Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science and Media

Mining Relevant Arguments At Web Scale

Master's Thesis

Yamen Ajjour
Born Aug. 28, 1989 in Krakow

Matriculation Number 115082

First Referee: Prof.Dr. Benno Stein
Second Referee: Dr. Andreas Jakoby

Submission date: January 9, 2017

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, January 9, 2017

. . . .
Yamen Ajjour

Abstract

With the goal of developing an argument search engine, a novel approach to estimate the relevance of the arguments on the Web is proposed. Argument relevance is a quality criterion which indicates how much an argument contributes to a specific discussion. We model an argument as a conclusion and a set of premises which support or attack the conclusion, all called argument units. The approach relies on constructing an argument graph which represents the arguments and their relations on the Web. PageRank is then used to score the argument units in the graph according to the attention they get as premises. The relevance of an argument is then estimated using the PageRank scores of their units. We use AIFdb to construct a ground-truth argument graph and apply our approach to rank the arguments in it. Then, we evaluate the rankings of a set of arguments by comparing them to rankings which are generated by experts. The experiment reveals a positive Kendall's correlation of 0.28. Starting from these results, we bring our approach to the Web by developing a cross-domain argument mining approach. A cross-domain argument web corpus is constructed to represent three different web domains. A domain is any set of documents on the Web which share the same source or the topic. The corpus is used to develop and evaluate the ability of the cross-domain argument mining approach to carry their knowledge over different collections of documents on the Web. To evaluate the ability of the cross-domain approach to generalize over domains, We develop an in-domain approach which is trained and tested on the same corpus. The effectiveness of both approaches is found to be relatively close with regard to a minority baseline but indicates the need for improvements.

Contents

Chapter 1

Introduction

With the rapid expansion of the content on the web, it became an invaluable source of information for billions of users around the globe. Contemporary search engines continuously mine the web and provide the users with the most relevant information for their queries. As the content of the web grows larger, the information needs of search engine users expand both in complexity and variety, making the task of mining the web even more complicated. An information need can be seen as a topic about which a user desires to know more [27].

Social media and debate forums enabled humans to discuss and debate about controversial ideas, adding new type of content to the web known as argumentative content. *Argumentation* can be defined as a process whereby arguments are constructed, exchanged and evaluated in the light of their interactions with other arguments. An *argument* is a set of *premises* advanced for or against some *conclusion* (we give a general term *argument unit* for a premise or a conclusion). People engage in argumentation as an integral part of their daily-life, usually to defend and exchange opinions. One type of information need, which is highly required in this process, is the user's wish to find relevant arguments to a conclusion or hypothesis he/she has, such as "we should ban homeworks". The major challenge would be to mine the Web for arguments and then to return to the user the subset of arguments which are more likely to convince him/her of the acceptance or rejection of the conclusion he/she is searching for. We refer to a search engine which will provide relevant arguments for a user's hypothesis as *argument search engine.*

In recent years, we experienced the emergence of a new research area called *argument mining* which aims to automatically detect the arguments of a document, their structure and their relations. In a typical argument mining task, a set of documents referred to as *corpus* is typically annotated manually by humans to constitute what we call *ground truth.* Later, supervised machine

learning classifiers are trained and tested on the annotated corpora and then applied in real world.
Machine learning is a field of computer science that enables programs to improve over some task from experience with respect to an effectiveness measure. Effectiveness quantifies the extent of which the output of an input instance is correct in comparison to the annotated corpus. Usually, an input instance is a span of text and the output is a class from a predefined set of classes (e.g. argumentative and non-argumentative). A program which is trained on an annotated corpus to predict the class of an input instance is called a *classifier*. To assess the effectiveness of such a classifier, precision and recall measures are used. For a specific class, *precision p* quantifies the ratio of the input instances which are correctly classified with that class to the all classified input instances
.
Earlier research on argument mining concentrated on classification of textual segments (e.g. sentences) into argumentative and non-argumentative ([30] and [26]), classification of the argumentative type of the argument units into premise or conclusion ([23], [32] and [38]), and the classification of the argumentative relation between each pair of argument units into support, attack or non-argumentatively related ([32], [41] and [33]). An *argumentative relation* refers to a pair of argument units (a premise and a conclusion) and indicates that the premise is supporting or attacking the conclusion. Each classification task is typically based on a single corpus in a k-fold cross-validation evaluation setting, where the corpus is divided into k parts and the effectiveness of the classifier is evaluated on each fold after being trained on the other k-1 folds, and then the total effectiveness measure is averaged over all folds.
The characteristics that distinguish good arguments have been investigated since humans started to think how they can effectively persuade their communities of a certain idea. Early work on persuasive argumentation goes back to Aristotle [8] where he introduced the principles of successful persuasion in public speaking. Further research has been done by Toulmin to introduce formal models for good arguments [43], representing an argument as a conclusion, supporting premises, warrants which justify the inference from the premises to the conclusion, backing that serves to support the warrant and counter argument statement known as rebuttal.
Recently, the criteria of argument quality have been analyzed and divided into: argument acceptability, argument relevance, and argument sufficiency [10]. According to these criteria, a good argument should have premises that are singly or in combination *relevant* to the conclusion, i.e. contribute to the acceptance or the rejection of the conclusion, *acceptable* for a reasonable person and *sufficient* as a ground to draw the conclusion. While most of these criteria are studied theoretically, only argument acceptability has been mod-

eled computationally. We call the automatic evaluation of argument quality *argument analysis*.
Dung [15] defined a mathematical representation of the acceptability of arguments by considering an argument accepted if all the arguments attacking it are rejected and considering it rejected if it has at least an attacking argument, which is accepted. It is easy to notice that argument acceptability is orthogonal to argument relevance, because an argument can be accepted but still not relevant to a proposition a user has.
Motivated by the idea of an argument search engine, in my thesis I investigated the following research questions: ***how to model argument relevance computationally on the Web?*** Argument relevance is the most important quality criteria for an argument search engine, since it matches the concept of relevance in regular web pages. In [45] we suggested a framework to assess the relevance of arguments by using PageRank. Originally, PageRank is used in information retrieval to assess the objective relevance of a web page, by using the number and quality of the referring web pages [31]. Analogously and based on argument attack/support relations, we estimated the relevance of an argument, hypothesizing that a conclusion is more relevant, the more it is used as premise by other arguments.
To evaluate this idea, I built an argument graph from existing argument maps on AIFdb [25]. An argument map is a set of arguments and the attack/support relations between. AIFdb is a database which allows for the storage and retrieval of arguments in a standard format. It has 50,000 argument units, 57 corpora and about 10,000 argument maps.
The constructed argument graph contains all the arguments on AIFdb and the attack/support relations between them, after merging the duplicate argument units across different argument maps. Next, I used PageRank to estimate the arguments' relevance in the constructed graph. Later on, we compared the rankings produced by PageRank with rankings produced by experts. We found that there is a positive correlation between the annotated ranking and PageRank scores, which outperforms several intuitive baselines.
These positive results in assessing argument relevance motivated me to realize our approach on the Web. The Web, characterized by its scale and heterogeneity, constituted a challenge to the existing argument mining approaches. The main reason is that most of the developed argument mining classifiers are usually tested and trained on one corpus in a supervised settings. Consequently, they tend to capture domain-specific properties which makes it ineffective on other domains. We call this difficulty, which an argument mining classifier faces, the *domain-effect*. A *domain* is a set of documents that share a common characteristic (e.g topic or source). Due to its increasing growth in terms of size and content in the last two decades, the Web covers now a plethora of

domains. This leads us to the second research question of ***how to domain-robustly mine the Web for arguments ?*** *Domain-robustness* of a classifier can be understood as the ability of a classifier to generalize over domains. To guarantee domain-robustness while developing our argument mining approach on the Web, I conducted a cross-domain experiment whereby the effectiveness of our argument mining classifiers was tested on one corpus, after being trained on the other corpora. To evaluate the effectiveness of our classifiers in guaranteeing domain-robustness, we carried out an in-domain experiment where we trained and tested our classifiers on the same corpus, thus excluding the domain-effect. The corpora which were chosen for the experiments include Araucaria [39], WebDiscourse [19] and AIFdb (without Araucaria) [25].
The three corpora were annotated with argument units and argumentative relations and constituted our *cross-domain argument web corpus.* We labeled the sentences in each corpora with the type argumentative and non-argumentative, representing a real or a fake argument unit. Similarly, we labeled ordered pairs of argument units in the corpora with related or non-related indicating a holding or a nonexistent argumentative relation between the pair. In the cross-domain and in-domain experiments, we developed classifiers to distinguish whether a sentence is argumentative or non-argumentative. We call such a classifier an *argument unit classifier.* In addition, we developed classifiers to identify whether an ordered pair of argument units are related or no-related. We call such a classifier an *argumentative relation classifier.* The count of the sentences in our corpus which are labeled with the type argumentative is smaller in size than those labeled with the type non-argumentative which makes it hard for a classifier to distinguish them. Therefore, we focused while evaluating our argument unit classifiers on the argumentative class. For the same reason, we concentrated on the related class in the evaluation of the classifiers which will be developed to predict the class of an argumentative relation. We refer to each class as the *positive* class.

In the cross-domain experiment, our argument unit classifiers achieved a positive precision of 0.43, in comparison to 0.56 in the in-domain experiment. Additionally, Our argumentative relation classifiers accomplished a precision of 0.33 in the cross-domain experiment in comparison to a precision of 0.48 in the in-domain experiment. The relatively close effectiveness of the classifiers in both experiments indicates a positive result of for our approach to develop a cross-domain argument mining approach. Nevertheless, it indicates the inadequacy of the existing approaches, which rely solely on existing corpora for training classifiers, to extract the arguments on the Web.

This thesis is structured as follows: In Chapter 2 we give an overview of machine learning and argument modeling. Additionally, we list the research related to argument analysis and argument mining. In Chapter 3 we introduce our approach to construct an argument graph from the Web and to estimate argument relevance and elaborate on the process in which we built an argument graph from AIFdb for the evaluation of our PageRank for argument relevance. In Chapter 4 we explain how we modeled arguments on the Web and the process of creating our cross-domain argument web corpus that we used later on to develop our domain-robust argument mining approach. Finally, in Chapter 5 we summarize the content of this thesis and give an outlook on further work.

Chapter 2

Background

Argumentation which is a verbal activity for which the goal consists of convincing the listener or the reader of the acceptability of a standpoint by means of a constellation of propositions justifying or refuting the proposition expressed in the standpoint [44], has been studied since the early work of Aristotle. It has been heavily investigated from different point of views, such as logic, psychology, philosophy, linguistics and computer science.

Recent work on argumentation in computational linguistics is concentrated on argument mining, which aims at the automatic extraction of arguments and their structure from a document. The emergence of this field was mainly motivated by previous work by Toulmin [43] on the modeling of persuasive arguments. An *argument model* is an abstraction from the language level to a more formal level, where the constituting units of an argument, their relations and their types are described. An argument model is crucial for the mining of arguments since it specifies the granularity of the constituting units (e.g. on sentence level), the types of the argument units and their relations. An approach for argument mining extracts the argument units and the relations of an argument as specified by the used argument model. These specifications are also essential for an analysis of the quality of an argument and especially for this work since, as described in Chapter 1, our approach relies on a structural analysis of the arguments to assess their relevance.

The influential work of Toulmin [43] was followed by more research in the direction of finding what are the criteria of a good argument, such as [10] and [17]. In addition, the advancements in technology in the 20th century and the wider understanding and developments in machine learning contributed to the recent progress in argument mining.

Machine learning, is a field which enables algorithms to improve over one task with experience with regard to an effectiveness measure. In argument mining, such a task can be the classification of the type of an argument unit, and an experience for this task can be an example of an argument unit labeled with its correct type. To use machine learning algorithms in argument mining, a *corpus*, which is a set of documents, is usually utilized to provide the machine learning algorithm with the required set of experiences to learn the task. A corpus is usually created by letting experts, students or normal humans manually locate the arguments in a set of documents. This is usually done by creating annotations to specify the argument units and the relations between them according to a predefined argument model. An annotation is a span of text labeled with some meta data, e.g. the type of an argument unit. Later on, the extent of consensus between the annotators over the annotations is calculated since different annotators can annotate the same segment with a different type value. A low consensus indicates the difficulty of the task and consequently a lower effectiveness by any machine learning algorithm in learning the task is expected. In this chapter, I will first introduce a brief overview of machine learning. Later on, I will briefly discuss existing argument models and state their main differences to our suggested argument model. Next, I will introduce a brief background about *argument analysis* which aims at the automatic evaluation of argument quality. Subsequently, I will summarize related work to argument mining, highlighting the main motivation and the differences to my approach. Finally, I will report on existing corpora, which are annotated with arguments to give the user an idea why and how I created a new corpus as described in Chapter 4.

2.1 Overview of Machine Learning

Most of the research done in the field of argument mining is based on machine learning algorithms. *Machine learning* is a field of computer science that enables algorithms (classifiers) to improve over some task (learn) from experience with respect to an effectiveness measure [28]. In contrary to classification where the output predicted by a classifier for an input vector is discrete, in *Regression* the predicted output for a given input vector as a continuous value. Machine learning can be divided into supervised and unsupervised learning. As will be explained in a later section, a typical task in argument mining is to classify a text segment into a class from a predefined set of classes (e.g. premise or conclusion). For this task, we call the text segment as *input instance.* In *unsupervised learning*, a classifier obtains the required knowledge to perform such a task by discovering patterns in given unlabeled input instances. On the other hand in *supervised learning*, a classifier is confronted with a set of input instances labeled with different classes to acquire that knowledge. Both approaches aim to generalize over a given set of input instances and their associated classes, called *ground truth*, to be able to classify unseen input instances into the correct output. To simulate this contrast in supervised learning, the given input instances and class pairs are divided into training and testing sets. A classifier has access to the class of the input instances only in the training set, while its effectiveness is measured on the training set by comparing its output to the associated class of the input instance in the ground truth. The *effectiveness* of a classifier quantifies its ability to predict the correct class of an input instance with regard to the ground truth.

Let us assume that a set of n input instances labeled with k class $D = (x_1, y_1), ...(x_n, y_n)$ is given where x_i represents an input instance and y_i represents its class. let D_i be the set of the input instances labeled with the ith class. A supervised classifier depends on two functions to perform its task:

- *model formation function*: a function which projects the input instances into a *model space* where interesting aspects of the input instances are quantified by means of a *computer*. Notice that a computer doesn't denote a regular machine but an abstract mathematical function that computes interesting features of such an object.These easily measurable properties called *features* represents the dimensions of the model space. Features are particular instances of a *Feature Type*. A feature type for example can be bag-of-words where each feature means the frequency of a word. We call the projection of an input instance into the model space as *feature vector*.

- *model function*: which maps all the feature vectors in the model space to the all possible classes.

Depending on the classifier type, the model function is designed to assume one hypothesis from a hypothesis space. A hypothesis is a formal representation that maps a feature vector into its class. Usually, the *classifier type* states also how the hypothesis space should be searched in a way that guarantees high effectiveness and tractability. The classifier type which we will use in this work is Random Forest which is a set of Decision Tree classifiers. Decision Tree is a classifier type whose model function assumes a hypothesis that is designed as a tree of nodes. Each node represents a decision which forks to a successor node depending on the possible value of a specific feature. The leaves of the tree decides for the class of the input instance.

All of the classifiers developed in this thesis are limited to two classes (e.g. argumentative vs non-argumentative). Usually, one class is called the *positive* class and the other class is called the *negative* class, depending on which class

is of interest for us. Given a testing set, a classifier separates its instances into *positives* and *negatives*. We call all the positives which were labeled as positive in the ground truth as *true positives* (TP), while we call are all others as *false positives* (FP). Similarly, we call all negatives which were labeled as negative in the ground truth as *true negatives* (TN), while we call all other negatives as *false negatives* (FN).
To evaluate the effectiveness of a classifier, Precision and Recall measures are used. For a given positive class, *precision* p quantifies the ratio of positives that are classified correctly to all the classifier input instances, while *recall* r refers to the ratio of all positive instances that were correctly classified to the all positive instances in the ground truth. Formally :

$$p = \frac{|TP|}{(|TP| + |FP|)}$$

$$r = \frac{|TP|}{(|TP| + |FN|)}$$

Generally, achieving either high precision or high recall is very easy, e.g. perfect recall can be obtained by always producing the positive class. If both measures are desired their harmonic mean can be computed, called the *F1-score* (F).

$$F = \frac{2 \cdot p \cdot r}{p + r}$$

All these measures concentrate on one positive class, giving it more importance than the other negative class. When all classes are of equal importance, *accuracy* (a) is used. Accuracy estimates the ratio of correct decisions taken by the classifier to the all taken decisions.

$$a = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|}$$

Alternatively, F1-scores can be weighted over classes. $\mathsf{Micro\text{-}f1}$ weights the F1-scores of a class by the count of the input instances labeled with that class in the testing split. On the other hand, $\mathsf{Macro\text{-}f1}$ weights the F1-scores of all classes equally. Provided that F_i is the F1-score of the class i, the micro-averaged F1-score and the macro-averaged F1-score can be calculated as follows:

$$\text{Macro-f1} = \frac{\sum_{i=1}^{|D|} F_i}{k}$$

$$\text{Micro-f1} = \frac{\sum_{i=1}^{|D|} F_i \cdot |D_i|}{|D|}$$

Usually during evaluation, the given labeled input instances are divided in multiple splits into different training and testing sets. In each split, the testing sets are usually exclusive and equal in size, i.e. no common input instances exists among them. The effectiveness of a classifier is measured over the testing set for each split, after it is trained on its corresponding training set. This evaluation setting is usually called *k-fold cross-validation* where a fold stands for a testing set and k is the count of the splits. The used effectiveness measure is then averaged over all splits.

All the aforementioned effectiveness measures rely on a ground truth, usually created by humans, to assess the extent of which a classifier is able to perform a task in the real world. Given a ground truth, a baseline is usually used to assess the difficulty of the task and the novelty and the need for a sophisticated machine learning classifier. A *baseline* is a lower bound of effectiveness achieved by a trivial approach, such as counting the number of words in an input instance. *Token n-grams* represents the percentage of every 1-N possible sequences of tokens.

In Regression, however, the predicted value of an input instance is continuous, i.e. a rational number. Typically, the model function is fit over the ground truth which represents the relation of the input-output pairs. The fitting is done by minimizing the average error between the output predicted by the function and the output associated with the input in the ground truth. A regression algorithm then predicts an output e_i for a given input instance x_i. In regression, the effectiveness is estimated in terms of a mathematical error. *Mean Absolute Error* (MAE) measures the average distance between an algorithm's predicted output and the actual output in the ground truth. *Mean Squared Error* (MSE) measures the average square of the distance between an algorithm's score predictions and the actual scores in the ground truth. Provided a set of n input-output instances $D = (x_1, y_1), (x_2, y2)....(x_n, y_n)$:

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |y_i - e_i|$$

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - e_i)^2$$

2.2 Argument Modeling

As discussed in the introduction, we aim in this thesis at estimating the relevance of the arguments on the Web. The relevance of an argument represents the benefit it brings to a debate or a discussion. Argumentation occurs in everyday life between two or more sides who have different views on a controversial topic. The dialectical view of argumentation represents it as a process, whereby a proponent and an opponent exchange their conclusions and the reasons behind them over the topic. Another view on argumentation focuses on arguments as the *products* of this process.

To analyze the quality of arguments usually an argument model is employed. An *argument model* abstracts from the language level to a higher level where the main units of the argument, their relations and their types are described. The *type* of an argument unit specifies the way it contributes to the persuasive strength of the argument from which the argument unit is part of, e.g. premise. The argumentative relations between the argument units of an argument represents the interactions between them that drive the persuasion of the argument. According to the argument model, an argumentative relation has a *source* and a *target* which specifies the flow of the interaction between them. Additionally, an argumentative relation has a *type* that implies the role it plays in the persuasive strength of the argument. The argument model can specify the types of the source and the target for an argumentative relation with a specific type to hold. For example, a support relation can exist between a source with the type premise and a target with the type conclusion and not the other way around. We call the argument unit types, the argumentative relations and their types used in an argument the *argument structure.*
On the language level, an argument model may describe the argument units and the argumentative relations as annotation categories, labeled with their types as metadata. An *annotation* is a span of text that represent a specific concept and labeled with some meta data, such as a tag or a reference to another annotation. An *annotation category* is a class of annotations which denote the same concept, e.g. "Title". We call this technical representation of an argument model on the language level as *annotation scheme.* As we will see in Subsection 2.5, an annotation scheme is of high importance during the creation of a corpus since it provides the required technical details to annotate the document with arguments.
Toulmin [43] was the first who modeled arguments mainly to evaluate their quality. The influential model of Toulmin formed a comprehensive abstraction of arguments on which different models and simplifications were based. Toulmin's model covers both the view of the argument as an individual product and as a part of a dialectical process.
As shown Figure in 2.1, Toulmin [43] defined six argument unit's types:

- Data: the facts or evidence used to prove the argument

- Claim: the conclusion of the argument
- Warrant: the general hypothetical logical unit that justifies the inference from the data to the claim
- Qualifier: units that limit the strength of the argument or propose the conditions under which the argument is true
- Rebuttal: a counter-arguments or an argument unit that indicates circumstances when the general argument doesn't hold true
- Backing: an argument unit that supports the warrant (i.e. an argument that doesn't necessarily justify the conclusion but does support the acceptability of the warrant

The rebuttal is used usually to state an exception to the generalization introduced by the warrant. Typically, an author tries to anticipate potential critics against his argument and try to preempt it by explicitly mentioning exceptions where the argument doesn't hold. This can be seen as a reflection of the dialectical aspect of argumentation in Toulmin's model. Even though Toulmin's model can be seen as an ideal model to represent arguments, the arguments exchanged in everyday life rarely contain most of the units described in Toulmin's model. Usually, an author of an argument drops the warrant since it is implicit or doesn't mention a rebuttal since he/she couldn't formulate one. A more coverage of the units in Toulmin's model can be expected in more formal domains such as law or scientific text.

