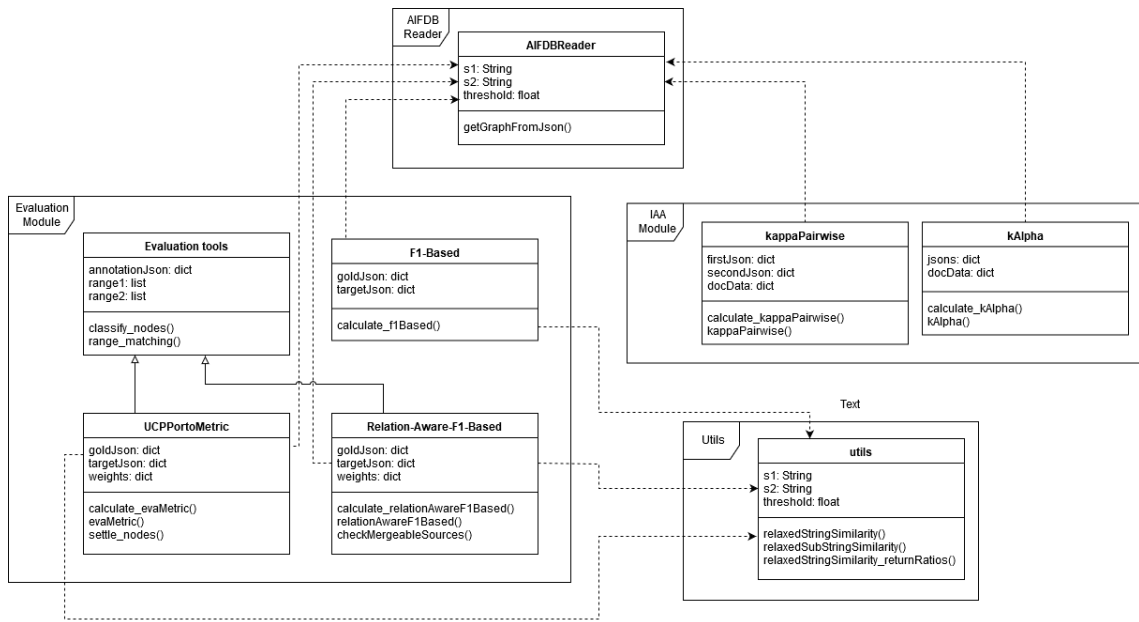


Figure 3.3: The ArgMine Argument Evaluation server architecture

In Figure 3.3, we can see that the evaluation of annotations can be divided into four modules: the AIFDB Reader, the Evaluation Module, the IAA module and the Utils Module. For the purposes of the ArgMine Platform itself, there is no use for the IAA module. But it is used when calculating the IAA of a corpus, and we will go into more detail about that in Chapter 5.

In the domain of the platform, we use all the other three modules, starting in the Evaluation module. What is used in this module depends, of course, on what metric the user chooses to use on their project. The UCPortoMetric and the f1Extra metric need the same three json files as input: a json file with the gold standard annotation, one with the annotation to be evaluated, and another one with the customizable evaluation settings that will be described in more detail in chapter 4. Both of these classes inherit methods from the Evaluation Tools class: *classify\_nodes()*, used to identify the role of each text node in the annotation, as well as each relation node's sources and destination, and *range\_matching()*, used to find out how much two ranges match.

In turn, the evaluation module makes use of the AIFDB Reader, an already existing class, to read the annotation json files, since their structure follows the Argument Interchange Format (AIF). This happens right at the beginning of the evaluation process, and then the metric's classes will make use of one of the methods in the Utils module. In this module, the "f1" class does not make use of the *classify\_nodes()* or *range\_matching()* methods, which is why it does not inherit them from the "Evaluation tools" class.

In the Utils module, what the *relaxedStringSimilarity()* method does is compare two strings, checking character by character to get the longest sub-string that is common to both the input strings. Then, as shown in Algorithm 3.1, it calculates the match percentage of the biggest sub-string with the two input strings and returns "True" or "False" depending on whether those match percentages are higher than a certain threshold.

```

1 if (longestSubstringLength / inputString1_length) >= threshold and (
    longestSubstringLength / inputString1_length) >= threshold :
2     return True
3 else:
4     return False

```

Listing 3.1: A portion of the *relaxedStringSimilarity()* method, in Python

The only difference between the *relaxedStringSimilarity()* and *relaxedStringSimilarity\_returnRatios()* methods is that the latter uses the code in Algorithm 3.2 instead of the one in Algorithm 3.1, thus returning the percentage of each string that was matched in the other string.

```

1 s1Ratio = longestSubstringLength / inputString1_length
2 s2Ratio = longestSubstringLength / inputString2_length
3
4 return {
5     "s1": s1Ratio,
6     "s2": s2Ratio
7 }

```

Listing 3.2: A portion of the *relaxedStringSimilarity\_returnRatios()* method, in Python

Which Utils method they use depends on the metric. The F1 metric uses the *relaxedStringSimilarity()*, and the UPC Porto and Relation-Aware-F1-Based metrics use the *relaxedStringSimilarity\_returnRatios()*.

Finally, the IAA module needs only the AIFDB Reader to read the json files in AIF. It has two classes: the "kappaPairwise" class and the "kAlpha" class, which are used to calculate the IAA metrics called "Cohen's kappa" and "Krippendorff's Alpha" respectively.

### 3.3 Interaction with the Platform

All the components of the architecture described in the previous section come together when the platform is used. We will go into more detail on this usage in this section.

To use the platform, users must first log in, which requires the users to be registered in the database. This process of registration is different for managers and for participants. While managers must be inserted directly in the database, participants are created indirectly when they are inserted into a project by a manager, which will also be described later on.

Logging in as a participant and as a manager are similar processes. In the form depicted in Figure 3.4, both types of users have to input their email and password (which is unique to the email, regardless of the type of user), then select their type of user to log in. An email may be registered as a manager and as a participant, but the user cannot access both these accounts at the same time. So, if this user logs in as a participant, they will only have access to the project they

## The ArgMine Platform

are participants in, with participant privileges. On the other hand, if the same user logs in as a manager, they will have access to manager privileges over the projects where they are managers.



Figure 3.4: The ArgMine Platform log in page with some credentials filled in

Common to both types of user is the account settings page:

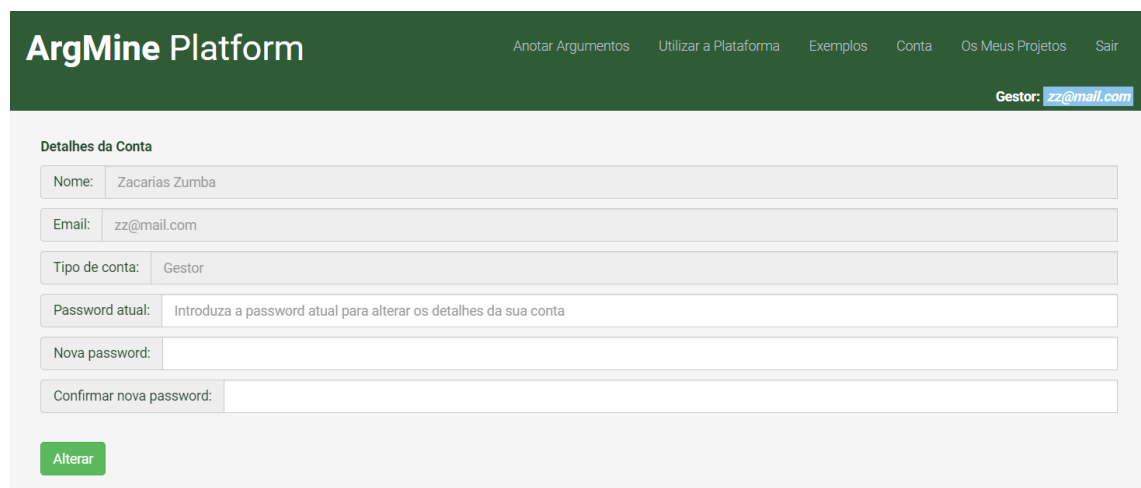


Figure 3.5: The settings page

Every user can go to this page using the header link “Account”. However, the only action they can take is to change their password, as all other changes must be done directly in the database.

### 3.3.1 The Participant User

After logging in as a participant, the user sees the list of projects to which they have been added, whether they are active or inactive. In either case, after clicking the project’s link, the students can view the list of documents, along with a set of information about the documents as seen in Figure 3.6.

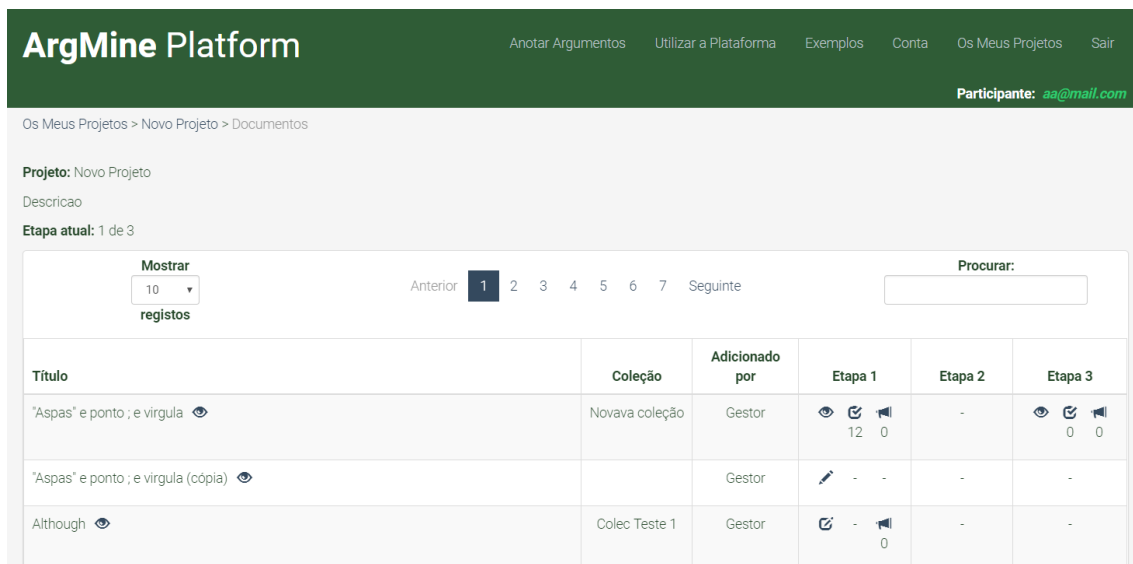


Figure 3.6: The participant documents list

The information that the participant can see, in each column, is about the collection which the document belongs to, about the user who added the document to the project, and then information about each stage of the current open project. The user who added the document can be either “Manager”, independently of which manager added the document to the project, or it can be the name of the current user, as each user can only see the documents which they added themselves. In the subsequent columns, one for each stage in the project, there is a link to open the annotator and, if the project manager allows, the automatic and manual score attributed to the stage’s annotation. All these elements can be seen in Figure 3.6.

Naturally, once the annotator’s link is clicked, the annotator window will open. Whether the annotation is editable or not depends on whether the stage is the current stage of the project, if the project is currently active and if that stage has been manually graded already. In the case that the participants are allowed to add their own documents, there is a link at the bottom of the document list. After clicking this link, the participant is led to the page in Figure 3.7.

In this page, the participant can insert a name and a body of text to the new document. However, this is not the only way to add documents to the project. The two alternate ways to do this can be accessed by clicking the other two tabs visible in Figure 3.7.

In the “Documents from Collection” tab, the participant can choose to add a whole collection to the project, or pick which documents to add from a collection. The key difference is that this way the participant does not have to create the document, but rather reuse a previously created one



## The ArgMine Platform

The screenshot shows the ArgMine Platform interface. At the top, there is a dark green header with the logo "ArgMine Platform" on the left and navigation links: "Anotar Argumentos", "Utilizar a Plataforma", "Exemplos", "Conta", "Os Meus Projetos", and "Sair" on the right. Below the header, the user's email "Participante: aa@mail.com" is displayed. The main content area has a breadcrumb trail: "Os Meus Projetos > Novo > Adicionar Documento". Underneath, it says "Projeto: Novo". There are three tabs: "[ Novo Documento ]" (which is active), "[ Documentos de Coleção ]", and "[ Documentos de Ficheiro ]". Below the tabs, there is a form with a "Título" label and a text input field containing "Título". Below that is a "Conteúdo:" label and a large text area containing "Conteúdo". At the bottom of the form is a green button labeled "Adicionar".

Figure 3.7: Participants can add documents to a project. Notice that there is no place for them to add this document to a collection

that they chose to add to one of their collections. If a document was not added to any collection upon creation, then it will not be selectable using this method, and the ones which were added to a collection, even though they may already be in the project, will be added without a gold standard annotation or participant annotations.

The screenshot shows the ArgMine Platform interface. At the top, there is a dark green header with the logo "ArgMine Platform" on the left and navigation links: "Anotar Argumentos", "Utilizar a Plataforma", "Exemplos", "Conta", "Os Meus Projetos", and "Sair" on the right. Below the header, the user's email "Participante: aa@mail.com" is displayed. The main content area has a breadcrumb trail: "Os Meus Projetos > Novo Projeto > Adicionar Documento de Coleção". Underneath, it says "Projeto: Novo Projeto". There are three tabs: "[ Novo Documento ]", "[ Documentos de Coleção ]" (which is active), and "[ Documentos de Ficheiro ]". Below the tabs, there is a "Coleção:" label and a dropdown menu showing "Coleção 3". Below that is a checkbox labeled "Escolher documentos" which is currently unchecked. At the bottom of the form is a green button labeled "Adicionar".

Figure 3.8: Participants can add documents from collections

As depicted in Figure 3.8, the participant can choose to select which collection documents to add to the project. If they do not select any, the whole collection will be added.

## The ArgMine Platform

The third way to add a document is in the last tab, “Documents from file”. Clicking this tab will show the page in Figure 3.9 to the participant.

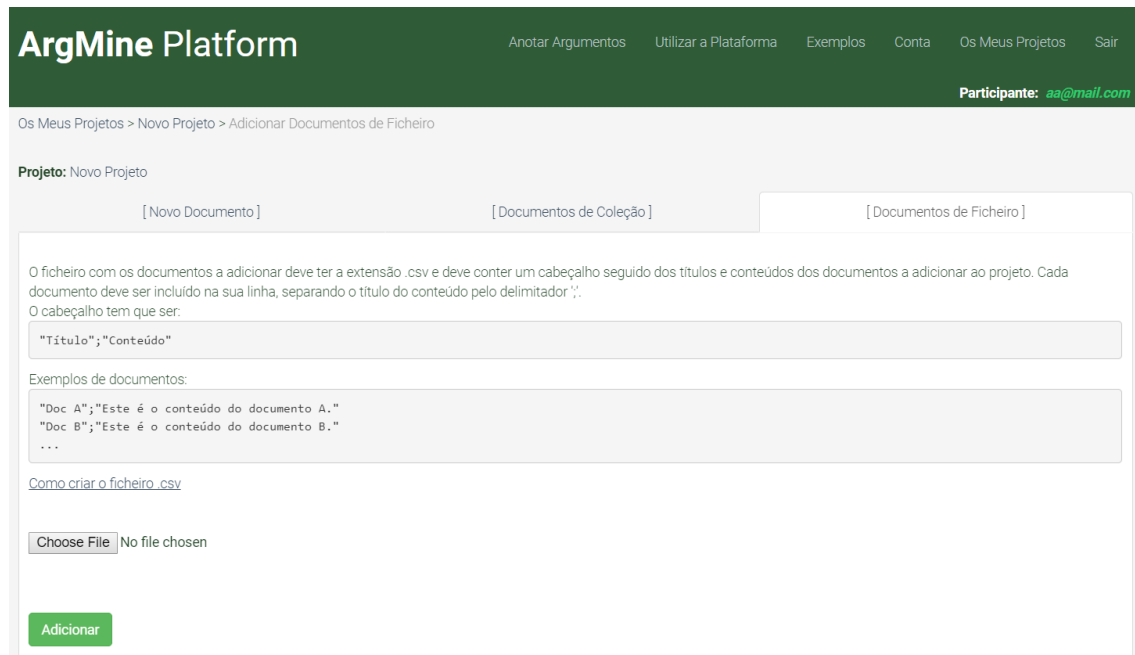


Figure 3.9: Participants can add documents in bulk to the project, using a properly formatted .csv file

As we can see from Figure 3.9, there are detailed instructions as to how to create the .csv file that should be uploaded to create the documents. The purpose of this way of creating files is to save the users the trouble of creating each one individually, thus improving their user experience on the platform. In addition, it makes it much easier to reuse documents. Like when adding from collections, the documents created from .csv files will be clean of any annotations, either the gold standard or the participants’ annotations.

### 3.3.2 The Manager User

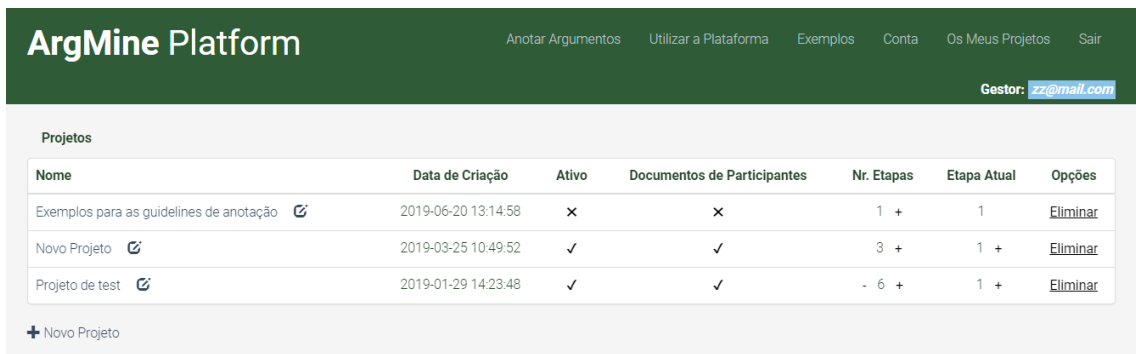
#### 3.3.2.1 The Projects List

After logging in as a manager, the list of associated projects appears as follows:

As is depicted in Figure 3.10, all the projects where the manager is associated are listed, regardless of whether the manager is an active manager of that project or whether the manager only has viewing permissions in it. The active managers can alter the project status, allow or disallow participants to add their own documents to the project, alter the stage, increase or decrease the maximum number of stages, as well as delete the project. If the manager chooses to delete the project, all the annotations created in the context of it will also be deleted. In addition, documents with no annotations in other projects will be deleted alongside the project.

