Imperial College London
Department of Computing

Using Argumentation to improve classification in Natural Language problems

Lucas Carstens

February 2016

Supervised by Professor Francesca Toni

Submitted in part fulfilment of the requirements for the degree of Doctor of Philosophy in Computing of Imperial College London and the Diploma of Imperial College London

Copyright declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work

Abstract

Argumentation has proven successful in a number of domains, including multi-agent systems and decision support in medicine and engineering. We propose its application to a domain yet largely unexplored by argumentation research: computational linguistics. Over the past decade or so advances in this field have commonly relied on data-driven solutions, i.e. machine learning. Recently, however, there appears to be a growing consent that, in order to achieve significant advances in certain areas of artificial intelligence, in general, and computational linguistics, in particular, we may need to consider data-driven approaches in unison with reasoning-, logic- and rule-based solutions. To this end we have developed a novel classification methodology that incorporates reasoning through argumentation with supervised classifiers. We train classifiers and then *argue* about the validity of their output. To do so we identify arguments that formalise prototypical knowledge of a problem and use them to correct misclassifications. We thus have at our disposal multiple ways of incorporating knowledge in classification and are able to integrate knowledge as it becomes available, without the need to retrain classifiers.

We illustrate our methodology on two tasks. On the one hand we address *binary cross-domain sentiment polarity classification*, where we train classifiers on one corpus, e.g. Tweets, to identify positive/negative polarity, and classify instances from another corpus, e.g. sentences from movie reviews. On the other hand we address a form of argumentation mining that we call *Relation-based Argumentation Mining*, where we classify pairs of sentences based on whether the first sentence attacks or supports the second, or whether it does neither. Whenever we find that one sentence attacks/supports the other we consider both to be argumentative, irrespective of their stand-alone argumentativeness. For both tasks we improve classification performance when using our methodology, compared to using standard classifiers, only.

Acknowledgements

The impact that the people around me have had on my work, and the very possibility of pursuing a PhD, cannot be overstated. Above anyone else I need to express my gratitude to Professor Francesca Toni. As my supervisor she has tirelessly supported me in my academic endeavors, expending time, effort, and expertise far beyond any reasonable expectations. Without her encouragement and support embarking on this journey would have been impossible, much less completing it.

As my second supervisor, Marc Deisenroth has offered his guidance on many occasions, and many of the details of my work would not be in their proper place without his help. The depth of his knowledge has driven my aspirations to continually learn more about the fields I work in.

The entire Computational Logic and Argumentation Group, and especially Valentinos Evripidou and Antonio Rago, deserve my thanks in equal measure. Much of my work has relied on their research and expertise, and without them, I would have surely produced a much more shallow exploration of the field of argumentation. Just the same my work has depended on the help of countless friends, who, time and again, were inundated with my requests to annotate text, without which none of my work would have been possible. I would also like to express my gratitude to Anthony Hunter and Chris Reed, who have so readily agreed to read and examine my thesis, and who have provided invaluable feedback on my work.

Ulrich Schaechtle has proven to be a reliable constant in my academic career. From taking my very first steps as a Bachelor student in the idyllic town of Duisburg, to collaborations in group projects, to proof-reading every single thesis I have ever put to paper, he has been a great friend and colleague.

Finally, I want to thank my parents, Maya, Simon, Kevin and Eric! You cannot know how important your support, your company and your inspiration are to me!

Contents

1. Introduction **16**

1.1. Cross-domain Sentiment Polarity Classification 17
1.2. Relation-based Argumentation Mining (RbAM) 18
1.3. Main contributions 20
1.4. Thesis structure 21
1.5. Publications 22
1.6. Statement of Originality 23

2. Background **24**

2.1. Argumentation 24
2.1.1. Abstract Argumentation 25
2.1.2. Bipolar quantitative Argumentation Frameworks . . . 26
2.1.3. Other Argumentation Frameworks 31
2.1.4. Argument Interchange Format 33
2.2. Corpus-based Computational Linguistics 35
2.2.1. Preprocessing 35
2.2.2. From text to numbers 38
2.2.3. Domain dependence 43
2.3. Rhetorical Structure Theory 44
2.4. Summary & Discussion 45

3. Classification enhanced with Argumentation (CleAr) **47**

3.1. Why Join Argumentation & Computational Linguistics? . . . 48
3.1.1. Knowledge consolidation 48
3.1.2. Accessibility for non-experts 48
3.1.3. Extensibility 49
3.2. Classification procedure 50
3.2.1. Step (1): Instance classification 50
3.2.2. Step (2): Argument identification 53
3.2.3. Step (3): Answer strength calculation 55

3.2.4. Step (4): Final classification decision 56
3.3. Related work 57
3.4. Summary & Discussion 60

4. CleAr for binary cross-domain Sentiment Polarity Classification 61
4.1. Applying CleAr to binary Polarity Classification 62
4.1.1. Step (1): Instance classification 62
4.1.2. Step (2): Argument identification 63
4.1.3. Step (3): Answer strength calculation 68
4.1.4. Step (4): Final classification decision 69
4.2. Corpora 69
4.2.1. Sanders Twitter corpus 71
4.2.2. STS Twitter corpus 73
4.2.3. Movie review corpus 74
4.3. Evaluation 74
4.3.1. Sanders vs. STS 76
4.3.2. Sanders vs. movie reviews 77
4.3.3. STS vs. movie reviews 81
4.3.4. Adding rhetorical structure arguments 88
4.3.5. Evaluation Discussion 89
4.4. Related work 96
4.5. Summary & Discussion 98

5. Relation-based Argumentation Mining 100
5.1. Why *Relation-based* Argumentation Mining? 101
5.1.1. This is a fact - why does it attack this other sentence? 101
5.1.2. Relating found arguments is a three-class problem . . 102
5.2. RbAM overview 102
5.3. Sentence pair construction & representation 104
5.4. Argument tree construction & visualisation 106
5.5. Corpora 109
5.5.1. Internet Argument Corpus 111
5.5.2. Argument Interchange Format corpus 113
5.5.3. News articles corpus 117
5.5.4. Other corpora 123

5.6. Evaluation . . . 126
5.6.1. Internet Argument Corpus . . . 126
5.6.2. AIFdb . . . 127
5.6.3. News articles corpus . . . 128
5.7. Related work . . . 132
5.8. Summary & Discussion . . . 134

6. CleAr for Relation-based Argumentation Mining 136
6.1. Step (1): Instance classification . . . 136
6.2. Step (2): Argument identification . . . 137
6.3. Step (3): Answer strength calculation . . . 141
6.4. Step (4): Final classification decision . . . 142
6.5. Evaluation . . . 142
6.5.1. Internet Argument corpus . . . 143
6.5.2. AIFdb corpus . . . 144
6.5.3. News articles corpus . . . 145
6.5.4. Evaluation Discussion . . . 147
6.6. Summary & Discussion . . . 149

7. Future work & Conclusion 150
7.1. Future work . . . 151
7.1.1. General challenges . . . 151
7.1.2. Sentiment Analysis challenges . . . 154
7.1.3. Argumentation Mining challenges . . . 154
7.1.4. Argumentation Mining and Sentiment Analysis . . . 157
7.2. Conclusion . . . 158

References 158

Appendices 185

A. RbAM Argument base 186

B. Discourse markers 191

C. News articles corpus annotation instructions 193

D. Machine Learning 195

List of Figures

2.1. Example BAF, comprised of arguments α, β and γ, represented as a graph, where β attacks α and γ supports α . . . 30

3.1. Tree view of the output of step (1) for the hotel example, where to root node is h and the second level of the tree denotes the possible class labels, with their scores shown as superscripts. 52
3.2. Tree view of the output of step (2) for the hotel example . . 55
3.3. Tree view of the output after executing step (3) for the hotel example, where only α is applicable to h 56
3.4. Tree view of the output after executing step (3) for the hotel example, where both α and β are applicable to h 56
3.5. Tree view of the output after executing step (3) for the hotel example, where both α, β and γ are applicable to h 57

4.1. Tree view of the output of step (1) for binary polarity classification for example sentence m 62
4.2. Tree view of the output of step (2) for arguments and relations applicable to example sentence m 66
4.3. Tree view of the output of step (2) for arguments and relations applicable to example sentence t 67
4.4. Tree view of the output of step (2) for arguments and relations applicable to example sentence m (when considering the extended argument base) 69
4.5. Tree view of the output of step (3) for arguments and relations applicable to example sentence m 70
4.6. F_1 scores for the Sanders vs. STS experiment, where the x-axis labels combinations of $Classifier/Semantics/AB$. The best performing combination is shown in bold. 77