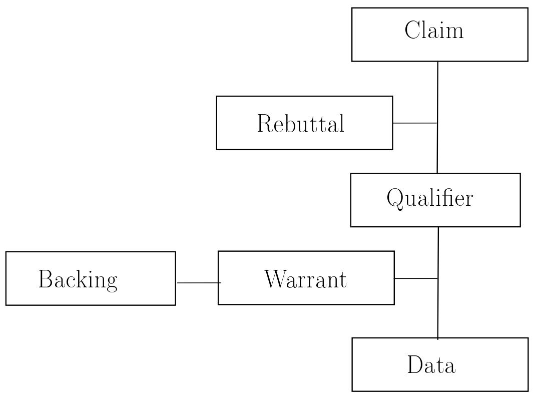


Figure 2.1: Toulmin's model

Since the focus of this thesis is to mine arguments on the Web and to analyze their relevance and owing to the heterogeneity of the Web, we opted for a simpler model as I will introduce in Subsection 4.3. In their work on annotating argument structure in the domain of essays, Stab and Gurevych [40] introduced a novel model to represents the arguments in argumentative essays. The argument model consists of argument units and the argumentative relations between them. In comparison to Toulmin's model, this model explicitly defines an annotation scheme which includes two annotation categories: argument unit and argument relation. For the argument units, the annotation scheme specifies the a granularity of the argument units in an essay which is any sequence of words. These technical details are important since Stab and Gurevych [40] used this argument model to create a corpus from a set of essays after manually annotating the arguments in the essays with the argument model. As shown in Figure 2.2, Stab and Gurevych [40] introduced three argument unit types:

- Premise: an evidence which underpins the validity of a conclusion
- Claim: A conclusion that is either true or false and shouldn't be accepted by the reader without additional support
- Major Claim: the author's stance with respect to the prompt against which the essay is written (also called thesis)

Compared to this model, our suggested model doesn't include a Major Claim argument unit type. The main reason is that while the author of an essay is inclined to state his thesis against the prompt, the argumentation which takes place on the Web doesn't necessarily be guided with a prompt. This makes it difficult to distinguish between a regular conclusion and a thesis in terms of generality. Another difference between our model and this model is that we define the granularity of an argument unit to be the sentence level. While this model's assumption of the argument units' granularity is more general, our decision not to choose a more fine-grained granularity is mainly for efficiency. In this work, we aim to mine arguments from the Web with the goal of assessing their relevance. Owing to the tremendous size of the Web, opting for a more fine-grained granularity like the clause level to define the boundaries of argument units in requires parsing all the documents on the Web, which is rather a computationally expensive task. Similar to our model, Stab and Gurevych [40] used argumentative relations which indicate either a support or an attack relation between either:

- a Premise and a Premise
- a Premise and a Claim
- a Premise and a Major Claim
- a Claim and a Major Claim

Argumentative relations constitute the main component of the structure of an argument. They models the inference or the refutation between the premises and the conclusion of an argument as a directed edge. Similarly, in our model we use argumentative relations to model the argument structure; however, we don't distinguish between an attack or a support relation. The main reason is that our approach estimates the relevance of an argument based on the amount of the involvement of its premises in different arguments regardless whether the premise was used in an attack or a support relation.

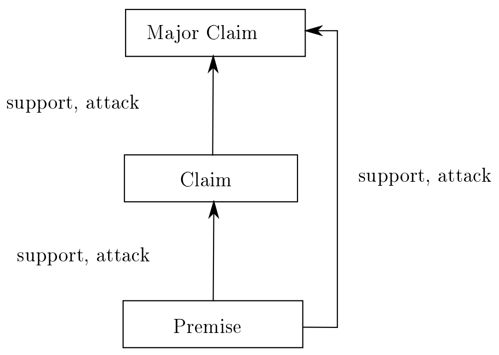


Figure 2.2: An argument model to represent arguments in the argumentative essays domain

2.3 Argument Analysis

What makes a good argument is a question which has been studied thoroughly in the field of logic and humanities. In his book [8], Aristotle introduced what is possibly persuasive in every given case. Motivated by the emergence of democracy, the main work of Aristotle on rhetoric concentrates on improving the ability of public speakers to convince their audience of a certain idea. His research focused on arousing and exploiting the emotions of the audience and studying the influence of a speaker's character. The theory introduced by Aristotle laid general rules about persuasion, concentrating more on the production of persuasive speech than a deep analysis of argumentation. Aristotle introduced a psychological analysis of human's emotions and character and how a speaker can exploit them to convince an audience with his stance on a certain topic.
The work of Aristotle is hard to model computationally since an explicit profiling of the speaker, his/her character, and his/her credibility lacks on the Web, owing to the anonymous nature of the web. For the same reason, it is difficult to identify a potential audience of an argument and their current emotional status. Additionally, a complete understanding of emotions and a computational analysis of it is still lacking. Aristotle's work on rhetoric, however, motivated recent studies on argument analysis.
Blair [10] for example, introduced three criteria of a good argument : *argument relevance*, *argument acceptability* and *argument sufficiency*. Damer [14] adds to these criteria the existence of a rebuttal. Argument sufficiency assesses whether an argument is based on enough amount of premises. For Argument relevance, Walton [46] differentiates between *probative relevance* and *dialectical relevance*. Probative relevance captures the contribution a set of premises brings to the acceptance or rejection of its conclusion. Whereas dialectical

relevance estimates how much does an argument contribute to settling a debate or a discussion. Argument acceptability, however, is concerned with the magnitude of consensus an argument achieves.
Even though the three quality criteria are well understood theoretically, no computational approach has been introduced to automatically assess them apart from the work of Dung [15] on argument acceptability. A computational approach to assess the quality criteria of arguments is needed to develop an argument search engine. The reason is that a search engine is expected to process a tremendous amount of data and an enormous number of queries in limited time. Therefore, an approach which is able to be implemented on computers is needed to process the large amount of data. A computational approach to assessing the quality of an argument should automatically retrieve the arguments on the Web that have enough support, widely accepted and relevant to the topic the user is arguing about.

Starting from a set of arguments and the attack/support relations among them, Dung [15] defined an argumentation framework which allows detecting which arguments are accepted. Roughly speaking, an argument is accepted, if all arguments attacking it are rejected, and it is rejected if it has at least an argument attacking it which is accepted. An argument which is not attacked at all is considered to be accepted. The work of Dung [15] was the first effort to introduce a computational model to assess a quality criterion of arguments. Cabrio and Villata [12] created a training set of arguments and their attack/support relation from Debatepedia [2] and used textual entailment to model support/attack relationships between the arguments. *Textual entailment* is a directional relation between two text segments which holds when the truth of the second text segment follows from the first. Cabrio and Villata [12] tried to identify accepted arguments on the set of arguments and their relations from Debatepedia, based on the mathematical models introduced by Dung [15]. First, they evaluated the effectiveness of EDITS(Edit Distance Textual Entailment Suite), which is an open-source software package for recognizing textual entailment [7]. They used it in modeling the relations between the arguments on debatepedia and achieved an accuracy of 0.69 on the training set they created. To evaluate the performance of EDITs on argument acceptability, they compared the accepted arguments, which are identified based on the gold-standard relations in the training set to the accepted arguments based on the argument relations classified by EDITs. The system achieved a precision of 0.74 and a recall of 0.76. Cabrio and Villata [12] intro Though argument acceptability is an important aspect of argument quality, it is orthogonal to argument relevance since an argument can be accepted, but still irrelevant to the proposition it is supporting or attacking.

In the domain of Essay Scoring, Persing and Ng [35] tried to model argument strength on essays, which he defines as the strength an essay makes for its thesis [35]. A thesis is the overall message of the whole essay. Persing and Ng [35] created a corpus of 1000 essays from International Corpus of Learner English (ICLE) [18]. The 1000 essays were written in response to 10 prompts, where a prompt is a proposition which the writers should either support or attack. Next, they let 6 out of 30 human annotators, who were most consistent with the expected scores, score the essay using a scoring rubric. The scoring rubric ranges from 1 to 4 by a difference of 0.5, where 4 is the best grade. This corpus is the first ground truth for the task of estimating argument strength of student essays..
After the creation of the annotated essay corpus, they used regression to automatically score the essays. They used syntactical, semantical and lexical features to map an input instance (here the essay) to the input space. In addition to the mentioned categories, the features Persing and Ng [35] used relied on the argument units and argument unit types annotated by the argument mining classifier created by Stab and Gurevych [41]. The classifier was trained on the created ICLE corpus after annotating the corpus heuristically with argument units. To evaluate the effectiveness of their approach, Persing and Ng [35] used Mean Absolute Error (MAE) and Mean Squared Error (MSE). In a five-fold cross-validation test, Persing and Ng [35] achieved an MAE of 0.392 and MSE of 0.244.
Compared to our approach to assessing argument relevance, the concept of argument strength attributes a complete essay and doesn't attribute an individual argument which makes it hard to be adapted in an argument search engine. Moreover, the approach they implemented is specifically tailored to argumentative essays domain. As a consequence, the effectiveness of their approach will most likely decrease severely when applied on the Web. The reason is that the argumentative content on the Web is very wide in terms of topics and platforms.
Habernal and Gurevych [20] introduced the concept of argument convincingness and proposed a method to automatically assess it. First, they crawled 16 debates from two debate portals [5] and [1] and modeled convincingness as an ordered relation between two arguments. In their experiment, they considered an argument to be a comment which supports or attacks the debate prompt and ended up with 16,927 argument pairs. A debate prompt is a controversial conclusion which the user of the debate portals should argue for or against. Second, they used crowdsourcing to decide for each pair which argument is more convincing or whether they are equally convincing. *Crowdsourcing* is the process of obtaining data by soliciting the contribution from a large group of

people usually called the crowd. Later on, they created an argument graph whose nodes are all the arguments and whose edges are the agreed convincingness relations from the crowd . Next, PageRank [31] algorithm was used to rank the dominant arguments in the argument graph according to their convincingness. PageRank was originally used on the Web to score the relevance of a web page by using the count and scores of the web pages referring to it in a. Recently, it has been used in different applications to find the dominant nodes in a graph by exploiting its structure.
Subsequently, they used different classifiers to predict the convincingness of a pair of arguments based on a large set of features and achieved an accuracy of 0.78. Addressing the problem as a regression problem, they used the same feature set with the same algorithms to automatically rank the convincingness of an argument and achieved a Spearman's correlation of 0.4.
The concept of argument convincingness is rather subjective, especially that they don't report an inter-annotator agreement for the crowd's annotations on which their approach is based. Moreover, it is completely decoupled from previous literature on argument modeling and argument quality since they represent an argument as a mere comment on a certain debate, implicitly ignoring the argument's structure and its inference. Nevertheless, their research is the first approach which relies on PageRank algorithm to assess a quality criterion of arguments.

2.4 Argument Mining

Recently, a subfield of natural language processing called argument mining emerged, which focuses on the automatic detection of argumentation structure in documents. Existing work on argument mining concentrates on the identification of argumentative from non-argumentative text segments (argument unit identification), the classification of argument units into premise or conclusion (argument unit classification) and finally the identification of argumentative relations between argument units .i.e support or attack (argumentative relation classification).
Since the motivation of this work is to evaluate the relevance of arguments on the web, developing argument mining classifiers which are domain-robust is important, because of the heterogeneous and noisy aspects of the web. A domain-robust classifier should be able to achieve good effectiveness on a new domain, on which it has not been trained.

2.4.1 Argument Unit Identification

In argument unit identification, text segments which might contain an argument unit are identified. The first step is to split a given text into segments that are likely to contain an argument unit. While some approaches make the assumption that such segments should match specific syntactic levels such as the sentence level or the clause level, other approaches consider all possible spans within a specific syntactic level. After splitting the text into segments, each segment is classified into a set of classes (e.g. argumentative or non-argumentative). In the domain of essays scoring, Persing and Ng [36] used heuristics to detect the beginning and the end of argument units. A *heuristic* is a practical method or a rule of thumb acquired by experience and used to solve problems. Their heuristics relied on the parse trees of each sentence and the position of connectives and commas. Later on, each combination of a beginning and an end are used to define a candidate argument units. According to the evaluation they conducted, they were able to find exact matches for 92.1% of all argument units in their ground truth. Next, they classified each candidate argument unit into a premise, conclusion, thesis or non-argumentative and achieved an F1-score of of 0.47. This work was the first systematic approach to solve the problem of argument unit identification. Nonetheless, this approach is rather hard to adopt for mining arguments on the Web because of the high computational effort required by the heuristics to parse the sentences which can be an obstacle, considering the enormous size of the Web. Additionally, the effectiveness they achieved is low even in the domain of essay, which is more likely to drop when applied it on the Web, taking into account the variety of content existing there.

On their corpus, which consists of 400 essays manually annotated with argument units and argument structure, Stab and Gurevych [42] used IOB-tagset to label argument units on the word level. *IOB-tagset* is a tag set, consisting of I (inside), B (begin) and O (outside) which are used to mark the words of an annotation according to their position with regard to the annotation. Later on, they classified the words of each essay into one of the tags and achieved an accuracy of 0.895. Since each word is classified only with one tag, the information about a token which occur in the intersection of two adjacent argument units will be lost. Similar to our approach, Moens et al. [30] classified sentences in AraucariaDB [39] into argumentative and non-argumentative and obtained an accuracy of 0.738 in a 10-fold cross-validation test using feature types that rely on word pair, text statistics such as sentence length, average word length and the number of punctuation marks, verb and keyword features. Despite its simplicity, the main disadvantage of this approach is that multiple argument units which occur in a sentence are impossible to locate. Addition-

ally, argument units which cross sentence boundaries cause both sentences to be classified as argumentative, adding some ambiguity to the model regarding the number and the location of the original argument unit.
In the task context-dependent claims detection, Levy et al. [26] suggested a 3-steps funnel to precisely allocate a claim which is relevant to a given topic. It starts by identifying sentences that might contain a claim and be related to the topic. Secondly, several sub-sentences are generated automatically by a boundary component which relies on a Maximum Likelihood model. A Maximum Likelihood model is a probability distribution whose parameters are optimized to best generate the set of observations it is trying to model. Next, the most probable sub-sentence is selected by a classifier. Lastly, another regression algorithm ranks all the candidate for one topic by the using as input the scores generated in the previous steps in the funnel and additional features.

2.4.2 Argument Unit Classification

In this step, the type of the argument units (e.g conclusion or premise) identified in the previous step is classified. The type of an argument unit is seen here as a class among a set of classes. While some approaches identify the argument units before classifying their types as we discussed in the previous step, other approaches merge both tasks in one task. This is done by adding extra classes that indicate whether a text segment is argumentative or not to the considered types.
Alternatively, some approaches skip the task of argument unit identification and classify a text segment into one of the considered types of the argument units. For example, Kwon et al. [23] identified the thesis of a document in public comments on government regulations by using a classifier. They classified each sentence into a thesis or not using token n-grams, the position of the thesis in the paragraph and the position of the paragraph in the document, the subhead of the sentence and a trained topical feature to identify whether the subtopic of the sentence is about policy and achieved an F1-score of 0.55. *Token n-grams* represents the percentage of every 1-n possible sequences of tokens. Later on, they classified the thesis with regard to the topic into support, oppose or propose a new idea via subjectivity clues, headword of the sentence (main predicate), FrameNet Baker et al. [9] frames which are semantic ways to generalize the main predicate, the parsing rule that expands the sentence and token n-grams.
Rooney et al. [38] classified text segments into conclusion, premise and non-argumentative and achieved an accuracy of 0.65. Palau and Moens [32] classified argument units in ECHR into premise or conclusion. ECHR is a set of documents extracted from legal documents of the European Court of Human

Rights (ECHR). They achieved an F1-score of 0.741 for conclusions and 0.681 for premises by using location features, sentence length, the tense of the verb, a binary feature type indicating that the sentence includes the definition of an article and a binary feature type indicating whether the sentence contains a reference to an article of the law.
Stab and Gurevych [41] classified text segments in persuasive articles into non-argumentative, premise, conclusion or thesis and achieved an accuracy of 0.773 by using boolean 1-3 n-grams feature types, three boolean feature types indicating the existence of a verb, an adverb or a modal verb. For each boolean feature type, they generated a boolean feature for each verb, adverb or a model verb from a predefined set. The boolean n-grams feature type they used is a set of boolean feature that indicate the existence or the absence of a sequence of n tokens. In addition, they used boolean syntactic feature types to represent the occurrence of each production rule in the parse tree of the sentence. As indicator feature types, they modeled the existence of a set of discourse markers, first person prepositions, and pronouns. Additionally, they relied on contextual features such as the number of punctuations, the number of tokens and the number of sub-clauses.
As explained in Chapter 1, one of the goals of this thesis is to develop argument mining algorithms which allows extracting the arguments on the Web as a former step to a structural analysis of their relevance. The structural analysis suggested by us relies on an argument graph where the arguments on the Web get connected to each other depending on the semantic similarity of their composing argument units. In this graph, an argument unit can be a premise for one argument and at the same time, it can serve as a conclusion for an another argument. More precisely, the type of an argument unit relies solely on the argument from which it is a unit of. Subsequently, in this work we are not classifying the type of the argument units into premise or conclusion; however, we classify sentences into argumentative or non-argumentative, leaving the task of identifying the type of an argument unit to the argumentative relations it participates in.

2.4.3 Argumentative Relation Classificaiton

Seen as the main component of the argument structure, the main goal of this step is to identify the argumentative relations between argument unit pairs. According to the used argument model, this task is turned into a classification problem by considering an ordered pair of argument units (the source and the target of the argumentative relation) as an input instance and the types of the argumentative relations specified by the model as the output classes. A corpus annotated with arguments according to the argument model is used to develop

the classifier to perform this task. When the corpus is not annotated with negative argumentative relations, this is done automatically by generating them in a systematic way. A negative argumentative relation is an ordered pair of argument units for which there is no argumentative relation between them. Having negative argumentative relations is important in order to learn the classifier to recognize nonexistent relations. As we showed in Subsection 2.2, an argument model can specify the types of the source and the target of an argumentative relation. This must be taken in consideration upon choosing an ordered pair for classification and upon generating negative argumentative relations .
Early work on this task done by Palau and Moens [32], however, did not rely on machine learning to perform it. Instead, they modeled the argumentation structure via a predefined Context-Free Grammar (CFG). Using these grammers they parsed the articles in ECHR after identifying the premises and conclusions in it. Even though the main goal of this model was to recognize premises and conclusions, it allows as well identifying argumentation structure as trees. The accuracy achieved using this approach is an accuracy of 0.6. Since we aim in this thesis to extract the arguments from the web to evaluate our approach on argument relevance, the approach introduced by Palau and Moens [32] to mine argumentative relations is rather hard to adopt. The reason is that they relied while defining the CFG on the formality which characterizes legal text, a characteristic which the heterogeneous content Web lacks. Another reason for excluding such an approach while developing argument mining algorithms on the Web is its low efficiency.
Stab and Gurevych [41] modeled argumentative relation as an ordered pair of argument unit and casted the problem of identifying argumentative relations to a binary classification into support or not-argumentatively related. They used a classifier with different features such as indicators, structural, lexical and syntactic features in addition to the predicted argumentative type and obtained an F1-score of 0.722.
The structural features they used include the number of tokens of the source and the target, and their difference and the number of punctuation marks in the source, the target and their difference. Additionally, they used a structural feature type which rely on the distance between the covering sentences and the position of the paragraph where the argument unit has occurred. The lexical feature types are based on all combinations of word pairs, first words and the existence of modal verbs. Syntactic features and indicators are the same ones used in argument unit classification.
In this thesis, we are only interested in whether there is a relation between two argument units or not. The reason is that, as described in Chapter 1, our approach for assessing argument relevance relies on constructing an argument

graph from the arguments on the Web by merging argument units across arguments which have the same meaning. Subsequently, PageRank algorithm is used to rank the argument units in the argument graph according to their usage by other arguments as premises, thus further information about the type of the relation is not needed for our approach to assess argument relevance. However, such information can be useful for presenting the results.
In his work on modeling argument structure in German microtexts, Peldszus [33] encoded argumentative relations with a tag for each segment which states the target of the relation, the dialectical role (proponent or opponent), the general type, whether the segment represents the central claim (thesis), supports or attacks another claim, the offset of the target of the relation, the type of attack (rebuttal or undercut), the type of support (normal or example) and whether the relation of the segment is simple or combined with another segment.
While Peldszus [33] achieved an accuracy of 0.39 for the whole tagset including 48 tags, their approach is tailored to microtext and cannot be applied to the Web, because of the possible unlimited tagset for long texts which are common on the Web.
Later on, Peldszus and Stede [34] used Minimal Spanning Tree (MST) to model several aspects of global argumentation structure and achieved an F1-score of 0.72 in identifying argumentative relations on their English microtext corpus. Even though the results seems promising, their data set is heavily unbalanced and contains relatively many attack relations (178 attack relations compared to 286 support relations) [33] which is usually not the case in real data. Furthermore, their approach doesn't tackle the problem of segmentation and assumes that each text contains only one argument, which makes it hard to adapt on the Web.