Every project’s setting page is also accessible through the project list:

## The ArgMine Platform

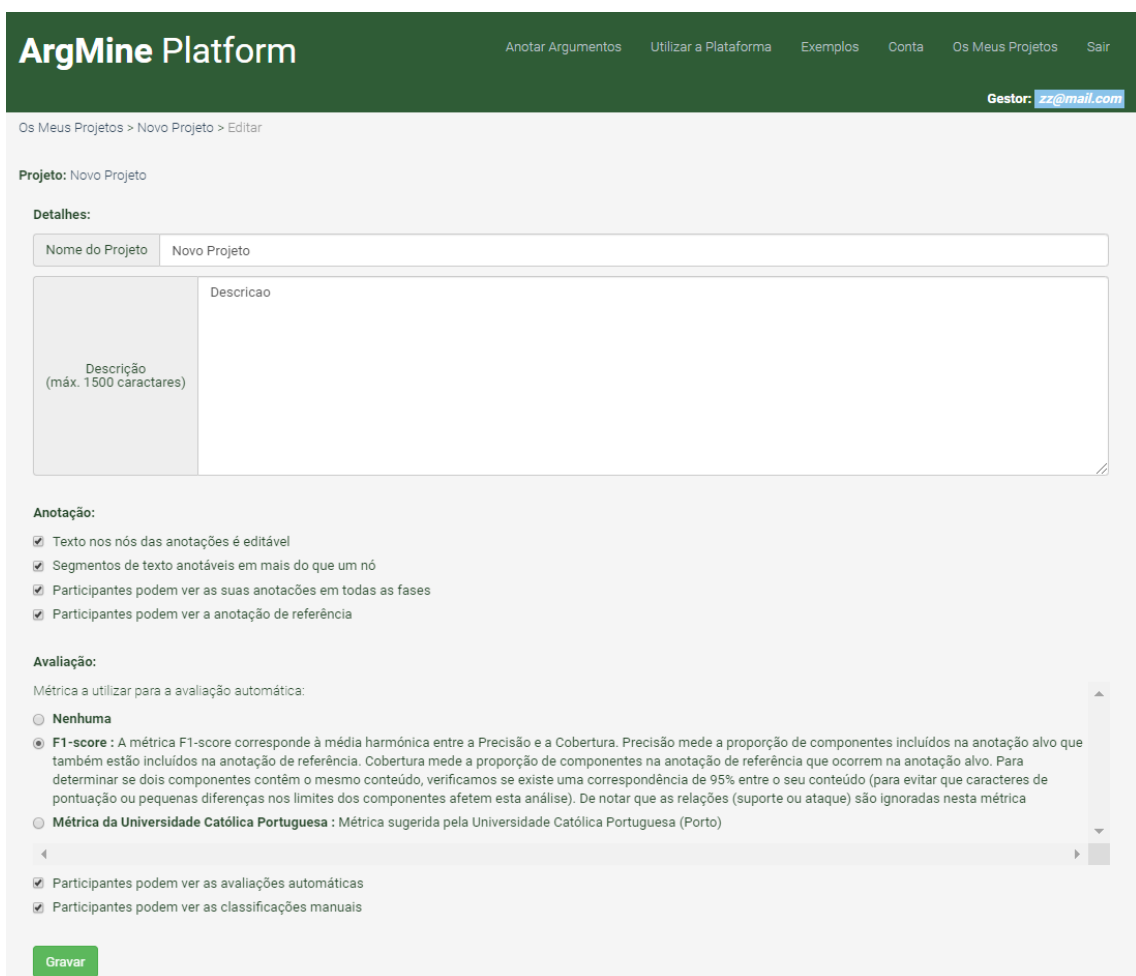


The screenshot shows the ArgMine Platform interface. At the top, there is a navigation bar with the platform name and several menu items: 'Anotar Argumentos', 'Utilizar a Plataforma', 'Exemplos', 'Conta', 'Os Meus Projetos', and 'Sair'. The user's name 'Gestor: zz@mail.com' is displayed on the right. Below the navigation bar, there is a section titled 'Projetos' containing a table with the following data:

Nome	Data de Criação	Ativo	Documentos de Participantes	Nr. Etapas	Etapa Atual	Opções
Exemplos para as guidelines de anotação	2019-06-20 13:14:58	✘	✘	1 +	1	<a href="#">Eliminar</a>
Novo Projeto	2019-03-25 10:49:52	✔	✔	3 +	1 +	<a href="#">Eliminar</a>
Projeto de test	2019-01-29 14:23:48	✔	✔	- 6 +	1 +	<a href="#">Eliminar</a>

Below the table, there is a '+ Novo Projeto' button.

Figure 3.10: The manager project list



The screenshot shows the 'Os Meus Projetos > Novo Projeto > Editar' page. The project name is 'Novo Projeto'. The 'Detalhes' section includes a text input for the project name and a large text area for the description, with a note that the description is limited to 1500 characters. The 'Anotação' section has four checked options: 'Texto nos nós das anotações é editável', 'Segmentos de texto anotáveis em mais do que um nó', 'Participantes podem ver as suas anotações em todas as fases', and 'Participantes podem ver a anotação de referência'. The 'Avaliação' section has a dropdown menu for the automatic evaluation metric, currently set to 'F1-score'. Below the dropdown, there is a scrollable area containing the description of the 'F1-score' metric and the 'Métrica da Universidade Católica Portuguesa' metric. At the bottom, there are two checked options: 'Participantes podem ver as avaliações automáticas' and 'Participantes podem ver as classificações manuais'. A green 'Gravar' button is located at the bottom left.

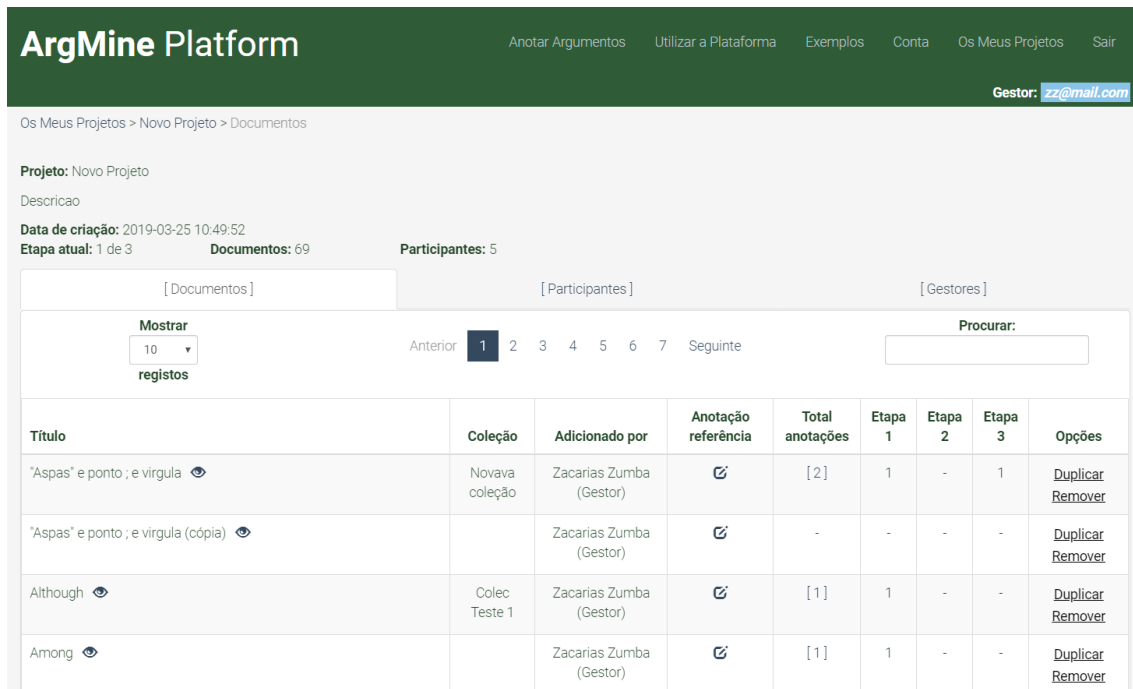
Figure 3.11: A project's settings page, accessible only to its active managers

In the page in Figure 3.11, accessible only to the active managers, it is possible to change the project name, change the project description, allow participants to view their automatic or manual grades, allow them to see all stages' grades. As of annotator settings, the manager can allow the participants to annotate the same portion of text more than once and allow them to create nodes without selecting any portion of text. Finally, the manager can choose what automatic grading

## The ArgMine Platform

metric to display for all the users to see. Currently, these metrics are the F1 metric and the UCP Porto metric. Once the manager changes the metric, every annotation's grade will be recalculated, which may take some time depending on the metric, and on the number and complexity of the annotations.

Upon clicking the project name in the list in Figure 3.10, the manager will be directed to the respective project's document list. As is the case with the project list, the manager has more options and information accessible than the participant in this list:



The screenshot shows the ArgMine Platform interface. The header includes the platform name and navigation links: "Anotar Argumentos", "Utilizar a Plataforma", "Exemplos", "Conta", "Os Meus Projetos", and "Sair". The user is logged in as "Gestor: zz@mail.com". The breadcrumb trail is "Os Meus Projetos > Novo Projeto > Documentos". The project details are: "Projeto: Novo Projeto", "Descrição", "Data de criação: 2019-03-25 10:49:52", "Etapa atual: 1 de 3", "Documentos: 69", and "Participantes: 5". The document list is displayed in a table with the following columns: "Título", "Coleção", "Adicionado por", "Anotação referência", "Total anotações", "Etapa 1", "Etapa 2", "Etapa 3", and "Opções". The table contains four rows of documents.

Título	Coleção	Adicionado por	Anotação referência	Total anotações	Etapa 1	Etapa 2	Etapa 3	Opções
"Aspas" e ponto ; e vírgula	Novava coleção	Zacarias Zumba (Gestor)		[ 2 ]	1	-	1	<a href="#">Duplicar</a> <a href="#">Remover</a>
"Aspas" e ponto ; e vírgula (cópia)		Zacarias Zumba (Gestor)		-	-	-	-	<a href="#">Duplicar</a> <a href="#">Remover</a>
Although	Colec Teste 1	Zacarias Zumba (Gestor)		[ 1 ]	1	-	-	<a href="#">Duplicar</a> <a href="#">Remover</a>
Among		Zacarias Zumba (Gestor)		[ 1 ]	1	-	-	<a href="#">Duplicar</a> <a href="#">Remover</a>

Figure 3.12: A project's document list, as seen by an active manager

Firstly, aside from the document list, the manager has access to two more lists: the participant list and the manager list.

### 3.3.2.2 The Documents Tab

On the document list, the manager can also see which collection the document belongs to and the user who added it. In the case of managers, they can see every document which has been added to the project and the exact name of the user who added it, be it themselves, another manager or a project participant.

The next column on the document list shows a link to open the annotator, and managers can use it to save the document's gold standard annotation. Since there can only be one gold standard annotation per document in the project, if a manager saves an ongoing annotation, this save will always overwrite the previous annotation. To add a bit more transparency and allow the managers to better keep track of this process, hovering the link to the annotator will reveal the name of the last manager who saved the gold standard and the time when it happened. While this can help

with knowing who last altered the gold standard annotation, there is no extended log beyond the last save.

Then, the managers can see a column with the total number of annotations of that document, followed by a stage-by-stage breakdown of that number. The total number of annotations is a link that will be described later, and the breakdown numbers are just text meant to give a better idea of the distribution of the annotations.

These columns are followed by the options column. It allows managers to delete or duplicate a document. Deleting a document will delete all annotations associated with it. This action will also delete the document from any collection it is in, if there are no more remaining annotations associated with the document across all the other projects. On the other hand, duplicating a document will create another document similar to the original one. Upon duplication, it is possible to change the title of the document but its body of text is only editable if there is no gold standard annotation associated with that document in the project. This is because the annotation's integrity is at stake when the original text is altered, as the annotation includes information on the portion of text that was selected to create the nodes. Furthermore, upon duplicating a document, the gold standard annotation is also duplicated along with the information about the last manager who altered it and when they did so. However, none of the participant annotations is carried over to the new document, and it cannot be added to any collection.

At the end of the document list, all active managers will see a link to add a document. This link will lead to a page much similar to the page that participants see to create a document, but with a few other options, such as being able to add documents to and from collections. Managers also have two alternate ways of adding documents aside from inputting the title and body manually, the same ways that are available to the participants and where described in Section 3.3.1. Because these methods were already described, we will not go into detail again and we will simply show the page in Figure 3.13. As we can see, there is an extra option of adding the new document to a collection. If the manager chooses to do so, they will be able to select one of their previously created collections or create a new one.

After adding documents to the project, the manager will be redirected to the project's document list once more.

### 3.3.2.3 The Participants Tab

From the documents tab, they can also access the Participants tab:

# The ArgMine Platform

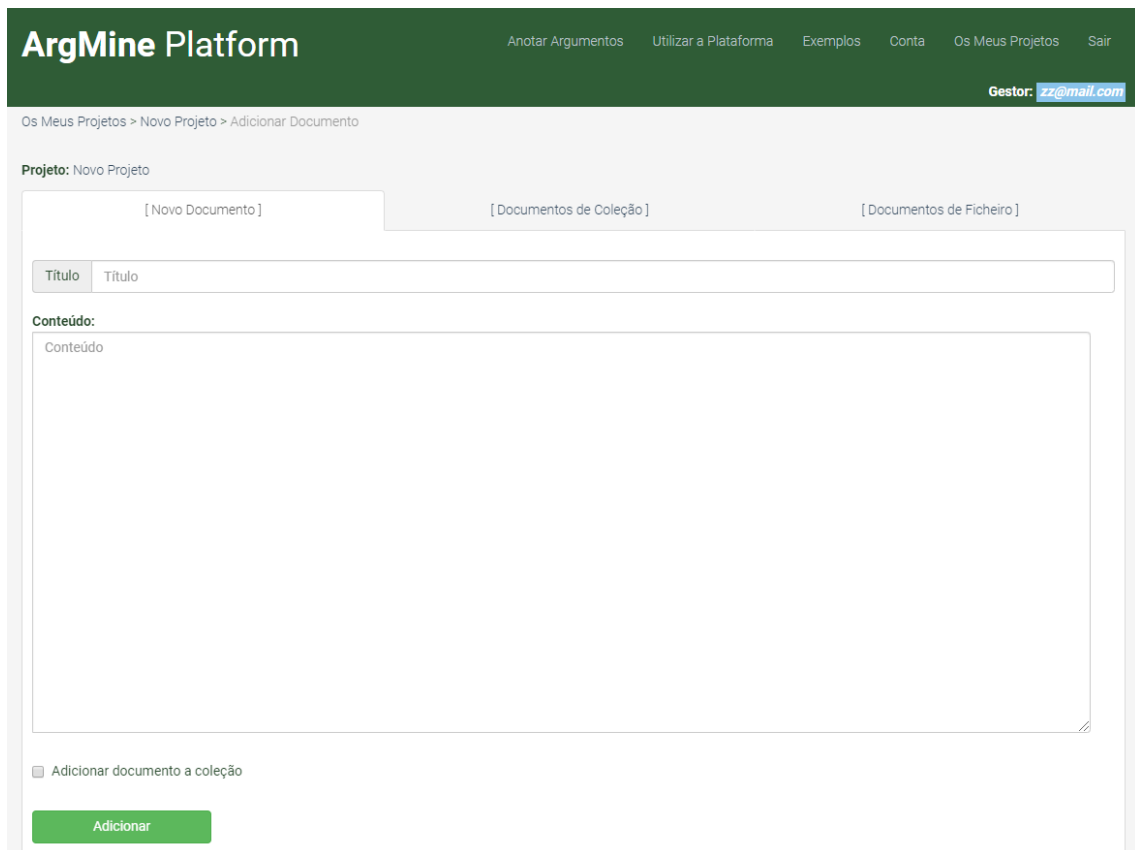


Figure 3.13: Managers can add documents to the project

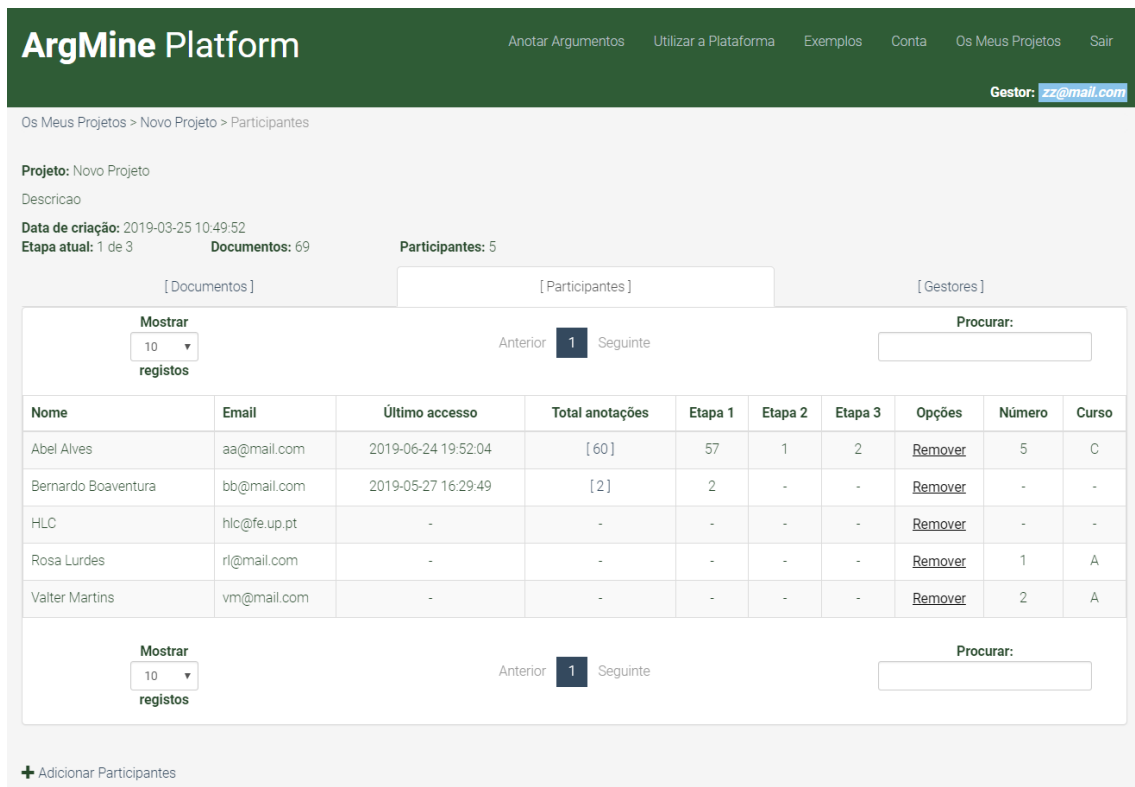
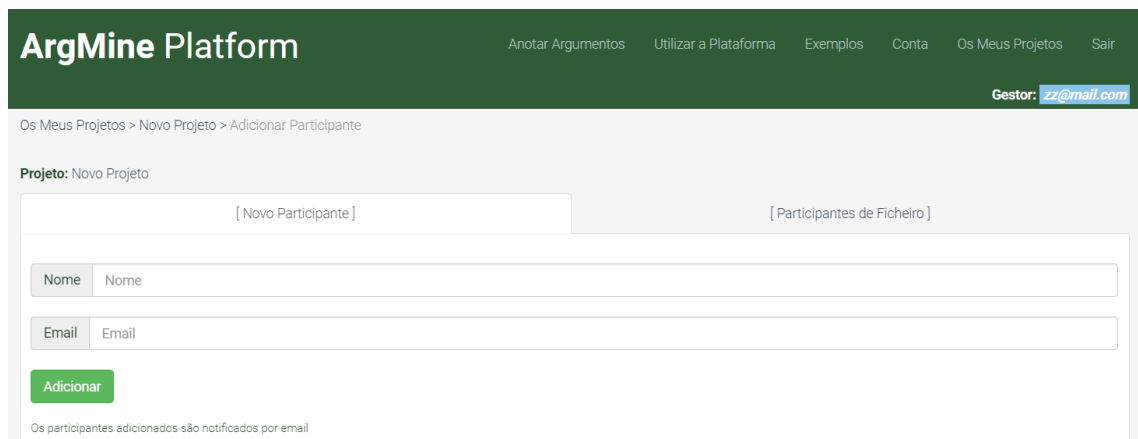


Figure 3.14: A project's participant tab, as seen by an active manager

## The ArgMine Platform

In the participants tab, the managers can see which participants have been associated to the project. This includes not only their name and email address, but also the number of annotations which they have saved across the project, the breakdown of this number across the several project stages, some further information that the managers may have chosen to add about them. In the case of active managers, there will be an Options column with a link to remove the participant from the project. Removing a participant from the project will result in the deletion of all of that participant's annotations in that project. Since the participants cannot add annotations to other participant's documents, removing a participant will also lead to the deletion of all the documents that that participant added to the project.