4.7. F_1 scores for the Sanders vs. movie reviews experiment, where the x-axis labels combinations of $Classifier/Semantics/AB$. The best performing combination is shown in bold. 81
4.8. F_1 scores for the STS vs. movie reviews experiment, where the x-axis labels combinations of $Classifier/Semantics/AB$. The best performing combination is shown in bold. 88
4.9. F_1 scores for the *RST annotated movie reviews corpus*, where the x-axis labels combinations of $Classifier/Semantics/AB$. The best performing combination is shown in bold. 91
4.10. Summary of evaluation results of applying CleAr to cross-domain sentiment polarity classification, where we compare the F_1 scores of applying models trained on either the STS or Sanders corpus to the RST annotated Movie reviews corpus, where the x-axis labels *Training corpus/Classifier* combinations, using three classifiers, RF/SVM/NB 92
4.11. Average classification performance over all corpus & argument base combinations 93
4.12. Summary of evaluation results of applying CleAr to cross-domain sentiment polarity classification, where we compare the F_1 scores across all three baseline classifiers for each corpus combination with the results of applying ESAA together with the basic argument base, where the x-axis labels corpus combinations as *Training corpus/Test corpus*. The baselines are shown as dotted bars while the results of applying CleAr are shown in solid bars. 95

5.1. Example argument tree created from sentence pairs that share common arguments, using strategy (2). 106
5.2. Example argument trees resulting from employing strategy (1). 108
5.3. Example argument tree resulting from employing strategy (2). 109
5.4. Schematic view of Quaestio-it visualisation, joining example argument subtrees from figures 5.1 and 5.3 110
5.5. Screenshot of visualisation provided by AIFdb [109, 110] . . . 115

6.1. Tree view of output of step (1) for RbAM for example pair *pc* 137
6.2. Tree view of output of step (2) for RbAM for example pair *pc* 140

6.3. Tree view of output of step (3) for RbAM for example pair *pc* 141
6.4. Summary of evaluation results of applying CleAR to RbAM, where we compare the F_1 score of the baseline with the application of CleAr using ESAA or DF-QuAD for each corpus 148

D.1. Illustration of support vectors. "The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points" [20]. The support vectors are those data points that lie on the decision boundaries. 197

List of Tables

2.1. Examples of positive and negative words from [235] 41
2.2. Examples of *explicit discourse connectives* that are listed in the *Penn Discourse Treebank (PDTB)* manual [173] 41

3.1. Examples of rules as they may be used to identify sentiment polarity of a Tweet *tw* 49

4.1. Example arguments taken from the sentiment analysis argument base 65
4.2. Corpora overview with counts for positive, negative and total instances 71
4.3. Class and category distribution of the Sanders Twittter corpus 72
4.4. Description of the Sanders Twitter corpus classes 72
4.5. Example Tweets taken from the Sanders Twitter corpus . . . 72
4.6. Description of the STS corpus categories and the most frequent entities for each category 73
4.7. Example Tweets taken from the STS Twitter corpus 73
4.8. Example sentences taken from the movie review corpus 74
4.9. Classification results for the Sanders vs. STS experiment, where *Acc.* stands for *Accuracy*, *Prec.* for *Precision* and *Rec.* for *Recall*. The best performing combinations of semantics and *AB* for each classifier are shown in bold. 78
4.10. Confusion matrices for classification of classifiers *trained on the Sanders corpus* and *tested on the STS corpus* 79
4.11. Confusion matrices for classification of classifiers *trained on the STS corpus* and *tested on the Sanders corpus* 80
4.12. Classification performance for the Sanders vs. movie reviews experiment, where *Acc.* stands for *Accuracy*, *Prec.* for *Precision* and *Rec.* for *Recall*. The best performing combinations of semantics and *AB* for each classifier are shown in bold. . . 82

4.13. Confusion matrices for classification of classifiers *trained on the Sanders corpus* and *tested on the movie reviews corpus* . . 83
4.14. Confusion matrices for classification of classifiers *trained on the movie reviews corpus* and *tested on the Sanders corpus* . . 84
4.15. Classification performance for the STS vs. movie reviews experiment, where *Acc.* stands for *Accuracy*, *Prec.* for *Precision* and *Rec.* for *Recall*. The best performing combinations of semantics and *AB* for each classifier are shown in bold. 85
4.16. Confusion matrices for classification of classifiers *trained on the STS corpus* and *tested on movie reviews corpus* 86
4.17. Confusion matrices for classification of classifiers *trained on the movie reviews corpus* and *tested on the STS corpus* 87
4.18. Classification performance for the *RST annotated movie corpus*, where *Acc.* stands for *Accuracy*, *Prec.* for *Precision* and *Rec.* for *Recall*. The best performing combinations of semantics and *AB* for each classifier are shown in bold 90
4.19. Average classification performance over all corpus & argumentation semantics combinations 94
4.20. Average F_1 scores, split according to whether both training and test data were Twitter corpora or one of the two was the movie review corpus 94
4.21. Summary of best performing combinations of corpora, classifiers, argumentation semantics and argument bases, according to F_1 scores 96

5.1. Overview of features used to represent sentence pairs 107
5.2. Corpora overview with counts split by class labels 111
5.3. Example pairs from the IAC 112
5.4. Class distribution of AIFdb corpora according to S-node types (see section 2.1.4 for more details) 114
5.5. Example pairs from the AIFdb corpus, where the RST labels are given, with class labels (Att,Sup,N) in brackets 116
5.6. Example pairs from the News articles corpus 120
5.7. Example pairs from the Quaestio-it corpus 124
5.8. Example pairs from the Customer reviews corpus 125

5.9. Classification performance for the *IAC*, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold. . 127
5.10. Confusion matrices for classification of the *IAC* 127
5.11. Classification performance for the *Two-class AIFdb corpus*, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold. 128
5.12. Confusion matrices for classification of the *Two-class AIFdb corpus* 128
5.13. Classification performance for the *Five-class AIFdb corpus*, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold. 128
5.14. Confusion matrices for classification of the *Five-class AIFdb corpus* 129
5.15. Classification performance for the *Two-class News articles corpus*, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold. 129
5.16. Classification performance for the *Two-class News articles corpus* when representing instances with features instead of a bag-of-words, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold. 130
5.17. Confusion matrices for classification of the *Two-class News articles corpus* 131
5.18. Classification performance for the *Three-class News articles corpus*, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold. 131
5.19. Confusion matrices for classification of the *Three-class News articles corpus* 131
5.20. Summary of best performing RbAM classifiers to be used as baselines in chapter 6 134

6.1. Example arguments taken from AB_{RbAM}, where Sim stands for Similarity. I is a pair $(Parent, Child)$ 139
6.2. Classification performance for the *IAC*, comparing the Random Forest baseline from section 5.6 with results of applying CleAr. The highest F_1 score is shown in bold. 143

6.3. Confusion matrices for classification of the *IAC*, comparing the Random Forest baseline from section 5.6 with results of applying CleAr 143
6.4. Classification performance for the *Two-class AIFdb corpus*, comparing the Random Forest baseline from section 5.6 with results of applying CleAr. The highest F_1 score is shown in bold. 144
6.5. Confusion matrices for classification of the *Two-class AIFdb corpus*, comparing the Random Forest baseline from section 5.6 with results of applying CleAr 144
6.6. Classification performance for the *Two-class News articles corpus*, comparing the Random Forest baseline from section 5.6 with results of applying CleAr. The highest F_1 score is shown in bold. 145
6.7. Confusion matrices for classification of the *Two-class News articles corpus*, comparing the Support Vector Machine baseline from section 5.6 with results of applying CleAr 145
6.8. Classification performance for the *Three-class News articles corpus*, comparing the Support Vector Machine baseline from section 5.6 with results of applying CleAr. The highest F_1 score is shown in bold. 146
6.9. Confusion matrices for classification of the *Three-class News articles corpus*, comparing the Support Vector Machine baseline from section 5.6 with results of applying CleAr 146
6.10. Summary of best performing combinations of corpora, classifiers and argumentation semantics, according to F_1 scores . . 147

A.1. Arguments comprising the RbAM argument base (1/4) 187
A.2. Arguments comprising the RbAM argument base (2/4) 188
A.3. Arguments comprising the RbAM argument base (3/4) 189
A.4. Arguments comprising the RbAM argument base (4/4). Here we only show a generic negation argument $\nu_{Keyword}(I, I')$. However, AB_{RbAM} contains one negation argument for each keyword argument listed, as described in section 6.2. 190

B.1. All *Explicit Discourse Connectives* that are listed in the *Penn Discourse Treebank (PDTB)* manual [173], as described in section 2.2.2, grouped by classes *Attacking*, *Supporting* and *Neither/Ambiguous* (1/2) 191
B.2. All *Explicit Discourse Connectives* that are listed in the *Penn Discourse Treebank (PDTB)* manual [173], as described in section 2.2.2, grouped by classes *Attacking*, *Supporting* and *Neither/Ambiguous* (2/2) 192

C.1. Instructions that were given to annotators to label sentence pairs as (Att, Sup, N) to create the corpus described in section 5.5.3 (1/2) 193
C.2. Instructions that were given to annotators to label sentence pairs as (Att, Sup, N) to create the corpus described in section 5.5.3 (2/2) 194

1. Introduction

We argue every day; we attempt to impart our views and understanding of things on others, and we use language as the prime vehicle of doing so. Accordingly, argumentation has been studied extensively by a variety of scholars, not least by philosophers and computational linguists, who, amongst other problems, have studied the way arguments are communicated, and how they interact in language, e.g. as in [212, 219]. The field of *computational argumentation* (henceforth called just *argumentation*), on the other hand, has emerged as a discipline concerned with developing more formal, logic-based accounts of arguments, e.g. as in [57, 107, 141]. Since its inception in the late 1980s this stream of research on arguments has had a substantial impact on a number of domains. This includes, but is not limited to, *multi-agent systems (MAS)*, e.g. [30, 69], and *decision support* in domains such as medicine, e.g. [68, 72], and engineering, e.g. [13, 64]. In this thesis we study how these two treatments of arguments, in argumentation and in computational linguistics, may be combined to benefit one another.