2.5 Argument Corpora

As discussed previously in this chapter, argument mining relies on machine learning algorithms to extract arguments from a given text. Usually, a typical step in argument mining, such as argument unit identification is realized by classifying a span of text into one of a set of classes (e.g. argumentative and non-argumentative). For this sake, different corpora have been created which includes a set of documents annotated with arguments according to predefined argument model. We call the process of annotating a set of documents with arguments the *annotation process*. One research question tackled in this thesis is how to effectively mine arguments on the Web. Therefore, a corpus that is

a good representative of the content on the Web is needed. The reason behind this is that having a representative corpus to helps us to better develop and evaluate our classifiers, as the corpus will contain a more similar input instances to those on the real Web.
As we defined in Subsection 2.2, the technical representation of an argument model on the language level is called *annotation scheme*. An annotation scheme specifies the annotation categories used to represent an argument on the language level. An annotation is a span of text which represents a concept and is labeled with some meta data (usually a type from a predefined set of types). The annotation scheme includes also further information about the granularity of a certain annotation category, e.g. sentence level. The technical informations provided by the annotation scheme is crucial in any annotation process because it defines a systematic rules to annotate arguments, which all parties involved in the annotation process should obey.
Usually, more than one annotator are employed to annotate a given corpus according to a predefined annotation scheme. Depending on the annotation task, the agreement between the annotators may vary.*The agreement* is the degree of consensus over annotations produced by the annotators and usually measured by κ. κ's value indicates complete agreement when $\kappa = 1$ and no agreement when $\kappa < 0$.
One of the most famous corpora is Araucaria [39] which includes structural annotations of arguments created with a graphical annotation tool. The corpus includes approximately 660 documents from different domains like newspaper editorials, parliamentary records, judicial summaries, and discussions. The documents are annotated with argument structure by using a graphical annotation tool. The authors of Araucaria did not report any agreement on the annotated arguments. Nevertheless, Araucaria is one of the most used corpora in argument mining.
Mochales and Moens [29] carried out an argument annotation study in legal cases of the European Court of Human Rights (ECHR). They experimented with a small corpus of 10 documents and obtain an inter-rater agreement of $\kappa = 0.58$. In a similar subsequent study, they achieved an inter-rater agreement of $\kappa = 0.75$ on a corpus of 47 documents. The annotation scheme in this corpus doesn't contain argumentative relation but contains only argument unit types annotations. Since we aim at mining the arguments on the Web to assess their relevance, such a corpus can't be used to develop a classifier to identify argumentative relations which constitute the core component of the argument structure our approach relies on.
Peldszus [33] created a corpus of 115 German microtexts which were annotated with argument units, support and attack relations in addition to the dialectical role of the argument unit (proponent or opponent). In the initial annotation

study, they let 26 students annotate 23 microtext with the described annotation scheme and reported an agreement of κ=0.38. In a subsequent study, they let three expert annotators annotate 115 German microtexts yielding an agreement of κ =0.83. The created corpus, though translated into English, lack non-argumentative text since it was created with specific rules which forced the annotators to generate only argumentative content.
[40] conducted an annotation study of argument units and argumentative relations in the domain of persuasive essays. The annotation scheme includes arguments units, argument unit types (premises, claim, and major claim) and argumentative relations (support and attack). The corpus contains 90 essays from [3] which were annotated by three annotators with percentage inter-rater agreement of 98%, 86 and %86 for major claims, claims, and premises respectively. For argumentative relations, they achieved an inter-rater agreement of 92% and 98% for support and attack relation respectively. The corpus is unbalanced with regard to argumentative relations as only 3.1% of all possible relations are classified as attack relations, 25.5 % were classified as support relations, while 72.4 % were classified as non-argumentatively related. Even though the annotation scheme used to create this corpus contains the enough specification needed by our argument mining approach which I will introduce in Chapter 4, we did not use to train or evaluate our developed classifier. The main reason is that essays domains is not a good representative of argumentative content on the Web, since the content of an essay tends to be formal and targeted against a specific prompt.

As we showed, most of the existing corpora are either constructed from documents from a specific domain, e.g. essays or lack argumentative relation annotations. Therefore and given the wide variety of content on the Web, a standalone corpus is not representative enough to achieve our goal to develop a domain-robust argument mining approach. As we will see in Chapter 4, we will construct a cross-domain corpus which contain multiple subcorpora one of which is Araucaria.

Chapter 3

Objective Assessment of Argument Relevance

3.1 Overview

As explained in Chapter 1, argument relevance, which is an argument quality criteria among others, can be seen from two different perspectives: dialectical and probative [46]. The probative (local) relevance of an argument captures the contribution a set of premises brings to the acceptance or rejection of its conclusion, whereas dialectical (global) relevance refers to the benefit of an argument in a discussion [46]. Dialectical argument relevance is the main quality criteria that an argument search engine should consider while retrieving the most relevant argument for a conclusion. The reason is that it represents the human interest in an argument. Given a set of arguments, an objective method to automatically assess the relevance of arguments to the conclusion is needed.
In this chapter, I will introduce our approach for computationally assessing argument relevance which relies on PageRank algorithm. PageRank is an algorithm which was developed by Google for assessing the relevance of web pages by recursively exploiting the structure of the hyper-links among them, assuming that a web page is more relevant the more it gets referenced by other web pages. A hyper-link which refers from one web page to another is seen as an indicator of the relevance of the referred web page to the referring web page.
To adopt PageRank to estimate argument relevance I will introduce our model which consists of three parts: A canonical argument structure to model arguments, an argument graph whose nodes are the arguments and a reuse interpretation function which defines equivalence between two argument units and thereby introduces reuse relations among the arguments in the argument graph. To simulate PageRank on the web graph, we hypothesized that a con-

clusion is more relevant, the more it is used as a premise for other arguments. This structural analysis of the argument graph should allow us to obtain an objective assessment of argument relevance. An objective assessment of argument relevance is important, because of the anonymous nature of the web, where we would like to return to a user the most relevant results while knowing nothing about his/her background. Based on this analysis, we calculate the PageRank scores of the argument units in the graph and then estimate the relevance of an argument based on the PageRank scores of its premises as we will see in the following subsection 3.2.5.
After introducing our approach for assessing argument relevance, we conduct an experiment to realize the approach and evaluate it. In the experiment, we construct an argument graph from AIFdb Lawrence and Reed [24] which constitutes of small argument graphs, called argument maps, that are created mostly by humans and thus, we call this graph a *ground-truth* argument graph. I will elaborate on the process of creating the ground-truth argument graph and provide statistics about its size and structure to give the reader an idea about its adequacy to evaluate our approach.

Subsequently, I will report on the ground-truth benchmark rankings obtained for a set of conclusions from the ground-truth argument graph by letting experts in computational linguistics rate the relevance of the arguments given for them. These benchmark rankings will be used to assess the effectiveness to our approach in estimating argument relevance. Lastly, I will elaborate on the experiment we carried out to evaluate the rankings produced by our PageRank for Argument Relevance approach against the previously created benchmark rankings.

3.2 PageRank for Argument Relevance

3.2.1 PageRank on The Web

PageRank is a link analysis algorithm which is used by search engines to estimate the relevance of web pages by simulating a random surfer. Consider a surfer who is browsing the Web. Starting from a random page, the surfer can either click on a random link on that page or browse to a random page on the Web. Let us assume that the probability with which the user stays on the page is α and the probability that he browses randomly somewhere else is $1 - \alpha$.
Now imagine that we let the surfer runs for an infinite time. The *PageRank* of a web page is the probability that we find the random surfer at that page. Thus, PageRank algorithm generates a probability distribution over the

web pages which represents their importance by making use of the hyperlinks' structure among them.
Given D a set of web pages, the PageRank of a web page $d \in D$ is computed recursively as:

$$p(d) = (1 - \alpha) \cdot \frac{1}{|D|} + \alpha \cdot \sum_i \frac{p(d_i)}{|D_i|}$$

$p(d_i)$ is the PageRank of a web page $d_i \in D$ that links to d and D_i is the set of all web pages that d_i links to. In this equation, the left summand represents the probability that the surfer browses the page d randomly while the right summand represents the probability that the surfer browses d starting from a referring pages d_i. According to the right summand, a page d_i linking to d contributes to $p(d)$ more the higher PageRank it gets and the less outgoing links it has.
PageRank is an algorithm which revolutionized web search, as it is *"a method for rating web pages objectively and mechanically effectively measuring the human interest and attention"* [31].

3.2.2 The Web as an Argument Graph

In [45], we introduced an argument graph model which provides a global and local formal representation of the arguments on the Web with the aim of estimating their relevance. Globally, the model provides an abstraction of the argumentation on the Web by viewing it as an argument graph. The argument graph represents all the arguments on the Web as its nodes. An edge between two arguments indicates the relevance of the source to the target. The relevance of the source argument is the extent of contribution it adds to the the target and is determined by the units and the structure of both arguments. Locally, the argument graph model represents an argument as a set of premises and a conclusion (called argument units), defining what we call a *canonical argument structure.*
The construction of an argument graph relies on detecting the reuse of the conclusions for some arguments in this graph as premises for others. A conclusion of an argument that is reused as a premise for another argument indicates the relevance of the former argument to the later. Given a set of arguments, a *reuse interpretation function* is used on the local level to merge their units which are semantically equal. On the global level, we create an edge between two arguments if the conclusion of the source argument is used as a premise by the target. Let us introduce our argument graph model more formally:

3.2.3 The Argument Graph Model

Let $D = d_1, d_2, \ldots$ be the set of all considered web pages. Each web page in D may contain zero, one, or more arguments. Given D, our model consists of three blocks:

- **Canonical Argument Structure**: which represents each argument in D as a tuple $a = (c, P)$ where c denotes the conclusion of a and $P = c_1, \ldots, c_k$ its premises, $k \geq 0$. Figure 3.1 depicts the canonical argument structure.

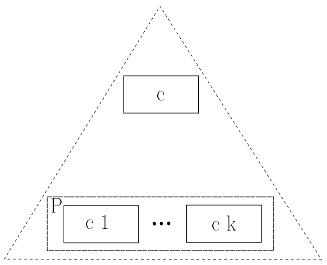


Figure 3.1: Canonical argument structure

- **Reuse Interpretation Function**: An equivalence interpretation function $\mathcal{I}$ that assigns a label from the set {" $\approx$ "," $\not\approx$ "} to each pair of argument units (c, c') of all arguments in D.
- **Argument Graph**:An argument graph $G = (A, E)$ such that $A = a_1, \ldots, a_n$ is a set of nodes where each $a \in \mathbb{A}$ corresponds to one argument in D $E \subseteq A \times A$ is a set of edges where $(a, a') \in E$ $\texttt{iff.}$ $\exists c_i \in a'.P : \mathcal{I}(a.c, c_i) = " \approx "$

Figure 3.2 illustrates a part of an argument graph where the focus is drawn to on an edge (a, a') between two arguments a and a'. The argument a has a conclusion c which is detected by a reuse interpretation function $\mathcal{I}$ to be semantically equivalent to a premise ci of a'. Thus, an edge was constructed from a to a' which indicates the relevance of a to a'.

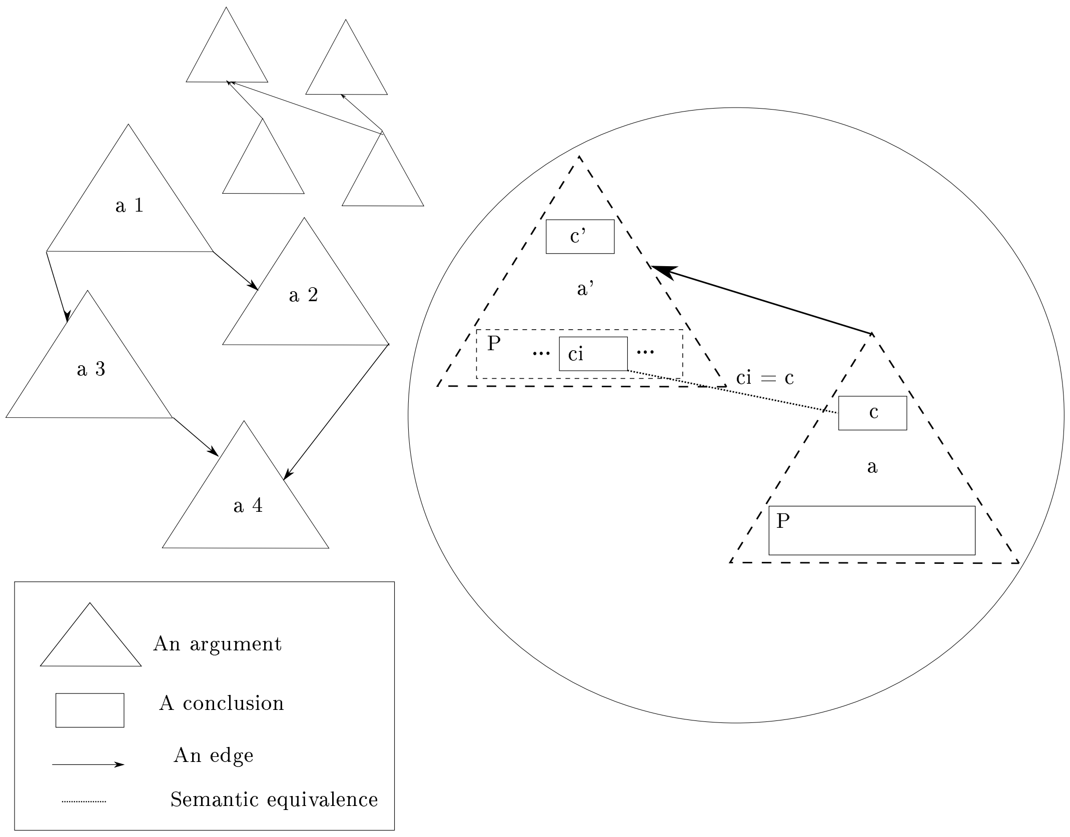


Figure 3.2: An argument graph example

3.2.4 PageRank for Conclusion Relevance

Given an argument graph $G = (A, E)$, we modify PageRank to score the conclusions in the argument graph. As stated in the previous subsection, an edge $e = (a, a_i) \in E$ indicates that the conclusion of a is equal to a premise of a_i with regard to the reuse interpretation function used to build the argument graph G. Now we make the assumption that a conclusion c is given a higher PageRank $\hat{p}(c)$ the more it serves as a premise for other conclusions and the higher PageRanks these conclusions get.
To simulate the random walk on the argument graph, we use the original PageRank $p(d)$ score of the document where an argument is found. The assumption here is that the more relevant a document is the more relevant conclusions found on it are likely to be. To maintain a sum of 1 over all arguments, we normalize $p(d)$ by the average number of arguments in each web page. This results in the ground relevance $\frac{p(d) \cdot |D|}{|A|}$
In contrast to the web graph, where a document d_i contributes more to the PageRank of a document it refers to d the less outgoing links d_i has, a conclu-

sion c_i contributes more to the PageRank of a conclusion it uses as a premise c, the less conclusions c_i uses as premises. Following this logic, we normalize the contribution of each conclusion c_i's relevance $\hat{p}(c_i)$, for which c serves as premise, by the number of premises $|P_i|$ given for c_i. The recursive PageRank formula is given below:

$$\hat{p}(c) = (1-\alpha) \cdot \frac{p(d) \cdot |D|}{|A|} + \alpha \cdot \sum_i \frac{\hat{p}(c_i)}{|P_i|}$$

3.2.5 From Conclusion Relevance to Argument Relevance

For a conclusion c all arguments whose conclusion is c are relevant to it. To estimate the relevance of an argument we aggregate the PageRank scores of an argument's premises to estimate its relevance, ignoring the inference used in the argument. This fits our model which aims mainly to estimate dialectical relevance, i.e. the profit an argument contributes to a certain discussion. We use four premise aggregation methods in this work, namely, (1) the *minimum* (2) the *average*, (3) the *maximum* and (4) the *sum* of the PageRank values of the argument's premises.

3.3 A Ground-truth Argument Graph and Benchmark Rankings

In this section, I report on the argument graph we created in the pursuit of evaluating our PageRank for Argument Relevance introduced in the previous subsection. First, I will introduce AIFdb [25], which we used to construct an argument graph as formalized in the argument graph model explained in the previous section. AIFdb consists of a set of argument maps which are small argument graphs attached to the documents from which they were extracted. Second, I will thoroughly demonstrate the process in which we created a ground-truth argument graph from AIFdb by merging all its argument maps, considering two argument units are equal if they have the exact text. Since the suggested PageRank for conclusion relevance approach relies on the usage of a conclusion as a premise by multiple conclusions, I will present statistics about the usages of argument units as premises in different arguments in the graph. This should give the reader an idea about the structure of the constructed ground-truth argument graph which our PageRank approach exploits. In addition to this and to basic statistics about the constructed graph and its arguments, I will introduce statistics about the count of arguments per each conclusion, as an evaluation for the relevance of the arguments can only be carried out on these argument units which serve as a conclusion for multiple arguments. Lastly, I will introduce the benchmark rankings we created by letting experts in computational linguistics sort the arguments given for or against a set of conclusions according to their relevance. These benchmark rankings will serve as ground-truth rankings associated with the ground-truth argument graph against which we will evaluate our PageRank for Argument Relevance approach.

3.3.1 AIFdb

AIFdb is a database implementation of AIF, allowing for the storage and retrieval of AIF complaint argument structures [25]. AIFdb contains about 70 corpora and about 10k argument maps, making the largest existing argument database. AIFdb also provides part some of documents from which the argument maps were extracted. The count of these documents and other statistics about the corpora are shown in Table 3.1.
An argument map consists of a set of argument units and the arguments in which they are used as premises or conclusion all modeled in a standard format Argument Interchange Format (AIF). The main aims of AIF were to facilitate the development of (closed or open) multi agent systems capable of argumentation-based reasoning and interaction using a shared formalism and to facilitate data interchange among tools for argument manipulation and argument visualization [13]. AIFdb offers an interactive web interface with which user can upload their corpora or create a new corpus or a new argument map.
According to AIF, an argument map is a directed graph which consists of two kinds of nodes, information nodes and scheme nodes. *Information nodes* represent argument units in a certain discussion whereas *scheme nodes* are specific applications of schemes. *Schemes* are patterns of reasoning which are typically used to draw the conclusion of an argument from its premises. An edge can follow from an information node to a scheme node, which means that the information node has been used in the inference made by the scheme nodes. These edges are called data edges. Alternatively, scheme edges emanate from scheme nodes and meant to support conclusions that follow from the scheme node.
To give an intuition of what is an argument map ? we present here an example of one as shown in Figure 3.3. The map consists of three information nodes and two scheme nodes. According to our terminology, these two scheme nodes correspond to two arguments which have the same conclusion and two different individual premises.
AIF format's representation of an argument can be mapped easily to the ar-

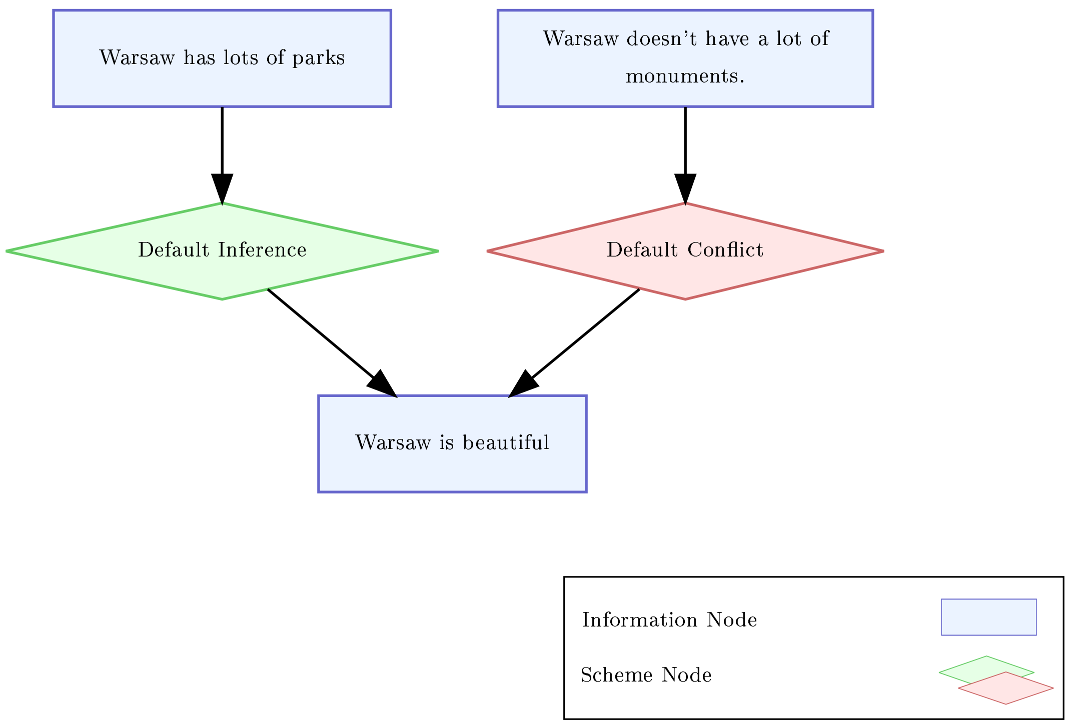


Figure 3.3: An argument map example

gument canonical structure we introduced in Subsection 3.2.3, since a scheme node clearly specifies the conclusion and the premises of an argument. Thus, the argument maps in AIFdb exhibit exactly the argument structure needed by our approach to assess argument relevance. The large size of AIFdb and the equivalence of its representation of argument structure to the structure needed by our PageRank approach to assess argument relevance made it a suitable candidate to be used to realize and evaluate the approach. In the next Subsection, we will show the process we conducted to construct an argument graph from AIFdb.

Statistic Name	Statistic Value
Count of corpora	57
Count of argument maps	8,479
Count of documents	5,421
Count of information nodes	49,504
Count of scheme nodes	26,012

Table 3.1: General statistics about AIFdb

3.3.2 Construction of Ground-truth Argument Graph

In the last subsection, we introduced AIFdb and elaborated on its adequacy to be used in the construction of an argument graph as modeled in Subsection 3.2.3. The argument graph will be constructed from all the argument maps in AIFdb which are mostly annotated by humans and covers multiple domains. Therefore, it constitutes a ground-truth representation of argumentative content on the Web.