Also in the case of active managers, there will be a link to add participants to the project, leading to the page in Figure 3.15.



The screenshot shows the ArgMine Platform interface for adding a participant. The header is dark green with the 'ArgMine Platform' logo on the left and navigation links ('Anotar Argumentos', 'Utilizar a Plataforma', 'Exemplos', 'Conta', 'Os Meus Projetos', 'Sair') on the right. A user profile 'Gestor: zz@mail.com' is visible in the top right. The main content area has a breadcrumb trail 'Os Meus Projetos > Novo Projeto > Adicionar Participante'. Below this, the project name is 'Projeto: Novo Projeto'. There are two tabs: '[ Novo Participante ]' (active) and '[ Participantes de Ficheiro ]'. The active tab contains a form with two input fields: 'Nome' (with placeholder 'Nome') and 'Email' (with placeholder 'Email'). A green 'Adicionar' button is positioned below the fields. A small note at the bottom of the form states 'Os participantes adicionados são notificados por email'.

Figure 3.15: Managers can add participants to the project

So, there are two methods for adding participants to a project, akin to the methods for adding documents. A manager can do this individually by inserting the name and email address of the participant that they wish to add, in the page depicted in Figure 3.15. Alternately, the manager may choose to mass-add the participants using a .csv file. This method is accessible in the tab "Add Participants from File", as seen in Figure 3.16.

Adding participants from file is not only the way to add more than one participant at a time, but also the way to attach some extra information about each participant. This is an option that makes it much easier for managers to browse and filter the great number of participants that may be associated to a project. To make use of this, they simply have to add all the extra columns they may deem necessary in the properly formatted participant .csv file. For that, they have access to detailed instructions in the page in Figure 3.16 and by clicking on the link "How to create the .csv file" on that same page.

After uploading a file, the list of participants in that file is displayed along with all the information that will be added if the manager chooses to add the participants. This list also includes some feedback to help the manager with checking which participants they are adding into the project. As shown in Figure 3.17, there are three types of feedback:

# The ArgMine Platform

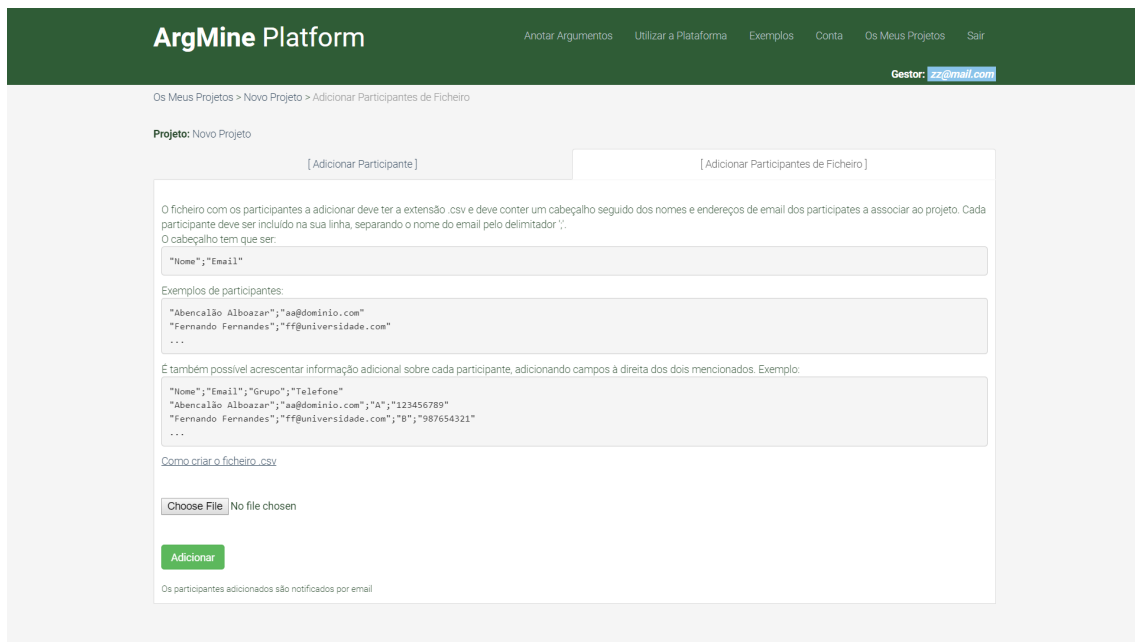


Figure 3.16: Managers can add participants in bulk to the project, using a properly formatted .csv file

- A tick if the participant will be created and added to the project;
- An "i" if the participant is already associated to the project;
- An exclamation mark in case the future participant's email address is already registered in the platform under a different name. In this last case, the participant will be added under the previously registered name and the new name will be assumed wrong and ignored.

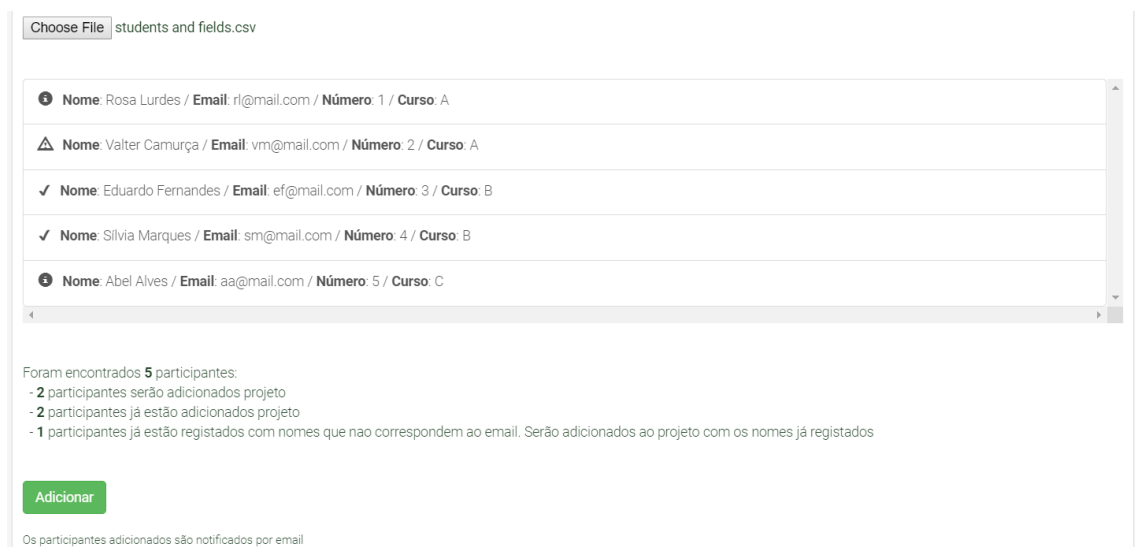


Figure 3.17: The list that appears in the page on figure 3.16, showing the the participants to be added. The feedback described in the previous paragraph is visible on the left-hand side



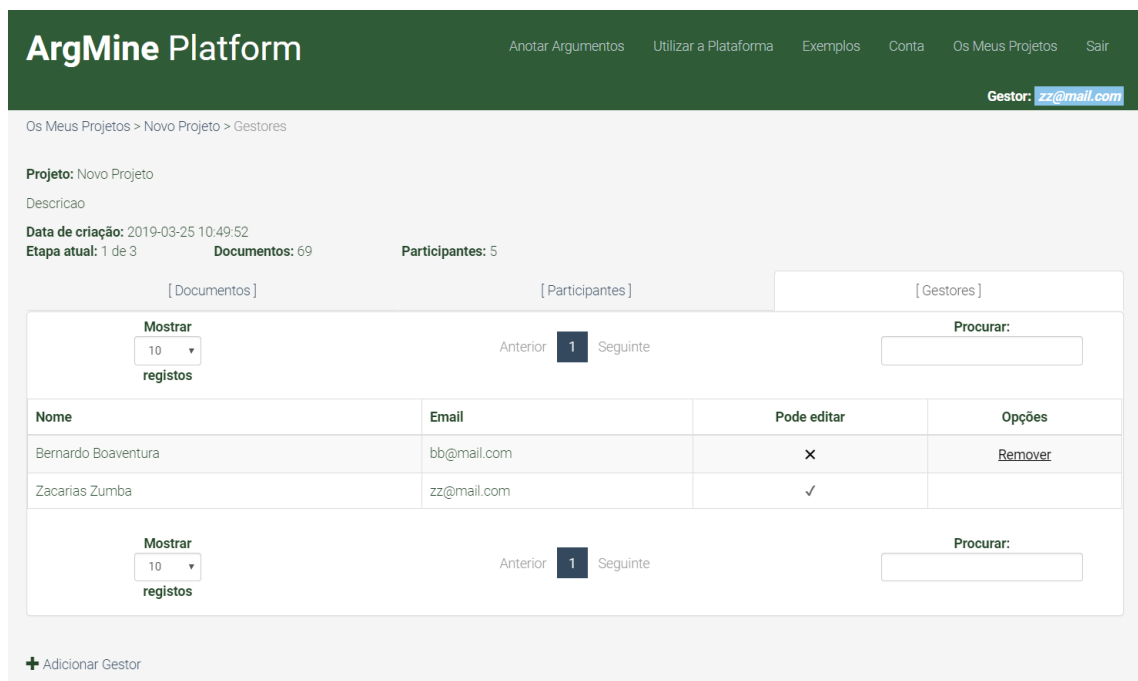
## The ArgMine Platform

Using any of these methods to add a participant will result in an email being sent out to all the email addresses of the users that are being added. This will serve as a registration process for participants in the platform. The email will inform the users that they have been registered in the ArgMine Platform, and provides them with a temporary password which they are urged to change as soon as they first log in. In the case that the email address has already been used for a participant account in the platform, the email will simply inform the user that they have been added to a new project.

Finally, after adding the participants, the manager will be redirected to the participant list which will then show the full and updated list of project participants. From this tab, the manager can also access the Managers tab.

### 3.3.2.4 The Managers Tab

All managers can access this tab and see which other managers have been associated to the project. There is little information on this table, as can be gauged from Figure 3.18.



The screenshot displays the 'Managers Tab' in the ArgMine Platform. The header shows the platform name and navigation links. The breadcrumb trail indicates the current location: 'Os Meus Projetos > Novo Projeto > Gestores'. The project details include 'Projeto: Novo Projeto', 'Data de criação: 2019-03-25 10:49:52', and 'Participantes: 5'. The main content area shows a table of managers with the following data:

Nome	Email	Pode editar	Opções
Bernardo Boaventura	bb@mail.com	×	Remover
Zacarias Zumba	zz@mail.com	✓	

The interface also includes pagination controls (Anterior 1 Seguinte) and a search bar (Procurar:).

Figure 3.18: The managers tab, as seen by an active manager

Only active managers can alter other manager's permissions and remove managers from the project, and there are little restrictions to this. Active managers may remove any manager from the project, including the project creator, and they may also make any manager active or inactive. The only exception to this is that active managers cannot remove themselves from the project, or remove their own permissions in the project. This guarantees that there is always at least one active manager per project, regardless of whether it is the original project creator or not.

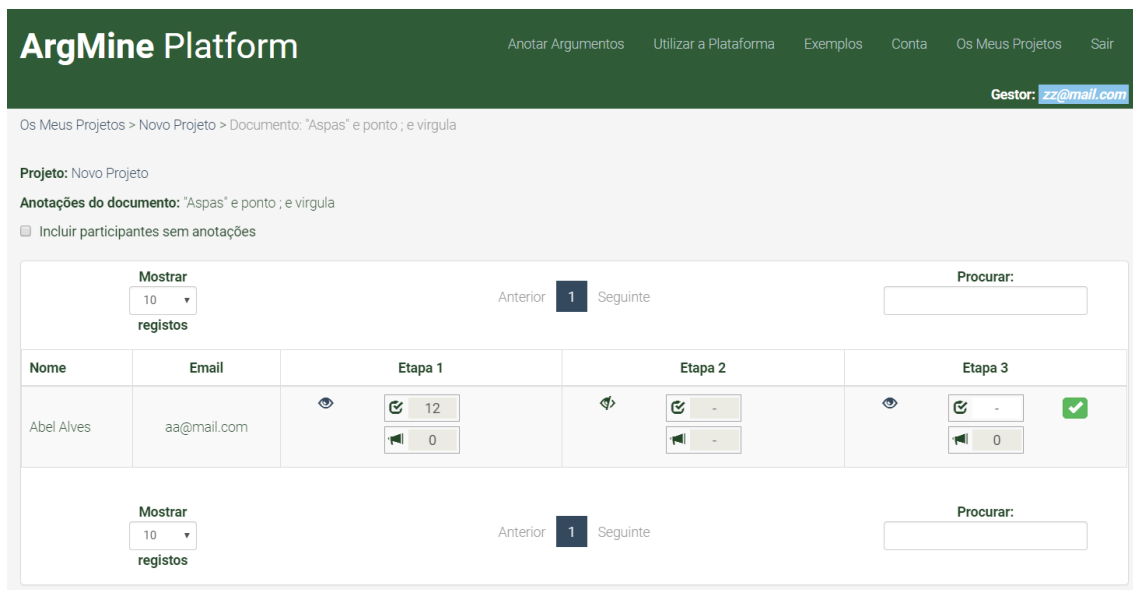
## The ArgMine Platform

To add project managers, any active manager can click the link at the bottom of the list. Then, the manager just has to insert the name and email address of the manager to be added to the project. As is the case with adding participants, any email which is not associated to a manager-type account will be used to create a new user of this type, and emails previously registered under different names will be added to the project using the already registered email. Creating a manager account will also trigger an email informing the user that they have been registered in the ArgMine Platform and provide them with a temporary password that they are asked to change as soon as they first log in. Or, if the user had already been registered, the email will inform them that they have been added as manager to another project.

### 3.3.2.5 Attributing Grades

Managers, as mentioned before, have access to links to see all the participants' annotations. The link with the total number of annotations of each document on the Documents list (Figure 3.12) and the link with the total number of annotations of each participant on the Participants list (Figure 3.14) are the two links which lead the managers to pages where they can see the annotations.

If the manager clicks one of these numbers in the Documents list (Figure 3.12), they will be redirected to the page in Figure 3.19.



The screenshot displays the ArgMine Platform interface for a document titled "Aspas" e ponto ; e virgula. The page shows a list of annotations for a participant named Abel Alves. The table below summarizes the data shown in the screenshot:

Nome	Email	Etapa 1	Etapa 2	Etapa 3
Abel Alves	aa@mail.com	12	-	-

Figure 3.19: Managers can see a list of annotations of a certain document, and grade some stages

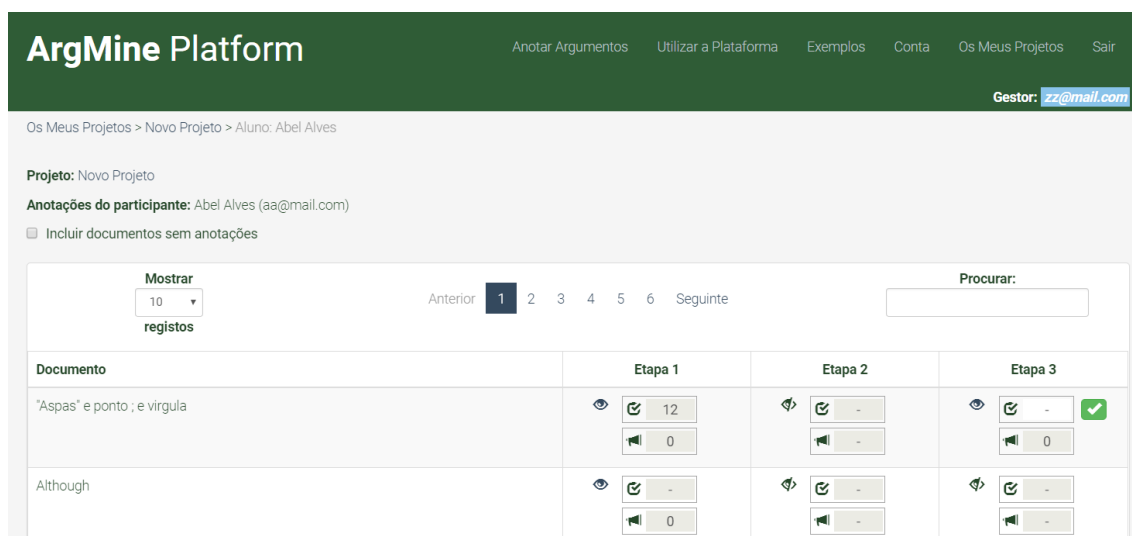
In this page, they can see a list with the participants who annotated the corresponding document. If the manager wishes to, they may extend this list to include the participants who did not annotate any document at all by checking the box "Include participants without annotations".

There is a column for the participant's name, another for their email and then a column for each stage of the project. In these columns there is a link to the annotator and two text boxes. The link to the annotator will naturally show the annotation that the participant saved for that document

## The ArgMine Platform

at that stage. As of the text boxes, the lower one is never editable and it shows the automatic grade of that participant's annotation to the corresponding document in that stage. But this will only happen if a grading metric was selected and if the participant annotated that stage. On the other hand, the upper text box may be editable or not. It is editable if the participant annotated the document in that stage and if that stage is not the current stage, because that is the box where the manager will input the participant's grade. If the participant can still edit that stage's annotation, the manager cannot attribute a grade to it.

However, if the manager clicks on the number in the Participants list (Figure 3.14), they will be redirected to a similar but slightly different page, as seen in Figure 3.20.



The screenshot shows the ArgMine Platform interface. At the top, there is a green header with the logo "ArgMine Platform" and navigation links: "Anotar Argumentos", "Utilizar a Plataforma", "Exemplos", "Conta", "Os Meus Projetos", and "Sair". The user's role is "Gestor: zz@mail.com". Below the header, the breadcrumb trail reads "Os Meus Projetos > Novo Projeto > Aluno: Abel Alves". The main content area shows "Projeto: Novo Projeto" and "Anotações do participante: Abel Alves (aa@mail.com)". There is a checkbox for "Incluir documentos sem anotações". A pagination control shows "Mostrar 10 registos" and "Anterior 1 2 3 4 5 6 Seguinte". A search box labeled "Procurar:" is on the right. The main table has columns for "Documento", "Etapa 1", "Etapa 2", and "Etapa 3".