Over the past decade or so advances in computational linguistics have commonly relied on *data-driven* solutions, as in e.g. [50, 133, 199], while argumentation research has largely yielded *knowledge- and logic-based* solutions, as in e.g. [57, 80]. Recently, however, as illustrated by e.g. [93, 129], there appears to be growing consent that, in order to achieve significant advances in certain areas of artificial intelligence, in general, and computational linguistics, in particular, we may need to consider data-driven approaches in unison with ones that are reasoning-, logic- and rule-based. For a number of problems, both within computational linguistics and beyond, purely data-driven approaches appear to be reaching limits in performance. For example, Manning [129] highlights how progress in *part-of-speech* tagging appears to have stalled and how the probabilistic classifiers used may be improved upon by incorporating a set of deterministic rules.

In this spirit we propose a novel classification methodology in which we use

argumentation, together with supervised learning, to integrate knowledge from different sources and modes of classification into a single classification procedure for computational linguistics problems. We consult knowledge specific to the problem at hand to *argue* about the validity of class labels proposed by a supervised classifier. Whenever we identify sufficiently strong arguments, arguing against a class label suggested by a classifier or for a different label than that suggested, we overrule the classifier and instead base our classification decision on what the arguments tell us. We build concrete solutions based on this methodology to tackle challenges in two widely studied application areas in computational linguistics, namely sentiment analysis and argumentation mining. Specifically, we apply our methodology to the following two problems:

1. *Binary cross-domain sentiment polarity classification*
2. *Relation-based Argumentation Mining (RbAM)*

Below we describe these two challenges in detail. For argumentation mining we propose a novel approach to identifying arguments in text, entirely based on argumentative relations that hold between sentences. Here we not only describe a novel methodology, we also provide a purpose-built, hand annotated corpus of around $2,200$ sentence pairs.

1.1. Cross-domain Sentiment Polarity Classification

Sentiment analysis is generally concerned with (1) identifying opinionated text, e.g. [12, 27, 88, 104, 182, 233, 245], and (2) identifying the polarity of text that is deemed to be opinionated, e.g. [55, 162, 206, 217, 243]. These tasks may be defined as a three-class classification problem, classifying instances in a single pass as *positive*, *negative* or *neutral*, as in e.g. [106]. Another popular approach has been to split the problem into two binary classification problems, where we first extract opinionated text and then classify its polarity, as in e.g. [101]. Say we want to determine the sentiment of the following sentence:

> With his constant bad temper and the endless scandals he has proven to be entirely unelectable.

To do so we need to (1) identify this sentence to be opinionated and (2) identify it to be expressing negative sentiment. We focus on task (2), which is referred to as *polarity classification*. More specifically we address *binary cross-domain polarity classification*, where we train models on corpora that assign sentences, or sentence-like structures, to one of two classes: positive or negative. We then use this model to classify unseen instances from a different domain; we may, for example, train a model on a corpus of Tweets and then use this model to classify sentences from movie reviews. Using models trained on data from a certain domain to classify instances from a different domain tends to yield poor results, even for low-level processing tasks such as parsing [195]. We refer to this lack of transferability of models to new domains as *domain dependence*. This problem afflicts not only sentiment analysis, but many other corpus-based computational linguistics challenges, e.g. [46, 130]. Our methodology allows us to integrate generic knowledge about sentiment with domain-dependent models, which, in turn, lowers the negative impact domain dependence has on classification performance.

1.2. Relation-based Argumentation Mining (RbAM)

Argumentation mining is generally understood as a problem of (1) identifying arguments in text, as in e.g. [24, 142, 157, 163], and (2) connecting said arguments through attack and support relations, as in e.g. [105, 163]. Continuing our above example, say we wanted to analyse the following excerpt of a fictional news article:

> With his constant bad temper and the endless scandals he has proven to be entirely unelectable. After he himself had highlighted his marriage of 21 years, the latest revelations in the yellow press have been particularly damning.

In addition to identifying our original example sentence as an argument, to complete step (1) we would need to do the same with the sentence that follows. In step (2) we would then set the two sentences in relation and e.g. determine whether the second sentence attacks or supports the preceding sentence, or whether it does neither.

With ***R**elation-**b**ased **A**rgumentation **M**ining (RbAM)* [42] we propose an alternative view, in which we focus solely on (2). We create sentence pairs without classifying individual sentences and directly classify the pairs as belonging to one of three classes, *Attack*, *Support* or *Neither*, where Neither denotes everything other than an attack or support relation. Whenever we identify a sentence as an attack or support on another, we consider both sentences to be argumentative, irrespective of their individual argumentative quality. With our approach we thus forego the initial classification and instead classify the sentence together with another sentence according to how one relates to the other.

As part of our work on argumentation mining we need to reflect on what type of text we consider to be an argument, and on what level of granularity we want to analyse text. Take the following definition of an argument from the Oxford online dictionary[1]:

> An argument is a reason or set of reasons given in support of an idea, action or theory.

We are provided with the two main components of an argument, a set of *reasons*, which we shall refer to as *premises*, that support an *idea, action or theory*, which we shall call the *conclusion*. To illustrate consider again our example sentence, which states that some person (he) is unelectable, and, conveniently enough, gives two reasons why this is so. Below we restate the sentence, marking the premises (p) and the conclusion (c) within brackets:

> With $_p$[his constant bad temper] and $_p$[the endless scandals] $_c$[he has proven to be entirely unelectable].

To the dismay of computational linguists and argumentation researchers alike, arguments in natural language are commonly not as neat and well structured as the one in our example. Arguments may stretch across sentences or a single sentence may contain multiple arguments. In either of these situations, an analysis in which we attempt to identify arguments within sentences will prove rather difficult.

Another common problem is the use of *enthymemes* in language, information that is left implicit and for a reader or listener to infer (see e.g.

[1] `www.oxforddictionaries.com`

[227]). Text written or spoken by humans commonly contains a plethora of such enthymemes and, many times, humans have little trouble inferring the implicit information. We are able to do so because we have access to a vast array of *common sense knowledge* (or *world knowledge*), and we often have some sort of *common ground* with the person who has written a text or who we are speaking with. Machines, however, are generally not privy to that same knowledge. Though work has progressed significantly in this area, as illustrated by e.g. [33, 197], no knowledge base today is able to model human grasp of this common sense knowledge. Additionally, as these knowledge bases grow, so does the computational burden of working with them.

Based on these issues, trying to identify neatly formed arguments, each comprised of premises and a conclusion, can be a daunting task. Because our focus lies on the identification of relations between texts we have opted to take a view of arguments and their relations in natural language that is akin to that of abstract argumentation [57]. Abstract argumentation treats arguments as *black boxes*, making no claims as to how individual arguments ought to be structured or what components they should be made up of. Instead, it focuses on the way arguments interconnect, and on determining the impact arguments have on each other through the relations that hold between them. By not attempting to accept only fully fledged arguments and avoiding the classification of individual sentences we both simplify the problem at hand to a more manageable one, as we do not deal with the internal structure of arguments, and we are able to consider a more varied selection of sentences to be argumentative. In addition we analyse text at sentence level. This is a further simplifying assumption, which we discuss further in section 7.1.3, that allows us to focus on identifying relations between clearly defined text spans. We discuss our approach to doing so in detail in chapter 5.