The process of building an argument graph from AIFdb started be deleting all duplicate argument maps. This was important since we noticed that some corpora existed as parts of larger corpora in AIFdb. We considered two argument maps duplicate if the documents associated with them contains the exact text. The count of duplicate argument maps amounts to 3,058 and the count of the remaining argument maps is 5,421.
Next , I converted all the arguments in the left argument maps to the canonical argument structure introduced in Subsection 3.2.3.
Given an argument map, I carried out the following steps :

- an information node is converted to an argument unit.
- a scheme node is converted to an argument, whose premises are the referring information node of the scheme and conclusion is the referred information node of the scheme node

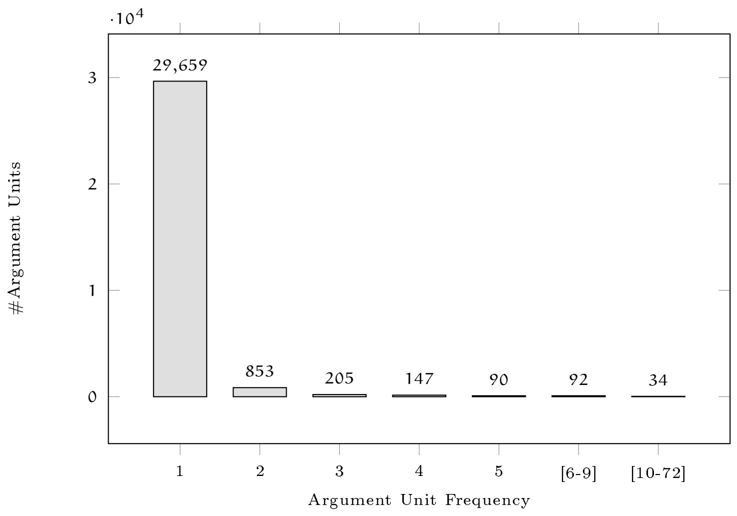
We ignored all the argument maps which have a scheme node without referring information nodes or referred argument maps. Additionally, we excluded all the arguments maps which had a scheme node that refers to another scheme node for simplicity.
After the conversion of the arguments on AIFdb to the canonical argument structure, all the arguments were merged into an argument graph. As described in Subsection 3.2.3, in our model a reuse interpretation function is used for deciding which argument units are semantically equal. In this case, we used a reuse interpretation function which decides that two argument units are semantically equal if they exactly have the same text after converting all its letters to lower case. Let us call this function *exact-match reuse interpretation function.* By comparing each possible pair of argument units in all the arguments via the described function, we merged those argument units which were decided to be equal. Figure 3.4a shows a histogram of the frequency of occurrences for all argument units in the argument graph in AIFdb. Noticeably, most of the argument units occurred only once in the original corpora and no matches were found for them. A reason for this can be the simplicity of the exact-match reuse interpretation function, which checks for the exact

Statistic Name	Statistic Value
Count of arguments	26,012
Count of argument units	31,080

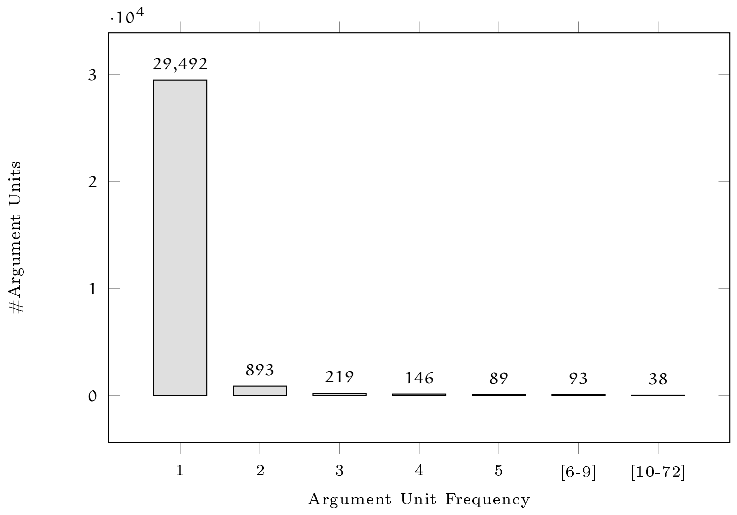
Table 3.2: General statistics about the ground-truth Argument Graph

equivalence between a given pair of argument units.
As explained in Subsection 3.2.5, our PageRank for Argument Relevance gives a conclusion a higher score the more it serves as a premise by different arguments. As a consequence, using a reuse interpretation function which decides for a higher count of matches between conclusions is of high importance while building the graph, as it fosters the structure in the argument graph our approach exploits for estimating argument relevance.
For this purpose, we experimented with a heuristic-based reuse interpretation function. A *heuristic* is a practical method or a rule of thumb acquired by experience and used to solve problems but without a guarantee. This function uses heuristics to ignore in its input punctuations, capital letters, all white spaces, a set of connectives and expressions which are usually used at the beginning of a conclusion, before checking if the text of the given pair of conclusions are identical. A full list of the ignored expression is given in the Appendix.
As shown in Figure 3.4b shows a histogram of the frequency of occurrences of the argument units in the argument graph constructed with the heuristics-based reuse interpretation function in AIFdb. As shown, this argument graph exhibits a similar structure to the argument graph constructed with the exact-match reuse interpretation function, as only few more matches between the argument units were achieved with the heuristics-based reuse interpretation function.
Though the structure of the argument graph obtained by using the exact-match reuse interpretation function doesn't bring a massive amount of discovery of reuse among different arguments, I decided to carry on further experiments using the argument graph constructed while using it. The main reason for this decision was the negligible amount of new matches achieved with the more sophisticated approach heuristics-based reuse interpretation function and because of its simplicity. We leave the development of more sophisticated reuse interpretation functions to future work, due to the limitation of time allocated for this thesis.

Table 3.2, shows general statistics about the count of arguments and the argument units of the constructed ground-truth argument graph as described previously.



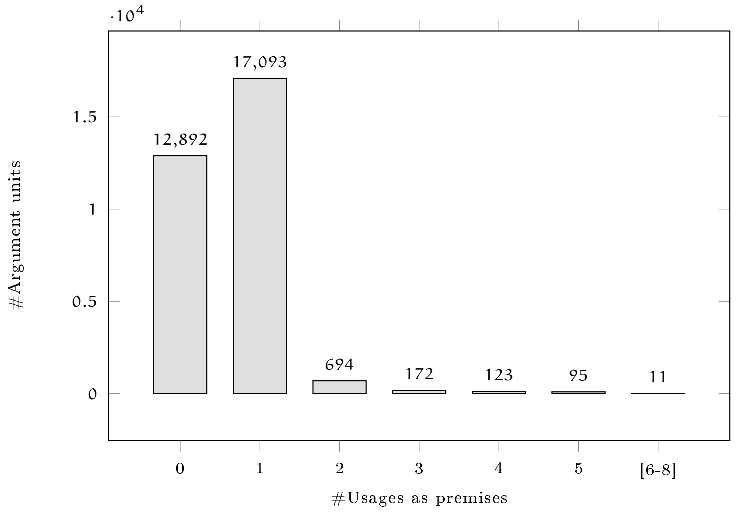
(a) Histogram of the frequency of occurrences of the argument units in AIFdb with an **exact-match reuse interpretation function**



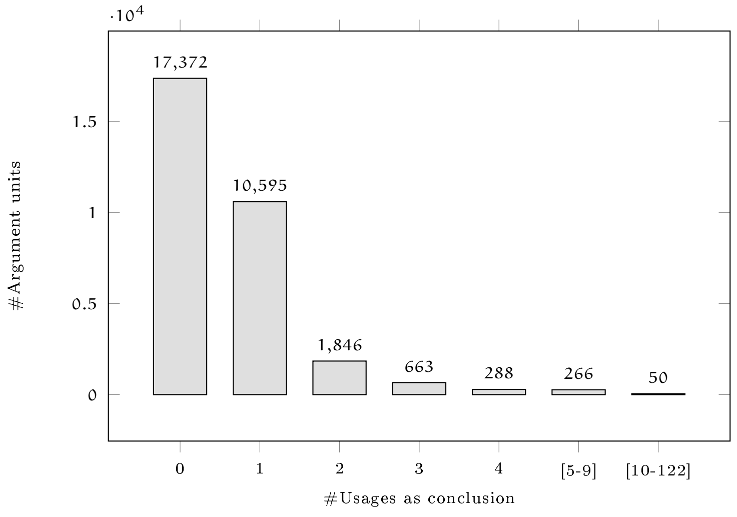
(b) Histogram of the frequency of occurrences of the argument units in AIFdb with a **heuristics-based reuse interpretation function**

Figure 3.4: Construction of ground-truth argument graph

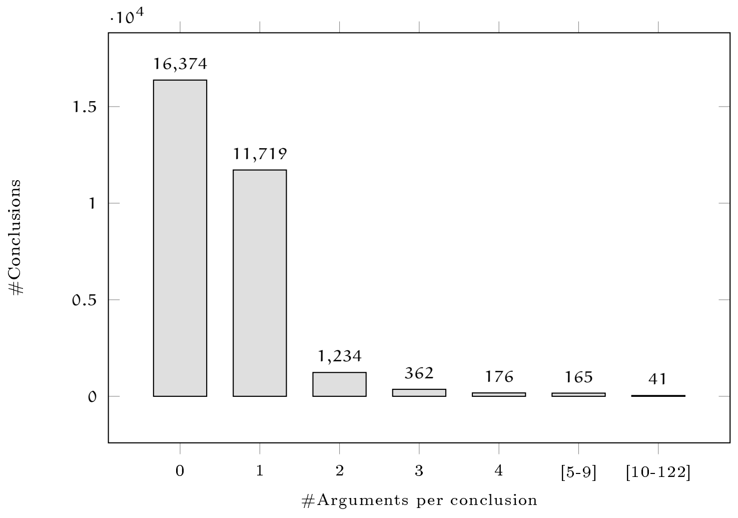
The main motivation for constructing this argument graph is to evaluate our PageRank for Argument Relevance approach. Following our suggested model in Subsection 3.2.3, the exact match reuse interpretation function was utilized to merge argument units in the whole AIFdb which have similar meanings to construct an argument graph. As described in Subsection 3.2.5, our suggested approach estimates the relevance of an argument based on the PageRank scores of its premises. Accordingly, PageRank is used to exploit the structure of a graph by recursively giving a conclusion a higher score the more arguments have it as a premise.
The count of argument units' usages as premises by multiple arguments, shown in Figure 3.5a for the constructed argument graph, represents exactly the structure exploited by our approach. The histogram shows that approximately the smallest part of the argument units 1,000 serve as premises more than once, in comparison to 17,093 argument units which are used by multiple arguments as premises.
To evaluate our PageRank for Argument Relevance approach, the set of argument units which serve as conclusions for multiple arguments are of high interest since only for or against these conclusions various argument are pushed that have contrast in relevance. Figure 3.6a shows the distribution of conclusions against the count of arguments for each conclusion. The histogram shows that the majority of conclusions serve for only one argument (11,719), while approximately 2,000 conclusions serve for more than one argument.



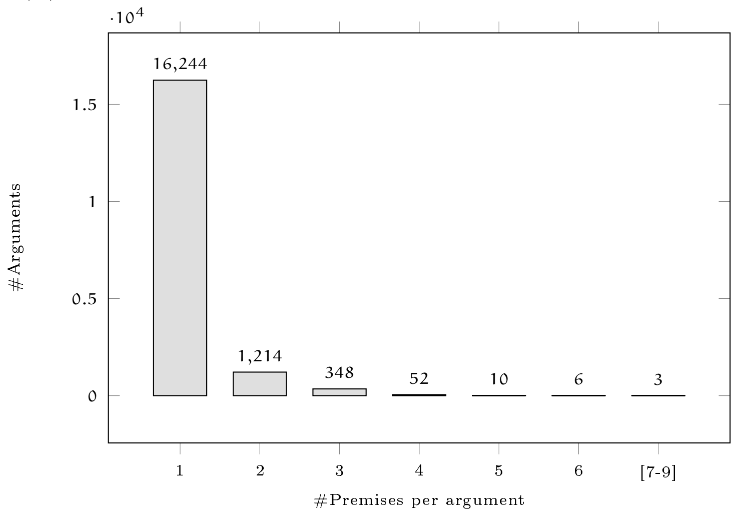
(a) Histogram of the usages of all argument units as premises



(b) Histogram of the usages of all argument units as conclusions



(a) Histogram of conclusions for each arguments count



(b) Histogram of arguments for each premises count

Figure 3.6: Ground-truth argument graph

3.3.3 Creation of Benchmark Argument Relevance Rankings

As described in the previous subsection, the ground-truth argument graph will be used to implement and evaluate our PageRank for Argument Relevance approach. For this purpose, we created benchmark rankings for a set of conclusions and all the arguments in the argument graph pushed against or for them. These arguments' rankings, sorted by experts in computational linguistics, will serve as ground-truth rankings against which we will evaluate

the ranking produced by our approach. All the conclusions which have multiple arguments supporting or attacking it were candidates for the selection process. Only this set of conclusions is interesting for us since the arguments which are attacking or supporting a given conclusion can be sorted according to their relevance. We call this conclusions set *candidate set* while we call the output of the selection process *final set.*
The selection process from the candidate set was conducted in two phases. In the first phase, we retrieved all the conclusions for which at least one argument has multiply used premises. All other arguments don't exhibit any structural difference in the graph in terms of their premises' usages and hence they will have the exact PageRank scores. In the second phase, two experts in computational linguistics classified each conclusion of the set of conclusions produced from the first phase, amounting to 498, into 2 classes: a candidate claim and not a candidate claim. The mean reason for carrying out the second phase is the noisiness of AIFdb. As the arguments advanced for the conclusions in the final set will be sorted by humans, we wanted to keep in the final set only the conclusions which are readable and valid.

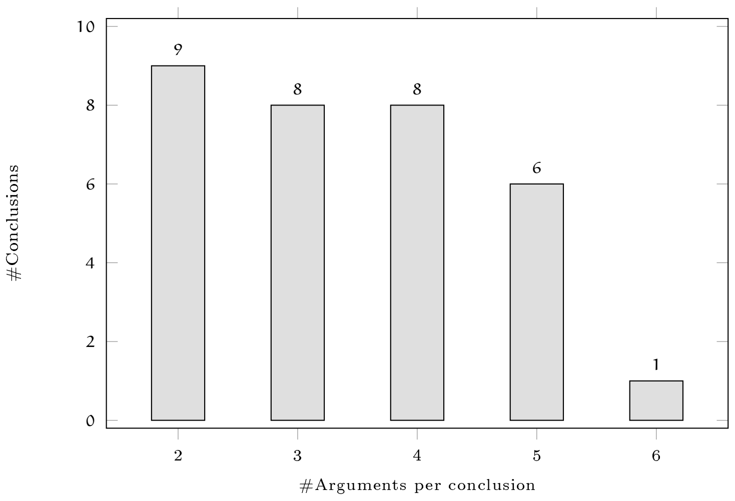
- A candidate claim: a claim that Web users might search arguments for
- Not a candidate claim owing to one of these reasons
  - It is not of general interest but comes from a specific or personal context, e.g., "viv needs to be allowed to prove herself"
  - Its meaning is not clear, e.g., "we need to get back to the classics"
  - It is not English
  - It combines multiple conclusions, e.g., " google wants everyone to move towards doing everything on their apps in the cloud. apple, as they made clear with their overly-long iwork for ipad demo on wednesday, wants every one to move towards using iphone os-based apps.
  - It is not a real conclusion but a topic, anecdote, question or description, e.g., "fingerpriting at the airport"

The experts were allowed to access all the premises of any argument in the candidate set to resolve unclear references or disambiguations. The two experts agreed 451 (90.6 %) of the cases with a substantial Cohen's κ agreement of 0.69. Cohen's κ is a statistic which measures the degree with which two raters agree on classifying a set of items into mutually exclusive classes, taking into account the probability that they agree coincidently. In 136 cases (27.4 %) both experts chose the class not a real conclusion which reflects the noisy

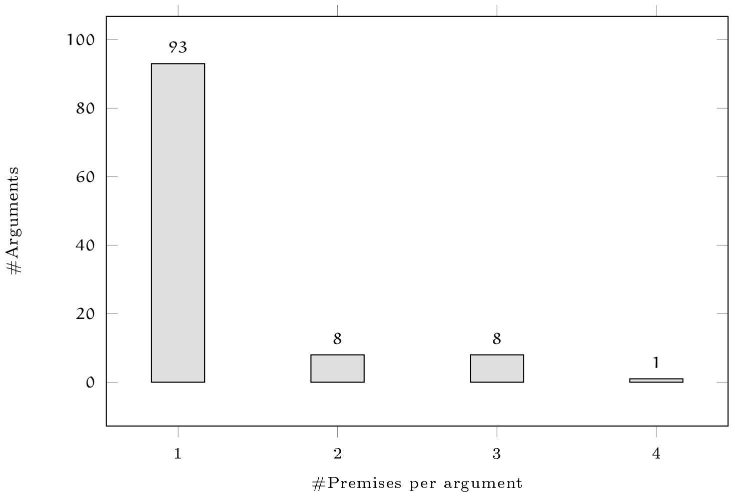
aspect of AIFdb. That's why we chose the conclusions on which both experts classified as a candidate claim which amounts to 70 conclusions.
The same experts classified the 264 arguments for the left 70 conclusions into 3 classes:

- Candidate argument
- Candidate counter-argument
- None

The experts agreed in 201 cases (76.1 %) with $\kappa = 0.63$. To allow for a reasonable and tractable rankings, we kept the conclusions on which both experts agreed on at least two arguments or counter-arguments. At the end, the final set included 32 conclusions and 110 attacking or supporting arguments.
Figure 3.7a shows the arguments distribution for each conclusion in the final set, while Figure 3.7b shows the distribution of arguments over each count of premises.



(a) Histogram of conclusions for each arguments count



(b) Histogram of arguments for each premises count

Figure 3.7: Benchmark argument relevance dataset

All the arguments given or for each conclusion in the final set were sorted by seven experts in computational linguistics and information retrieval according to their relevance. Since our approach aims to adapt PageRank for the objective estimation of argument relevance and provided the possible subjective understanding of it by humans, a large number of experts was needed to have as various range of views about the ranking tasks as possible. A large set of experts should allow us to approximate an objective perception of argument relevance during the ranking process. An objective estimation of argument relevance is important in the ground-truth benchmark rankings because our approach against which this benchmark ranking set will be evaluated should be objective as well. During the manual ranking process, we defined the relevance of an argument for a given conclusion as *the degree by which it contributes to the acceptance or rejection of the conclusion.* The experts were informed that they have to give a strict ordering for the arguments, i.e. each arguments should be unique ranking. To evaluate the consensus between each pair of experts over the rankings of a set of arguments, we used Kendall's τ. Kendall's τ measures the ordinal similarity between two rankings. In comparison to Spearman correlation, Kendall's τ is not affected by how far from each other the rankings are but only whether there is an inversion between them. To better understand Kendall's τ let us introduce it more formally:

Definition: Let $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ be a set of observations of the joint random variables X and Y respectively. Any pair of observations (x_i, y_i), where $i \neq j$, are said to be *concordant* if the ranks for both elements agree, i.e., if both $x_i > x_j$ and $y_i > y_j$ or if both $x_i < x_j$ and $y_i < y_j$. They are said to be discordant, if $x_i > x_j$ and $y_i < y_j$ or if $x_i < x_j$ and $y_i > y_j$. Let us call the count of concordant pairs C and the count of discordant pairs D. Furthermore, let K and L be the count of of the tied groups of observations for X and Y respectively. A group of observations is said to be tied if they have the same value. Additionally, assume that t_k and u_l are the size of a tied group of observations for X and Y respectively.

$$\tau = \frac{C - D}{\sqrt{(n_0 - n_x)(n_0 - n_y)}}$$

Where

$$n_0 = \frac{n(n-1)}{2}$$

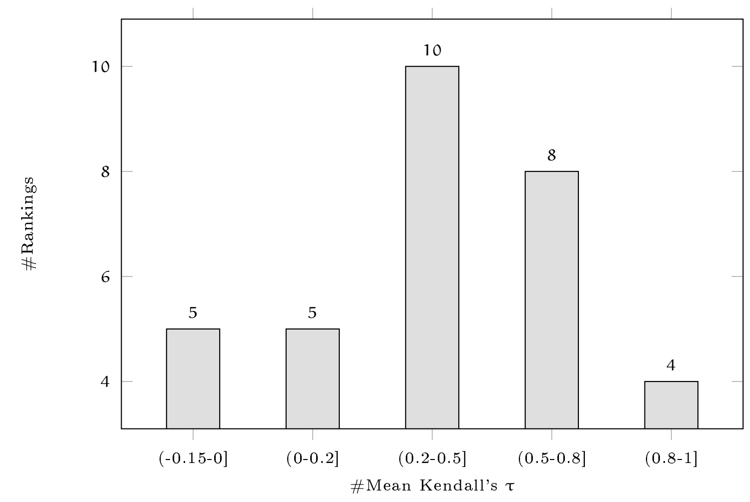
$$n_x = \sum_{k=1}^{K} \frac{t_k(t_k - 1)}{2}$$

$$n_y = \sum_{l=1}^{L} \frac{u_l(u_l - 1)}{2}$$

- if $\tau = 1$ then the two rankings are the same
- if $\tau = -1$ then the two rankings are the reverse of each other
- if $\tau = 0$ then X and Y are independent

For each conclusion, the average Kendall's τ of the argument rankings for all experts' pairs was calculated. The distribution of the conclusions over average Kendall's τ is shown in Figure 3.8a. In 22 cases, the Mean Kendall's

τ was greater than 0.2. Thus, we believe that the resulting rankings qualify as benchmark rankings. The mean Kendall's τ over all conclusions and all experts is 0.35, while the lowest agreement among experts' pairs is 0.1 and the highest agreement is 0.59, all give in Kendall's τ correlation values. This contrast between the experts' pairs supports our hypothesis that the notion of argument relevance is perceived subjectively.