Documento	Etapa 1	Etapa 2	Etapa 3
"Aspas" e ponto ; e virgula	<input type="text" value="12"/> <input type="text" value="0"/>	<input type="text" value="-"/> <input type="text" value="-"/>	<input type="text" value="-"/> <input type="text" value="0"/>
Although	<input type="text" value="-"/> <input type="text" value="0"/>	<input type="text" value="-"/> <input type="text" value="-"/>	<input type="text" value="-"/> <input type="text" value="-"/>

Figure 3.20: Managers can see a list of annotations that a certain participant has saved, and grade some stages

In this page, they will see a list of documents instead of a list of participants because this list shows a single participant's annotations in the whole project. The manager may again choose to see a full list of documents, annotated by the participant at hand or not, by checking the box "Include documents without annotations".

In this page there is a column for the document name, and then one column for each stage. Like with the page in Figure 3.19, there is a link to view the corresponding annotation in the annotator and then two text boxes. These text boxes function in exactly the same way as in the page in Figure 3.19, and they have the same restrictions.

The ArgMine Platform also has a user manual, describing how to use the platform, that can be found at <https://web.fe.up.pt/argmine/platform/tools.php>.

### 3.3.3 The ArgMine Annotator Tool

The ArgMine annotator tool is accessible through in ArgMine Platform for every document in an active project. In an inactive project, or for stages that are not the current stage that the project is in, the annotator tool may also be used to see annotators.

## The ArgMine Platform

So, the annotator also has modes:

- Active mode: where the user may use all the functionalities that the annotator provides;
- Read-only mode: where the user is able to see the annotation and change it in all the ways that are also permitted in the active mode. The key difference is that this annotation cannot be saved.

We will now go into detail about how the annotator in active mode.

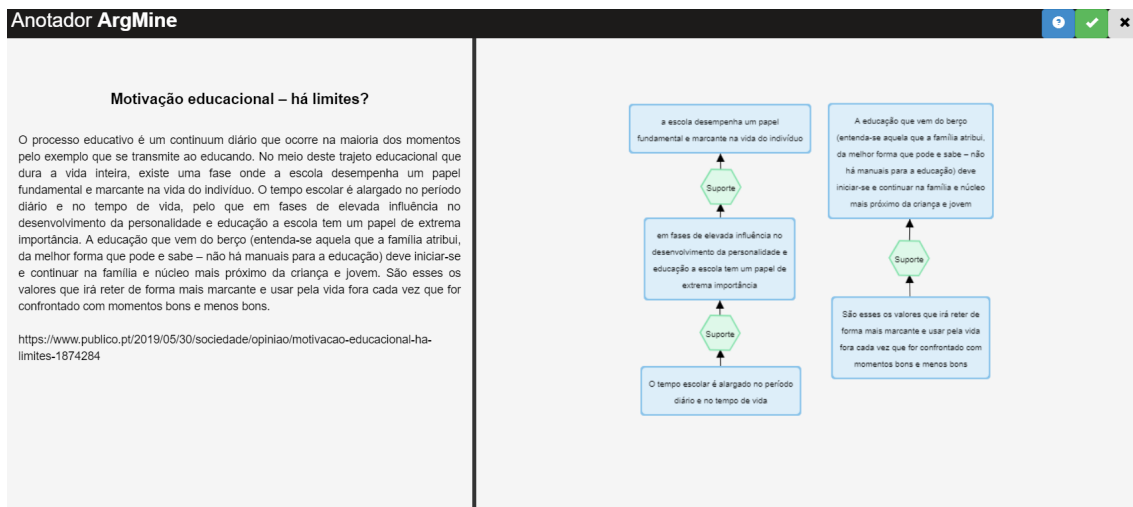


Figure 3.21: The ArgMine annotator with an annotation

Once the annotator is opened, there are two visible panels, as we can see in Figure 3.21: the one on the left-hand side is scrollable and shows the text at hand, and the one on the right-hand side is draggable and will show the annotation as the user creates it.

Saving on the annotator is very straightforward. If there have been changes on the annotation, the box with the “check” symbol will become orange. To save the changes, the user must simply click this button, which will become green if the save was successful or red otherwise. The save button is the same across all annotator windows. As is the exit button, marked by a “cross” symbol. Where appropriate, there is a third button with a “question mark” symbol, which will open the help text. If the user chooses to open the help text, the box in Figure 3.22 will pop up, explaining how to create, delete or edit all types of nodes.

There are two ways to create a text node. The way that the node was created is crucial when calculating the automatic grade of an annotation, so users should be aware of the kind of node they are creating.

The first and more traditional way is for the user to select the text that they want to include in the node and then clicking the annotation pane or dragging the text to it. The node will automatically appear on the right pane. This will result in a node with information attached to it about the portion of the text that was used to create it.

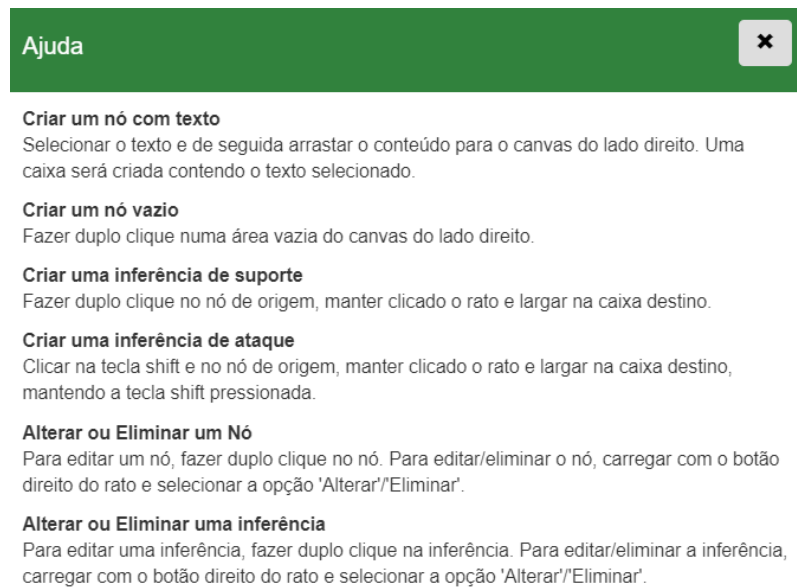


Figure 3.22: The annotator tool's "Help" pop-up

The second way is to double-click the right pane. A new node will appear with some default placeholder text between "lesser than" (<) and "greater than" (>) symbols. This kind of node will not have a portion of text attached to it, so it will be a "0-range" node, which is again a significant detail to the grading of the annotations. 0-range nodes exist to allow the user to create nodes with whatever text they want, but it should be noted that this kind of node will only be taken into account for grading purposes once it does contain the user's own text. Consequently, if the default placeholder text is never altered, the node will be ignored from the annotation, even if it is connected to another node.

Another functionality of the annotator tool is allowing the user to edit the text within any text node, whether it was created using a portion of text or by double-clicking the left pane. To do this, the user must simply double-click the node and alter the text as they wish.

Any node with altered text will be marked not only in the annotation data itself but also visually in the annotator. As we can see in Figure 3.23, there are two nodes with a striped border. One of those nodes was created by double-clicking the right pane and its text is still between the "<" and ">" symbols, but it is still considered a node with edited text. The other one has its text only slightly altered. Any node that is at some point considered to have been altered will always remain marked as altered, even if the original text is rewritten on it. If the user wishes to create an unedited node with that portion of the text, they must create it using the traditional way described above.

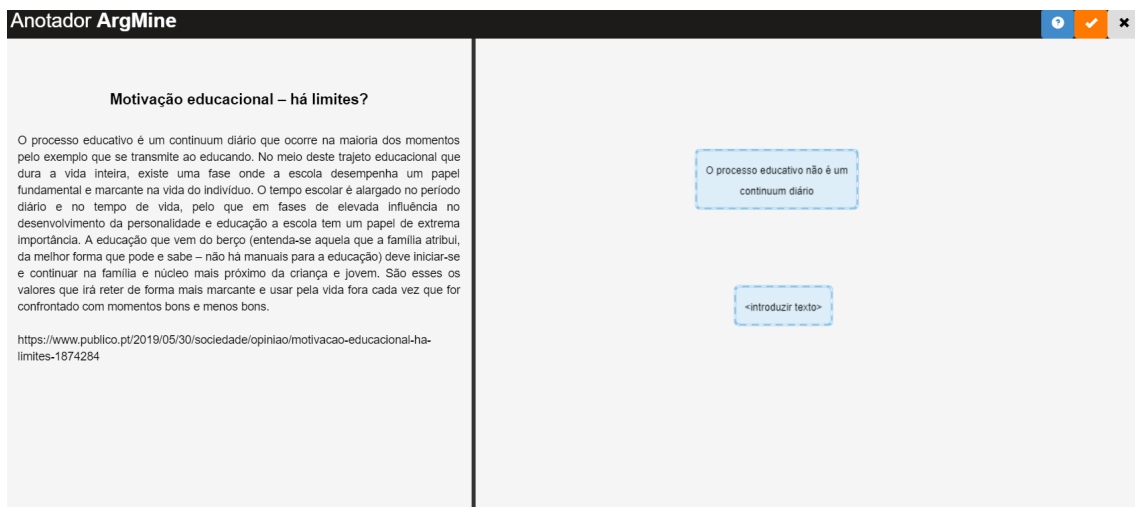


Figure 3.23: The ArgMine annotator with an annotation with the two types of edited nodes. These are marked by stripes in their borders

In the ArgMine annotator tool, as mentioned before in this chapter, it may be possible to create more than one node using the same portion of text. That means that the scenario in Figure 3.24 is possible:

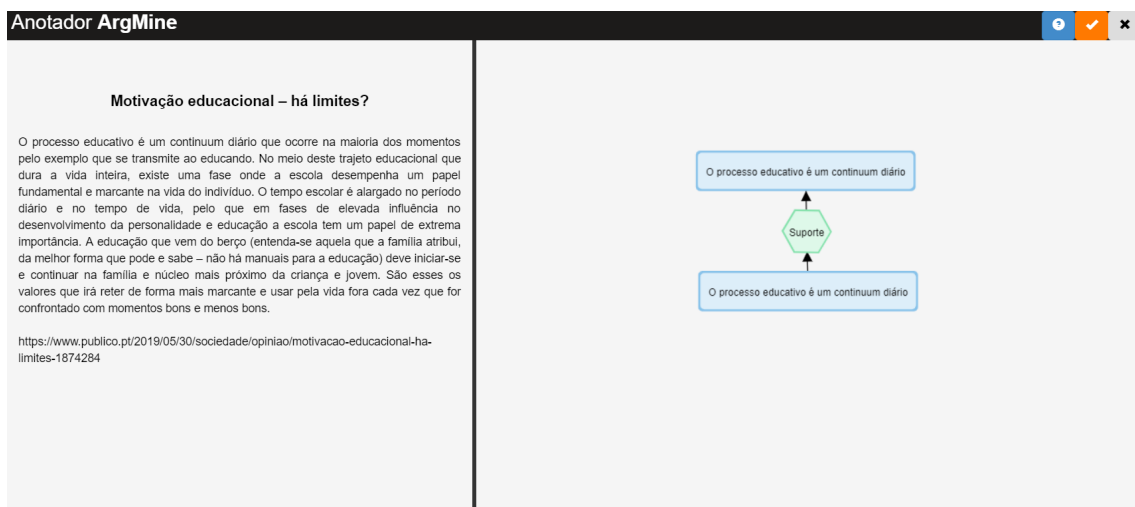


Figure 3.24: The ArgMine annotator with an annotation with two nodes with the same text

Unlike the case of nodes with edited text, there is no way to visually distinguish two nodes created by selecting the same portion of the text. Because the ArgMine annotator strives to be a good visual annotation tool, we must be aware of the details that are not important enough to warrant a visual distinction. This is one of those details, because these kinds of nodes (such as the ones in Figure 3.24 are completely interchangeable for the automatic evaluation metrics. So even though we want to make the annotator as visual as possible, we chose not too clutter the annotation graph with the visual distinction of these kinds of nodes.

## The ArgMine Platform

The user can also choose the text node's color as well. This may help when a user wants to annotate different arguments within a text, showcase the several levels of premises of an argument, highlight a node for some reason, etc. While the color set is small, we feel that it is already enough to bring a huge advantage to visualizing annotations in the ArgMine annotator. To change the color of the text nodes, the user simply has to double-click the node that they want to change, choose a color from the drop-down list and click the save button.

Besides the text nodes, there are two other types of nodes: support relation nodes and attack relation nodes. The user can effortlessly tell these nodes apart from the text nodes, as relation nodes have a hexagonal shape and the text nodes have a rectangular shape. In a large annotation, this becomes instrumental in interpreting the annotation graph with ease and speed.

To create a support relation node, the user can:

1. click the Shift key, then
2. double-click the source text node, then
3. drag the arrow that will appear on the screen to the intended destination text node, and finally
4. release the Shift key before releasing the mouse over the destination text node

And to create an attack relation node, the user can use the same process but let go of the Shift key while dragging the arrow to the destination text node. An alternative to this is double clicking the source text node and dragging the arrow to the destination text node, which will create an attack relation node.

The type of relation node can always be switched by double-clicking an existing relation node, altering its type on the drop-down box and saving.

When the user chooses to save the annotation, it will be stored in the ArgMine database in the AIF format, which, as introduced in Chapter 2, is a standard format for saving annotations meant to facilitate their exchange across the scientific community. As a result, the ArgMine annotator tool can interpret any corpus that comprises of annotations in the AIF format.

## The ArgMine Platform



## Chapter 4

# Automatic Evaluation of Annotations

This chapter will describe and analyze the two evaluation metrics that were created during this thesis project. Both of them compare and grade an annotation with a gold standard which is considered correct, and both of them have configurable parameters.

### 4.1 Problem and Motivation

Grading annotations in whatever context is a cumbersome task that requires a human evaluator to compare two annotations that may have nothing to do with each other, or that differ only in infinitesimal details and then grade one of them considering certain standards. Keeping consistency alone can be a problem, since the same exact grading scenario may appear more than once but if the evaluator does not remember it, it may be graded differently. Also, since comparing complex arguments can take a long time, a human is likely to get tired and make grading mistakes through no fault of their own. Automatically evaluating annotations can solve this problem. Moreover, in a classroom setting, automatically getting this can help students become more interested in annotating, thus making them more interested in argumentation and incentivizing them to engage with arguments more often. With this, we can foment a more critical youth generation. Finally, automatically evaluating annotations has utility from the point of view of Argumentation Mining in two ways:

- **Potentiate growth in number of annotations** — if we are able to provide a comprehensive grade of an annotation against another, it means that a new range of people will be interested in annotating arguments in platforms such as ArgMine. For example, with the proposed evaluation metric, teachers would be able to grade their students automatically with a grading system that can be adapted to fit their teaching methods and pace, as will be explained later. So, it is easy to see how providing automatic ways to evaluate annotations can bring more users to annotation platforms and increase the number of annotated documents available for Argumentation Mining;

- **Use as an agreement metric** — a grading metric is, in its core, evaluating the agreement between one annotation and another annotation considered correct. So, a grading metric can also double as an agreement metric between two or more annotations, if we then average the results for each pair of annotations. This concept will be further explored in Chapter 5.

When in the context of these evaluation metrics, we will refer to the "gold standard" annotation and to the "target" annotation. Because, to grade annotations, one of them (the "gold standard" annotation) must be considered correct, and the other (the "target" annotation) is the one to evaluate.

In this thesis, two different annotation evaluation metrics will be presented. One of them, F1-Extended, is based on the F1-Based metric, and the other, the UCP Porto metric, was developed in collaboration with Universidade Católica. One common concern addressed in both metrics is matching annotated ranges rather than annotated words. A range, as explained in Chapter 3, is the interval between the first and last indexes selected to create a node. Similarly, the annotated words (or text) are the actual characters inside the text nodes that are visible in annotations like the one shown in Figure 3.21.

We decided also matching node ranges was a better way of evaluating annotations because the context from which a certain premise is annotated can determine important aspects, such as if it is an argumentative part of the text, or if it belongs to an argument and not another. Given that the final goal of evaluating and finding the agreement of a corpus is to then be able to use it for machine learning tasks, it becomes essential that we make sure that the right sections of the text are being annotated. This problem is illustrated in Figure 4.1, of an annotation of the the following text:

I agree with the opponents of nuclear energy, so I don't believe Portugal should follow that path.

But just because I agree with the opponents of nuclear energy, it doesn't mean that I am not open to discussing the issue.

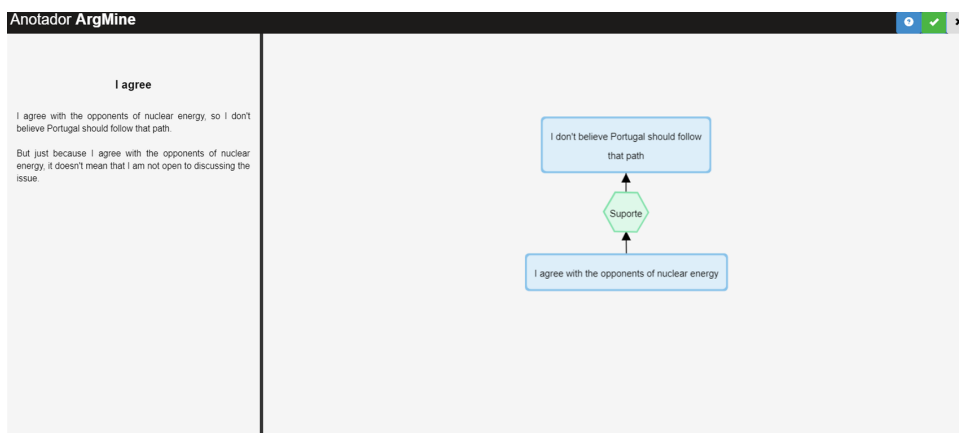


Figure 4.1: The ArgMine annotator showing a node that could have been annotated in two different contexts

In the annotation in Figure 4.1, there is a node with the text "*I agree with the opponents of nuclear energy*". Without looking at the ranges that were selected to create this node, there is no way to tell if the node was annotated in the context of the first paragraph or of the second paragraph. And it is not correct to interchange them because they belong in different argumentative contexts. So, this can clearly be a problem when grading the annotations and when using them in a machine learning setting. Because on the one hand we want to make sure that the annotator had the right argumentative context in mind, and on the other, we do not want the algorithms to learn to annotate text that actually has no argumentative value or context (even if it is textually similar to the parts that do have argumentative value).

## 4.2 New Approach to Node Matching

With the range problem (illustrated in Figure 4.1) in mind, we implemented an approach to node matching that takes into consideration firstly the annotated range, to account for the context in which they were created, and secondly the annotated text. The last part became necessary because the ArgMine platform allows the annotated text to be edited.

Also, by allowing a more flexible match we liken the evaluation metrics that use this kind of matching to more a human evaluation of annotations. Which is good because these annotations were made by people and we do not want to discard good annotations just because of slightly different ADU boundaries. In addition, the fact that we will take the percentage of range and text match into account leaves us with the possibility for more customizable matching. The higher we set the threshold, the more the ranges will have to overlap for us to pair up any two nodes.

We will now go into more detail on this kind of matching. As described in Section 3.3.3, the range is the interval between the index of the first annotated character and the last annotated character, and every node created by selecting text in the left-hand side panel has its own annotated range. On the other hand there may also be 0-range nodes that were created by double-clicking the right-side panel in Figure 3.21. Another thing to keep in mind before starting the explanation itself is that we decided that the comparison between ranges should not be entirely strict. As with comparing words, we want to allow a certain flexibility to encompass slight differences in ADU boundaries (introduced in Chapter 2).