1.3. Main contributions

The main contribution of the work described in this thesis is fourfold:

1. We introduce ***Cl**assification **e**nhanced with **Ar**guments (CleAr)*, a classification methodology for corpus-based computational linguistics problems that uses argumentation in unison with supervised learning. We

take a supervised classifier, such as random forests [28], and use argumentation to enhance its performance. As a first step we classify an instance, using the model built by the classifier, e.g. determining a sentence to exhibit negative or positive polarity. We then use argumentation to *argue* on the validity of this class label. We do this by consulting a collection of arguments, an *argument base*, to identify arguments *applicable* to the instance in question.

2. We propose ***R**elation-**b**ased **A**rgumentation **M**ining (RbAM)*, an approach to identifying argumentative sentences and relations between them. Instead of attempting to identify individual arguments, and then relating them, we classify *sentence pairs* according to a directed relation that holds from one sentence to another. Whenever this relation is classified to be an attack or support we consider *both* sentences to be *argumentative*, irrespective of their individual argumentativeness.

3. We provide a hand-labeled corpus of around $2,200$ sentence pairs, based on which we can build RbAM solutions.

4. We build instantiations of CleAr for sentiment polarity classification and RbAM. To study the usefulness and effectiveness of CleAr we have tested our instantiations, combining a variety of classifiers, argumentation semantics and corpora. We have also developed different argument bases for each instantiation, formalising prototypical knowledge for both polarity classification and RbAM. Based on these parameters we have conducted extensive experimental studies on both instantiations of CleAr, and we show that the application of CleAr yields improvements across the board compared to using classifiers, only.

1.4. Thesis structure

The thesis is structured as follows. We first provide relevant background on argumentation and corpus-based computational linguistics, as well as *Rhetorical Structure Theory*, in chapter 2. In chapter 3 we describe how we join argumentation and corpus-based computational linguistics to form CleAr. One of the two challenges for which we instantiate CleAr is cross-domain sentiment polarity classification. We describe how we apply our

methodology to this problem in chapter 4. Secondly we discuss RbAM and how we instantiate CleAr to be applied to this challenge in chapters 5 and 6, respectively. We conclude the thesis with a discussion of our work, as well as some pointers towards future challenges, in chapter 7.

1.5. Publications

Part of this thesis has been published, or is under review, as follows:

> This paper summarises the main ideas described in chapters 4 to 6: Lucas Carstens & Francesca Toni - Using Argumentation to improve classification in Natural Language problems (under review for *ASM ACM TOIT 2017 : Argumentation in Social Media*)
>
> This paper describes part of chapter 4: Lucas Carstens & Francesca Toni - Improving out-of-domain Sentiment Polarity Classification using Argumentation, *Proceedings of the 5th ICDM Workshop on Sentiment Elicitation from Natural Text for Information Retrieval and Extraction (SENTIRE), 2015* [40]
>
> This paper describes preliminary work for chapter 4: Lucas Carstens & Francesca Toni - Enhancing sentiment extraction from text by means of arguments. *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining (WISDOM), 2013* [39]
>
> This paper conceptually introduces the methodology described in chapter 5: Lucas Carstens & Francesca Toni, Valentinos Evripidou - Towards Relation-based Argumentation Mining, *Proceedings of the 2nd Workshop on Argumentation Mining, 2015* [41]
>
> This paper describes the application in which we first introduced the analysis of sentence pairs, as in chapter 5: Lucas Carstens, Francesca Toni & Valentinos Evripidou - Argumentation Mining and Social debates (Demonstration), *Proceedings of the 5th International Conference on Computational Models of Argument (COMMA), 2014* [42]

The following two papers put our work on argumentation mining in broader context:

> Lucas Carstens, Valentinos Evripidou & Francesca Toni - Abstract argumentation and social networks (to appear in *Arguing on the Web: Theory, Analysis, and Applications, 2016*)

> Lucas Carstens, Xiuyi Fan, Yang Gao & Francesca Toni - An overview of Argumentation Frameworks for Decision Support, *Proceeding of the 4th International Workshop on Graph structures for Knowledge Representation and Reasoning (GKR), 2015* [38]

1.6. Statement of Originality

I declare that this thesis was composed by myself and that the work it presents is my own, except where otherwise stated.

2. Background

Our work draws primarily on *argumentation* and *corpus-based computational linguistics* and here we elaborate on relevant background from the two fields. In section 2.1 we discuss relevant argumentation background, particularly *abstract argumentation* [25, 107, 223], *bipolar argumentation frameworks* [5, 44, 43] and the *Argument Interchange Format (AIF)* [19, 47, 176, 179].

In section 2.2 we discuss relevant background from the field of corpus-based computational linguistics. We start by describing common preprocessing tasks and ways of representing text numerically in sections 2.2.1 and 2.2.2, respectively. In section 2.2.3 we describe challenges that domain dependence in computational linguistics entails. Additionally we review *Rhetorical Structure Theory (RST)* [127] in section 2.3, a popular framework for representing relations in text. We conclude in section 2.4 with an outlook on where certain aspects of the background discussed are of particular relevance. We review more related work throughout the remaining chapters where appropriate, i.e. for the consolidation of argumentation and machine learning in section 3.3, for sentiment analysis in section 4.4 and for argumentation mining in section 5.7.

2.1. Argumentation

From philosophers, musing on how we argue and on argumentation's prerequisites, e.g. [167], to linguists, developing formalisations of argumentative structure, e.g. [212, 219], argumentation has been the focal point of much research. While aspects of natural language arguments have often been the subject of choice, a more formal interpretation has emerged over the past decades to form the field of (computational) argumentation see e.g. [57, 107, 141]. Broadly, argumentation is concerned with the development of *semantics* under which we may create, use and evaluate arguments in *argumentation frameworks (AFs)*. These may be graphs of arguments linked

through attack relations [57], or through attack and support relations, as in e.g. [5, 43]. Below we discuss those frameworks that have been particularly relevant to our work. Additionally, in section 2.1.3 we provide a brief overview of other frameworks that have been developed, but that we did not make use of in the context of our work.

2.1.1. Abstract Argumentation

In his discussion on the acceptability of arguments, Dung introduces the notion of *abstract AFs* [57]. An *AF* consists of a set, whose elements are referred to as arguments, and a binary *attack* relation between arguments, representing conflicts. Formally, an AF is a pair

$$< Args, Attack >$$

where

$$Attack \subseteq Args \times Args$$

and $(\alpha, \beta) \in Attack$ is read 'α *attacks* β'. AFs may be represented as directed graphs, where arguments constitute the nodes and the edges denote attack relations from one node to another. Attacks can be made, for example, as rebuttals or by undercutting. Rebutting amounts to presenting opposing conclusions, e.g.[1]

α = I admire him for his ability to keep things simple.

rebuts (and thus attacks)

$\beta = I$ do not like him because his views are too simplistic.

and, vice versa, β rebuts (and thus attacks) α. Abstract argumentation makes no claims regarding the internal structure of the arguments, themselves. Arguments may have any structure or syntax, so they need not be sentences as in our examples.

[1] Here, and subsequently, we use natural language sentences to illustrate formal notions of argumentation. These may or may not be full arguments, as in section 1.2, where we show a fully fledged argument, but also describe the notion of sentences being *argumentative* despite not being fully fledged arguments.

Undercutting, on the other hand, represents asymmetric conflicts, which may take one of two possible forms. First, a counter-argument may undercut an argument by proving a statement that is claimed not to be provable by the attacked argument. For example, the argument

> He has no idea of foreign policy and he will never be able to prove otherwise.

is undercut (and thus attacked) by the argument

> On the Syrian crisis he has shown his foreign policy acumen.

Secondly, an argument may attack the link between premises and the associated conclusion. For example, the argument

> He is a great politician because he is knowledgable in all tasks associated with his office.

is undercut by the argument

> He is the Jon Snow of energy policy; he knows nothing!

As for arguments, AFs make no assumptions on how attacks are obtained. There are multiple criteria for selecting *winning* arguments in an AF, which are known as *argumentation semantics* [43, 57]. Some of these semantics are defined as *dialectically acceptable* sets of arguments [57], where each set is known as an *extension*. For example, for an AF $< Args, Attack >$, $\mathbf{args} \subseteq Args$ is an *admissible* extension iff **args** is *conflict-free* (i.e. for any argument $\alpha \in \mathbf{args}$, α is not attacked by any argument in **args**) and can *defend* all its member arguments (i.e. for any $\alpha \in \mathbf{args}$, if there exists an argument $\beta \in Args \setminus \mathbf{args}$ that attacks α, there is some argument in **args** attacking β). Other, *quantitative semantics*, described below, instead assess the *dialectical* strength of arguments numerically.