(a) Histogram of conclusions for each arguments count

Figure 3.8: Benchmark argument relevance dataset

3.4 Assessing Ground-truth Argument Relevance

In this section, I will introduce the experiment we carried out to test the effectiveness of our approach to estimate argument relevance on the Web. As described in Subsection 3.2.5, our approach relies on a model which projects the argumentative content on the Web into an argument graph whose nodes are the arguments on the Web and whose edges are the reuse relations among them. PageRank algorithm is adopted then to estimate the relevance of a conclusion by recursively and structurally measuring the attention it gets as a premise by all the arguments. Subsequently, the relevance of a given argument in the argument graph is estimated by aggregating the PageRnak scores of its premises. The structural analysis of a representative argument graph should allow us to objectively estimate the relevance of the arguments found in it.
In the last previous subsection, I introduced the benchmark rankings we created by letting 7 experts in Computational Linguistics sort the given arguments for each conclusion from a systematically selected conclusions set by

their relevance. A relatively large number of experts was needed to approximate an objective understanding of argument relevance during the ranking process. This benchmark rankings set which consists of a set of conclusions, their supporting or attacking arguments and the relevance rankings of the arguments serves as ground-truth dataset for argument relevance.
The experiment I report on here was mainly conducted to compare the agreement between the benchmark rankings and the rankings produced by PageRank to the agreement between the benchmark rankings and the rankings produced by different baselines. A *baseline* is an intuitive approach that provides a lower bound of effectiveness with which the effectiveness of the suggested approach is compared during evaluation. Most of the used baselines follow our approach by estimating relevance of its premises before aggregating it with a Minimum (Min), Average (Avrg), Maximum (Max), Sum aggregation function to assess the relevance of the arguments

3.4.1 Baselines to Assess Argument Relevance

- **Frequency**: This baseline captures the popularity of its premises in the original corpora, i.e. AIFdb. The relevance of an argument is reduced to the frequency of its premises in the original corpora, i.e. AIFdb.

- **Similarity**: The relevance of an argument is reduced to the similarity of its premises to its conclusion. We use Jaccard similarity between the words in the premises and the conclusion. This baseline assumes that the more common words between the premise and the conclusion the more relevant the premise to the conclusion is.

- **Sentiment**: An argument's relevance corresponds to the positivity of its premises. We use SentiWordNet [16] to quantify the positivity of a premise by summing all the positive values and subtracting all negative values generated by SentiWordNet for its words. SentiWordNet learns a set of classifiers to give a positive and a negative score from 0 to 1.0 for a token by using the WordNet database. WordNet is a lexical database that groups English words into semantic groups, such as synonyms.

- **Most premises**: This baseline reduces the relevance of an argument to the count of its premise, hypothesizing that the larger the amount of premises an argument relies on the more relevant it is.

- **Random**: The relevance is decided randomly.

3.4.2 Experiment

For each conclusion in the benchmark ranking dataset, we calculated the relevance scores produced for the arguments given for the conclusion by our approach and with the five baselines described above. For the first three baselines and for our approach, the relevance was first calculated for the premises before aggregating them to estimate the relevance of the argument through 4 aggregation methods, namely Minimum (Min), Average (Avrg), Maximum (Max) and Sum. Next, each set of arguments was sorted according to the scores produced by the approach. Later on and for each conclusion, Kendall's τ has been calculated to estimate the agreement between the rankings produced by each baseline for each argument given for the conclusion and the rankings existing int the benchmark ranking dataset. Subsequently, the average Kendall's τ was calculated over all conclusions. Similarly, the agreement between the rankings produced by our approach and the rankings existing in the benchmark ranking dataset was calculated in terms of mean Kendall's τ. The results for our approach and for each baseline is summarized in the Table 3.3.

Baseline	**(a) Min**	**(b) Avrg**	**(c) Max**	**(d) Sum**	**Best of a-d**
Frequency	-0.1	-0.03	-0.01	0.1	0.1
Similarity	-0.13	-0.05	0.01	0.02	0.02
Sentiment	0.01	0.11	0.12	0.12	0.12
Most premises	-	-	-	-	0.19
Random	-	-	-	-	0
Approach					
PageRank	0.01	0.02	0.11	0.28	**0.28**

Table 3.3: (a-d) Mean Kendall's τ correlation for all 32 benchmark argument rankings. (e) Best observed results for each baseline/approach

3.4.3 Results

As shown in Table 3.3, PageRank with the aggregation method Sum achieves the highest correlation with the benchmark rankings with a Kendall's τ of 0.28. Even though the correlation value is not so high, it is close to the mean Kendall's τ of all the experts 0.36 as reported in Subsection 3.3.3. The results clearly show that the PageRank approach outperforms the baselines Frequency and Similarity in all the aggregation methods. Matching the concept that popularity is not matched with merit [17], Frequency doesn't achieve any significant correlation with the benchmark rankings. Comparably, the Similarity

baseline also barely achieves any correlation with the benchmark rankings. A justification for that might be that this baseline rewards redundancy, i.e. a longer premise might get a higher score even though it is not so relevant to the conclusion.
The only baseline which achieves a close correlation with the benchmark rankings to our approach is the Most Premises baseline. However, as shown in Figure 3.7b, the majority of the arguments have only one premise and more importantly all the arguments given for 22 conclusions of the 32 conclusions have only one premise and thereby all were given the same ranking. To conclude, we see these positive results as evidence for the appropriateness and impact of our PageRank approach to estimate argument relevance.

Chapter 4

Domain-Robust Mining of Arguments from The Web

An argument search engine, as we introduced in Chapter 1, will retrieve all arguments on the Web whose conclusions are semantically equal to a given user's query and sort them according to their quality. In my thesis, I will concentrate on argument relevance as the main argument quality criteria, since it matches the notion of relevance in regular web search engines. Prior to retrieving relevant arguments to a given user's query, all arguments on the Web should be mined and analyzed to allow for a fast fulfillment of the user's information need. Argument mining, which is a sub field of natural language processing, aims at the automatic extraction of arguments and its structure from a given a text. As we introduced in Chapter 2, most of the approaches developed in argument mining train supervised machine learning classifiers on annotated corpora, before using them to extract the arguments from a given text. In this process, An argument model is used to abstract from the language level to a higher level where the units of an argument, their types, and their relations are described.

The Web, characterized by its enormous scale and heterogeneity, will constitute a major challenge in the process of releasing such a search engine. The main reason is that the majority of the classifiers developed in the area of argument mining are trained and tested on a corpus from a specific domain, e.g. law text. While trying to maximize its effectiveness, a classifier will capture domain-specific features, such as tokens which occur in annotations labeled with a certain class. Thus, the effectiveness of this classifier is more likely to decrease upon applying it on another domain, because it will contain different usage of language, for example, the usage of different senses of a word. We call the difficulty faced by a classifier trained on one domain and tested on a different domain the *domain effect.* As we aim at mining arguments from the

Web and provided the variety of content on it, developing argument mining approaches which are domain-robust is essential for extracting arguments on the Web.

In Chapter 3, we introduced our approach to assessing argument relevance using PageRank. Given a set of arguments, modeled as a conclusion and set of premises, we first build an argument graph whose nodes are the given arguments. An edge between two arguments is constructed when the conclusion of one of them is semantically equal to a premise of the second one with regard to a given reuse interpretation function. Ideally, this argument graph abstracts the whole argumentative content on the Web in a structure that highlights the attention an argument unit gets as a premise. The argument graph is structurally analyzed by PageRank, which gives a score distribution over the argument units of the argument graph, recursively giving higher scores to those conclusions which are used as a premise by more arguments. Finally, the relevance score of an argument is aggregated using the PageRank scores of its premises.

To evaluate the effectiveness of our approach in assessing argument relevance on the Web, we constructed an argument graph from the corpora of AIFdb [25]. Next, we used our suggested approach to scoring the relevance of all arguments in the argument graph. Later on, we systematically selected a set of valid and readable conclusions for or against which more than one argument existed with different argument relevance scores. For each conclusion, we let 7 annotators rank the relevance of all the retrieved arguments, creating the first objective benchmark argument relevance rankings. By comparing the benchmark argument relevance rankings and the rankings produced by our approach we found a positive correlation in terms of Kendall's τ, indicating the effectiveness and impact of our approach in assessing the relevance of the arguments on the Web.

The evidence of the appropriateness of our approach to assessing argument relevance on the Web motivated us to study the question of how to effectively and efficiently mine the Web for arguments, before preparing them for further analysis to assess their relevance according to our approach. As described previously, the main issue to consider while developing classifiers for argument mining on the Web is how to guarantee domain-robustness, i.e. the ability of a classifier to generalize over domains and to achieve an acceptable effectiveness on a not seen domain. Another question to answer is how to model the arguments on the Web in a way that allows effective mining and meanwhile provides enough information for the argument analysis steps, in my case, this is our approach to assessing argument relevance using PageRank. The canonical argument structure we introduced in Chapter 3 represents an argument as a conclusion and a set of premises, both seen as argument units, and the

argumentative relations between them.
In this Chapter, I will first introduce the simple argument model we used to model the arguments on the Web, which exactly matches the canonical argument structure. Subsequently, I will introduce the annotation scheme we used to represent an argument defined according to our argument model on the language level. This argument model will be used in our domain-robust argument mining approach which will be broken up into classification tasks to distinguish the types of the annotations specified by the argument model. Next, in Subsection 4.2, I will introduce the corpus that I created and on which the domain-robust argument mining approaches will be based and evaluated. The corpus is constructed from three different subcorpora which are Araucaria [37], AIFdb [25] and WebDiscourse [21] to allow for evaluating the effectiveness of a classifier on one corpus after training it on the other corpora, simulating a cross-domain testing scenario, as each corpus represents a different domain. For each subcorpus, I will introduce its argument model and how it is mapped to our argument model. We call the argument model which is used originally to represent the arguments in each subcorpus the *original argument model*, to distinguish it from our simple argument model to which it will be mapped. Additionally, I will provide statistics about the arguments in it and in the whole constructed corpus. These statistics should give the reader an idea about the count of the argument units, their types and the argumentative relations existing in the corpus which reflects the argumentative content on the Web and helps the reader to judge and understand the results of the experiments.
The domain-robust argument mining approach I developed is divided into two sequential tasks: argument unit classification and argumentative relation classification, closely matching our suggested simple argument model which consists of the argument units and the argumentative relations between them. This modular break-up of the main task of mining the arguments on the Web into argument unit classification and argumentative relation classification allows for a separate development and evaluation of both tasks, before cascading them into a pipeline that performs the whole task. In each task, a classifier will be developed to distinguish between positive and negative instances of both argument units and argumentative relations.
For each task, I will conduct two experiments: an in-domain experiment and a cross-domain experiment. In the in-domain experiment, a classifier will be trained and tested on the same subcorpus, whereas in the cross-domain experiment a classifier will be tested on a subcorpus and trained on the other subcorpora. The cross-domain experiments aim at developing and evaluating classifiers which are domain-robust by testing the classifier on a completely unseen subcorpus. To distinguish between the general difficulty of both clas-

sification tasks and the difficulty introduced by using a subcorpus from a different domain as a testing set, we conduct the in-domain experiments where the training set and testing set belong to the same corpus and hence excluding the domain effect.
In Subsection 4.3, I will present the design of both experiments: the in-domain experiment and the cross-domain experiments and how did we create the training and testing splits for both of them. Later on, I will introduce our approaches to solving the tasks of argument unit classification and argumentative relation classification and the results of both approaches in the in-domain and cross-domain experiments. Finally, I will report on the effectiveness of the whole approach after cascading both tasks into a pipeline and discuss the results.

4.1 Models For Argument Mining on The Web

The simple argument model represents an argument on the Web as a conclusion and a set of premises pushed to support or attack the conclusion. As explained previously, our domain-robust argument mining approach will rely on this representation to extract arguments from the Web. This model exactly matches the canonical argument structure needed by our approach to assessing the relevance of an argument as we discussed in Subsection 2.2. The equality between the two representation is important because in this thesis we aim at assessing the relevance of the arguments on the Web and hence, the mined arguments will be subjects to the argument relevance analysis we introduced in Chapter 3. As depicted in Figure 4.1 In our model, the conclusion and the premises of an argument are both seen as argument units. An argumentative relation which is an ordered pair of two argument units specifies the conclusion as the target of the edge and the premise as the source of it. According to this model, the type of an argument unit is either: argumentative or non-argumentative. Further information about whether an argument unit is a premise or a conclusion is determined by each argumentative relation it is involved in. Thus, an argument unit can be a premise of one argumentative relation and a conclusion of another. Additionally, in this model we don't distinguish between an attack or support relation since these details are not needed by our PageRank approach to assessing argument relevance. Our approach which estimates the relevance of an argument based on a structural analysis of the usage of its premises in different arguments doesn't differentiate whether a premise was used in an attack or a support relation.

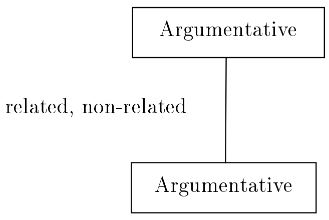


Figure 4.1: The simple argument model, including argument units and argumentative relations.

Annotation Scheme As we discussed in Subsection 2.5, an annotation scheme is a technical representation of an argument model which is used to map an argument on the language level by means of annotations. After we defined our simple argument model, we present here the annotation scheme which we adopted to annotate our cross-domain web argument corpus with arguments. Our annotation scheme specifies two annotation categories:

- *ArgumentUnit* with two types: *argumentative* and *non-argumentative.*
- *ArgumentativeRelation* with two types: *related* and *non-related.* Each ArgumentativeRelation refers to a *premise* ArgumentUnit annotation and a *conclusion* ArgumentUnit annotaiton, both must be labeled as argumentative.

The annotation scheme defines the granularity of an ArgumentUnit annotation to be the sentence level. While a more fine-grained granularity, such as clause level can result in a better accuracy to represent argument units, we excluded this option because it requires parsing all the sentences on the Web which is a computationally expensive task. Hence, we leave the decision of the appropriate granularity to model argument units on the Web for further research. An ArgumentUnit annotation labeled with the type argumentative specifies a real argument unit, while an ArgumentUnit annotation labeled with the type non-argumentative specifies a fake argument unit. An ArgumentativeRelation annotation labeled with the type related specifies a real argumentative relation, while an ArgumentativeRelation annotation labeled with the type non-related specifies a nonexistent argumentative relation.

4.2 A Cross-domain Web Argument Corpus

As we discussed previously, a major challenge in developing argument mining approaches on the Web is the domain effect, as most of the approaches rely on annotated corpora to train classifiers before using them to mine arguments from a real document. Consequently, the performance of such a classifier will decrease when applied to a new domain where people use the language differently, e.g. the usage of different senses of a word. This is an important challenge while developing argument mining approaches on the Web since the Web is characterized by its heterogeneous content. To guarantee domain-robustness when developing our argument mining approaches for the Web, I constructed a *cross-domain web argument corpus* which includes three different subcorpora, namely Araucaria [37], AIFdb [25] and WebDiscourse [21], each representing a different domain.

The constructed corpus is annotated with arguments as specified by the simple argument model we introduced in the previous Subsection 4.3. This model represents an argument as a conclusion and a set of premises, all seen as argument units whereby argumentative relations specify a conclusion and a premise as the target and the source of the relation respectively. On the language level, the annotation scheme of the simple argument model uses ArgumentUnit annotation categories to represent argument units and ArgumentativeRelation to represent argumentative relations between them.

For each subcorpus, we will map each argument in it from its original argument model to our argument model, generating the corresponding ArgumentUnit and ArgumentativeRelation annotations for the argument in two steps as follows:

Step 1 Given a document annotated with a set of arguments, we segment its text into sentences and do the following:

- Generate an ArgumentUnit annotation a labeled wtih the argumentative type for each sentence if:
  - It contains an annotation a' corresponding to an argument unit as defined in the original argument model regardless of its type, e.g. a premise or a conclusion.
  - It is contained in an annotation a' corresponding to an argument unit as defined in the original argument model regardless of its type.
  - It intersects with an annotation a' corresponding to an argument unit as defined in the original argument model.

  For the second step we keep a map between each annotation a and the original annotation a' from which it was created

- Generate an ArgumentUnit annotation a labeled the with non-argumentative type for all sentences which don't fall in one of the previous categories.

Step 2 Depending on the annotation(s) and the types which the original argument model of a subcorpus uses to represent the structure of an argument, we create our ArgumentativeRelation annotations. Given two annotations a'_1 and a'_2 which correspond to two argument units with a directed support/attack relation as defined in the original argument model of the subcorpus. Let us assume that a'_1 corresponds to the premise of the relation and a'_2 to the conclusion of the relation. Let also A_1 and A_2 be the set of ArgumentUnit annotations that are mapped to a'_1 and a'_2 respectively according to the map we created in Step 1. We do the following:

- Generate an ArgumentativeRelation annotations between each $a_1 \in A_1$ as a premise and each $a_2 \in A_2$ as a conclusion and we label it with the related type, regardless to the type of the argumentative relation between a'_1 and a'_2.

Step 3 For each pair of ArgumentUnit annotations a_1 and a_2 if there is no ArgumentativeAnnotation created in Step 1 whose premise is a_1 and whose conclusion is a_2, we generate an ArgumentativeRelation annotation and set its premise to a_1 and its conclusion to a_2 and set its type to non-related.

Algorithm 1: General algorithm for mapping arguments to our argument model

4.2.1 AIFdb

As we introduced in Subsection 3.3.1, AIFdb contains about 70 corpora and about 10k argument maps. An argument map consists of a set of argument units and the arguments in which they are used as premises or conclusion all modeled in a standard format Argument Interchange Format (AIF). AIF format represents an argument as a set of nodes which can be one of two types: information node or scheme node. An information node represents a content unit which is used in an argument, corresponding to argument unit in our terminology. A scheme node represents the type of reasoning which is used to draw the conclusion of an argument from its premises. AIF format uses directed edges to represent the connection between the information nodes and the scheme node of an argument. A directed edge from an information node to a scheme node indicates its usage as a premise in the argument, while a directed edge from the scheme node to an information node indicates its usage as a conclusion in the argument. Thus, an argument modeled in AIF format can be mapped to our simple argument model by converting the information nodes to argument units. Subsequently, a scheme node linking the premises and the conclusion of an argument can be mapped to a set of argumentative relations between each premise and the conclusion. Thus, AIF format provides the adequate specification needed by the simple argument model to represent arguments.
While I used the argument maps of AIFdb in Chapter 3 to construct a ground-truth argument graph with the purpose of evaluating our PageRank approach to assessing argument relevance, I will reuse it here to construct a subcorpus of our cross-domain web argument corpus which I will utilize to develop a domain-robust argument mining approach. Some of the argument maps in AIFdb are associated with the documents from which they were created, technically stored in plain text format. As AIFdb contains argument maps in 10 different language [24] and since we target only English language in this thesis, we excluded all documents which are written in a non-English language. Additionally, we excluded all documents which have the exact text. All documents of Araucaria corpus, which is part of AIFdb, were excluded from this subcorpus and from this statistics as it will constitute an independent subcorpus on its own as we will see in the next Subsection 4.2.2. As shown in Table 4.1, the count of the valid documents which are left and which will constitute our AIFdb subcorpus is 1,000 document.

Statistic Name	Statistic Value
Count of all documents	1,742
Count of non-english documents	190
Count of invalid documents	122
Count of duplicate documents	859
Count of valid documents	571

Table 4.1: General statistics about the documents in AIFdb subcorpus

The separation between the arguments' structure and the documents from which they were extracted in AIFdb required creating an annotation scheme to represent argument maps on the language level. An argument map is stored technically in JSON format, specifying the information nodes and the scheme node of each argument. The documents associated with the argument maps were stored in plain text and without any metadata that links a node in an argument map to its corresponding text segment in the associated document. To annotate the documents with the arguments defined in the associated argument maps we defined a new annotation scheme.
The annotation scheme we used in the automatic annotation process contains two annotation categories: *InformationNode* and *SchemeNode*, closely matching the specification of AIFdb. An InformationNode annotation spans a sequence of tokens and represents an argument unit. A SchemeNode annotation refers to a *conclusion* InformationNode annotation and to an array of InformationNode annotations called *premises*. Additionally, a scheme node has a type which takes either a support or attack value.
The automatic annotation process relied on a content-based matching of the information nodes in the documents. Since there was no metadata about the location of an information node in the original document, we used a string matching algorithm to locate an information node's content in the document and generate an InformationNode annotation for it at that location. Subsequently, for each scheme node in an argument map, a SchemeNode annotation was generated, specifying its conclusion and premises.
After annotating each document with all the arguments defined in its associated argument map, we used Algorithm 1 to map the arguments to the simple argument model. We considered an InformationNode annotation to represent an argument unit and hence, we performed Step 1 accordingly. Additionally, for a SchemeNode annotation we considered each InformationNode of its premises to have an argumentative relation with its conclusion and performed Step 2 & 3 accordingly. The output of this step constituted our final AIFdb subcorpus annotated with arguments as modeled by the simple argu-

ment model.

4.2.2 Araucaria

As we introduced in Chapter 2, Araucaria is one of the most used corpora in argumentation mining and constitutes of approximately 660 documents annotated with argument structure. The documents come from different sources, such as discussion forums, newspapers, and weekly magazines; however, the majority of the documents come from unclassified sources. The original argument model of Araucaria represents an argument as a conclusion and a set of premises. Since Araucaria is a subcorpus of AIFdb, we handled it the same way we handled AIFdb but as a separate subcorpus. Araucaria, being a subcorpus of AIFdb, consists of about 660 argument maps modeled with AIF format associated with the documents from which they were extracted. Therefore, we carried out the same steps we did for AIFdb. First, we excluded all the duplicate and all the empty documents in Araucaria. Table 4.2 shows the count of all the documents, the empty documents, the valid documents and the left valid documents which will constitute our Araucaria subcorpus.

Statistic Name	Statistic Value
Count of all documents	661
Count of duplicate documents	69
Count of valid documents	592

Table 4.2: General statistics about the documents in Araucaria subcorpus

Since Araucaria is a subcorpus of AIFdb, we carried out the exact automatic annotation process to what we conducted on AIFdb. The goal of this automatic process was to annotate the valid documents of Araucaria with the arguments' structure defined by the argument maps associated with them. Just like the automatic annotation process conducted on AIFdb, the content of an information node in an argument map was located in the document associated with the argument map by a string matching algorithm. In this automatic process, we used the same annotation scheme we created to represent arguments modeled by AIF format. As we described in Subsection 4.2.1, the annotation scheme consists of two annotation cases: InformationNode and SchemeNode. The output of this automatic process were the valid documents of Araucaria annotated with the annotation scheme, which is created to represent the arguments' structure in AIF argument maps on the language level.