Once the matching process has started, it first decides which ranges will be compared. As described in Section 3.3.3, annotations may have 0-range nodes. Since we want to use the selected range to convey context, it would go against our goal to match 0-range nodes with non-0-range nodes, which therefore were annotated in a context. For this reason we disregard the 0-range nodes in the beginning.

So, to compare the ranges of the non-0-range nodes, we compute a matching percentage between each pair, which will represent how much they intersect, using Algorithm 4.1. Here, *targetStart* and *targetEnd* are the start and end of the target annotation node's range, and *goldStart* and *goldEnd* are the start and end of the gold standard annotation's node range.

## Automatic Evaluation of Annotations

```
1
2 goldLength = float(goldEnd - goldStart)
3 targetLength = float(targetEnd - targetStart)
4
5 if(goldStart == targetStart and goldEnd == targetEnd):
6     return 1
7
8 matchPercentage = 0
9
10 if(goldEnd > targetStart and goldStart < targetEnd):
11
12     if(goldStart <= targetStart):
13
14         if(goldEnd >= targetEnd):
15             # target range is contained in the gold range
16             # check the percentage of match
17
18             matchPercentage = targetLength / goldLength
19
20         else:
21             # target range starts within gold range but ends outside
22             # we'll check the match percentage considering this
23
24             matchingLength = goldEnd - targetStart
25             matchPercentage = matchingLength / goldLength
26
27         else:
28             if(goldEnd > targetEnd):
29                 # target range starts outside but ends within the gold range
30                 # check percentage of match
31
32                 matchingLength = targetEnd - goldStart
33                 matchPercentage = matchingLength / goldLength
34
35             else:
36                 # target range starts outside and ends outside of the gold range
37                 # in this situation we check the match "in reverse" - how much of
38                 # the target is in gold
39                 matchPercentage = goldLength / targetLength
40
41     elif(goldEnd == targetStart or goldStart == targetEnd):
42         # the target range starts exactly where the gold range ends OR ends exactly
43         # where the gold range starts
44         # for small ranges this might mean high match percentage, so we calculate
45         # it here
46         # only one character in common with the gold range
47         matchPercentage = 1 / goldLength
48     else:
```

```

46     # the target range starts after the gold range ends - they're completely
         separate
47     matchPercentage = 0
48
49     return matchPercentage

```

Listing 4.1: The algorithm that checks the match percentage between two ranges, in Python

After two nodes' ranges have been matched, we must confirm the node match by taking a look at their text. The range matching alone does not guarantee that the nodes match because, as mentioned in Section 3.3.3, the ArgMine annotator offers the possibility to create nodes without selecting any text at all. This means that a node could be created using a certain range and subsequently edited beyond recognition, or that there are nodes created with no selected text (and thus no context) which can be edited to perfectly match a node with a range. In these cases, we do not want the nodes to be matched.

So, to definitely match the already paired nodes, we compare their texts in a similarly flexible way, described in Section 3.2. The texts only have to match up to a certain percentage, which may also be configured. This offers all the benefits that the flexible range matching also offers.

Then, as a final step in the matching, we attempt to match the 0-range nodes. Since they do not have ranges, they can only be matched with each other through the matching text percentage. If two 0-range nodes do match in text, they are added to the rest of the matched nodes. Otherwise, they are left unmatched.

## 4.3 F1-Based Metric

The F1-Based metric had already been implemented by the time this thesis project started, but it was not given a lot of use. When the opportunity arose to create an evaluation metric in collaboration with Universidade Católica, we looked back into it as a way to test how things would work in terms of coordination between the ArgMine platform and the evaluation server. However, it ended up being inserted into the platform.

### 4.3.1 Explanation

The F1-Based metric is very naive. It takes a gold standard annotation that is considered correct and a target annotation that will be graded. Then, it simply tries to match all the nodes, a pair at a time, by comparing their text. If the text matches beyond a certain threshold, which is customizable, the two nodes will be matched.

Then, to grade the target annotation, the F1-Based metric will calculate the *precision* and *recall* of the number of matched nodes. The harmonic mean between these values, which we consider to be a comprehensive view of the performance of the target annotation considering the gold standard annotation, will be the final grade:

$$\textit{Precision} = \frac{\textit{Number of matching nodes in the target annotation}}{\textit{Total number of nodes in the target annotation}}$$

$$\textit{Recall} = \frac{\textit{Number of matching nodes in the target annotation}}{\textit{Total number of nodes in the gold standard annotation}}$$

$$\textit{Harmonic mean} = \frac{2 \times \textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$

### 4.3.2 Shortcomings

As it is such a simple metric, it does come with a lot of room for improvement. One of the main problems we found with this metric was that it did not account for context when matching nodes, which is to say it only compares nodes by their text. As was exposed in this chapter, with the example in Figure 4.1, this was something we wanted to avoid in order to later use the metrics and annotations for machine learning purposes.

Another shortcoming of the F1-Based metric is that it is not an extensive evaluation of the annotation, mostly because it does not even glance at the relations between the nodes. In an extreme case, the target annotation could have unconnected nodes with exactly the same text as the gold standard annotation nodes, which would have connections between the nodes. And this target annotation could be graded as 100% matching the gold standard annotation, when really it does not even annotate a single argument.

## 4.4 Relation-aware-F1-based Metric

The Relation-Aware-F1-Based is based on the F1-Based metric, described in Section 4.3. Since it was essentially an extension of the F1-Based metric, we decided to call it F1-Extended.

The key differences between the Relation-Aware-F1-Based metric come as a result of trying to make up for some of the shortcomings mentioned about the F1-Based metric in section 4.3.2. These differences include matching the ranges that were used to create each node to match them, and taking a look at the connections between the nodes to account for nodes that may represent the same train of thought but were annotated slightly differently in the two annotations. Two differences that will be further explained in this section.

While this metric does have the potential to raise agreement between annotations, that is not its main purpose. Instead, we hope to provide a more forgiving approach to matching nodes that mimics the way a human would look at an argument, as we will exemplify in this section.

### 4.4.1 Explanation

As stated before, we were concerned with bringing context into the evaluation metrics, which also includes differentiating nodes with edited text from nodes with text annotated from the original text. The already implemented F1-Based metric did neither of these things, and that is one of

the differences between it and the newly developed metric, F1-Extended. The other difference lays in looking at the relations between nodes within an annotation to have a more comprehensive matching of nodes.

### 4.4.1.1 Part 1 - Node Matching

As for taking into account the context of the annotated text, we use the approach explained in Section 4.2. Then, as explained in that section, we get a list of all the pairs of nodes that matched firstly by range, and then by text.

### 4.4.1.2 Part 2 - Node Merging

At this point in the code, there a list of matching nodes between the two annotations, but each node matches only one of the nodes of the other annotation. However, as addressed in Chapter 2, different people may have different annotation styles even if they agree on a certain argument. An example of this will be discussed in Section 4.4.2, where we can see two annotations of the same argument that differ only slightly. From this example we can infer that the two annotators had the same argument in mind, and the grading of one of these annotations against the other should reflect this, we implemented the next step in this metric, where the algorithm will match not only nodes, but also sets of nodes.

This means that two or more nodes in an annotation may match just one node on the other, provided that they fulfill a set of requirements that guarantee that the thinking behind both annotations would be interpreted as similar if a human looked at both annotations, as happens in Figure 4.2. In this case, we aimed at accommodating the case when one of the annotators created one big node to support or attack a proposition but the other annotator split that same range and text into two or more smaller nodes to attack or support the same proposition. One thing to note is that whether the connection is of attack or support is not taken into account when matching the nodes, just as it isn't with the original F1-Based metric.

If we take a look at the example Figure 4.2, we can see that nodes 2 and 3 combined have the same text as node 5, so we will call them "smaller nodes". Conversely, the node that contains the combined text of the smaller nodes will be called the "bigger node" henceforth.

We will only merge the smaller nodes (nodes 2 and 3 in Figure 4.2) and match them to a bigger node (node 5 in Figure 4.2, if:

- All them, smaller and bigger, are connected to matched nodes. In the annotation in Figure 4.2, this means that the smaller nodes (2 and 3) and the bigger node (5) must be connected to nodes 1 and 4, respectively. Because node 1 matched node 4 in the previous part of this metric, this ensures that both annotators had the same support/attack relation in mind
- The smaller nodes are in a linked connection. A linked connection, as shown in Figure 2.1, means that the two premises must be considered together in order to prove a conclusion (as opposed to each of them being able to independently prove the conclusion). In the

annotation in Figure 4.2, nodes 2 and 3 must be in a linked connection to their conclusion, node 1. This means, of course, that the annotator of the smaller nodes (2 and 3) thought that neither of them could support the conclusion individually. Since the annotator of the bigger nodes decided not to split them, we must assume that they also thought that that piece of text needed to be considered together. So, we take this as further evidence that both annotators had the same argument in mind when annotating;

If these requirements are met, the next step is to check how much the ranges of the nodes match, and for this the algorithm transcribed in Algorithm 4.1. There are two range match percentages to look at:

- How much of each smaller node matches the bigger node (percentage *A*) – In the annotation in Figure 4.2, it simply means getting the range match percentage of node 2 and node 5, and getting the range match percentage of node 3 and node 5;
- How much of the smaller nodes, together, match the bigger node (percentage *B*) – That is, adding up the *A* percentages .

And after getting these percentages, we will check if they are higher than the configurable thresholds for each percentage. We established that there should be two percentages to weed out two cases:

- The bigger node contains a lot more text than the combined text of the smaller nodes (dealt with by percentage *B*);
- Each smaller node barely matches the bigger node individually, but all of the smaller nodes combined surpass the threshold (dealt with by percentage *A*)

Provided that percentages *A* and *B* are over their respective thresholds, the same percentages are calculated but using the text in the nodes instead of their ranges. Again, we must take this extra step because the text in the nodes may be editable in the ArgMine platform. However, if we choose to configure it that way, the thresholds for the *A* and *B* percentages in ranges can be different from the thresholds for the *A* and *B* percentages in text. This is for the sake of flexibility only.

Finally, if the nodes have passed all these steps, they will be merged. In the annotation in Figure 4.2, for example, this would mean that nodes 2 and 3 would be disregarded and replaced by a new node – which we can call 6 – that matches node 5. This means that, for the purposes of grading, the two smaller nodes will disappear and one other node will be added as a match to the bigger node. As this is as far as the Relation-Aware-F1-Based metric goes, we only altered the total number of nodes instead of actually creating a new merged node (hypothetical node 6). However, if the metric did go on, we would also have to add the new node 6 and node 5 to the list of matched nodes and possibly change the argument json to accommodate for these changes, always keeping in conformity with the AIF format.



#### 4.4.1.3 Part 3 - Grading

The grading step in this metric is exactly the same as the one in the F1-Based metric. We take the following formulas:

$$\text{Precision} = \frac{\text{Number of matching nodes in the target annotation}}{\text{Total number of nodes in the target annotation}}$$

$$\text{Recall} = \frac{\text{Number of matching nodes in the target annotation}}{\text{Total number of nodes in the gold standard annotation}}$$

$$\text{Harmonic mean} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

And we consider the harmonic mean to be the Relation-Aware-F1-Based grade.

#### 4.4.2 Example

The following is a very simple example tailored specifically for the purpose of featuring here as an example. The full text is:

John likes all movies with his favourite actor, and this movie stars his favourite actor.  
So, John likes this movie.

Depending on the annotator, this text can lead to two annotations which are mostly in agreement, shown in Figure 4.2.

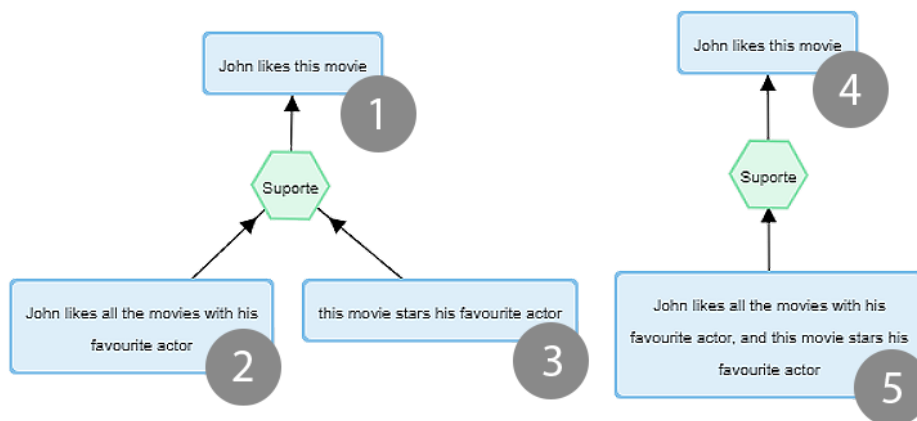


Figure 4.2: The two annotations that will be graded in this example. We may consider the right-hand side annotation as the gold standard annotation

In these annotations all the nodes were created by selecting and dragging the text, so all the nodes have corresponding ranges. For the purpose of this example, we use the settings in Table 4.1.

Threshold for text matching (part 1 - node matching)	60%
Threshold for percentage <i>A</i> in text matching (part 2 - node merging)	95%
Threshold for percentage <i>B</i> in text matching (part 2 - node merging)	95%
Threshold for range matching (part 1 - node matching)	60%
Threshold for percentage <i>A</i> in range matching (part 2 - node merging)	95%
Threshold for percentage <i>B</i> in range matching (part 2 - node merging)	95%

Table 4.1: The thresholds used in this example. The lighter shaded rows show the thresholds used for testing the text matching percentages, while the darker shaded rows show the thresholds used for testing the range matching percentages

After Part 1 (explained in Section 4.4.1.1), nodes 1 and 4 are already matched, as they have the same range and text. However, all the other nodes would be ignored because neither node 2 or node 3 makes up 60% of the range and text of node 5. When striking for comprehensive grading, this is not desirable.

So, continuing on to part 2 (explained in 4.4.1.2), we see that the relation starting in nodes 2 and 3 and the relation starting in node 5 both end in nodes that matched each other (nodes 1 and 4). So, nodes 2 and 3 can potentially be equivalent to node 5.

The next step is checking the relations. Since both relations are support relations, we go on to check if the relation of the smaller nodes, 2 and 3, is a linked relation. The requirement states that the smaller nodes be connected by a linked connection, and that the bigger node be connected by a simple convergent relation. That is exactly the case, so we can proceed with the grading. If the relation requirements were not met, the nodes would not be matched and the grading process would end here.

Now, when comparing the ranges, percentage *B* is 91.1% . This is enough to pass the "Threshold for percentage *B* in range matching (part 2 - node merging)" in Table 4.1. In addition, percentage *A* is 100% , which satisfies the "Threshold for percentage *A* in range matching (part 2 - node merging)" in Table 4.1.

As the final step, we calculate percentages *A* and *B* for the text in these nodes, instead of doing it for their ranges as we did just now. This gives us roughly the same values for *A* and *B* that comparing the ranges did, because the text in the nodes was not edited. These results are enough to pass the "Threshold for percentage *A* in text matching (part 2 - node merging)" and the "Threshold for percentage *B* in text matching (part 2 - node merging)" from Table 4.1.

And finally, nodes 2 and 3 will be merged and considered as one in the grading of the target annotation. The substitution of nodes 2 and 3 with a new node will allow the newly created node to match with node 5, bringing this annotation's Relation-Aware-F1-Based grade to 100%.

### 4.4.3 Shortcomings and Possible Improvements

A shortcoming of this metric is that part 2 (node merging) only works if the nodes to be merged are attacking/supporting another node. So, even though part 1 (node matching) works normally, part 2 (node matching) will only work with nodes that are not conclusions. It may also yield different

results depending on the percentage thresholds provided, but that is not as much a shortcoming as it is a necessary trade-off for flexibility and personalization of the metric.

It should be noted that another trade-off for the flexibility we allow in the Relation-Aware-F1-Based, this that this metric will not be as strict as the F1-Based metric to measure the ADU quality. Comparing with the F1-Based metric, this one has a layer that allows nodes to match when their range or text match percentage is lower than that allowed in the F1-Based metric (if it was higher, the nodes would already have been matched). That is the function of the parameters in Table 4.1, for example.

Moreover, this evaluation metric is still not very comprehensive. It only takes range and text into account in most cases, as it only looks at relations when merging nodes. From this comes the disadvantage of grading annotations with only unconnected nodes very highly, if most nodes match more than 60% (as per the example).

So, a big improvement opportunity is starting to look at relations, for example to only match nodes that are connected in the same way in the gold standard annotation and in the target annotation. Another improvement can be to check if the smaller node ranges overlap, because that could cause some false positives when calculating percentage  $B$ .

### 4.5 UCP Porto Metric

This metric was as a result of the ongoing collaboration with Universidade Católica during the development of this thesis project. The professors at Universidade Católica (UCP Porto) planed to have their students use the platform and provide us with annotations, while we tailored the ArgMine platform to their needs. Because of some unforeseen circumstances, namely a few bugs in the platform at the time they first tried to use it in their classes, they ended up not making as much use of the platform this year as we would have hoped. However, we still continued developing and implementing their requirements and suggestions. Among them was this evaluation metric, which we tentatively called UCP Porto metric because of this.

The suggestions that we got from the professors at UCP Porto served as guidelines for the kind of grading metric they wanted. It should be highly flexible and comprise of three steps:

1. Textual matching – when the algorithm should check how many nodes are correct on the annotation to grade;
2. Usage of nodes – when the algorithm should checks if the nodes were used correctly;
3. Relation matching – when the algorithm should check if the relations connecting the nodes are correct.

All of these steps are evaluated with the UCP Porto metric, as we will describe in the next section.

This metric, given its well-rounded scope of evaluation which considers not only the text in the nodes but also the relations connecting them, can be comparable to evaluation techniques like

CASS (described in Chapter 2). However, the two belong to different realms of the evaluation process. Especially if you consider the fact that the CASS technique is designed for annotations with dialogical data.

Since the UCP Porto metric offers a general grade for annotations with no dialogical data, it is a candidate to be used as a metric in some of the CASS process steps. The "segmentation" and "propositional content relations" steps of the CASS, described in Chapter 2, can be graded using the UPC Porto metric. And the adjustments that have to be made for that are simple, due to the metric's flexible grading system: to get the grade of just one of the levels that will be described in the next section, we just have to give a weight of 0 to the other levels in the final grade.

## 4.5.1 Explanation

Given the steps that the professors felt were important to grade, the UCP Porto able to grade on three levels: the node matching level, the role level, and the relation level, each of which will be further detailed below. Even though each level is graded individually, they do have implications on each other, particularly the first one. We will now explain each level and its grading in more detail.

### 4.5.1.1 Level 1: Text Matching

As exposed in Section 4.1, we decided that matching the pure text would not be enough to guarantee that two nodes represented the same portion of argumentative text. Instead, the range of text that was selected to create that node is much more informative of the mindset of the annotator than the words themselves. So, for the sake of both grading a student and of machine learning, in this metric we will also consider the range of the nodes to pair them up.

To match the ranges, as in the Relation-Aware-F1-Based metric, we use the approach described in Section 4.2.