2.1.2. Bipolar quantitative Argumentation Frameworks

When extracting arguments and their relations to each other from human interaction, we need to represent their interplay as accurately as possible. By defining only arguments and attacks, leaving out the possibility of arguments

supporting each other, Dung's definition of AFs covers only part of such human argumentation. *Bipolar argumentation frameworks (BAF)* [4, 5, 43] include a *support* relation to complement the original definition by Dung. Formally, a BAF is a tuple

$$< Args, Attack, Support >$$

where

$$< Args, Attack >$$

is and AF and

$$Support \subseteq Args \times Args$$

is an additional binary relation compared to Dung-style AFs. BAFs can also be represented as directed graphs, in which nodes correspond to arguments and directed arcs correspond to attacks as well as supports.

We focus here on approaches to BAFs that assign strength values to arguments. We do this because they offer a natural link to the numerical output of machine learning solutions with which we seek to combine argumentation, and also because of desirable properties that these approaches fulfill by incorporating numerical strength. This, for example, entails that adding attacks against an argument decreases the argument's strength [13, 175]. This focus does not imply, however, that other interpretations of bipolar argumentation, summarised in section 2.1.3, may not also offer fruitful ways of interfacing with machine learning. The *Extended Social Abstract Argumentation (ESAA) framework* [65], as well as the *Quantitative Argumentation Debate (QuAD) framework* [13] and its extension, *Discontinuity-Free QuAD (DF-QuAD)* [175], are instances of such *quantitative* semantics for BAFs. They all assume that arguments are equipped with a *base score*, i.e. a real number in $[0, 1]$. The base score amounts to an intrinsic (non-dialectical) strength of arguments, which is then altered to give the final (dialectical) strength, based on the (dialectical) strength of attacking and supporting arguments. The resulting strength of arguments is determined by aggregating the strength of attackers against and supporters for these arguments, for restricted types of BAFs in the form of trees. Formally an ESAA framework

is a tuple

$$< Args, Attack, Support, S^0 >$$

where $< Args, Attack, Support >$ is a BAF, extended with a function

$$S^0 : Args \rightarrow [0, 1],$$

assigning a base score to arguments. A QuAD framework is a tuple

$$< Ans, Pro, Con, R, S^0 >$$

where Ans is a set of possible answers to a question and Pro and Con are two types of arguments, attacks and supports, respectively. A QuAD framework can be understood as an ESAA framework by setting

$$Args = Ans \cup Pro \cup Con$$

and

$$Attack = \{(X, Y) | X \in Con, (X, Y) \in R\}$$
$$Support = \{(X, Y) | X \in Pro, (X, Y) \in R\}.$$

R is a relation between arguments as follows:

$$R \subseteq (Pro \cup Con) \times (Pro \cup Con \cup Ans)$$

and

$$S^0 : Ans \cup Pro \cup Con \rightarrow [0, 1]$$

is a function that associates a base score with answers and arguments. Both approaches then define dialectical strength

$$S : X \rightarrow [0, 1]$$

where X is $Args$ in ESAA and $Ans \cup Pro \cup Con$ in QuAD, assigning a score to answers and/or arguments, taking S^0, as well as supports and attacks into account. In ESAA we may additionally ascribe positive and negative votes

to arguments, which produce modifications of their base scores (see [38] for details).

Let v_0 be the base score of an argument α, and v_{att}, v_{sup} the combined strength of all attackers against α and supporters for α, respectively. Then the dialectical strength of α is given by $g(v_0, v_{att}, v_{sup})$ defined as follows:

$$g(v_0, v_{att}, v_{sup}) = \begin{cases} v_{att} & \text{if } v_{sup} = nil \text{ and } v_{att} \neq nil; \\ v_{sup} & \text{if } v_{att} = nil \text{ and } v_{sup} \neq nil; \\ v_0 & \text{if } v_{att} = v_{sup} = nil; \\ \frac{(v_{att}+v_{sup})}{2} & \text{otherwise.} \end{cases}$$

Here $v_{att} = nil$/ $v_{sup} = nil$ if there are no attackers against/supporters for (respectively) α or all such attackers/supporters have strength 0 (and are thus ineffective [13]). The combined strength v_{att}/v_{sup} of (the sequence Seq of the strength of) all attacks against/supports for α is computed as $\mathcal{F}_*(v_0, Seq)$, for $* = att$ or $* = sup$ (respectively), defined recursively as follows, in both QuAD and ESAA:

$$\begin{aligned} \text{if } Seq \text{ is ineffective } : \quad & v_* = nil \\ \text{if } Seq = (v) : \quad & \mathcal{F}_*(v_0, Seq) = f_*(v_0, v) \\ \text{if } Seq = (v_1, \ldots, v_n) : \quad & \mathcal{F}_*(v_0, (v_1, \ldots, v_n)) = \\ & f_*(\mathcal{F}_*(v_0, (v_1, \ldots, v_{n-1})), v_n) \end{aligned}$$

where in both QuAD and ESAA:

$$f_{att}(v_0, v) = v_0 - v_0 \cdot v = v_0 \cdot (1 - v)$$

in QuAD:

$$f_{sup}(v_0, v) = v_0 + (1 - v_0) \cdot v = v_0 + v - v_0 \cdot v$$

and in ESAA:

$$f_{sup}(v_0, v) = min(v_0 + (v_0 - v_0 \cdot (1 - v)), 1)^2$$

For example, given the BAF of Figure 2.1, assuming that all arguments

[2] as defined in [38], correcting a typo in the original publication [65]

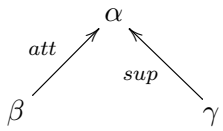


Figure 2.1.: Example BAF, comprised of arguments α, β and γ, represented as a graph, where β attacks α and γ supports α

have a base score of 0.5, the dialectical strength of α is given, in both QuAD and ESAA, by

$$g(0.5, f_{att}(0.5, v_\beta), f_{sup}(0.5, v_\gamma))$$

where v_β / v_γ is the dialectical strength of β / γ, respectively. Since there are no attacks against or supports for β and γ, their strength is their base score (the third case in the definition of g applies). Thus, in both approaches, the dialectical strength of α is given by

$$g(0.5, f_{att}(0.5, 0.5), f_{sup}(0.5, 0.5)).$$

In both approaches $f_{att}(0.5, 0.5) = 0.5 \cdot (1 - 0.5) = 0.25$. In QuAD, $f_{sup}(0.5, 0.5) = 0.5 + (1 - 0.5) \cdot 0.5 = 0.75$. In ESAA, $f_{sup}(0.5, 0.5) = min(0.5 + (0.5 - 0.5 \cdot (1 - 0.5)), 1) = min(0.75, 1) = 0.75$. Thus, in both approaches, the dialectical strength of α is $(0.25 + 0.75)/2 = 0.5$. It is easy to see, however, that the two approaches give different strengths in general.

DF-QuAD [175] is an extension of QuAD, which addresses certain discontinuities that may arise during calculation with QuAD. Discontinuities in QuAD arise, for example, when an argument is attacked by a large number of arguments and has no supports. In this situation the argument's score may be close to zero, yet adding a single support to the argument causes its strength to increase to a comparatively high value. An argument's score drops analogously when we add a single attack against an argument that has a score close to one, prompted by a large number of supports. DF-QuAD rectifies these discontinuities as follows. Whereas both in ESAA and QuAD the strength of attacks (f_{att}) and supports (f_{sup}) are aggregated using separate functions, DF-QuAD defines a single strength aggregation function; given $S = (v_1, ..., v_n),\ n \geq 1$

$$
\begin{aligned}
&\text{if } n = 0 \ : \quad \mathcal{F}(S) = 0 \\
&\text{if } n = 1 \ : \quad \mathcal{F}(S) = v_1 \\
&\text{if } n = 2 \ : \quad \mathcal{F}(S) = f(v_1, v_2) \\
&\text{if } n > 2 \ : \quad \mathcal{F}(S) = f(\mathcal{F}(v_1, ..., v_{n-1}), v_n)
\end{aligned}
$$

where

$$f(v_1, v) = v_1 + (1 - v_1) \cdot v = v_1 + v - v_1 \cdot v.$$

Once strengths of arguments for and against an argument have been aggregated using $\mathcal{F}$, the dialectical strength of an argument is calculated as follows:

$$
\begin{aligned}
g(v_0, v_{att}, v_{sup}) &= v_0 - v_0 \cdot |v_{sup} - v_{att}| \quad if \ \ v_{att} \geq v_{sup} \\
g(v_0, v_{att}, v_{sup}) &= v_0 + (1 - v_0) \cdot |v_{sup} - v_{att}| \quad if \ \ v_{att} < v_{sup}
\end{aligned}
$$

For the example BAF in figure 2.1, we would thus have

$$g(v_0, v_{att}, v_{sup}) = 0.5 - 0.5 \cdot |0.5 - 0.5| = 0.5$$

Refer to [175] for more detailed examples and proofs of how discontinuities are avoided.