In the next step, we used Algorithm 1 to map the arguments, which were annotated in the automatic process described previously, to the simple argument model. We took the same steps we took in creating the final AIFdb subcorpus, by considering an InformationUnit to represent an argument unit and performing Step 1 based on this assumption. Additionally, for a SchemeNode annotation we considered each InformationNode of its premises to have an argumentative relation with its conclusion and performed Step 2 & 3 accordingly. The output of this step constituted our final Araucaria subcorpus annotated with arguments as modeled by the simple argument model.

4.2.3 WebDiscourse

The WebDiscourse corpus consists of 340 documents covering six domains (homeschooling, private vs. public schools, mainstreaming, single-sex education, prayer in school, and redshriting) [19]. The argumentation model used in the corpus is based on an extension of Toulmin's model [43]. The argument model consists of several argument unit types: claim, premise, backing, rebuttal and refutation. The argumentative relations used in this model have two types: attack or support. Habernal and Gurevych [19] used the annotation categories ArgumentComponent and ArgumentRelation to model both argument units and argumentative relations respectively. An ArgumentRelation annotation has a *source* and a *target* each of which refers an ArgumentComponent annotation. This argument model provides a similar, though more detailed, representation of the arguments on the Web to the one we introduced at the beginning of this Chapter. Both this reason and the fact that corpus contains documents taken from different domains the Web made this corpus a suitable candidate to be a part of the cross domain web argument corpus.
To integrate WebDiscourse in the corpus we are constructing, we had to map its arguments to the simple argument model, which we proposed to represent the arguments on the Web. For this purpose, we used Algorithm 1 by considering an ArgumentComponet to represent an argument unit and did Step 1 accordingly. Similarly, we assumed that an ArgumentRelation represents an argumentative relation whose premise is the source of the ArgumentRelation and whose conclusion is the target of it. We carried out Step 2 & 3 to generate the ArgumentativeRelation annotations with the mapped type. The output of this process is the WebDiscourse subcorpus after being annotated with arguments, as modeled by the simple argument model we proposed.

4.2.4 Corpus Statistics

Here we present statistics about the type distribution of the ArgumentUnit and ArgumentativeRelation annotations generated for the three subcorpora. The corpus will be used mainly to develop classifiers for our argument mining approach on the Web, where the annotations of each annotation category will be treated as input instances and their types as the classes to be predicted. In the cross-domain experiment, we will divide the corpus into three splits. Each split divides the corpus into a testing set that is one subcorpus and a training set which is the remaining subcorpora. For each annotation category, we will evaluate the effectiveness of our classifier in classifying its type for each split and compare it to the effectiveness achieved by a baseline. We will use a trivial baseline that chooses always the same type which we are interested in. The effectiveness of this baseline can be calculated for each split by using the type distribution of the annotation categories which the classifier is targeting.
Table 4.3 shows the type distribution of ArgumentUnit annotations for each subcorpus in the cross-domain argument web corpus. For the ArgumentUnit annotation category, the count and percentage of its annotations with the type argumentative and non-argumentative is shown. Similarly , the count and the percentage of ArgumentativeRelation annotations with the type related and non-related is presented in Table 4.3. The tables also show the count of all the annotations in the third column of each table. For the ArgumentativeRelation annotations, we see that there is a big difference of the count of the annotations between the WebDiscourse corpus and the Araucaria corpus.
Table 4.5 summarize the statistics for both annotation categories in the three corpora the main corpus .These statistics should give the reader a rough idea about the approximate count and percentage of the argument units for represented by our argument model on the Web. As shown the percentage of the argument units with the type argumentative, i.e. real argument units amount to 33% approximately, reflecting a slightly skewed distribution. For argumentative relations, however, we see here a more skewed distribution with almost 11 % real argumentative relations compared to 88% fake argumentative relations.

Subcorpus	Argumentative	Non-argumentative	Count of annotations
AIFdb	822 (32.88 %)	1,678 (67.12 %)	2,500 (100 %)
Araucaria	890 (27.47 %)	2,350 (72.53 %)	3,240 (100 %)
WebDiscourse	1535 (39.27 %)	2,374 (60.73 %)	3,909 (100 %)

Table 4.3: Type distribution of the ArgumentUnit annotations for each subcorpus

Subcorpus	Related	Non-related	Count of annotations
AIFdb	654 (11.75 %)	4,912 (88.25 %)	5,566 (100 %)
Araucaria	366 (15.84 %)	1,944 (84.16 %)	2,310 (100 %)
WebDiscourse	869 (8.43 %)	9,438 (91.57 %)	10,307 (100 %)

Table 4.4: Type distribution of the ArgumentativeRelation annotations for each subcorpus and for the whole corpus

ArgumentUnit		ArgumentativeRelation	
Argumentative	**Non-argumentative**	**Related**	**Non-related**
3247 (33.65 %)	6402 (66.35 %)	1889 (10.39 %)	16294 (89.61 %)

Table 4.5: Class distribution of the ArgumentUnit and ArgumentativeRelation annotations for the cross-domain web argument corpus

4.3 Mining Arguments Across Web Domains

Our approach to mine arguments from the Web constitutes of two sequential tasks: argument unit classification and argumentative relation classification, which are arranged in a pipeline to extract the arguments as represented by the argument model we introduced previously in Subsection . According to our model, an argument constitutes of a set of argument units (a conclusion and a set of premises) and a set of argumentative relations between the premises and the conclusion. On the language level, argument units are represented by an annotation category called ArgumentUnit which is labeled by the argumentative or non-argumentative type. Similarly, argumentative relations are represented by an annotation category called ArgumentativeRelation which is labeled with the related or non-related type and refer to an ArgumentUnit annotation which specifies the premise of the relation and another one which specifies the conclusion of the relation. These two annotation categories, one referring to the another (ArgumentUnit referred to by ArgumentativeRelation), justify the sequential design of our approach as two tasks one piped in the other task (argument unit classification piped in argumentative relation classification).

For each task, I developed a classifiers set which uses the same classifier type and a set of feature types. A *classifiers set* consists of multiple classifiers which are instances of the classifier type but each is evaluated in a different experiment (cross-domain or in-domain). We call the classifier set developed for the argument unit classification task the *argument unit classifiers set* and the classifier set developed for the argumentative relation classification task the *argumentative relation classifiers set.* A classifier uses features types to quantify interesting aspects of an input instance that allow for distinguishing its class. In our case, an input instance is an annotation (ArgumentUnit or ArgumentativeRelation) and its class is its type, e.g. argumentative. The feature types, which are developed for each classifier, measure syntactic, structural or lexical properties of the annotation that are likely to correlate with its type.

The different design of the in-domain and the cross-domain experiments can affect the knowledge a feature type brings to a classifier for a specific task. In the in-domain experiment, a classifier is exposed to input instances (annotations) in the training and testing sets which come from the same domain and thus exhibit similar usage of language. In the cross-domain experiment, a classifier is confronted with input instances in the testing sets which belong to a different domain from the ones the classifier were exposed to in the training set and hence, have a more different usage of language. Since our feature types depend on linguistic properties of an input instance, the effectiveness of a classifier which uses it in both experiments are likely to vary. More importantly, the best feature types combination for each classifier in the classifiers set can be different, because each one of them is developed in a different experiment. To have a plausible way of comparison between the classifier members in a classifiers set, we a adopt a systematic approach to select the best feature type combination for each classifier.

4.3.1 Experiment Design

Two main experiments were carried out to evaluate the classifiers set developed for each of the argument unit classification and argumentative classification tasks, namely the cross-domain experiment and the in-domain experiment. Each experiment splits the corpus we presented in Subsection 4.2 into testing and training sets differently. The aim of both experiments is to develop our domain-robust approach for argument mining and evaluate its effectiveness across different web domains.

Cross-Domain Experiment In the cross-domain experiment, we developed a classifier for each task (argument unit classification and argumentative relation classification) and evaluate its ability to generalize over domains. The ability of the classifiers to generalize over domains is important since their ultimate goal is to extract an argument from the Web, which is characterized by its wide variety of content. For this purpose, We split the corpus into three testing sets which correspond to the three subcorpora of which our corpus consists, namely AIFdb, Araucaria, and WebDiscourse. For each task in our approach, we test the classifier we developed for the task on one testing set (subcorpus) after training it on the other subcorpora. Later on, we weight the effectiveness of the classifiers over the three testing sets by their size.
The main reason for this is the large difference among the subcorpora in terms of the count of their input instances. As we showed in Subsection 4.2.4, the count of the ArgumentativeRelation annotations in the Araucaria subcorpora is only 20 % of the ArgumentativeRelation annotations in the WebDiscourse subcorpora. This large difference between the two subcorpora will make the split in which the WebDiscourse subcorpora functions as a testing set easier than the one where the Araucaria subcorpora functions as a testing set because there are more different input instances to learn from in the first case.

In-Domain Experiment In the in-domain experiment, we developed a classifier for each task (argument unit classification and argumentative relation classification) and evaluate its effectiveness within each domain represented by our cross-domain web argument corpus. Even though we won't use the developed classifiers in this experiment for mining the arguments on the Web, their achieved effectiveness helps us have a basis of comparison to the classifiers developed in the cross-domain experiment. The goal of conducting this experiment is to estimate the effectiveness of the classifiers in each task while excluding the domain-effect. By excluding the domain-effect, we can differentiate between the effectiveness of the classifier to perform the task itself and its effectiveness in performing the same task in an unseen domain.

The in-domain experiment is broken up into three experiments one for each subcorpora in our cross-domain web argument corpus, assuming that each subcorpus represents a specific domain. For each experiment, we develop a classifier for each task in our approach (argument unit classification and argumentative relation classification) and evaluate its effectiveness it in 5-folds cross-validation setting. Thus, for each experiment the involved subcorpus is split into 5 testing sets and the classifiers are tested on each set after training it on the other sets. Then, the effectiveness of the classifiers is averaged over the 5 testing sets.

We split each subcorpus by randomly and uniquely assigning an annotated document to a testing set while trying to have a similar count of the ArgumentUnit and ArgumentativeRelation input instances in each of them. The random assignment of the documents helps to have a class distributions of the testing set that resembles the class distributions of the subcorpus. A class distribution of a testing set, training set or a corpus is the percentage of the input instances for each class in it. The class distributions of the subcorpus are our best knowledge about the percentage of the input instances for each class in the real world. Hence, having testing sets with a class distribution that is similar to that of the corpus guarantees us more reliable results in our experiment. The unique assignment of the documents to a specific testing set helps us to have testing sets that are exclusive, i.e. no common input instances exists in two testing sets. An input instance existing in two testing sets decrease the reliability of the results in our experiment since the input instance will occur in the training set for both testing sets. The final criterion which we tried to meet is having equal count of the ArgumentUnit and ArgumentativeRelation input instances in each testing set. Since the documents in the corpus contain a different count of the input instances, assigning only an equal count of documents to each testing set might lead to having a different count of the ArgumentUnit and Argumentative input instances in each of them. This can affect the results in our experiment as the effectiveness of a classifier is likely to decrease on the biggest testing set, as it will have fewer input instances to learn on in that split.

The tables 4.6, 4.7 and 4.8 show the class distribution of ArgumentUnit and ArgumentativeRelation input instances for the 5 splits of each subcorpus AIFdb, Araucaria and WebDiscourse respectively. Here, we show the class distribution only for testing sets for brevity and space limitations. When needed, the class distribution for a training set can be calculated by summing the class distributions for all other testing sets except the one with which the training set is associated.

Split	**ArgumentUnit**		**ArgumentativeRelation**	
	Argumentative	Non-argumentative	Related	Non-related
1	180 (38.54 %)	287 (61.46 %)	128 (12.81 %)	871 (87.19 %)
2	152 (32.34 %)	318 (67.66 %)	108 (6.54 %)	1567 (93.55 %)
3	162 (29.29 %)	391 (70.71 %)	141 (14.39 %)	839 (85.61 %)
4	159 (29.12 %)	387 (70.88 %)	151 (17.50 %)	712 (82.50 %)
5	169 (36.42 %)	295 (63.58 %)	126 (12.01 %)	923 (87.99 %)

Table 4.6: Class distribution of the ArgumentUnit and ArgumentativeRelation input instances for each split of AIFdb in the in-domain experiment

Split	**ArgumentUnit**		**ArgumentativeRelation**	
	Argumentative	Non-argumentative	Related	Non-related
1	159 (24.16 %)	499 (75.84 %)	71 (19.35 %)	296 (80.65 %)
2	185 (29.65 %)	439 (70.35 %)	95 (20.00 %)	380 (80.00 %)
3	175 (29.17 %)	425 (70.83 %)	65 (13.49 %)	417 (86.51 %)
4	193 (28.26 %)	490 (71.74 %)	72 (14.23 %)	434 (85.77 %)
5	178 (26.37 %)	497 (73.63 %)	63 (13.13 %)	417 (86.88 %)

Table 4.7: Class Distribution of the ArgumentUnit and ArgumentativeRelation annotations for each Split of Araucaria in the in-domain Experiment

Split	**ArgumentUnit**		**ArgumentativeRelation**	
	Argumentative	Non-argumentative	Related	Non-related
1	336 (35.44 %)	612 (64.56 %)	178 (6.35 %)	2624 (93.65 %)
2	290 (39.51 %)	444 (60.49 %)	165 (10.19 %)	1455 (89.81 %)
3	288 (35.47 %)	524 (64.53 %)	166 (9.25 %)	1629 (90.75 %)
4	333 (42.86 %)	444 (57.14 %)	193 (7.79 %)	2228 (92.03 %)
5	288 (45.14 %)	350 (54.86 %)	167 (10.01 %)	1502 (89.99 %)

Table 4.8: Class Distribution of the ArgumentUnit and ArgumentativeRelation annotations for each split of WebDiscourse in the in-domain Experiment

Figure 4.2 shows the classifiers' names developed for the cross-domain and in-domain experiments for each task in our approach. Having developed three classifiers for each task in the in-domain experiment we number them from 1 to 3, representing the subcorpora on which the experiment was based on: AIFdb, Araucaria and WebDiscourse respectively. The classifiers developed for the same task in the in-domain and the cross-domain experiments are grouped in classifiers sets. We call the classifier sets developed for the argument unit classification task as the *argument unit classifiers set* and the classifiers set developed for the argumentative relation classification task the as *argumentative relation classifiers sets.*

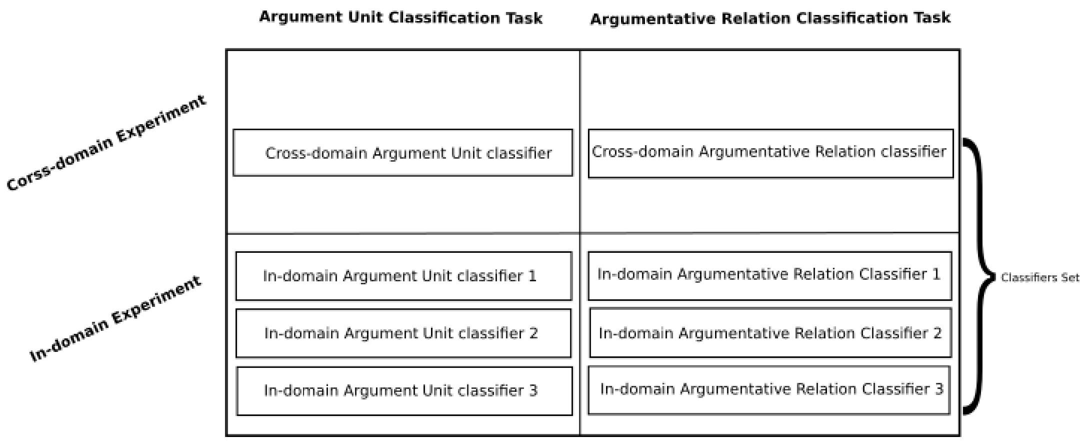

Argument Unit Classification Task
Argumentative Relation Classification Task
Corss-domain Experiment
In-domain Experiment
Cross-domain Argument Unit classifier
Cross-domain Argumentative Relation classifier
In-domain Argument Unit classifier 1
In-domain Argument Unit classifier 2
In-domain Argument Unit classifier 3
In-domain Argumentative Relation Classifier 1
In-domain Argumentative Relation Classifier 2
In-domain Argumentative Relation Classifier 3
Classifiers Set

Figure 4.2: The classifiers developed for each task and each experiment

We chose RandomForest [11] as the classifier type for all the classifiers developed in our approach. A *RandomForest* classifier uses a set of DecisionTree classifiers where each of them is trained on a randomly selected sample set of size n from the whole available training set. The size of the DecisionTree classifiers set in our experiments is 100.

Evaluation As we stated previously, in the cross-domain experiment we develop and evaluate our cross-domain argument unit and argumentative relation classifier which we will use to mine the arguments on the Web. In the in-domain experiment, the effectiveness of the developed classifiers is used for the comparison with the effectiveness of the cross-domain classifier in the same classifiers set. Since we developed three in-domain classifiers for each cross-domain classifier we average their effectiveness to compare the effectiveness of a cross-domain classifier with them.
As effectiveness measures, we use precision, recall, and F1-score. During evaluation, we focus more on the argumentative class in the argument unit classification task and on the related class in the argumentative relation classification. As we saw in Subsection 4.2.4 and 4.3.1, the class distributions for the the ArgumentUnit and ArgumentativeRelation input instances are skewed in all the splits the in-domain experiment and the cross-domain experiment, i.e. the percentage of the input instances with the negative classes (non-argumentative and non-related) is higher. Therefore, a classifier which is trained on such a split can achieve better effectiveness by always favoring the negative class since this decision is less risky. To avoid this problem, we always oversample the input instances labeled with the positive class by doubling them in the training set till the the class distribution is equally balanced. In the testing set, however, we don't oversample the input instances labeled with the positive class to preserve the original class distribution which is more similar to the one a classifier will face in the real world. Additionally, we focus more on the effectiveness achieved by the classifier in identifying the positive class by reporting the positive precision, recall, and F1-score.
During the evaluation of our developed classifiers sets, we concentrate more on positive precision than positive recall. The reason for this is that the argument mining approach we developed will be used mainly on the Web with the aim of assessing of their relevance according to the approach we discussed in Chapter 3. Our approach constructs an argument graph from the extracted arguments by merging those argument units used in multiple arguments if they are semantically equal. Later on, PageRank algorithm is used to rank the argument units in the graph according to the attention they get as premises. An invalid argumentative relation can severely affect the PageRank scores of its premise. Similarly, an invalid argument unit in the graph can severely affect

the PageRank scores of all the premises with which it is used as a conclusion in an argument. For this reason, we are more interested in avoiding classifying negative input instances as positive than classifying all of the positive instances on the Web to exclude invalid argumentative content or argument structure in our graph. In other words, the big size of the Web offers us the opportunity to trade the quality of the input instances classified as positive for their quantity.

4.3.2 Greedy Feature Type Selection

In the cross-domain and in-domain experiments, we developed a classifiers set for each classification task in our approach. A classifiers set consists of a cross-domain classifier and three in-domain classifiers. All the classifiers are of the same classifier type Random Forest, however, each classifier uses a different feature type combination. As we discussed previously, the achieved effectiveness of a classifier that uses an individual feature type or a combination of them will vary according to the experiment on which it is developed.
Finding the best combination of feature types that maximizes the effectiveness of a classifier developed in an experiment can be seen as a search problem. Given a set of feature types for the classifiers set, our approach represents all the possible combinations of it as a search space and searches for the combination that maximizes the effectiveness of the classifier. A *search space* is a directed graph whose nodes are all the possible solution candidates (in our case feature type combinations) and the edges between them are called operators. An *operator* is a transition between a solution candidate to another. With each solution candidate for the search problem we are trying to solve, a measure of merit is associated. In our case, the merit associated with each solution candidate (feature type combination) is the effectiveness achieved by the classifier using the feature type combination in the experiment.
A search space has an initial solution candidate and one or multiple best solution candidates. The initial solution candidate has a merit of zero while the best solution candidate has the maximum merit. When the search space is small, all the solution candidates can be traversed to find the best solution candidate; however, this is usually not the case. If we have for example 10 feature types, then the number of the possible feature type combinations is 2^10. Another factor for not visiting each solution candidate is the limitation of the resources existing on a computer. Therefore, a *search algorithm* is usually used to traverse a small part of the search space in a way that guarantees tractability and optimality or near optimality. *Optimality* is the quality of the solution candidate which the search algorithm chooses as the best solution candidate. By near optimality we mean that the best solution candidate found by the search algorithm might be not the best solution candidate but its merit

is very close to its merit.
To find the best feature type combination for a classifier in an experiment, we use a greedy search algorithm. A *greedy search algorithm* traverses the search space by always choosing the operator at each solution candidate that leads to the maximum gain in terms of merit.
Given a set of developed feature types, we generate all the possible combinations of them as solution candidates. We define our set of operators to match exactly the set of the given feature types. A solution candidate in our constructed search space has as many operators as the feature types it doesn't have, each leading to a solution candidate with the same feature types in the previous solution candidate in addition to the feature type represented by the operator. If there are no such an operator the algorithms stops and returns the last visited solution candidate as the best solution candidate.
We use the greedy search algorithm in choosing the best feature combination during the development of each classifier in the argument unit classifiers set and the argumentative relation classifiers set. This systematic development of the classifiers in both classifiers sets in addition to the usage of the same classifier type and the set of feature types during development helps us to perform a consistent comparison of their effectiveness in the both experiments we will conduct.

4.3.3 Argument Unit Classification

The goal of the argument unit classification task is to extract the argument units in a document on the Web as a prior step to classifying the potential argumentative relations between them. According to our simple argument model, an argument unit can have either the type argumentative or non-argumentative. An argument unit with the argumentative type indicates a valid argument unit, whereas an argument unit with the type non-argumentative denotes an invalid one, i.e. a non argumentative text segment. Technically, an argument unit is represented as an ArgumentUnit annotation category which spans a sentence. In Subsection 4.2, we created a corpus which covers three different domains and which is annotated with arguments as modeled with the simple argument model. Using this corpus, we designed two experiments with the purpose of developing an argument unit classifier that is able to extract argument units in a domain-robust fashion. In the cross-domain experiment, we divided our cross-domain argument web corpus into three splits whose testing sets are its three subcorpora. In this experiment, we developed an argument unit classifier and tested its effectiveness on the three splits. The in-domain experiment consists of three experiments.