Much like with the final grades in the F1-Based metric and the Relation-Aware-F1-Based metric, this level's grade is based on the harmonic mean of precision and recall of the target annotation in relation to the gold standard annotation:

$$Precision = \frac{\text{Number of matching nodes in the target annotation}}{\text{Total number of nodes in the gold standard annotation}}$$

$$Recall = \frac{\text{Number of matching nodes in the target annotation}}{\text{Total number of nodes in the gold standard annotation}}$$

$$Harmonic\ mean = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

To reach the actual grade, this level's result is sifted through a grading scale. This grading scale is meant to bring together the metric's result with a more human way of grading, especially because

it is also completely configurable. It is noteworthy that this is the only level where the grading scale is put to use.

Finally, the way the grading scale works is by creating a series of thresholds and steps. For an example, we can look at Table 4.2.

Grading Scale Grade 1	0
Grading Scale Threshold 1	60%
Grading Scale Grade 2	25%
Grading Scale Threshold 2	70%
Grading Scale Grade 3	50%
Grading Scale Threshold 3	80%
Grading Scale Grade 4	75%
Grading Scale Threshold 4	90%
Grading Scale Grade 5	100%

Table 4.2: An example grading scale to use in this level. The "Grading Scale Grade"s will be the "gradeNodeMatching" in Algorithm 4.2

In Table 4.2, if the harmonic mean is less than 60%, the grade will be zero. While at the other side of the spectrum any harmonic mean beyond 90% will be given the maximum score. We can see the code for this example in Algorithm 4.2. In the code this scale is customizable, so all the numeric values in the example are actually variables.

```

1 gradeNodeMatching = 1
2
3 if (harmonicMean <= 0.6):
4     gradeNodeMatching = 0
5 elif (harmonicMean < 0.7):
6     gradeNodeMatching = 0.25
7 elif (harmonicMean < 0.8):
8     gradeNodeMatching = 0.5
9 elif (harmonicMean < 0.9):
10    gradeNodeMatching = 0.75

```

Listing 4.2: The code implementation of the example in Table 4.2, in Python.

#### 4.5.1.2 Level 2: Node Role

In the next level, we will evaluate the role that the nodes perform in the annotation as a whole. In the context of this metric, we consider that a node can perform one of the following three possible roles:

- **Conclusion** — a node that does not have an outgoing edge. It is not required that there be an incoming edge to a conclusion, so a lone node will be considered a conclusion;

- **Premise** — a node with an outgoing edge pointing to a conclusion. Even if this node has another outgoing edge that does not point to a conclusion, the node will still be considered a premise;
- **Justification** — a node with at least one outgoing edge pointing to a premise. This kind of node must only have outgoing edges pointing to premises, because if any outgoing edge points to a conclusion the node will be considered a premise.

So, to grade the node roles, we classify each node on the annotations into one of these three roles. It is important to notice that we only consider the nodes that have been matched between the two annotations, so those are the only ones that are classified. Furthermore, in this level we are concerned with whether matching nodes also have the same roles. This means that after classifying all nodes into the roles, we check if the matching nodes (determined in level 1) have the same role in both annotations. If they do not, they will not count as correct nodes.

After classifying the nodes, we progress to the grading stage. To grade this level we calculate three components: the justifications grade component, the premises grade component and the conclusions grade component:

$$\begin{aligned} & \textit{Justification grade component} = \\ = & \frac{\textit{Number of target justification nodes that match gold justification nodes}}{\textit{Number of gold justification nodes}} \end{aligned}$$

$$\begin{aligned} & \textit{Premise grade component} = \\ = & \frac{\textit{Number of target premise nodes that match gold premise nodes}}{\textit{Number of gold premise nodes}} \end{aligned}$$

$$\begin{aligned} & \textit{Conclusion grade component} = \\ = & \frac{\textit{Number of target conclusion nodes that match gold conclusion nodes}}{\textit{Number of gold conclusion nodes}} \end{aligned}$$

We calculate each component by dividing the target node count by the gold standard node count because the gold standard one is the one that we must consider correct. And it is safe to do so, as we only classify the nodes that matched in the first level, guaranteeing that the components will never be higher than 1.

Then, to finally calculate the role grade, we add up each component's grade divided by their weight, which is also configurable::

$$\begin{aligned} & \textit{Level 2 grade} = \\ = & \textit{Justification grade component weight} \times \textit{Justification grade component} + \\ & + \textit{Premise grade component weight} \times \textit{Premise grade component} + \end{aligned}$$

$$+ \textit{Conclusion grade component weight} \times \textit{Conclusion grade component}$$

### 4.5.1.3 Level 3: Relations

For the final level in this metric, we evaluate the relations between the nodes. So, each relation will be graded. While in the previous level we only looked at the incoming and outgoing relations of each node, in this level we will look at what kind of relations are connecting the nodes and whether they connect the right nodes. There are three evaluating steps to grading the relations, described below and shown in the code in Listing 4.3, and the relations must pass each step to get the score for the next one.

In the **first step, the Relation Matching Step**, we simply check if the nodes at the ends of the relation are matching nodes. This is how we match the relations between the two annotations, as we want to compare only relations that connect matching nodes from both annotations. In the code in Listing 4.3 the Relation Matching Step happens roughly from lines 3 to 42.

To evaluate the **second step, the Support/Attack Check Step**, we take a look at the kind of relation which connects the matching nodes on both annotations. That is, if the relations are of attack or support. In the code in Listing 4.3 the Support/Attack Check Step happens between lines 46 and 50.

In the **third step, the Convergent/Linked Check Step**, we check if the relations are of the same type. There are two possible types of argumentative relations in this case, which were also explained before in Chapter 2:

- **Convergent relation** — the relation starts in a single node that must independently attack or support a single final node of the relation. Note that if two nodes were supporting the same conclusion with convergent relations, these would be separate relations and each of them would have its own grade in this metric;
- **Linked relation** — the relation connects two or more nodes that need to be considered together to be able to attack or support a single final node of the relation. Contrary to what happens with convergent relations, if two nodes were supporting the same conclusion with a linked relation, there would be only one relation. And this relation would naturally only have one grade in this metric.

Checking if the relations are of the same type is very strict in the Convergent/Linked Check Step. Because we consider that even slight changes in the relations are a result of significant difference in thought. Here are some examples:

- The same three nodes are connected with a linked relation on the gold standard annotation but with separate convergent relations on the target annotation. So, the target annotator thought that all the three nodes proved the conclusion independently (convergent relations), which is wrong since the gold standard annotation established that they need to be looked at together to prove the conclusion (linked relation). In the end, this means that the target annotator did not provide even one good premise to support/attack the conclusion;

## Automatic Evaluation of Annotations

- A linked relation on the gold standard annotation has two source nodes, while on the matching linked relation on the target annotation has the same two source nodes and an extra one. This implies that the target annotator thought that only annotating the two nodes (the correct nodes) was wrong or insufficient to support/attack the conclusion;
- The same three nodes are connected with convergent relations on the gold standard annotation but with a single linked relation on the target annotation. While it is a small difference, the target annotator thought that the nodes were not enough to support/attack the conclusion independently. Which was not the case, as established by the gold standard annotation.

With this in mind, we decided that the relations should match exactly between the two annotations. Only in that case do we give the score for the Convergent/Linked Check Step. In Listing 4.3, we can find the Convergent/Linked Check Step in line 449, where all the matching relation sources are added up.

```
1
2 # if the target relation source is a matched node, we can further compare the two
   relations
3 for goldRelation in goldRelationList:
4
5     goldRelationTarget = goldRelation["targetNode"]
6     goldRelationSources = goldRelation["sources"]
7     # for STEP 3
8     numberOfSourcesOnGold = len(goldRelationSources)
9
10    # try to find the relation in the target annotation that points to the match of
       goldRelationTarget
11    for targetRelation in targetRelationList:
12
13        targetRelationTarget = targetRelation["targetNode"]
14        targetRelationSources = targetRelation["sources"]
15
16        # targetRelation's target node matched this goldRelation target node
17        if targetRelationTarget in matchedNodes:
18
19            targetRelationSources = targetRelation["sources"]
20            # for STEP 3
21            numberOfSourcesOnTarget = len(targetRelationSources)
22
23            # at this point we know that the gold and target relation are pointing
               to matched nodes, but we will only check the relation sources now
24            # note that a relation may have several sources, if it is of type "
               linked"
25
26            for targetSourceNode in targetSources:
27
```



## Automatic Evaluation of Annotations

```
28         # only proceed if the target relation's source node has been
           matched
29         if targetSourceNode in matchedNodes:
30
31             # and look for the matching node on the gold relation's source
               nodes
32             for goldSourceNode in goldRelationSources:
33
34                 if targetSourceNode == goldSourceNode:
35
36                     # finally identified a matching relation
37
38                     # for STEP 1
39                     # account for the matching relation
40                     goldAndTargetSourcesAndTargetsMatch = 1
41
42                     # for STEP 2
43                     # check if the relation type (attack/support) is the
                       same on both annotations
44                     targetRelationType = targetRelation["type"]
45                     goldRelationType = goldRelation["type"]
46
47                     if targetRelationType == goldRelationType :
48                         typeOfRelationMatches = 1
49
50                     # for STEP 3
51                     # count the number of matching sources
52                     matchingSourcesFound += 1
```

Listing 4.3: The simplified snippet of the code used to check and grade all three steps of this level, in Python

With this, we have checked all the relations and we have enough information about each of the relations to grade them. The code used for this is shown in Algorithm 4.4. As we can see there, the default grade is 0 and it is discrete. It can only be incremented step by step, if the conditions in each step described above are met. So, a grade will only receive the score for step 1 (Convergent/Linked Check Step) if it also received the score for step 2 (Support/Attack Check Step), and so on. The scores granted by each step are configurable.

```
1
2     # in case the gold relation does not match any target relation, the grade is 0
3     thisRelationGrade = 0
4
5     # final check: if the gold and target relations connect *at least one* matched
           source to the same matched target
6     if goldAndTargetSourcesAndTargetsMatch :
7
```

## Automatic Evaluation of Annotations

```
8      # STEP 1 GRADE is added to the final grade
9      thisRelationGrade = gradeRelationStep1Weight * 1
10
11     # final check: if relation is of the right kind (attack/support)
12     if typeOfRelationMatches :
13
14         # STEP 2 GRADE is added to the final grade
15         thisRelationGrade += gradeRelationStep2Weight * 1
16
17         # final check: all the relation sources on the target annotation are
18             matched to relation sources on the corresponding gold standard
19             relation (matchingSourcesFound == numberOfSourcesOnGold), and if
20             there are no extra sources on either side
21             (numberOfSourcesOnTarget == numberOfSourcesOnGold)
22         # note: if there are no extra sources on either side, as we cannot give
23             the grade if one of the relations has more nodes than the other
24         if(matchingSourcesFound == numberOfSourcesOnGold and
25             numberOfSourcesOnTarget == numberOfSourcesOnGold):
26
27             # STEP 3 GRADE is added to the final grade, always making it reach
28                 1
29             thisRelationGrade = 1.0
```

Listing 4.4: The simplified code used to grade each gold standard relation, after it has been put through the code in Listing 4.3 (and still inside the first for loop shown there), in Python

Finally, after all the gold standard annotation relations have been submitted to the chain of steps to grade them, they will each have their own independent grade. However, we need to get a total grade for this level. We use the following formula:

$$\text{Level 3 grade} = \frac{\text{Sum of all gold standard annotation relation's grades}}{\text{Total number of relations on the gold standard annotation}}$$

By looking at this formula it may seem like we are grading the gold standard annotation, but this is not the case. We use this formula in order not to penalize any extra relations between extra nodes that might have been annotated in the target annotation, but that have not matched any nodes on the gold standard annotation. Those aspects have already been graded and penalized in other levels, and in this one we focus on relations. So, here we consider only the number of nodes and relations that the target annotation was supposed to have, which is the total number of relations in the gold standard annotation. In addition, this does not keep missing relations (that did not pass the first step and thus received an independent grade of zero) from being duly penalized in this level.

### 4.5.1.4 Final Grade

To calculate the final grade of the target annotation, there is yet another layer of configurable weights to allow a more flexible metric. So, each level will be multiplied by a certain percentage

and the results of these multiplications will be added up to get the final grade:

$$\begin{aligned}
 \text{Final grade} &= \\
 &= \text{Level 1 weight} \times \text{Level 1 grade} + \\
 &+ \text{Level 2 weight} \times \text{Level 2 grade} + \\
 &+ \text{Level 3 weight} \times \text{Level 3 grade}
 \end{aligned}$$

#### 4.5.2 Example

As there are not many available evaluation metrics in the literature, we chose to show an example that was made specifically to exemplify and showcase this metric’s grading process and its levels. The text used in this example was written with this purpose and thus cannot be considered an argumentative text in its full merit.

The complete text reads as follows:

"I'm wondering if I should feed my cat now. I'm not going to be able to feed my cat later because I'm going to work. But I think the cat is not hungry because if the cat is hungry, it will meow and the cat is not meowing. The cat's food is tuna flavored and I still have a lot of cans left in the pantry..."

And from this text, the annotation in Figure 4.3 was produced to be considered the gold standard annotation in this example:

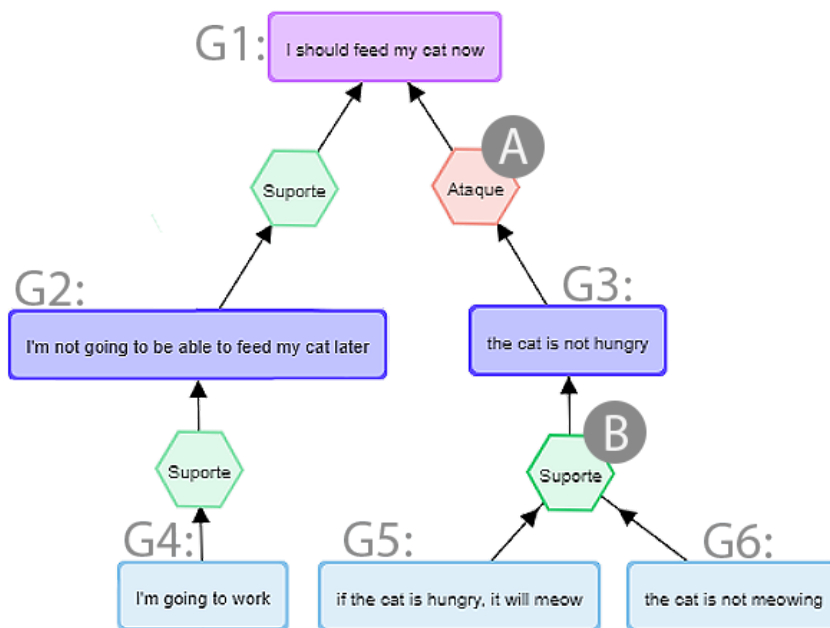


Figure 4.3: The annotation used as gold standard annotation for this example

And the annotation in Figure 4.4 was produced to act as the target annotation in this example:

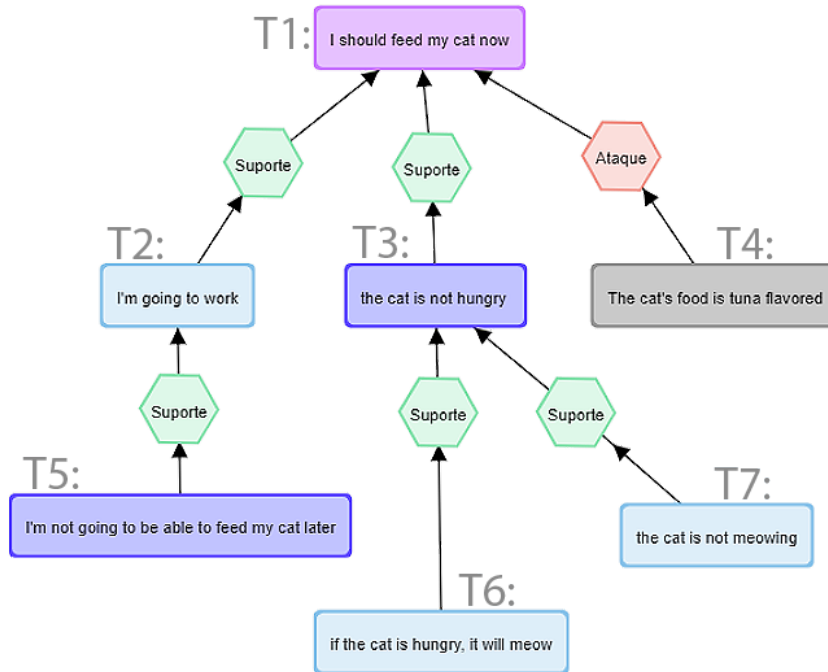


Figure 4.4: The annotation used as target annotation for this example

In both the gold standard and the target annotations, all the nodes were created in the more traditional way where the user selects the text and then drags it to the annotation pane. This means that all the nodes have ranges that will be taken into account when grading the first level, and that there are no 0-range nodes (introduced in Section 3.3.3).

To grade this target annotation, we have used the parameters in Table 4.3 for the UPC Porto metric:

Level 1 weight	40%
Level 2 weight	40%
Level 2: conclusion identification weight	50%
Level 2: premise identification weight	25%
Level 2: justification identification weight	25%
Level 3 weight	20%
Level 3: step 1 (Relation Matching Step)	20%
Level 3: step 2 (Support/Attack Check Step)	40%
Level 3: step 3 (Convergent/Linked Check Step)	40%

Table 4.3: The weights used in the UCP Porto metric for the sake of this example. The darker rows show the level's total weight in the grade, while the white rows show the weights used to grade within the levels

Grading Scale Grade 1	0
Grading Scale Threshold 1	60%
Grading Scale Grade 2	25%
Grading Scale Threshold 2	70%
Grading Scale Grade 3	50%
Grading Scale Threshold 3	80%
Grading Scale Grade 4	75%
Grading Scale Threshold 4	90%
Grading Scale Grade 5	100%

Table 4.4: The grading scale used in this example. The thresholds are shown in darker shade and the grades that are used up to that threshold are shown in white. For example, an annotations with 85% level 1 score will be graded

With all these parameters, we may proceed to the grading process.

#### 4.5.2.1 Level 1 grading

In the first level, the nodes will be matched in pairs where possible. As we can tell by looking at the text of both annotations, all the nodes in the gold standard annotation are paired with nodes on the target annotation, but the opposite is not true. The black colored node on the target annotation does not have a match on the gold standard annotation and so will not be taken into account when grading any of the subsequent levels of the metric. In Table 4.5 we show the final node matching.

Gold Standard Node	Target Node
G1	T1
G2	T5
G3	T3
G4	T2
G5	T6
G6	T7

Table 4.5: The matched nodes table for this example. Target node T4 does not figure in it, as it matched no node on the gold standard annotation. It will be ignored after this level

To grade this level we calculate the harmonic mean between the precision and recall of the matches. So:

$$Precision = \frac{6}{7} = 0.85$$

$$Recall = \frac{6}{6} = 1$$

$$Harmonic\ mean = \frac{2 \times 0.85 \times 1}{0.85 + 1} = 0.92$$

And according to the grading scale in Table 4.4, since 92%  $\geq$  90%, the grade for this level will be 100%.