2.1.3. Other Argumentation Frameworks

Beyond BAFs and quantitative semantics, many extensions to abstract argumentation have been proposed. The below treat arguments qualitatively instead of quantitatively, as is done e.g. in ESAA or DF-QuAD. *Value-based AFs (VAF)* [16, 17] are an extension of Dung-style AFs, where a VAF is a tuple

$$VAF =< Args, Attacks, V, val, valpref >$$

where $< Args, Attacks >$ is an AF, "V is a set of values, val is a function

which maps from elements of $Args$ to elements of V, and $valpref$ is a preference relation [...] on $V \times V$" [17]. The introduction of preference relations allows a distinction between an argument attacking another and that attack *succeeding*. Successful attacks are defined as *defeats*, where an argument α defeats and argument β iff

$$(\alpha, \beta) \in Attacks \quad AND \quad not\ valpref(val(\alpha), val(\beta)).$$

This means that, for the attack to be successful, the value promoted by the attacked argument must not be preferred to the value promoted by the attacking argument. An alternative way of incorporating preferences over arguments is described by Modgil [138], where attacks are allowed not only against arguments, but against attacks, as well. Here an *Extended AF (EAF)* is a tuple

$$< Args, Attack, D >$$

where <Args,Attack> is an AFs and $D \subseteq Args \times Attack$. In a further extension to traditional Dung-style AFs *probabilistic abstract argumentation* [59, 98, 99, 113, 210] assigns probabilities to arguments, so as to reflect uncertainty in AFs. For example, a probabilistic AF may be a tuple

$$< Args, Attack, p >$$

where, again, $Args$ and $Attack$ are defined as in abstract AFs and "$p \rightarrow [0, 1]$ is a probability function over arguments" [98]. Whereas in abstract AFs the internal structure of an argument is of no relevance, *structured argumentation* [18] covers a range of AFs where some sort of structure is imposed upon individual arguments. In *Assumption-Based Argumentation (ABA)* [58, 211] arguments are identified as deductions, supported by assumptions and rules, and attacks in terms of *contraries of assumptions*, where assumptions are premises that form the support in arguments for conclusions. An ABA Framework is a tuple

$$< \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- >$$

where $< \mathcal{L}, \mathcal{R} >$ is a *deductive system*, with language $\mathcal{L}$ and a set of rules $\mathcal{R}$. $\mathcal{A} \subseteq \mathcal{L}$ is a non-empty set, whose elements are the assumptions, and $^-$ is

a total mapping from $\mathcal{A}$ into $\mathcal{L}$, where $\bar{a}$ stands for the *contrary* of a [211]. AFs in $ASPIC^+$ [139, 140, 171] are defined as a tuple

$$< \mathcal{L}, ^-, \mathcal{R}, n >$$

where again $\mathcal{L}$ is a logical language and $^-$ maps contrary formulae, $\mathcal{R} = \mathcal{R}_s \cup \mathcal{R}_d$ is a set of strict ($\mathcal{R}_s$) and defeasible ($\mathcal{R}_d$) inference rules and $n : \mathcal{R}_d \rightarrow \mathcal{L}$ is a naming convention for defeasible rules [139].

Gorogiannis and Hunter [80] add expressiveness to abstract AFs by using propositional logic to represent the internal structure of arguments. Here, AFs are still tuples

$$< Args, Attack >$$

as defined by Dung [57], with elements of $Args$ defined as pairs

$$< \Phi, \phi >$$

where Φ is a consistent, finite set of formulae from a knowledge base Δ, i.e. $\Phi \subseteq \Delta$, and it entails ϕ, i.e. $\Phi \vdash \phi$. Φ constitutes the premises of such an argument, which entails the conclusion ϕ. To avoid superfluous premises no proper subset of Φ may entail ϕ. Attacks are then formalised as functions

$$D : Args \times Args \rightarrow \{\top, \bot\}$$

where $\top, \bot$ represent truth and falsum, respectively. In this context the notion of defeat then represents cases where, for two arguments α and β, $D(\alpha, \beta) = \top$. D, in turn, is evaluated by means of attack functions, e.g.

$$\alpha \; defeats \; \beta \; if \; C(\alpha) \vdash \neg \bigwedge S(\beta)$$

where $C(\alpha)$ is the claim of argument α and $\bigwedge S(\beta)$ is the conjunction of all premises in β. See [80] for other notions of defeat defined as attack functions, as well as the formalism through which to generate concrete AFs.

2.1.4. Argument Interchange Format

The *Argument Interchange Format (AIF)* [19, 47, 176, 179] provides a model upon which one may define argumentative interactions between agents, tools,

systems for argument manipulation and visualisation, etc. By postulating a number of principles, explained below, the AIF provides guidelines on both syntax and semantics of an AF and thus postulates a framework within which arguments and relations are to be treated. Chesnevar and colleagues [47] formulate an ontology of constraints and rules within whose boundaries a set of arguments and their relations can be represented. Additionally, AIF states rules of *communication* and *context*, guiding how an exchange between multiple parties may unfold and which role the environment plays in which an exchange occurs. More concretely, AIF organises related arguments as a directed graph where the nodes of the graph can be one of two types:

- *Information nodes (I-nodes)*, representing claims
- *Scheme nodes (S-nodes)*, representing the application of argumentation schemes, e.g. as in [226]

Scheme nodes belong to one of three categories, each of which may be divided into a number of subcategories, depending on how the AIF is applied:

- *Inference schemes* (*rule of inference application nodes (RA-node)*)
- *Preference schemes* (*preference application nodes (PA-node)*)
- *Conflict schemes* (*conflict application nodes (CA-node)*)

One of the shortcomings of AIF is the fact that it was originally only able to cope with monologic argumentation. This problem is addressed by Reed and colleagues [179], who introduce AIF^+. This extension of AIF's original ontology adds further argumentation schemes to cope with dialogue and also introduces new node types:

- *Transition Application (TA) nodes* [179]
- *Illocutionary Application (YA) nodes* [178].

TA-nodes are a subclass of RA-nodes that signify transitions between only a specific subset of I-nodes that contain locutions in dialogue, called *L-nodes*. YA-nodes make explicit the *illocutionary force* of arguments uttered in dialogue, which expresses the intention underlying a statement. A YA-node links a statement made with a formalisation of this intention, for example, the question

What should they do in order to look after their children?

taken from *AIFdb* [109, 110], connects via a YA-node to the statement

They should do xxx in order to look after their children.

The link established by the YA-node thus highlights that the question has an illocutionary force by which the question illustrates the intention towards doing something. AIFdb, which we describe in detail in section 5.5.2, is a corpus of so-called *argument maps*, graphs of arguments following the structure defined by AIF.

2.2. Corpus-based Computational Linguistics

The second major field of research we draw upon is that of computational linguistics. Specifically, the problem domain we are concerned with is *corpus-based computational linguistics*. In this thesis we focus on solving natural language problems through supervised learning approaches. This broadly means that we take a corpus of instances of text, e.g. sentences, each assigned to one of a predefined set of classes, and train and evaluate a statistical classification model on this corpus. This model is then used to classify unseen instances according to patterns that were identified in the corpus to occur more often in one class than any of the other classes.

Below we describe text processing tasks that are commonly involved in many computational linguistics problems. Specifically, in section 2.2.1 we describe preprocessing techniques frequently applied in computational linguistics to render text *cleaner*. In section 2.2.2 we review the two broad ways of representing text numerically to allow its classification: the *bag-of-words (BOW)* [189] representation and that based on *Feature Engineering* [194]. Additionally in section 2.2.3 we describe the challenges posed by domain dependence, a common problem in corpus-based computational linguistics.

2.2.1. Preprocessing

To process text computationally, extensive *preprocessing* may be needed to make text amenable to computational analysis. Though text generally exhibits some type of structure, guided by grammar, this is often not sufficient to perform useful computational analysis on it. Preprocessing enables us to

impose additional structure on text and to simplify it. Here we review the following four *low-level* text processing tasks commonly employed ahead of conducting more *high-level* processing, such as sentiment analysis:

1. Tokenisation
2. Sentence splitting
3. Part-of-speech tagging
4. Stemming/Lemmatisation

These four tasks are often strung together in a processing pipeline, where we (1) divide text up into its individual words, then (2) identify sentence boundaries, (3) determine word types and finally (4) break each word down to its stem or lemma, i.e. its basic form.

Tokenisation

In tokenisation we take a piece of text and divide it into a sequence of tokens, where tokens usually roughly correspond to words. This is often achieved applying a set of rules, such as splitting text along white space and considering every collection of contiguous letters or numbers a token. The *Penn Treebank tokeniser* [135] uses a simple set of rules to split text into tokens, using punctuation and whitespace to split text. Additionally the tokeniser breaks down verb contractions and genetive nouns into their constituents. For example, the word *won't* would be represented by its stem and the ending, giving *wo* and *n't*. Brackets are replaced with placeholders and double quotes with doubled single quotes. Based on this Manning and colleagues have built a more sophisticated tokeniser that is shipped with the *Stanford CoreNLP* library [128]. We use this tokeniser, as well as other facilities described below, provided by the Stanford CoreNLP library, for all our developments.