Each experiment is conducted on a subcorpus of our cross-domain argument web corpus in a 5-fold cross-validation evaluation setting. For each experiment, an argument unit classifier is developed and evaluated. The argument unit classifiers developed in the two experiments uses the same classifier type which is Random Forest and belong to the argument unit classifiers set.
We created an argument unit classifiers set by developing feature types that provide the classifier with interesting aspects about an ArgumentUnit input instance that help in identifying its class. Thus, an ArgumentUnit annotation is handed into a classifier as an input instance whose class is the type of the ArgumentUnit. We consider the argumentative type of an argument unit as the positive class and the non-argumentative type as the negative class. As we discussed in Subsection 4.3.1, We will report the effectiveness of the classifier in this experiment as its average effectiveness for each split in the cross-domain experiment. As effectiveness measures, we use positive precision, positive recall, and positive F1-score and weighted F1-score to assess the effectiveness of the argument unit classifier.
To choose the best feature combination for each classifier in our classifier sets, we use our greedy feature type selection we introduced in Subsection 4.3.2. As we discussed in Subsection 4.3.1, we will concentrate during the evaluation of our classifiers in each experiment on its positive precision, because we will use these classifiers to mine the Web for arguments whose size offers us the opportunity to trade the quantity of the ArgumentUnit input instances classified as positive for their quality. We concentrate on the positive class, because as we saw in Subsection 4.3.1 the positive class is less represented in the training set. This skewed class distribution makes it harder for a classifier to distinguish it in the testing set since there are less representative input instances of the minority class. Therefore, we will associate with each feature type combination the weighted positive precision of the classifier on each testing set in the experiment. We weight each testing set by the count of its input instances. The main reason for this is the large difference in the size of the domains. This Search-based feature type selection during the development of the classifiers in the argument unit classifiers set allows us to make a consistent comparison between their effectiveness in the two different experiments we will carry.

Feature Types

A complete list of the developed feature types in the argument unit classification is:

- **Lexical Feature Types** :

- **Keyword n-grams (1-3)**: Keywords refer to about 300 words or word phrases extracted from a list of indicative terms for argumentation [22]. Examples from the list are "but", "consequently", and "because of". This list is used by Moens et al. [30] to develop a feature type for the classification of argument units in Araucaria corpus. We generate a feature that represents the percentage of each possible 1-3 sequences of the elements of this list in an ArgumentUnit annotation normalized by its length.
- **TransitionalPhrasesType n-grams (1-3) (TPT n-grams)**: This feature type relies on a list compiled by Study Guides and Strategies [6] of 149 transitional phrases which are categorized in 14 phrase types. The same list is used by Persing and Ng [35] in his work on assessing the argument strength of an essay. While they use the number of transitional phrases in an essay for each phrase type in an essay, we use the percentage of each possible 1-3 sequences of an occurrence of a phrase type in an ArgumentUnit annotation normalized by length.
- **Token n-grams (1-3)**: represents the percentage of every 1-3 possible sequences of token normalized by the text's length.
- **TopKToken n-grams (1-3)**: This feature type relies on the k most common words in the ArgumentUnit annotations in the training set. We create a feature for each 1-3 possible sequences of each of these tokens that represent its percentage in the ArgumentUnit annotation normalized by its length.
- **FirstToken n-grams (1-3)**: This feature type generates a boolean feature for every possible first 1-3 sequences of tokens in the ArgumentUnit annotation in the training set. For an ArgumentUnit annotation, we set those feature that represents its first 1-3 sequences of tokens.

• **Syntactic Feature Types** :

- **Part-Of-Speech (POS) n-grams (1-3)**: a part-of-speech tag is a tag which a word is assigned depending on its use and function (e.g. noun, verb, adjective). This feature type represents the percentage of each possible 1-3 sequences of part-of-speech tags normalized by the ArgumentUnit annotation's length.

• **Structural Feature Types**:

- **Punctuations**: the relative percentage of the tokens that contain punctuation marks to the all tokens in an ArgumentUnit annotation. The punctuation marks we considered are (commas, exclamation marks, question marks, colons, semicolons and dots).
- **SentenceStatistics**: the minimum, average, maximum and the standard deviation of the length of each token in an ArgumentUnit annotation.
- **ArgumentUnitLength (AuLength)** : This feature type captures the length of an ArgumentUnit annotation in different ways. This include the count of characters, syllables, tokens and phrases of the ArgumentUnit annotation. Additionally, the count of syllables per character, tokens per character, phrases per character, tokens per syllable, phrases per syllable and phrases per token.
- **ArgumentUnitPosition (AuPosition)** : The position of the ArgumentUnit annotation in the document and in the paragraph in which it occurred. Additionally the position of the paragraph in the document.
- **StatementBoundaries**: This feature type captures the first token and the last token of the ArgumentUnit annotation and their POS in addition to the first three characters and the last three characters of the ArgumentUnit annotation. We generate a boolean feature for each encountered value for them for all the ArgumentUnit annotations in the training set. Given an ArgumentUnit annotation, we set those feature which has the same values in it.

• **Sophisticated Feature Types**:

- **SentiWords**: This feature type captures the distribution of the negative or positive tokens in an ArgumentUnit annotation using SentiWordNet [16]. The name of the feature comes from the word *sentiment* which is a measure to the polarity of a word, i.e. how negative or positive it is perceived. For example, the word "stormy" has a negative score of 0.75 according to SentiWordNet. SentiWordNet learns a set of classifiers to give a positive and a negative score from 0 to 1.0 for a token by using the WordNet database. WordNet is a lexical database that groups English words into semantic groups, such as synonyms. We generate a numerical feature for each token in an ArgumentUunit annotation in the training set that has a positive score or negative score higher than a 0.4. For a given ArgumentUnit annotation, we set the value of each feature to the count of the occurrences of the word which the feature represents in it normalized by the length of the ArgumentUnit annotation.

- **GeneralInquirerCateogories (GIC)**: The General Inquirer is a dictionary that stores the categories of about 11,788 word meanings [4]. The categories of the words are sorted by their lemmas. A *lemma* is the root of the word which is usually found in the dictionary. For example, the lemma of the word "running" is "run". This feature type generates a feature for each category for all the token in the ArgumentUnit annotations in the training set. The value of the features for a given ArgumentUnit annotations is the distribution of the categories of all tokens' lemmas in an ArgumentUnit annotation, that is it for each token we increase its General Inquirer category by one. All the features are normalized by the length of the ArgumentUnit annotation in tokens.

Cross-domain Results

Table 4.9 shows the effectiveness of our cross-domain argument unit classifier using each individual feature type in addition to the best feature type combination found by our greedy feature type selection in the cross-domain experiment. To estimate the novelty of our approach and the difficulty of the argument unit classification task, we adopt a baseline which always chooses the positive class. The effectiveness of this baseline can be calculated from the class distribution of the ArgumentUnit input instances in the three testing sets split in the experiment. The positive precision is the division of the count of the input instances in the testing set which are labeled as positive to the count of all the input instances. The positive recall is equal to 1.0 because we are able to retrieve all the positive input instances in the testing set. As shown in Table 4.9, our greedy feature type selection chose the feature type combination: ArgumentUnitLength, TranistionalPhrasesType n-grams, Punctuations and ArgumentUnitPosition to be the best feature type combination with a positive precision of 0.429. We notice that most of the feature types in this combination belong to the structural set which shows the superiority of structural feature types in the domain-robust mining of argumentative content on the Web. While the effectiveness of the cross-domain argument unit classifier doesn't seem very high, it manages to beat the baseline with about %20 effectiveness gain. The results of the in-domain experiment for the argument unit classification task will help us better evaluate our cross-domain argument unit classifier because in the in-domain experiment we exclude the domain-effect by developing and evaluating our classifiers on the same corpus.

Feature Type	**Positive Precision**	**Positive Recall**	**Positive F1-score**	**F1-score**
Baseline	*0.337*	*1.000*	*0.501*	-
Keyword n-grams	0.388	0.424	0.404	0.593
TPT n-grams	0.389	0.155	0.217	0.582
Token n-grams	0.375	0.410	0.391	0.584
TopKToken n-grams	0.373	0.374	0.367	0.580
FirstToken n-grams	0.363	0.294	0.315	0.576
POS n-grams	0.366	0.392	0.373	0.573
Punctuations	0.374	0.583	0.452	0.551
SentenceStatistics	0.365	0.438	0.393	0.568
AuLength	0.364	0.408	0.380	0.570
AuPosition	0.364	0.558	0.416	0.505
StatementBoundaries	0.363	0.423	0.389	0.572
SentiWords	0.378	0.315	0.343	0.593
GeneralInquirerFrequency	0.388	0.348	0.358	0.590
AuLength, TPT n-grams, Punctuations, AuPosition	**0.429**	**0.514**	**0.452**	**0.606**

Table 4.9: Cross-domain argument unit classifier 's effectiveness results for each feature type and for the best feature type combination: ArgumentUnitLength, TransitionalPhrasesType n-grams, Punctuations, ArgumentUnitPosition

In-domain Results

The in-domain experiment consists of three smaller experiments in which we develop an argument unit classifier for each subcorpus in our cross-domain argument web corpus. As we discussed in Subsection 4.3.1, we call the classifiers with the names: in-domain argument unit classifier 1, in-domain argument unit classifier 2, in-domain argument unit classifier 3 for the subcorpora AIFdb, Araucaria and WebDiscourse respectively. For each in-domain argument unit classifier, we list the effectiveness of our developed in-domain argument unit classifiers using each individual feature type in addition to the best feature type combination found by our greedy feature type selection on each the experiment. We use the same baseline we used in the cross-domain experiment which always chooses the positive class. The positive precision achieved by of the baseline for each experiment can be calculated from the class distribution of the ArgumentUnit input instances in the its five testing splits. Table 4.10, Table 4.11 and Table 4.12 shows the effectiveness of each of the in-domain

classifiers developed on AIFdb, Araucaria and WebDiscourse respectively. The best feature type combination for each experiment is listed at the end of each table.
Noticeably, all of the in-domain argument unit classifiers performed very well by using the TopkToken n-grams, achieving a positive precision of 0.536 on the in-domain experiment on the AIFdb subcorpus, however, the positive precision of the cross-domain argument unit classifier by using just this feature type is relatively low (positive precision of 0.375). The difference between the contribution of TopToken n-grams in the in-domain experiment and the cross-domain experiment is a logical consequence of the design of the in-domain experiment where we exclude the domain-effect by training and testing the classifier on the same subcorpora. As we have shown in Subsection 4.3.1, we created the testing sets in the in-domain experiment in a by assigning documents to different testing sets randomly while trying to have a similar distribution of the ArgumentUnit input instances in the testing sets. The similar usage of language in a specific corpus and the randomness used to create the testing splits in our corpus guarantees a similar usage of language in each testing set. For example, in the Araucaria subcorpus, which includes a fair amount of law text, the word "suit" is more likely to always be used to mean a law case while it might be used in a different domain (like AIFdb) as an outfit. Hence, a lexical feature type which relies on counting words like TopKToken n-grams is likely be less effective in mining argument units on the Web.

Feature Type	**Positive Precision**	**Positive Recall**	**Positive F1-score**	**F1-score**
Baseline	*0.328*	*1.000*	*0.493*	-
Keyword n-grams	0.426	0.424	0.417	0.614
TPT n-grams	0.409	0.188	0.257	0.596
Token n-grams	0.482	0.538	0.506	0.660
TopKToken n-grams	0.536	0.599	0.564	0.699
FirstToken n-grams	0.479	0.381	0.424	0.647
POS n-grams	0.494	0.516	0.503	0.666
Punctuations	0.403	0.544	0.461	0.593
SentenceStatistics	0.431	0.461	0.443	0.623
AuLength	0.447	0.490	0.464	0.631
AuPosition	0.405	0.632	0.491	0.580
StatementBoundaries	0.390	0.604	0.473	0.569
SentiWords	0.395	0.298	0.339	0.602
GeneralInquirerFrequency	0.459	0.454	0.451	0.635
TopKToken n-grams, AuPosition	**0.578**	**0.616**	**0.594**	**0.725**

Table 4.10: In-domain argument unit classifier 1 's effectiveness results for each feature type and for the best feature type combination: TopKToken n-grams, ArgumentUnitPosition (AIFdb subcorpus)

Feature Type	**Positive Precision**	**Positive Recall**	**Positive F1-score**	**F1-score**
Baseline	*0.275*	*1.000*	*0.431*	-
Keyword n-grams	0.392	0.410	0.400	0.665
TPT n-grams	0.269	0.693	0.367	0.385
Token n-grams	0.375	0.410	0.391	0.584
TopKToken n-grams	0.522	0.462	0.488	0.729
FirstToken n-grams	0.328	0.365	0.344	0.625
POS n-grams	0.490	0.438	0.461	0.714
Punctuations	0.440	0.658	0.527	0.692
SentenceStatistics	0.404	0.421	0.412	0.673
AuLength	0.404	0.404	0.402	0.672
AuPosition	0.373	0.623	0.466	0.629
StatementBoundaries	0.339	0.506	0.404	0.611
SentiWords	0.352	0.370	0.360	0.643
GeneralInquirerFrequency	0.456	0.378	0.412	0.695
TopKToken n-grams, SentenceStatistics	**0.564**	**0.502**	**0.530**	**0.752**

Table 4.11: In-domain argument unit classifier 2 's effectiveness results for each feature type and for the best Feature type combination: TopKToken n-grams, SentenceStatistics, (Araucaria Subcorpus)

Feature Type	**Positive Precision**	**Positive Recall**	**Positive F1-score**	**F1-score**
Baseline	*0.392*	*1.000*	*0.562*	-
Keyword n-grams	0.444	0.424	0.433	0.559
TPT n-grams	0.377	0.233	0.233	0.481
Token n-grams	0.444	0.451	0.447	0.559
TopKToken n-grams	0.467	0.487	0.476	0.579
FirstToken n-grams	0.436	0.442	0.424	0.532
POS n-grams	0.436	0.477	0.455	0.552
Punctuations	0.410	0.509	0.451	0.517
SentenceStatistics	0.403	0.444	0.421	0.521
AuLength	0.418	0.461	0.437	0.535
AuPosition	0.490	0.612	0.453	0.599
StatementBoundaries	0.426	0.505	0.460	0.538
SentiWords	0.436	0.330	0.374	0.551
GeneralInquirerFrequency	0.452	0.494	0.470	0.564
TopKToken n-grams, AuPosition	**0.558**	**0.570**	**0.562**	**0.651**

Table 4.12: In-domain argument unit classifier 3 's effectiveness results for each feature type and for the best feature type combination: TopKToken n-grams, ArgumentUnitPosition (WebDiscourse Subcorpus)

Comparison of the Cross-domain and In-domain Results

Table 4.13 shows the effectiveness of the cross-domain argument unit classifiers together with the three in-domain argument unit classifiers and their average. We weight the effectiveness of an in-domain argument unit classifier by the count of the ArgumentUnit input instances in the subcorpora on which it was developed for the consistency with the design of the cross-domain experiment. As we discussed in Subsection 4.3.1, we weight the effectiveness of the cross-domain argument unit classifier by the size of the testing set to balance the large difference of the ArgumentUnit input instances in the three subcorpora.
The weighted positive precision of the in-domain argument unit classifiers amounts to 0.565, compared to a positive precision of 0.429 achieved by our cross-domain argument unit classifier. Even though all of the in-domain argument unit classifiers are evaluated on the same corpus, their effectiveness in the classification of ArgumentUnit input instances labeled with positive stays only slightly higher than the effectiveness of the corresponding cross-domain classifier. This close performance of our argument unit classifiers set in the

cross-domain and in-domain experiments indicates the possibility of developing domain-robust argument unit classifiers and the appropriateness of our approach of guaranteeing domain-robustness. Nevertheless, the low effectiveness of the cross-domain classifier and the in-domain classifiers reflects the difficulty of extracting argument units on the Web.

Classifier Name	**Positive Precision**	**Positive Recall**	**Positive F1-score**	**F1-score**
Baseline	*0.337*	*1.000*	*0.501*	-
In-domain argument unit classifier 1	0.578	0.616	0.594	0.725
In-domain argument unit classifier 2	0.564	0.502	0.530	0.752
In-domain argument unit classifier 3	0.558	0.570	0.562	0.651
In-domain argument unit classifier (Weighted)	**0.565**	**0.559**	**0.560**	**0.704**
Cross-domain argument unit classifier	**0.429**	**0.514**	**0.452**	**0.606**

Table 4.13: Cross-domain vs in-domain argument unit classifiers effectiveness comparison

4.3.4 Argumentative Relation Classification

The second task in our domain-robust argument mining approach is the argumentative relation classification task. Seen as a sequential task to argument unit classification, this task aims at the extraction of the argumentative relations between argument unit pairs. As we showed in Chapter 3, argumentative relations constitute the main component of our argument structure, which our PageRank for argument relevance exploits to score the argument units by recursively measuring the attention they get as premises pushed for or against the arguments on the Web. Thus, the simple argument model we use as an abstraction for the arguments on the Web represents an argumentative relation as an ordered pair of argument units which has a premise, a conclusion, and a type. Technically, argumentative relations are represented by the ArgumentativeRelation annotation category which defines a premise and a conclusion that refer to ArgumentUnit annotations and have the type argumentative. The type of an ArgumentativeRelation can be either related, indicating an existing relation between the premise and the conclusion or non-related indicating a nonexistent relation.

In Subsection 4.3.1, we introduced the design of the cross-domain experiment where we developed our cross-domain argumentative relation classifier. In the cross-domain experiment, we divided our cross-domain argument web corpus, which constitutes from three subcorpora, into three splits. The testing set of each split is one of the subcorpora while the training set consists of the other two subcorpora. Parallel to the cross-domain experiment, we carry out the the in-domain experiment which is broken up into three experiments done on each of the subcorpora of the cross-domain argument web corpus in 5-folds cross-validation evaluation setting. The class distribution of the ArgumentativeRelation annotations in the corpus is severely skewed where the percentage of the argumentative relation annotations labeled as related is only about 10 %.

In the cross-domain experiment and the in-domain experiment we developed four argumentative relation classifier (a cross-domain classifier and three in-domain classifiers), comprising our argumentative relation classifiers set. The cross-domain classifier will be used to mine the arguments on the Web, while the in-domain classifiers will be used to evaluate the ability of our classifier to work robustly across domains. An ArgumentativeRelation annotation is handed into the argumentative relation classifiers as an input instance associated with the type of the annotation as its class as well as its conclusion and premises.

The dependency of an ArgumentativeRelation annotation category on the ArgumentUnit annotation category to indicate the premises and the conclusion of an ArgumentativeRelation annotation made us provide the ground-truth ArgumentUnit annotations in the training and the testing set for each classifier. The availability of correctly extracted argument units doesn't match a real working scenario of our pipeline since it assumes a perfect effectiveness of the previous task. This assumption, however, allows us to evaluate the argumentative relation classification task independently.

We created an argumentative relation classifiers set by developing feature types that provide the classifier with interesting aspects about an ArgumentativeRelation input instance that help in identifying its class. We consider the related type of an argument unit as the positive class and the non-related type as the negative class. As we discussed in Subsection 4.3.1, We will report the effectiveness of the classifier in this experiment as its average effectiveness for each split in the cross-domain experiment. As effectiveness measures, we use precision, positive precision, positive recall, and positive F1-score and weighted F1-score to assess the effectiveness of the argumentative relation classifier.

To choose the best feature combination for each classifier in our classifier sets, we use our greedy feature type selection we introduced in Subsection 4.3.2.

As we discussed in Subsection 4.3.1, we will concentrate during the evaluation of our classifiers in each experiment on its positive precision, because we will use these classifiers to mine the Web for arguments whose size offers us the opportunity to trade the quantity of the ArgumentativeRelation input instances classified as positive for their quality. We concentrate on the positive class, because as we saw in Subsection 4.3.1 the positive class is less represented in the training set (the count of ArgumentativeRelation annotations in the Araucaria subcorpus is 20 % of their count in WebDiscourse subcorpus). This skewed class distribution makes it harder for a classifier to distinguish the input instances labeled with it in the testing set since there are less representative input instances of the minority class. Therefore, we will associate with each feature type combination the weighted positive precision of the classifier on each testing set in the experiment. We weight each testing set by the count of its input instances. The main reason for this is the large difference in the size of the domains. This Search-based feature type selection during the development of the classifiers in the argumentative relation classifiers set allows us to make a consistent comparison between their effectiveness in the two different experiments we will carry.

Feature Types

Seen as a relation between two ArgumentUnit input instances (a premise and a conclusion), some of the feature types for ArgumentativeRelation input instances relies on the feature types defined for ArgumentUnit annotations in Subsection 4.3.3.

- **Lexical Feature Types**
  - **ArgumentTopKToken n-grams (ArTopKToken n-grams)** : The TopKToken n-grams feature type of the conclusion in addition to the TopKToken n-grams feature type of the premise.
  - **ArFirstToken n-grams(ArFirstToken n-grams)**: The FirstToken n-grams feature type of the conclusion in addition to the FirstToken n-grams feature type of the premise.
  - **JaccardSimilarity**: The count of the common tokens in the premise and in the conclusion of the ArgumentativeRelation annotation. Additionally, the count of the tokens in the premise which don't occur in the conclusion. The two features are normalized by the count of the tokens in the premise.
- **Syntactic Feature Types**

  - **ArgumentPOS n-grams**: The POS n-grams feature type of the conclusion in addition to the POS n-grams feature type of the premise.

- **Structural Feature Types**
  - **ArgumentBinaryPositionalOrder (ArBPO)**: Three boolean features indicating whether the premise precede the conclusion or whether the conclusion occurs after the premise or whether the argumentative relation is reflexive, i.e. if the premise and the conclusion is the same ArgumentUnit annotation.
  - **ArgumentPositionalOffset (ArPO)**: The count of the sentences between the conclusion and the premise weighted by the count of the sentences in the document. Additionally, The count of the argument units between the conclusion and the premise weighted by the count of the argument units in the document.
  - **ArgumentPunctuations**: The Punctuations feature type for the premise in addition to the Punctuations feature type for the conclusion.