### 4.5.2.2 Level 2 grading

In this case, despite the fact that all the gold standard nodes have matches in the target annotation, not all the nodes perform matching roles in the two annotations. The nodes marked G2 and G4 in the gold annotation are, respectively, a premise and a justification in the gold standard annotation. However, their matching nodes in the target annotation are performing the roles of justification and premise, respectively.

To grade this level, we must take into account the importance given to identifying conclusion, premises, and justifications in Table 4.3. According to the table, the weights for identifying these elements are, respectively, 50%, 25% and 25%.

So, the grading will occur as follows:

$$\textit{Conclusion score} = \frac{1}{1}$$

$$\textit{Premise score} = \frac{1}{2}$$

$$\textit{Justification score} = \frac{2}{3}$$

$$\textit{Level 2 grade} = 0.5 \times 1 + 0.25 \times \frac{1}{2} + 0.25 \times \frac{2}{3} = 0.79$$

And this level's final grade is 79%.

### 4.5.2.3 Level 3 grading

In this level the grading is incremental, since we add one of the "Level 3" scores in Table 4.3 at a time, in case the relation is correct. We should note that we will go through the relations from the standpoint of the gold standard, because of the grading issue explained at the end of Section 4.5.1.3. So we will only grade the four relations in the gold standard annotation, ignoring all the others.

The first step (Relation Matching Step) in this level looks at which relations connect the same nodes in the target annotation (Figure 4.4) as they do in the gold standard annotation (Figure 4.3). As is visible from the figures just mentioned, only relations A and B, as marked in Figure 4.3, connect nodes that are matched (Table 4.5 shows the matched nodes).

In this step, it may happen that a relation matches more than one relation on the other annotation, as is the case of relation B. This will be dealt with later, and the Relation Matching Step is considered 100% correct for these cases.

So, only these two gold standard relations will receive this step's score. The remaining relations, not having completed step 1, will all have a score of 0.

At this point, we have the score table as shown in Table 4.6:

## Automatic Evaluation of Annotations

Relation	Score
A	20%
B	20%

Table 4.6: The level 3 cumulative score table after the first step

Then, the second step (Support/Attack Check Step) will check if the relations that match the same nodes are also of the same kind. If a relation on the gold standard annotation is a support relation, the matching relation on the target annotation must also be a support relation. Naturally, the same applies to attack relations. From the gold standard annotation relations that have completed step 1, one of them is not of the same kind as its matching relation in the gold standard annotation. That relation, the one marked with a A in Figure 4.4, will not receive any score henceforth, while the other one will be given this step's completion grade as established in Table 4.3.

After the second step, the score table becomes as shown in Table 4.7:

Relation number	Score
A	20%
B	60%

Table 4.7: The level 3 cumulative score table after the second step

In the following and last step (Convergent/Linked Check Step), the target relations which completed the second step will be checked to see if their type matches their paired relations on the gold standard annotation. As we can see in the gold standard annotation in Figure 4.3 and in the target annotation in Figure 4.4, the gold relation marked B connects three nodes that have their matches connected by two different relations in the target annotation. This means that the type of relation is linked in the gold standard annotation but convergent in the target annotation, and that relation B will not receive the remaining 40% that would give it a full score. So, the score table after the third step will be as shown in Table 4.8, which is the same as Table 4.7 because none of the relations got any more score in this step.

Relation number	Score
A	20%
B	60%

Table 4.8: The level 3 cumulative score table after the third step

After scoring each gold relation, we can grade the third level by averaging the total score over the number of gold standard relations:

$$\begin{aligned}
 & \text{Level 3 grade} = \\
 \text{Level 3 grade} &= \frac{\text{Sum of all gold standard annotation relation's grades}}{\text{Total number of relations on the gold standard annotation}}
 \end{aligned}$$

$$= \frac{0 + 0.2 + 0.4 + 0.4}{4} = 0.35$$

And the third level grade is 35%

#### 4.5.2.4 Final grade

Now we will use the level weights that are highlighted with darker color in Table 4.3 to calculate the final grade:

$$\begin{aligned} \text{Final grade} &= \\ &0.4 \times 1 + 0.4 \times 0.79 + 0.2 \times 0.35 = 0.78 \end{aligned}$$

So, considering the gold standard annotation in Figure 4.3, the UPC Porto metric's grade for the target annotation in Figure 4.4 is 78%.

### 4.5.3 Shortcomings and Possible Improvements

Even though this is a highly comprehensive and customizable metric, there are obviously some aspects about it which can be improved.

One easy way to improve the UCP Porto metric is to use the Relation-Aware-F1-Based process of merging nodes in the first level of grading. The Relation-Aware-F1-Based code is easy to conciliate in this metric's code, and it was just a matter of time that kept us from doing this.

The shortcomings in using this metric as an automatic evaluation metric appear when the annotations are very simple. In these extreme cases, the resulting grades can be regarded as inflated. For example, if a gold standard annotation with unconnected nodes is used to grade an empty target annotation, the grade for level 3 will be 100%. This is because both the gold standard annotation and the target annotations have no relations, which is full agreement on that level.

## 4.6 Comparing the Evaluation Metrics

To some extent these metrics do complement each other, in the sense that both the Relation-Aware-F1-Based metric and the UCP Porto metric use the F1-Based metric in their initial node matching phase. Furthermore, the Relation-Aware-F1-Based metric is also a candidate to be used in the node matching level of the UCP Porto metric. Despite this, each metric may have its own place when evaluating different annotation tasks.

For example, if the annotation task is a simple one, such as Named-Entity Recognition (NER), the most appropriate of these metrics would be the F1-Based metric. In the case of NER there are no relations between annotated content, rendering the Relation-Aware-F1-Based and UCP Porto metrics irrelevant, as and both these metrics need relations to grade past the node matching phase. So, using them would only result in unnecessary efforts such as setting up threshold and grading weights.



## Automatic Evaluation of Annotations

As to the Relation-Aware-F1-Based metric, it is most effective when the node content is the primary focus of the annotation task. As explained in this chapter, it will merge nodes with similar content, unlike the F1-Based and UCP Porto metrics. Allowing the metric to selectively forsake the ADU boundaries creates more agreement, but, as the thresholds are completely customizable, it can be controlled to match the evaluator's needs.

Finally, the UCP Porto metric is the best fit for more complex annotating tasks, or when the evaluator wants a grade to reflect the annotations in a more comprehensive manner. This means that relations will be a big part of what needs to be evaluated, and the UCP Porto dutifully delivers this evaluation while the Relation-Aware-F1-Based and F1-Based metrics do not. Because it considers not only node matching but also the annotation's relations on two levels (attack/support and linked/convergent), the UCP Porto metric is the best evaluator of the meaning of the annotation out of the three metrics here exposed. In addition, and in contrary to the Relation-Aware-F1-Based metric, the UCP Porto metric is not as lenient with ADU boundaries, which may also be desirable in some situations.

## Automatic Evaluation of Annotations

## Chapter 5

# Inter-Annotator Agreement

As introduced in Section 2.5, measures of inter-Annotator Agreement (IAA) are widely used by the scientific community to test and report the reliability of annotation corpora created by multiple people [Art17]. These corpora can be found in a variety of different fields: legal (such as the one introduced in [GHH04]), medical (such as the one introduced in [NLP<sup>+</sup>18]), linguistic (such as the one introduced in [BEF<sup>+</sup>06]), etc. The field of AM is no exception. In the pursuit of the goal of automatically extracting arguments from text, any effort in the field of AM will need some input, which must come from people extracting arguments from text (or a smaller task related to this, such as syntactical elements) because that is what AM is trying to mimic. However, with any annotation effort with multiple it is common to compare the annotations, for a variety of reasons such as improving annotation schemes and guidelines, identifying ambiguities and difficulties in the source, calculating corpus reliability and reproducibility, among others [Art17]. This is where IAA measures come in.

In AM specifically, we need reliable and reproducible corpora of annotations done by different people, to serve as input in the machine learning process. The IAA of the corpora is what guarantees that the annotation process is sound. Because different people annotated a set of documents and their mostly agreed, which shows the consistency of the annotation process. And that is what is key to good AM results [Art17].

However, as was exposed Chapter 2, there is no perfect way to evaluate this yet. Many measures present inconsistencies and misrepresent the corpus where they are used [AP08]. Despite this, in lack of better solutions, there are some flawed metrics used throughout the field, such as Cohen's Kappa, Krippendorff's Alpha, and others.

In this chapter we will go into detail about these two metrics and their implementation in the context of this thesis. We will also describe the application of the evaluation metrics presented in Chapter 4 as IAA metrics, and how we went about implementing this application.

## 5.1 Problem and Motivation

The context of this thesis is inserted in a project which aims to advance in the field of Argumentation Mining.

With this in hand, we still needed to know for sure whether the data was usable or not. For that purpose, two IAA metrics were implemented and run on this corpus (Cohen’s Kappa, Krippendorff’s Alpha). In addition to these two widely used IAA metrics, we also used the F1-Based, Relation-Aware-F1-Based and UPC Porto annotation evaluating metrics, presented in Chapter 4, as IAA metrics.

We decided to use these two evaluation metrics as IAA metrics as well because it is well-documented, and exposed in Chapter 2, how the current set of IAA metrics is lacking in many ways. And while these metrics were created with evaluation in mind, we feel they could also provide representative results when it comes to agreement. Especially because they do not only take into account the annotated tokens but also how they are used in the annotation.

Another propelling factor is that the process of calculating an IAA score can actually accommodate any evaluation metric we may want to use. The adjustment we must make to calculate the evaluation metrics as IAA metrics is to change which annotation is considered the gold standard and then make the rounds of all the other annotations as target annotations and calculate their grade. After considering each annotation to be the gold standard and grading all the other annotations, we average all the grades, as is customary with established agreement metrics such as Kappa, and we get a meaningful agreement score.

However, evaluation metrics require a gold standard annotation, a target annotation, and cannot grade more than one annotation at a time. In the pair of annotations, the gold standard is considered the correct annotation and the target annotation is graded according to how much of it is in agreement with the correct annotation. On the other hand, in IAA there is no such thing as a gold standard annotation, which leaves us with a question of which annotation should be given to the evaluation metric as the gold standard one. We decided that the most fair way to resolve this question, and to adjust the evaluation metrics into being used as IAA, would be to run these evaluation metrics twice for each pair of annotations, and switch which one we consider the gold standard each time. This way we can account for the perspectives of all annotators in the final agreement score of an entire corpus. We show a code representation of this in Algorithm 5.1, which is run once per document in a corpus.

```

1
2   for annotation1 in docAnnotations:
3       for annotation2 in docAnnotations:
4
5           annotationPair = annotation1 + annotation2
6
7           if annotation1 != annotation2:
8
9               if not annotationPair in alreadyGradedPairs:
```

## Inter-Annotator Agreement

```
10
11     # found a new pair of annotations
12     alreadyGradedPairs.append(annotationPair)
13
14     grade = evaluationMetric(annotation1, annotation2,
15                               metricWeightSettings)
16     metricGrades.append(grade)
```

Listing 5.1: The simplified code used to adjust the evaluation metrics to function as IAA metrics, in Python

This is the basis of how we use the evaluation metrics as IAA - each annotation will have to be considered the gold standard annotation once, and all the other annotations will receive a grade considering that. The main consequence of this process is that each pair of annotations will have two grades. And we will have to account for that when we average the results of the metric to get the IAA score of an entire corpus, in the following formula:

$$\begin{aligned} & \textit{Evaluation Metric as IAA score} = \\ & = \frac{\textit{Sum of all documents' "metricGrades" list}}{2 \times \textit{Sum of pairs of annotations in each document}}, \end{aligned}$$

where "metricGrades list" is the "metricGrades" list found in Algorithm 5.1.

## 5.2 Kappa

The Kappa agreement metric, explained in Section 2.5 is obtained by averaging all the results of calculating Cohen's Kappa pairwise throughout a corpus. In turn, Kappa pairwise will give a score based on how much two annotators have annotated the same parts of the text, calculating observed agreement and expected agreement in the process. So, to calculate this metric we also used the full text of the documents themselves, in addition to the annotation's json files.

We use words as tokens, and we stripped them of some of the most common trailing punctuation. At the end of a word, the following characters are eliminated: period (.), comma (,), exclamation mark (!), question mark (?), semicolon (;), colon(:). The reason behind this is that we thought it was going to be a common problem that some annotators included the trailing punctuation in their annotations and some did not. By removing the most common punctuation marks from both the original text and the annotation node's text, we reduce the impact that this might have brought upon the agreement score. It is expected that the agreement will increase after doing this, but we justify increase by noting that two annotation that are exactly alike with the exception of trailing punctuation still agree on the most relevant part to Argumentation Mining: the annotated argument is the same.

After all the texts have been cleared, we will calculate the expected agreement and the observed agreement to be able to finally calculate the IAA according to Cohen's Kappa. These notions are explained in Chapter 2, so we will only lightly go over them in this chapter. In this case, we can

## Inter-Annotator Agreement

think that each token can be labeled as "argumentative" or "non-argumentative", as the annotators had only two options: to annotate the token or not to annotate it. The distribution of the possible annotation of tokens is shown in Table 5.1.

		A1	
		Arg.	Non-Arg.
	Arg.	a	b
	Non-Arg.	c	d

Table 5.1: Table showing the possible distribution of tokens across two annotations made by two annotators: A1 and A2

The observed agreement, which comes down to how many tokens both annotators annotated with a certain label, was calculated using the following formula:

$$\text{Observed agreement} = \frac{a + d}{a + b + c + d}$$

As to the expected agreement, it is calculated as the probability of both annotators labeling a token in a each of the categories, if they were annotating randomly. So:

$$\text{Probability of annotating "Arg."} = \frac{a + b}{a + b + c + d} \times \frac{a + c}{a + b + c + d}$$

$$\text{Probability of annotating "Non-Arg."} = \frac{c + d}{a + b + c + d} \times \frac{b + d}{a + b + c + d}$$

$$\begin{aligned} \text{Expected agreement} &= \\ &= \text{Probability of annotating "Arg."} + \text{Probability of annotating "Non-Arg."} \end{aligned}$$

To get the Kappa value for an entire corpus, we then average the Kappa result of each pair of annotations.

### 5.3 Krippendorff's Alpha

Unlike Cohen's Kappa, Krippendorff's Alpha bases its inter-annotator agreement value on expected and observed disagreement, rather than expected and observed agreement. However, this did not have much impact on our approach to processing the data to calculate Krippendorff's Alpha, as it was done in the same way as when processing the data for calculating Kappa.

So, we again considered words as tokens. And again the words in the original text and in the texts in each node were cleared of some common trailing punctuation marks (period (.), comma (,), exclamation mark (!), question mark (?), semicolon (;), colon(:)) to try to mitigate the effect of the lack of annotation guidelines.

After processing the texts we calculate the expected disagreement and the observed disagreement, which requires us to decide on a measure of severity of disagreement. This can be any

function ranging from 0 to 1, as mentioned in Chapter 2. Because there are only two possible labels in this case (labeling a token as argumentative or as non-argumentative) which are totally different, we had to consider the severity of disagreement to be 0 or 1 in all cases, with no possibility for anything in between.

The observed disagreement ( $P_\alpha$ ) and the expected disagreement ( $P_e$ ), this last one based on the disagreement expected if the annotator was choosing the label randomly, are then used to calculate the inter-annotator agreement of the corpus with:

$$\alpha = \frac{P_\alpha - P_e}{1 - P_e}$$

## 5.4 F1-Based Metric

Grading based on precision and recall, this evaluation metric must undergo the process of adjustment described in the final part of Section 4.2.

We should note that the F1-Based metric relies on a type of relaxed string matching, which takes a parameter in order to report two strings as matches or not. If two node's text strings match above this flexible threshold, the nodes are matched.

To calculate a general corpus score, we make use of the final formula in Section 5.1.

## 5.5 UCP Porto Metric as IAA

Since the UCP Porto metric is an evaluation metric, we will use the approach described in Section 5.1 to adjust it into giving us a global corpus score.

In this evaluation metric there is no need to process any text because the metric uses only the annotations to calculate the score.

To calculate the UPC Porto score as the IAA of a corpus, as mentioned before, every annotation will have its turn as the gold standard annotation and all the other annotations will be graded according to it. And we will have to calculate the final corpus' IAA score with the formula shown in Section 5.1.

## 5.6 Relation-Aware-F1-Based Metric as IAA

In much the same way that we can apply the UPC Porto evaluation metric as an inter-annotator agreement metric, we can do the same with the Relation-Aware-F1-Based metric. Much like the UCP Porto metric, with the Relation-Aware-F1-Based metric there is a difference in grade depending on which annotation is considered the gold standard. So, again, we will have to calculate two grades for each pair of annotations in order for each annotation to have its turn as the gold standard.

Like in the case of UCP Porto metric, there is no need to process any text because the metric uses only the annotations to calculate the score.

Finally we calculate the Relation-Aware-F1-Based score of a corpus using the same formula as when calculating the UCP Porto score: the one shown in Section 5.1.

## 5.7 IAA Metrics Application

### 5.7.1 The ArgMine Corpus

After implementing these metrics, we are going to apply them and check the results of the ArgMine corpus. This corpus comprises of a number of annotated documents, namely news articles. However, this annotating effort took place a few years ago, in 2016, and the standard for the quality of the annotations is questionable. The main reason for that is that the annotators, who did not have background in fields connected to linguistics and were annotating non-exclusively in their free time, were never guided on which documents to annotate, and did not follow rigorous guidelines (even though they were trained for the task with an introduction to the topic and detailed instructions on how to annotate).

The result was a somewhat unbalanced corpus, as will be reported in Section 5.7.2, with several documents annotated by only one annotator, a few annotated by more than one, and even less annotated by more than two. And in every case the annotations may differ greatly, even though they show the same argument, simply because there were no official and strict guidelines for annotating.

Unfortunately, the unruly annotation process resulted in many documents being annotated only once, which leaves us with only smaller numbers of documents and annotations to run these metrics. The structure of these annotations' json was different from the structure currently used to create annotation json files in the ArgMine platform. While both follow the Argument Interchange Format, the annotations in the ArgMine corpus did not have any information on the ranges of the text selected for the text nodes. That was the most important part of the restructuring that was necessary before we could run the evaluation metrics as IAA on these corpus. This factor did not influence the more traditional IAA metrics, namely Cohen's Kappa and Krippendorff's alpha.

### 5.7.2 Results

In this corpus, we were able to run the metrics on 28 documents, adding up to 13434 tokens. At approximately 2.32 annotations per document, we ran the metrics through 65 annotations. We should mention again that the evaluation metrics grade based on the assumption that one of the annotations is correct, thus making the grade between the same two annotations vary according to which one is considered correct and forcing us to grade each pair of annotations twice. On the other hand, the more classic IAA metrics do not vary between two annotations and only have to be calculated once for each pair.