Sentence splitting

Sentence splitting, or sentence segmentation, alludes to a similar process as tokenisation, though on a different level of granularity. Instead of splitting text into word-like constituents, here the aim is to divide text into its sentences. The Stanford CoreNLP library performs sentence splitting based

on tokenisation. Here a sentence is considered to end whenever a *sentence-ending* character, such as a *full stop (.)* or a *question mark (?)*, occurs that is not associated with other characters to form a token, such as dots in abbreviations.

Part-of-Speech tagging

Part-of-speech (POS) tagging is one of the fundamental ways in which we can abstract text away from word-level, representing every word in a text with its type, e.g. *proper noun* or *superlative adjective*. A commonly used tag-set is that developed as part of the *Penn Treebank Project*[3]. The tag-set is comprised of 36 unique tags[4] where, for example, the *VB* tag is assigned to verbs in their base form, the *RB* tag is assigned to adverbs, etc. Representing any text with at most 36 unique tags allows tremendous dimensionality reduction compared to treating individual words as features. It should be noted, however, that this reduction comes at the price of discarding information about the text that may be vital to its analysis. Though deterministic rule-based taggers have been proposed in the past, e.g. [29, 84], the majority of taggers today is based on statistical techniques, e.g. [192, 200, 216]. This includes the Stanford part-of-speech tagger [213, 214], which we use for our developments.

Lemmatisation/Stemming

An alternative way of abstracting away from the original text to reduce its dimensionality comes in the form of lemmatisation, e.g. as in [121, 169], and stemming, e.g. as in [123, 170, 220]. Working in a similar way, but producing somewhat different results, these techniques abstract words not to their word types, as is done in part-of-speech tagging, but rather reduce them either to their lemmata or their stems. While stemming generally heuristically shortens words to their stem, e.g. collapsing *seeing* to *see*, lemmatisation reduces words to their basic forms in a more sophisticated manner. For example, while stemming would not be able to process the past form *saw* of the verb *see*, correct lemmatisation would collapse all the following verbs to the same one, *see*:

[3] `www.cis.upenn.edu/~treebank/`

[4] see `www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html` for the full list

seeing, sees, saw, seen

Additionally, a well performing lemmatiser would leave the noun *saw* untouched during lemmatisation. Again, we use the Stanford CoreNLP lemmatiser [128]. We have opted to use this library throughout for the above preprocessing tasks because they both perform at state-of-the art levels and provide an easy-to-use *Application Programming Interface (API)* and Java libraries.

2.2.2. From text to numbers

To analyse text, using statistical classifiers, we need to come up with ways of representing it numerically. We can broadly do so in two ways. First we may represent our data using a *bag-of-words (BOW)*, described in detail below. In any of its variations this is generally the simplest way we can choose to represent text numerically.

Secondly we may represent text using features that reflect some of its semantic or syntactic properties, such as the length of a sentence, the occurrence of certain key words, or grammatical properties. We describe this approach below, as well. The process of devising features to represent some unit of text is referred to as *feature engineering*.

Each of the two approaches brings with it advantages as well as disadvantages. The simplicity of BOW representations often yields very sparse representations of instances. Feature engineering, on the other hand, generally allows us to create more dense representations of data. This, however, comes at the cost that is involved in identifying valuable features, which may entail extensive quantitative and qualitative analysis of text. Hence one needs to tailor the choice of representation according to various factors, such as the type of problem at hand, the resources available, the size and quality of training data, etc.

Bag-of-Words

Binary feature representation. When using a binary representation of text we simply track the occurrence of words in each instance. Every instance is represented by the same amount of features, one for each word appearing in the vocabulary of the training data used. If, for example, our training corpus contains $10,000$ unique words, the feature vector representing each instance

will be comprised of $10,000$ features, as well. Each feature is instantiated to a value of either 1 or 0, depending on whether the word represented by a feature occurs in an instance or whether it doesn't, respectively. This approach generally yields sparse feature vectors, as the majority of features will be set to 0. Assuming again a vocabulary of $10,000$ words, a sentence comprised of 20 unique words would be represented by $9,980$ features set to 0 and 20 features set to 1. This sparsity means that, during classification, it may be likely for us to encounter instances that were unseen during training, which, in turn, does not allow us to achieve unambiguous classification in many cases.

Term frequency - inverse document frequency (tf-idf). A slightly more sophisticated BOW representation is the *term frequency - inverse document frequency (tf-idf)*, where a *document* is an instance, e.g. a sentence or a Tweet. Instead of simply noting the occurrence of a word, tf-idf takes into account the importance of a word within a document relative to the corpus used as training data. Tf-idf assigns weights to words, or terms, according to how often they appear in a document, or sentence, but offsets the weights, depending on how often a word appears in the corpus, in general. This is done to account for the fact that some words simply appear more often than others, rendering their occurrence in any one instance less pertinent for class prediction. There are variations of tf-idf, but in its basic form, for term t let

$$tf = \frac{(Number\ of\ times\ t\ appears\ in\ a\ document)}{(Total\ number\ of\ terms\ in\ the\ document)}$$

and

$$idf = log_e \frac{(Total\ number\ of\ documents)}{(Number\ of\ documents\ with\ t\ in\ it)}.$$

Then

$$\text{tfidf} = tf \times idf.$$

Feature Engineering

Another option to represent text is to define features that describe traits of the text we aim to classify. Instead of encoding individual words, features

represent identifiable characteristics of a text that manifest themselves in possibly more than one word. Such characteristics may reflect the structure of the text, e.g. (parts of) their parse trees, their length, their similarity to other text, etc. In this section we describe those linguistic traits that we have used most frequently in our own work. This is not meant to serve as an exhaustive discussion on feature engineering, but rather as an illustration of what types of features we may come up with.

Word lists. While with the bag-of-words approach we construct vocabularies from the entirety of words occurring in a training corpus, we may also define just a certain category of words, whose occurrence in text is then used as a feature. Word lists are generally a collection of words that share certain characteristics, which, in turn, may tell us something about the nature of the text we are trying to classify. Importantly for our work, they have been popular in sentiment analysis, e.g. as in [96, 235], where many lists, grouping words of positive and negative polarity, have been developed. A feature vector representing text may then have either a single feature to indicate the occurrence of one or more words from a word list or it may have multiple binary features, one for each word in the list. The latter approach is somewhat similar to the bag-of-words approach, but restricted to a subset of the full vocabulary. Examples of positive and negative words are shown in table 2.1. An alternative to simple sentiment word lists is offered by *SentiWordNet* [63], a thesaurus based on WordNet [134] whose entries are labeled with three sentiment scores, one each for positivity, negativity and neutrality.

Another group of keywords that may help us to identify argumentative relations are discourse markers, a category of words or phrases that broadly signpost discourse, e.g. by indicating turns in a discussion, joining thoughts together or illustrating attitudes. As part of the construction of the *Penn Discourse Treebank (PDTB)* [172] a list of *explicit discourse connectives* [173] has been proposed. These are split into three categories, *Attacking*, *Supporting* and *Neither*. We show example connectives in table 2.2, while the entire list can be found in appendix B.

Similarity measures. When trying to classify text in comparison to other text, as we do in our work (see chapter 5), similarity measures can offer

Table 2.1.: Examples of positive and negative words from [235]

Positive	**Negative**
graceful	absurd
charming	chaos
secure	tired

Table 2.2.: Examples of *explicit discourse connectives* that are listed in the *Penn Discourse Treebank (PDTB)* manual [173]

Attack	**Support**	**Neither/Ambiguous**
however	additionally	by then
instead	thus	until
otherwise	hence	while

handy ways of examining the relation between texts. Some of the more prominent approaches to measuring similarity between words, phrases, etc. have been

- WordNet measures [11, 92, 100, 112, 115, 164, 181, 165, 237];
- Word vector measures [132, 133]; and
- Word overlap measures.

WordNet [134] is a lexical database of the English language, providing definitions of words, as well as their relations to other words. WordNet links words via *synsets*, or cognitive synonyms, grouping different words that describe the same concept, such as *politician* and *legislator*. Using synset relations in WordNet we can calculate distance measures between words. One measure commonly used is a count of how many *steps* one needs to traverse through WordNet to connect two words. If, for example, a word is a synonym of another, the distance between these words is simply 1. If two words are not synonymous, but share a common synonym, the distance between the two words is 2, and so on.