- **Sophisticated Feature Types**
  - **ArgumentGeneralInqurierCategories (ArGIC)**: The GeneralInqurierCategories feature type of the conclusion in addition to the GeneralInqurierCategories feature type of the premise.

Cross-domain Results

Table 4.18 shows the effectiveness of our cross-domain argumentative relation classifier for each developed feature type. Additionally, we show the effectiveness of our baseline which amounts to 0.104 positive precision. The best feature type combination we found by using our greedy feature type selection is: ArgumentPunctuations, ArgumentBinaryPositionalOrder and Argument-TopKToken n-grams. Despite the severely skewed distribution of the ArgumentativeRelation input instances, our cross-domain argumentative relation classifier is able to achieve a positive precision of 0.330 which is two times higher than the effectiveness achieved by the baseline. Nevertheless, these results implies that 67% of the ArgumentativeRelation input instances classified as related are incorrect.
As we showed in Chapter 3, our approach for assessing the relevance of an argument relies on argumentative relations to model the argument structure needed by our PageRank algorithm to score its argument units. Thus the low

effectiveness achieved by our cross-domain argumentative classifier is likely to severely affect the PageRank scores. To better analyze the contribution of the domain-effect on the argumentative relation classification task we introduce the results achieved by our argumentative relation classifiers set in the in-domain experiments.

Feature Type	Positive Precision	Positive Recall	Positive F1-score	F1-score
Baseline	0.104	1.000	0.187	-
ArTopKToken n-grams	0.148	0.026	0.041	0.821
ArFirstToken n-grams	0.110	0.347	0.113	0.581
JaccardSimilarity	0.113	0.441	0.179	0.604
ArgumentPOS n-grams	0.168	0.034	0.056	0.823
ArBPO	0.141	0.680	0.231	0.552
ArgumnetPositionalOffset	0.136	0.521	0.212	0.604
ArgumentPunctuations	0.147	0.200	0.167	0.781
ArGIC	0.260	0.027	0.045	0.824
ArgumentSentenceStatistics	0.121	0.045	0.063	0.814
ArgumentPunctuations , ArBPO, ArTopKToken n-grams	**0.330**	**0.062**	**0.096**	**0.828**

Table 4.14: Cross-domain argumentative relation classifier 's effectiveness results for each feature type and for the best feature type combination: ArgumentPunctuations, BinaryArgumentPositionalOrder, ArgumentTopKToken n-grams

In-domain Results

In the in-domain experiment, we develop three argumentative relation classifiers on the three subcorpora of our cross-domain argument web corpus. As we showed in Subsection 4.3.1, we split each subcorpora in 5 testing sets and conduct an experiment on it in a 5-folds cross-validation evaluation setting, resulting in three experiments. In each experiment, we develop an in-domain argumentative relation and number them from 1 to 3 according to the subcorpora on which it is developed AIFdb, Araucaria and WebDiscourse respectively. The distribution of the ArgumentativeRelation annotations in each subcorpora is highly skewed but vary according to corpora. Thus, the effectiveness of the baseline we suggested when we designed will vary as well. We use the same feature types we used for the cross-domain experiment and use the greedy feature type selection approach to choose the best feature type combination for each classifier. This will allow us to carry out a consistent comparison of the

effectiveness of the in-domain classifiers to the effectiveness of the cross-domain classifier.
Table 4.15, 4.16 and 4.17 show the effectiveness of the three in-domain experiments for the subcorpora AIFdb, Araucaria and WebDiscourse respectively. In comparison with the first two classifiers, the argumentative relation classifier developed on WebDiscourse achieves a relatively lower positive precision of 0.403; however as we showed in Subsection 4.2.4, the distribution of the ArgumentativeRelation input instances in this corpus is more skewed and amounts to only 8% of the whole count of the input instances. In the three experiments, our in-domain argumentative relation classifiers achieve 4 times higher positive precision than the listed baseline. The feature type ArgumentTopKToken n-grams performs very well in the tree experiment similar to TopKToken n-grams in the in-domain experiment for argument unit classification, emphasizing the superiority of lexical feature types in the in-domain experiments on argument mining in general to other feature types.

Feature Type	**Positive Precision**	**Positive Recall**	**Positive F1-score**	**F1-score**
Baseline	0.117	1.000	0.208	-
ArTopKToken n-grams	0.405	0.190	0.255	0.843
ArFirstToken n-grams	0.013	0.200	0.024	0.636
JaccardSimilarity	0.167	0.580	0.258	0.657
ArgumentPOS n-grams	0.446	0.191	0.264	0.848
ArBPO	0.182	0.862	0.298	0.573
ArgumnetPositionalOffset	0.221	0.722	0.336	0.710
ArgumentPunctuations	0.281	0.321	0.299	0.818
ArGIC	0.488	0.195	0.276	0.852
ArgumentSentenceStatistics	0.358	0.230	0.378	0.841
ArgumentPunctuations, ArGIC	**0.581**	**0.215**	**0.308**	**0.859**

Table 4.15: In-domain argumentative relation classifier 1 's effectiveness results for each feature type and for the best feature type combination: ArgumentPuctuations, ArgumentGeneralInqurierCategories (AIFdb subcorpus)

Feature Type	**Positive Precision**	**Positive Recall**	**Positive F1-score**	**F1-score**
Baseline	0.158	1.000	0.272	-
ArTopKToken n-grams	0.491	0.256	0.330	0.814
ArFirstToken n-grams	0.155	0.598	0.223	0.407
JaccardSimilarity	0.175	0.664	0.277	0.508
ArgumentPOS n-grams	0.489	0.236	0.316	0.813
ArBPO	0.175	0.429	0.247	0.640
ArgumnetPositionalOffset	0.188	0.772	0.302	0.486
ArgumentPunctuations	0.270	0.342	0.298	0.759
ArGIC	0.449	0.201	0.276	0.806
ArgumentSentenceStatistics	0.279	0.245	0.260	0.775
ArGIC ArTopKToken n-grams	**0.616**	**0.250**	**0.348**	**0.826**

Table 4.16: In-domain argumentative relation classifier 2 's effectiveness results for each feature type and for the best feature type combination: ArgumentGeneralInqurierCategories, ArgumentTopKToken n-grams (Araucaria subcorpus)

Feature Type	**Positive Precision**	**Positive Recall**	**Positive F1-score**	**F1-score**
Baseline	0.084	1.000	0.155	-
ArTopKToken n-grams	0.250	0.037	0.064	0.873
ArFirstToken n-grams	0.155	0.256	0.193	0.835
JaccardSimilarity	0.092	0.451	0.152	0.658
ArgumentPOS n-grams	0.129	0.016	0.029	0.869
ArBPO	0.156	0.753	0.257	0.703
ArgumnetPositionalOffset	0.135	0.461	0.209	0.758
ArgumentPunctuations	0.135	0.184	0.154	0.837
ArGIC	0.272	0.030	0.054	0.873
ArgumentSentenceStatistics	0.131	0.049	0.071	0.863
ArgumentPOS n-grams, ArGIC	**0.403**	**0.038**	**0.069**	**0.875**

Table 4.17: In-domain argumentative relation classifier 3 's effectiveness results for each feature type and for the best feature type combination: ArgumentPOS n-grams, ArgumentGeneralInqurierCategories (WebDiscourse subcorpus)

Comparison of the Cross-domain and In-domain Results

Table 4.18 shows the effectiveness of the cross-domain argumentative relation classifier together with the in-domain classifiers. The effectiveness of the three in-domain classifiers is weighted with the size of the corpus on which they were developed. The main reason for this is to balance the large difference of the ArgumentativeRelation input instances' count in the three subcorpora (the count of ArgumentativeRelation input instances in WebDiscourse is about 2,000 while its count is about 10,000 in Araucaria). Similar to the argument unit classification task, the positive precision of our cross-domain argumentative relation classifier appears to be fairly close to the weighted positive precision of the in-domain classifiers. This relatively close performance indicates the possibility of developing argumentative relation classifiers that are able to generalize over domains and the appropriateness of our approach for developing it. Nevertheless, the low performance of the cross-domain and in-domain classifiers shows the difficulty of extracting argumentative relations on the Web.

Classifier Name	**Positive Precision**	**Positive Recall**	**Positive F1-score**	**F1-score**
Baseline	*0.104*	*1.0*	*0.187*	-
In-domain argumentative relation classifier 1	0.581	0.215	0.308	0.859
In-domain argumentative relation classifier 2	0.616	0.250	0.348	0.826
In-domain argumentative relation classifier 3	0.403	0.038	0.069	0.875
In-domain argumentative relation classifier (Weighted)	**0.485**	**0.119**	**0.178**	**0.864**
Cross-domain argumentative relation classifier	**0.330**	**0.062**	**0.096**	**0.828**

Table 4.18: Cross-domain vs In-domain Argumentative Relation Classifiers ' Effectiveness Comparison

Chapter 5

Conclusion

In this work, we took the first steps on the way of developing an argument search engine whose goal is to retrieve the arguments on the Web which are relevant to a user's information need. Argument relevance is a quality criterion, among others, which attributes an argument and indicates locally how much its premises contribute to its conclusion and globally how much it is useful for a specific discussion.
In the last two decades, the Web has become a favorite place for people to discuss different topics on different platforms, such as social media. The wide spectrum of content on the Web in terms of topic or platform makes it hard to mine the Web for arguments since it covers collections of different documents (domains) on different levels. Nevertheless, the enormous size of the Web makes it an invaluable knowledge source for mining relevant arguments to a user's information need.
Motivated by these rewards and to tackle these difficulties, we studied the following two research questions:

1. ***How to model argument relevance computationally on the Web?***
2. ***How to domain-robustly mine the Web for arguments ?***

5.1 Contributions

Given the success of PageRank algorithm [31] in assessing relevance of web pages, we introduced a novel approach to assessing argument relevance computationally by using PageRank algorithm [45]. Our approach relies on building an argument graph model to represent argumentative content on the Web. In this argument graph, we create a node for each argument on the Web and construct a directed edge between two arguments if the conclusion of the target argument is semantically equal to a premise of the source argument. This

structure of the argument graph is used by PageRank algorithm to score the contribution of an argument unit in it according to the extent with which it gets used as a premise.
The second contribution in this direction is the construction of a ground-truth argument graph from the arguments on AIFdb [25], considering two argument units to be semantically equal only if they have the same text. The argument graph contains about 26,000 arguments and 30,000 argument units. We created a ground-truth benchmark argument relevance ranking by applying our PageRank approach on the constructed graph. To evaluate our approach, we compared the ground-truth rankings for a set of arguments in the constructed graph with a set of rankings produced by experts in computational linguistics. The evaluation revealed a positive correlation of 0.28 between the two rankings in terms of Kendall's τ, indicating the impact and the correctness of our PageRank for argument relevance approach and the reliability of our ground-truth rankings.
To bring our approach to the Web, we introduced a simple argument model to represent arguments on the Web in accordance with the argument structure needed by the approach. The model represents an argument as a set of argument units and a set of argumentative relations. Technically, an argument unit is considered to be a complete sentence and an argumentative relation is considered to be a directed pair of argument units and states the premise and the conclusion of the relation.
In the pursuit of studying the second research question, I carried out two experiments to develop a domain-robust argument mining which made our main contribution in this direction. The approach is designed as a pipeline which consists of two sequential tasks, namely the argument unit classification and the argumentative relation classification. Most of the corpora used in developing conventional argument mining tasks cover only one domain because they are usually created using documents form the same source. The lack of corpora which cover multiple domains led to our second contribution which is the construction of a cross-domain argument web corpus. This corpus comprises of three subcorpora: Araucaria [39], WebDiscourse [19] and AIFdb (without Araucaria) [25] which are annotated with arguments according to our simple argument model.
The cross-domain web argument corpus was used to design the two experiments which we conducted to develop our domain-robust argument mining approach. In the first experiment, called cross-domain experiment, we developed and evaluated our classifier by testing it on a subcorpus after training it on the other subcorpora. Since each subcorpus represents a different domain, in the cross-domain experiment, we studied the ability of an argument mining classifier to carry out its learned knowledge across domains. The second ex-

periment, called in-domain experiment, was done to evaluate the effectiveness of our classifiers in the cross-domain experiment by training and testing them on the same subcorpora thus excluding the difficulty faced in the generalization over domains. In the cross-experiment, I achieved a positive precision of 0.429 in the task of argument unit classification, outperforming a baseline which achieved a positive precision of 0.337. In the same experiment, the positive precision for the argumentative relation classification task was 0.33 which outperformed the baseline whose positive precision amounted to 0.104. See Subsection 4.3.3 and 4.3.4 for more details.
In both tasks, the positive precision achieved in the in-domain experiment were 0.565 and 0.485 for the argument unit classification and the argumentative relation classification tasks respectively. These results indicate a positive outcome in developing an argument mining approach which is able to work effectively across domains. Nonetheless, it also shows that there is still room for improvement. In addition, the low effectiveness achieved in the in-domain experiments highlights the seriousness of the obstacles a conventional argument mining approach will encounter in extracting arguments from the Web.
To summarize, I list the contributions for each research question below:

1. ***How to model argument relevance computationally on the Web?***
    - An Approach to Assessing Argument Relevance on the Web
    - A ground-truth Argument Graph Together with a Ground-truth Benchmark Argument Relevance Rankings
2. ***How to domain-robustly mine the Web for arguments ?***
    - A Cross-domain Argument Web Corpus
    - A First Cross-domain Argument Mining Approach

5.2 Future Work

- Domain-robust Argument Unit Identification

  Our simple argument model represents an argument unit on the sentence level for simplicity. Moens et al. [30] took a similar decision during their work on argument mining in Araucaria. Despite its simplicity, the main disadvantage of this approach is that multiple argument units which occur in a sentence are impossible to locate. Additionally, argument units which cross sentence boundaries cause both sentences to be classified as argumentative. While there is some research on argument unit identification in the essays domain byPersing and Ng [36] and Stab and Gurevych

[42], the question of how to identify argument units on the Web is still open for research.
Most of the argument mining approaches, including ours, are designed as a pipeline, i.e. a sequence of tasks which depend on each other in a sequential order. The main disadvantage of this design is the propagation of errors increasingly through the pipeline [36]. Therefore, the simplification we did by segmenting a document into sentences can be one reason for the low effectiveness of our pipeline. Adding an argument unit identification task before the argument unit classification task can improve the results of the subsequent tasks.

- Domain-specific Argument Mining Approach

  A possible way to adapt our in-domain argument mining pipelines to handle a document from multiple domains is to add a domain classification task at the beginning of the pipeline. A domain classifier can be trained easily on our cross-domain argument web corpus to identify the domain of a given document. Our suggested domain-specific argument mining approach will thus constitute of a domain classification task which will identify the domain of an input document before handing it to the conventional in-domain argument mining pipeline which is responsible for this domain. This approach, however, is hard to use for mining arguments on the Web where we have a large and increasing growing number of domains.

- Argument Synthesis

  Our approach to assessing argument relevance uses PageRank to score an argument unit in an argument graph based on the magnitude of its usage as a premise in different arguments. The relevance of an argument is then estimated by aggregating the PageRank scores of the premises of the argument [45]. Our simple argument model represents an argument as a set of argument units and a set of argumentative relations which represent the structure of the argument. The cross-domain argument mining approach, however, doesn't determine which sets of argumentative relations establish a complete argument. Since humans tend to use multiple premises in their arguments (see Figure 3.6b), having an end-to-end argument mining approach which aggregates argumentative relations is crucial for our PageRank approach to fully work.

- Paraphrasing for Argument Graph Construction

  The PageRank for argument relevance approach [45] relies on constructing an argument graph for the estimation of the relevance of an argument. In the process of constructing our ground-truth argument graph, we created an edge between two arguments if the conclusion of the target argument is semantically equal to a premise in the source argument. We considered two argument units to be semantically equal if they have the same text. Humans, however, can and tend to express the same idea differently in the same language. We call the rewording of a sentence differently *paraphrasing*. In my thesis, we studied a simple approach to detecting paraphrasing while constructing our graph (see Subsection 3.3.2) which did not improve the structure of the argument graph needed by our approach. A more focused study with more sophisticated paraphrasing detection can bring more matches among the argument units, leading to better PageRank scores.

Chapter 6

Appendix

This list of expressions is used to detect statements which are semantically equivalent but are expressed differently by using an extra expression. A indicate a set of possible alternatives while ( ) indicate an optional word.

- i|you|he|she|it|we|they know|think|feel|believe|agree|maintain(s) (that)
- in (my|your|his|her|its|our|their) view|opinion|mind
- from (my|your|his|her|its|our|their) (perspective|point of view|viewpoint)
- therefore|thus|hence|so|as a consequence|consequently|as a result|as a matter of fact
- firstly|secondly|thirdly|finally|on the one hand|on the other hand
- also|moreover|in addition|likewise|besides|additionally|furthermore|plus
- still|however|but|yet|notwithstanding|nevertheless|nonetheless
- indeed|in fact|anyway|anyhow

Bibliography

[1] Createdebate. `www.createdebate.com`. Accessed: 2016-09-27.

[2] Debatepedia. `www.debatepedia.idebate.org`. Accessed: 2016-09-22.

[3] Essayforum. `essayforum.com`. Accessed: 2016-09-28.

[4] Study guides and strategies. http://www.wjh.harvard.edu/ inquirer/homecat.htm.

[5] Procon.org. `www.procon.org`. Accessed: 2016-09-27.

[6] Study guides and strategies. `www.studygs.net/wrtstr6.htm`. Accessed: 2017-01-01.

[7] Edits. `www.edits.fbk.eu/`. Accessed: 2016-09-22.

[8] Omer Aristotle and George A Kennedy. *On rhetoric: A theory of civic discourse*. Oxford University Press, 2006.

[9] Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

[10] J Anthony Blair. *Groundwork in the theory of argumentation: selected papers of J. Anthony Blair*, volume 21. Springer Science & Business Media, 2011.

[11] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[12] Elena Cabrio and Serena Villata. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 208–212. Association for Computational Linguistics, 2012.

[13] Carlos Chesñevar, Sanjay Modgil, Iyad Rahwan, Chris Reed, Guillermo Simari, Matthew South, Gerard Vreeswijk, and Steven Willmott. Towards an argument interchange format. *The Knowledge Engineering Review*, 21 (04):293–316, 2006.

[14] T Edward Damer. *Attacking faulty reasoning.* Cengage Learning, 2012.

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

[16] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422. Citeseer, 2006.

[17] Trudy Govier. *A practical study of argument.* Cengage Learning, 2013.

[18] Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. *International corpus of learner English.* Presses universitaires de Louvain, 2002.

[19] 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*, pages 2127–2137. Citeseer, 2015.

[20] Ivan Habernal and Iryna Gurevych. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

[21] Ivan Habernal and Iryna Gurevych. Argumentation mining in user-generated web discourse. *arXiv preprint arXiv:1601.02403*, 2016.

[22] Alistair Knott and Robert Dale. Using linguistic phenomena to motivate a set of rhetorical relations. *Human Communication Research Centre, University of Edinburgh*, 1993.

[23] Namhee Kwon, Liang Zhou, Eduard Hovy, and Stuart W Shulman. Identifying and classifying subjective claims. In *Proceedings of the 8th annual international conference on Digital government research: bridging disciplines & domains*, pages 76–81. Digital Government Society of North America, 2007.

[24] John Lawrence and Chris Reed. Aifdb corpora. In *COMMA*, pages 465–466, 2014.

[25] John Lawrence, Floris Bex, Chris Reed, and Mark Snaith. Aifdb: Infrastructure for the argument web. In *COMMA*, pages 515–516, 2012.

[26] Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. Context dependent claim detection. 2014.

[27] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[28] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[29] Raquel Mochales and Marie-Francine Moens. Study on the structure of argumentation in case law. In *Proceedings of the 2008 Conference on Legal Knowledge and Information Systems*, pages 11–20, 2008.

[30] Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. Automatic detection of arguments in legal texts. In *Proceedings of the 11th international conference on Artificial intelligence and law*, pages 225–230. ACM, 2007.

[31] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[32] Raquel Mochales Palau and Marie-Francine Moens. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 98–107. ACM, 2009.

[33] Andreas Peldszus. Towards segment-based recognition of argumentation structure in short texts. *ACL 2014*, page 88, 2014.

[34] Andreas Peldszus and Manfred Stede. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 938–948, 2015.

[35] Isaac Persing and Vincent Ng. Modeling argument strength in student essays. 2016.

[36] Isaac Persing and Vincent Ng. End-to-end argumentation mining in student essays. In *Proceedings of NAACL-HLT*, pages 1384–1394, 2016.

[37] Chris Reed and Glenn Rowe. Araucaria: Software for argument analysis, diagramming and representation. *International Journal on Artificial Intelligence Tools*, 13(04):961–979, 2004.

[38] Niall Rooney, Hui Wang, and Fiona Browne. Applying kernel methods to argumentation mining. In *FLAIRS Conference*, 2012.

[39] Glenn Rowe and Chris Reed. Argument diagramming: The araucaria project. In *Knowledge Cartography*, pages 164–181. Springer, 2008.

[40] Christian Stab and Iryna Gurevych. Annotating argument components and relations in persuasive essays. In *COLING*, pages 1501–1510, 2014.

[41] Christian Stab and Iryna Gurevych. Identifying argumentative discourse structures in persuasive essays. In *EMNLP*, pages 46–56, 2014.

[42] Christian Stab and Iryna Gurevych. Parsing argumentation structures in persuasive essays. *arXiv preprint arXiv:1604.07370*, 2016.

[43] Stephen Edelston Toulmin. 77ie uses of argument, 1958.

[44] Frans H Van Eemeren, Rob Grootendorst, and A Francisca Sn Henkemans. *Argumentation: Analysis, evaluation, presentation.* Routledge, 2002.

[45] Henning Wachsmuth, Benno Stein, and Yamen Ajjour. "PageRank" for Argument Relevance. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 17)(to appear)*, April 2017.

[46] Douglas Walton. *Fundamentals of critical argumentation.* Cambridge University Press, 2005.