Furthermore, for the evaluation metrics used as IAA it is necessary to report what weights we used to calculate the corpus score. For the relaxed string matching in the F1-Based metric we used a threshold of 75% to match the nodes. For the Relation-Aware-F1-Based metric we used the



## Inter-Annotator Agreement

values in Table 4.1, as shown in that section’s example. And for the UCP Porto metric we used the values in Table 4.3 and Table 4.4, also as used in the example in their corresponding section. With this, we obtained the results shown in Table 5.2:

Metric	Result
Krippendorff’s Alpha	0.336
Cohen’s Kappa	0.638
F1 score	0.359
Relation-Aware-F1-Based metric as IAA	0.385
UCP Porto metric as IAA	0.289

Table 5.2: Results of IAA metrics on the ArgMine corpus

As we can see from Table 5.2, the overall scores for this corpus are not very high: The metric showing the highest score is Cohen’s Kappa, at approximately 0.64, while the one with the lowest score is the UCP Porto metric as IAA, at roughly 0.29. Considering the criticism on Cohen’s kappa, namely in [Kri04], which states that Cohen’s kappa can actually count disagreement as agreement when a bias is present, it would not be wise to say that this corpus is reliable just based on the Kappa measure. Instead we must conclude the opposite, because all the other metrics are below the 0.40 mark, which indicates low agreement and low reproducibility [MG97]. And that means that this corpus is not very reliable for machine learning purposes.

Furthermore, it is unsurprising that the UCP Porto measured the lowest IAA, given that it takes a deeper look at the relations between the nodes while the first four only look at the text in the nodes. It is also expected that the Relation-Aware-F1-Based metric measured a slightly higher IAA than the F1 score, because it will take the extra step of merging and matching nodes that the F1 metric might have disregarded.

## Inter-Annotator Agreement

## Chapter 6

# Conclusion

Argumentation Mining is a field of Text Mining that aims at the automatic extraction of arguments from natural language text. This is a difficult task, given the ambiguity of natural language text that is brought about by different writing styles, cross referencing across a text, the need for context and real world knowledge that may not be contained within the text, etc.

But the budding field of Argumentation Mining is not challenged solely by the complexity of its goal. Perhaps because it is still in its infancy, Argumentation Mining faces issues such as:

- Lack of quality input data – it is difficult to find corpora of annotated argumentative texts that can be used as input for the machine learning process. The difficulties range from difference in language, in type and genre of text, in argumentation models, in annotation formats, among others. It is especially difficult to find quality corpora which is not in English. In fact, to the best of our knowledge, the ArgMine corpus used during this thesis project is the only resource of its kind in Portuguese;
- Lack of a widely accepted way to evaluate the quality of the input data – the scientific community collectively challenges the inter-annotator agreement metrics used to report the reliability of annotated corpora. However, those same metrics are extensively used for this purpose because there is no better alternative;
- Lack of standard metrics for reporting inter-annotator agreement – Tying in with the last point comes the fact that it is up to each researcher to decide which metric to use to report their results. This makes it even more difficult to compare corpora and the different annotation processes, which also hampers the advance in finding a reliable way to annotate corpora;

In an attempt to deal with these issues, we presented the ArgMine platform and its annotator tool. We also presented some evaluation metrics and the concept that led us to use them as IAA metrics.

## Conclusion

The ArgMine platform and its annotator aim to facilitate the annotation process and the management of annotation projects. Making annotation easier and integrating it in a project has the potential of encouraging people to move from manual annotation into annotating in the ArgMine Platform. In turn, this will hopefully increase the number of annotated corpora available to serve as input in the machine learning process, provided that the corpora are reliable. Which leads us to the importance of the evaluation metrics. We introduced some evaluation metrics in an effort to:

- Further motivate users into annotating in the ArgMine annotator tool – The opportunity to have annotations automatically graded will drive annotators to annotate more in the ArgMine platform because they can get instant feedback annotations they submit. Furthermore, it makes the platform a very attractive tool to be used in classes where annotating documents is a requirement, as it would save professors a lot of legwork when they could simply check if they agree with the grade or not. This is especially true since the metrics are customizable;
- More consistently and comprehensively calculate the reliability of a corpus – we also present these evaluation metrics as IAA metrics because they are more comprehensive than the current widely used IAA metrics in the sense that they take into account the relations between the nodes in an annotation graph rather than only the annotated text. They can also be more consistent because they can be used in a wide range of scenarios and models, whereas with the usual IAA metrics there is usually a more adequate metric for a corpus and another one for another corpus. This break in consistency makes the corpora difficult to compare.

We succeeded in implementing some new evaluation metrics that encompass all aspects of the annotations, and we also used these same metrics to calculate the inter-annotator agreement of a corpus. This leads to a more well-rounded representation of agreement than the typical agreement metrics that are widely used in research across the field.

Furthermore, we also present an annotating platform with a strong focus on project management and an annotator tool that can provide a certain amount a freedom. Two things which have the potential to drive people to use the platform and provide us with the much needed resource that is an annotated argumentative corpus (in Portuguese or even otherwise).

### 6.1 New Insights

While working on this thesis project, we came to experience first-hand some of the most common problems that researchers in Argumentation Mining have to face. We also had the opportunity to work in collaboration with the professors at UCP Porto, and so get new perspectives from people who are not of the same academic background as us. All in all, we gained valuable insights in overcoming these problems and when collaborating with the professors.

Namely, dealing with an inconsistent body of research where terminologies are not standardized across the field, and dealing with the freedom that each research team has to choose their own

## Conclusion

argumentation schemes or inter-annotator agreement metrics, has brought to light the importance of being able to keep track of the research we read over time. These irregularities in the naming of key concepts, which nevertheless are normal in a rapidly budding field such as Argumentation Mining, made it difficult not only to compare research works but also to write our own work. Because we found that we had to make sure that the way that each author employed each term was consistent with our idea and employment of the term, so as not to create confusion in our own dissertation.

We believe that all this confusion is slowing down the growth of Argumentation Mining, and the extra effort it takes for a new researcher to familiarize themselves with the terminology in the field may even keep them from actually researching this topic.

In addition to this, we also felt the difficulties of not having plentiful resources to carry on the research as we would have wanted. In this case, this pertains to the ArgMine corpus. Although it is a valuable resource and, to the best of our knowledge, the only one of its kind in Portuguese, it is still a fairly small and possibly unreliable corpus. Nevertheless, we presented our results as we would have done with a bigger and more high-quality corpus.

Finally, when working in collaboration with the professors at UCP Porto, we found that their priorities were different that we would have thought, and, understandably, completely different from ours. Even though both parts involved in the collaboration came from academic backgrounds, the prime concerns of the professors at UCP Porto were the project management features and the online annotator tool. This is natural, as those are the aspects that can bring them the most advantages and help them on their daily work. What we came to realise was that we should use these features in which they show more interest as tools in driving them to make more use of the platform, and migrate from annotating on paper into annotating online. And this applies not only to the professors at UCP Porto, but also other users that may want to create annotating projects online that they can manage with ease.

## 6.2 Future Work

The obvious next steps in continuing this thesis project's work are related to the continuous improvement of the ArgMine platform and the evaluation metrics. More specifically:

- Allow users to import and export annotations to their ArgMine projects – this improvement could also lead to an increase of use of the ArgMine platform and provide the ArgMine project with more input data to use;
- Improve the visual representation of the annotation process – the annotator tool is already focused on visually representing annotations, but it does have room for improvement. For example, showing the user what parts of the text have already been annotated;
- Provide help when using the annotator – by showing tips while the user is annotating, we could make the annotator more didactic and raise the agreement of the corpora. However,

## Conclusion

this is a sensitive topic, as we will have to find ways to give tips to the user that would not create biases in the annotations;

- Improve the evaluation metrics – as mentioned in Chapter 4, there are some possible improvements to be made in the evaluation metrics;
- Create new evaluation metrics – considering the newly implemented metric's improvements to be made and their shortcomings, there is also the possibility of creating new metrics to approach the evaluation of annotation in different ways.

# References

- [All19] Mike Allen. *The SAGE Encyclopedia of Communication Research Methods*. SAGE Publishing, February 2019.
- [AP08] R. Artstein and M. Poesio. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–96, 2008.
- [Art17] Ron Artstein. Inter-annotator agreement. In *Handbook of linguistic annotation*, pages 297–313. Springer, 2017.
- [AVL14] Jean-Yves Antoine, Jeanne Villaneau, and Anaïs Lefevre. Weighted krippendorff’s alpha is a more reliable metrics for multi-coders ordinal annotations: experimental studies on emotion, opinion and coreference annotation. In *EACL 2014*, pages 10–p, 2014.
- [BAG54] Edward M Bennett, R Alpert, and AC Goldstein. Communications through limited-response questioning. *Public Opinion Quarterly*, 18(3):303–308, 1954.
- [BBC93] Ted Byrt, Janet Bishop, and John B. Carlin. Bias, prevalence and kappa. *Journal of Clinical Epidemiology*, 46(5):423–429, May 1993.
- [BCMS99] Mousumi Banerjee, Michelle Capozzoli, Laura McSweeney, and Debajyoti Sinha. Beyond kappa: A review of interrater agreement measures. *Canadian journal of statistics*, 27(1):3–23, 1999.
- [BEF<sup>+</sup>06] Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. The salsa corpus: a german corpus resource for lexical semantics. In *LREC*, 2006.
- [BH08] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. The MIT Press, 2008.
- [Bra00] Thorsten Brants. Inter-annotator agreement for a german newspaper corpus. In *LREC 2000*, 2000.
- [Bud11] Katarzyna Budzynska. Araucaria-pl: Software for teaching argumentation theory. In *Proceedings of the Third International Congress Conference on Tools for Teaching Logic, TICTTL’11*, pages 30–37, Berlin, Heidelberg, 2011. Springer-Verlag.
- [Col02] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

## REFERENCES

- [CSH18] C. Chen, M. Song, and G. E. Heo. A scalable and adaptive method for finding semantically equivalent cue words of uncertainty. *Journal of Informetrics*, 12(1):158–180, 2018.
- [CV18] Elena Cabrio and Serena Villata. Five Years of Argument Mining: a Data-driven Analysis. In *IJCAI*, 2018.
- [CWB<sup>+</sup>11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.
- [DLBR16a] Rory Duthie, John Lawrence, Katarzyna Budzynska, and Chris Reed. The cass technique for evaluating the performance of argument mining. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 40–49, 2016.
- [DLBR16b] Rory Duthie, John Lawrence, Katarzyna Budzynska, and Chris Reed. The cass technique for evaluating the performance of argument mining. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 40–49, 2016.
- [Fre91] J.B. Freeman. *Dialectics and the Macrostructure of Arguments: A Theory of Argument Structure*. Pragmatics and Discourse Analysis Series. Foris Publications, 1991.
- [G<sup>+</sup>02] Kilem Gwet et al. Inter-rater reliability: dependency on trait prevalence and marginal homogeneity. *Statistical Methods for Inter-Rater Reliability Assessment Series*, 2002.
- [GHH04] Claire Grover, Ben Hachey, and Ian Hughson. The holj corpus. supporting summarisation of legal texts. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora*, 2004.
- [GLPK14] Theodosios Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. Argument extraction from news, blogs, and social media. In *Hellenic Conference on Artificial Intelligence*, pages 287–299. Springer, 2014.
- [GNP16] Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. Unsupervised text segmentation using semantic relatedness graphs. In *Association for Computational Linguistics*. Association for Computational Linguistics, 2016.
- [Gwe02] Kilem Gwet. Kappa statistic is not satisfactory for assessing the extent of agreement between raters. *Statistical methods for inter-rater reliability assessment*, 2002.
- [Gwe11] Kilem L Gwet. On the krippendorff’s alpha coefficient. *Manuscript submitted for publication*, 2011.
- [HEKG14] Ivan Habernal, Judith Eckle-Kohler, and Iryna Gurevych. Argumentation mining on the web from information seeking perspective. In *ArgNLP*, 2014.
- [HG15] Ivan Habernal and Iryna Gurevych. Exploiting debate portals for semi-supervised argumentation mining in user-generated web discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015. page 2127.



## REFERENCES

- [HJ17] Yufang Hou and Charles Jochim. Argument Relation Classification Using a Joint Inference Model. In *Proceedings of the 4th Workshop on Argument Mining*, pages 60–66, 2017.
- [HR05] George Hripcsak and Adam S Rothschild. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298, 2005.
- [KPK16] Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. Clarin-el web-based annotation tool. In *InLREC 2016*, 2016.
- [Kri80] Klaus Krippendorff. *Content Analysis: An Introduction to Its Methodology (Context Series)*. Sage Publications, 1980.
- [Kri04] Klaus Krippendorff. Reliability in content analysis. *Human communication research*, 30(3):411–433, 2004.
- [LBG<sup>+</sup>18] Ran Levy, Ben Bogin, Shai Gretz, Ranit Aharonov, and Noam Slonim. Towards an argumentative content search engine using weak supervision. In *Proceedings of the 27th International Conference on Computational Linguistics*, 2018.
- [LBH<sup>+</sup>14] Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500, 2014.
- [Lev66] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, 1966.
- [LGBR18] Nancy L. Green, Michael Branon, and Luke Roosje. Argument schemes and visualization software for critical thinking about international politics. *Argument & Computation*, 10:1–13, 10 2018.
- [LGP18] Anne Lauscher, Goran Glavaš, and Simone Paolo Ponzetto. An argument-annotated corpus of scientific publications. In *Proceedings of the 5th Workshop on Argument Mining*, pages 40–46, 2018.
- [LT15] Marco Lippi and Paolo Torroni. Argument mining: A machine learning perspective. In *International Workshop on Theorie and Applications of Formal Argumentation*, pages 163–176. Springer, 2015.
- [MB16] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*, 2016.
- [MG97] Annette M. Green. Kappa statistics for multiple raters using categorical classifications. In *Proceedings of the Twenty-Second Annual Conference of SAS Users Group*, January 1997.
- [MI09] Raquel Mochales and Aagje Ieven. Creating an argumentation corpus: do theories apply to real arguments?: a case study on the legal argumentation of the echr. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 21–30. ACM, 2009.

## REFERENCES

- [MM11] Raquel Mochales and Marie-Francine Moens. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22, 2011.
- [Mor19] Katsuhide Morio, Gaku Fujita. On the Role of Syntactic Graph Convolutions for Identifying and Classifying Argument Components. *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019), Poster, Honolulu, Hawaii*, 2019. page 1.
- [MRS10] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.
- [MSK<sup>+</sup>18] Elena Musi, Manfred Stede, Leonard Kriese, Smaranda Muresan, and Andrea Rocci. A multi-layer annotated corpus of argumentative text: From argument schemes to discourse relations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.
- [NL16] Huy Nguyen and Diane Litman. Context-aware Argumentative Relation Mining. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1127–1137, 2016.
- [NLP<sup>+</sup>18] Benjamin Nye, Junyi Jessy Li, Roma Patel, Yinfei Yang, Iain J Marshall, Ani Nenkova, and Byron C Wallace. A corpus with multi-level annotations of patients, interventions and outcomes to support language processing for medical literature. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2018, page 197. NIH Public Access, 2018.
- [OT69] Lucie Olbrechts-Tyteca. The new rhetoric: a treatise on argumentation. *Notre Dame, London*, 117, 1969.
- [PC19] J. Park and C. Cardie. A of erulemaking user comments for measuring evaluability of arguments. In *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, pages 1623–1628, 2019.
- [Pet14] Georgios Petasis. Annotating arguments: The nomad collaborative annotation tool. In *LREC 2014*, 2014.
- [Pow15] David MW Powers. What the f-measure doesn’t measure: Features, flaws, fallacies and fixes. *arXiv preprint arXiv:1503.06410*, 2015.
- [PS15a] Andreas Peldszus and Manfred Stede. An annotated corpus of argumentative micro-texts. In *Proceedings of the First Conference on Argumentation*, 2015.
- [PS15b] Andreas Peldszus and Manfred Stede. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948, 2015.
- [REE14] Mathilde JANIER John LAWRENCE Chris REED. Ova+: An argument analysis interface. In *Computational Models of Argument: Proceedings of COMMA*, volume 266, page 463, 2014.
- [RLC17] Gil Rocha and Henrique Lopes Cardoso. Towards a relation-based argument extraction model for argumentation mining. In Nathalie Camelin, Yannick Estève, and

## REFERENCES

- Carlos Martín-Vide, editors, *Statistical Language and Speech Processing: 5th International Conference, SLSP 2017, Le Mans, France, October 23–25*, pages 94–105, Cham, 2017. Springer International Publishing.
- [RLCT16] Gil Rocha, Henrique Lopes Cardoso, and Jorge Teixeira. ArgMine: A Framework for Argumentation Mining. In *Computational Processing of the Portuguese Language - 12th International Conference, PROPOR 2016, Student Research Workshop, Tomar, Portugal, July 13-15, 2016*.
- [RM99] Lance A Ramshaw and Mitchell P Marcus. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer, 1999.
- [Roc16] Gil Rocha. Argmine: Argumentation mining from text. Master’s thesis, University of Porto, Porto, Portugal, 2016.
- [RSLCG18] Gil Rocha, Christian Stab, Henrique Lopes Cardoso, and Iryna Gurevych. Cross-lingual argumentative relation identification: from English to Portuguese. In *Proceedings of the 5th Workshop on Argument Mining*, pages 144–154, Brussels, Belgium, November 2018. Association for Computational Linguistics.
- [Sco55] William A Scott. Reliability of content analysis: The case of nominal scale coding. *Public opinion quarterly*, pages 321–325, 1955.
- [SED<sup>+</sup>18] Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. Multi-Task Learning for Argumentation Mining in Low-Resource Settings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 35–41, 2018.
- [SG17] Christian Stab and Iryna Gurevych. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659, 2017. page 636.
- [Tou58] Stephen E Toulmin. *The Uses of Argument*. Cambridge University Press, 1958.
- [VEG09] Frans H Van Eemeren and Bart Garssen. Problems of argumentation: An introduction. *Pondering on problems of argumentation. Twenty essays on theoretical issues*, ed. FH van Eemeren, and B. Garssen, xi–xxi. New York: Springer, 2009.
- [Wal13] D. Walton. *Argumentation Schemes for Presumptive Reasoning*. Taylor & Francis, 2013.
- [WAMP14] Lars Wissler, Mohammed Almashraee, Dagmar Monett, and Adrian Paschke. The gold standard in corpus annotation. In *IEEE GSC*, June 2014.
- [Wha41] R. Whately. *Elements of rhetoric*. Longmans, Green, Reader, and Dyer, 1841.
- [WMGA14] Nina Wacholder, Smaranda Muresan, Debanjan Ghosh, and Mark Aakhus. Annotating Multiparty Discourse: Challenges for Agreement Metrics. In *Proceedings of LAW VIII - The 8th Linguistic Annotation Workshop*, pages 120–128. Association for Computational Linguistics and Dublin City University, 2014.

## REFERENCES

- [Xie] Qingshu Xie. Agree or disagree? a demonstration of an alternative statistic to Cohen's kappa for measuring the extent and reliability of agreement between observers. unpublished.
- [XYLL14] Junyi Xu, Li Yao, Le Li, and Jinyang Li. Multi-Agent Joint Learning from Argumentation. In Longbing Cao, Yifeng Zeng, Andreas L. Symeonidis, Vladimir Gorodetsky, Jörg P. Müller, and Philip S. Yu, editors, *Agents and Data Mining Interaction*, pages 14–25, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- [ZHHL17] Fan Zhang, Homa B Hashemi, Rebecca Hwa, and Diane Litman. A corpus of annotated revisions for studying argumentative writing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1568–1578, 2017.
- [ZLD13] Xinshu Zhao, Jun S Liu, and Ke Deng. Assumptions behind intercoder reliability indices. *Annals of the International Communication Association*, 36(1):419–480, 2013.
- [ZZF17] Meishan Zhang, Yue Zhang, and Guohong Fu. End-to-End Neural Relation Extraction with Global Optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1730–1740, 2017.