Word vector based measures determine the similarity between words, phrases etc. by performing arithmetic operations on the vectors that represent instances. The *word2vec* tool [132, 133, 180] provides an implementation of measuring word similarity based on the context in which a word appears. The general spirit of the word representations used, *continuous bag-of-words*

(CBOW) [132] and *skip-grams* [85], is that words that frequently appear in the same context are considered more similar than those that do not. Context here means being surrounded by certain words in a text. Both the CBOW and skip-gram models learn vector representations of words using *Neural Net Language Models (NNLM)* [144]. The learned word vectors can then be compared using simple arithmetic operations, such as addition and subtraction, to determine how closely related the vectors, and hence the words, are.

Word overlap measures are generally simple comparisons of word occurrences in the texts we compare. In the simplest case we count the amount of words that occur in both of the texts whose similarity we want to determine. *Edit Distance* (see e.g. [151]) gives us a similarity measure by counting the amount of *replace*, *add* and *delete* operations one needs to perform to turn one string into another. Depending on the use case, Edit Distance may be calculated at character or word level. For example, the Edit Distance at character level between the words *politician* and *politics* is 3, where, to turn the former into the latter, we need to replace one character, the last i in politician with an s, and delete the final two letters.

***Low-level* (Syntactical, lexical and grammatical) features.** The low-level analysis of text provides us with numerous features that may be conducive towards addressing more complex problems. Features deduced from such analysis may include, but are not limited to

- Part-of-speech tags;
- sentence statistics;
- (partial) parse trees;
- punctuation;
- tense; and
- location within a larger text.

These features are often easily attainable, e.g. by using toolboxes such as *Stanford CoreNLP* [128]. Recall that POS tags allow us to abstract from the word level by replacing concrete words with their types, such as *verb*,

pronoun, etc. One may then determine certain combinations of POS tags that occur more frequently in one class than in another. Another option may be to identify certain word types that appear with varying frequencies across classes. Say, for example, we find that opinionated sentences contain more *superlative adjectives*, e.g. worst or best, than non-opinionated sentences. We could then formalise this with a POS-based feature that identifies the occurrences of this particular type of words in the instances that we want to classify. Sentence statistics may include the word count of a sentence, the average word length, etc.

Other features. Certain corpora come with *meta data*, i.e. facts and knowledge about the text being analysed that are provided by some source other than the text itself. Such features may, for example, be provided by the author of the text or by its publisher. For example, product reviews often have ratings associated with them, which may help us determine a review's sentiment towards a product; a five star review ought to be more positive than a two star review. Text may be assigned to certain categories, such as the type of product that is under review. These features can be used in the same manner as any *engineered* feature. They have the advantage of being *free*, as they are readily available for all instances in a corpus. At the same time the use of meta data to build a classification model may negatively impact our ability to generalise this model to new data, as it may not come with the same set of meta data. A review posted on *Amazon* is likely to have a different set of meta data than a review posted on *IMDB*. We discuss such issues of *domain dependence* more generally below.

2.2.3. Domain dependence

Domain dependence is a challenge that has blighted developments in virtually all corpus-based areas of computational linguistics, as described in e.g. [8, 195]. It broadly refers to the problem that a classification model tailored to one domain will often not generalise well to other domains. Say we are trying to build a supervised classifier that identifies sentiment in Tweets, as in fact we describe in chapter 4. To do so we need to obtain a, preferably large, collection of Tweets and assign each one to the class it belongs to, say $C \in \{Positive, Negative\}$. Often labelling, or annotating, instances involves considerable manual labour. This is compounded because we gen-

erally need to obtain multiple independent annotations for at least a subset of instances in our corpus to ensure its quality by measuring *inter-annotator agreement*.

Once we have successfully completed our annotation we are able to build classification models based on the resulting corpus. Certain polarity classification problems have been solved with accuracy in the high seventies, e.g. [26]. These results, however, are obtained by taking a single corpus and either splitting it into training and test sets or using evaluation schemes like cross-validation[5]. This means that, although training and test data do not overlap, they are usually obtained from the same domain. Thus writing style, topic and other characteristics are likely to be more or less homogenous across training and test data.

Aue and Gamon [9] show in a case study of the portability of sentiment classifiers across domains that reasonable in-domain classification performance mostly does not translate into acceptable cross-domain classification performance. They propose four possible approaches to adapting existing corpora for the use of classifying instances from other domains, but each requires at least some annotated data to be available for all domains in question. In chapter 4 we describe our approach to increasing the portability of sentiment classifiers across domains using CleAr.

Domain dependence is an issue that affects many areas beyond sentiment analysis. Arguably, any task in which we build supervised classifiers from a limited amount of data will be faced with this issue. Sekine and colleagues describe decreases of performance when trying to perform parsing across domains [195]. Other work on domain dependence in computational linguistics, and attempting to overcome it, includes e.g. [8, 62, 143]

2.3. Rhetorical Structure Theory

Rhetorical Structure Theory (RST) [127] provides a framework for annotating text and has been widely used for identifying and characterising relations between units of text. In RST relations are defined between *nuclei* and *satellites*. While the nucleus holds the more central piece of information, the

[5]Cross-validation, also referred to as n-fold cross-validation, is a method of evaluating a classifier's performance, where a dataset is split into n folds. A classifier is then trained on $n-1$ of these folds and tested on the held out fold. This is repeated for each fold and classification performance is averaged across all evaluations.

satellite holds additional information that is generally only interpretable in context, together with the nucleus. In the example below the first part of the sentence is the satellite s while the second part is the nucleus n:

> [$_s$Though he did a *good* job at his latest public appearance] [$_n$ his overall popularity ratings are *abysmal*]

There are 23 relations, or *schemas*, that can hold between text spans, described in detail in [207], which usually exhibit a nucleus-satellite structure. Each of these relations is defined in terms of four fields:

1. Constraints on the nucleus
2. Constraints on the satellite
3. Constraints on the combination of nucleus and satellite
4. Effect (achieved on the text receiver)

For a practical example of how RST is applied to text, refer to [126]. Since it was first proposed, numerous parsers have been developed to automate the identification of rhetorical structure in text, such as [61, 70, 71, 90, 202]. To perform RST parsing one generally needs to perform discourse segmentation and discourse parsing. The former entails splitting text into its elementary discourse units (EDUs), which commonly correspond to clauses, while the latter then seeks to link these units up according to how they relate to one another. The RST Discourse Treebank [36] provides a corpus of linked EDUs and has become a common standard on which many RST parsers have been built, e.g. [90]. We describe how we use the distinction between nuclei and satellites to support sentiment polarity classification in section 4.3.4. Here we also describe options of using either a parser or hand labeled examples to accomplish this. We did not, however, make use of the Discourse Treebank, as it is not freely available.

2.4. Summary & Discussion

In this chapter we have provided background on the two research areas vital to the work described in the remainder of this thesis:

1. Argumentation

2. Computational linguistics

In the following chapter we propose CleAr, a classification methodology that uses argumentation, specifically bipolar argumentation frameworks described in section 2.1.2, to integrate knowledge with conventional supervised classifiers. We instantiate CleAr in chapters 4 and 6 to work on sentiment analysis and argumentation mining, respectively. We compare ESAA and DF-QuAD, described in section 2.1.2, as part of these instantiations to investigate how we may best integrate arguments via CleAr. Each instantiation of CleAr works on text that has been passed through the preprocessing pipeline described in section 2.2.1. Finally, we have also discussed Rhetorical Structure Theory, aspects of which we use to contribute arguments to the argument base built for sentiment polarity classification, described in chapter 4.

3. Classification enhanced with Argumentation (CleAr)

From its early days in the 1940's until well into the 1990's, computational linguistics was dominated by logic and rule-based solutions [35, 150]. With the advent of ever increasing computing power, however, more data-driven approaches came into focus. Building statistical models based on ever larger datasets gained traction and more knowledge-heavy solutions began to fall out of favour. More recently, however, a rekindling with the methods of old has been taking place [91, 93], indicating that, after a prolonged period of placing data at the centre of computational linguistics, it may be time to consolidate the two streams of research.

To this end we propose *CleAr (**Cl**assification **e**nhanced with **Ar**guments)*, a novel classification methodology that incorporates reasoning through argumentation with corpus-based computational linguistics. With CleAr we have at our disposal multiple ways of incorporating knowledge in a classification procedure and are able to flexibly integrate knowledge as it becomes available, without the need to retrain classifiers. A further advantage of using argumentation to represent knowledge is the ease of understanding that the format offers. Arguments come in an easy-to-grasp format that may possibly be more intuitively understood by non-experts than the numerical representation of data used to build classifiers. Finally, the use of arguments lends us some justificatory power over classification decisions. Each argument that contributes to classification can be consulted for its premise(s), giving a justification for why the argument has a particular conclusion.

In section 3.1 we motivate the approach in detail and, based on this, in section 3.2 we describe the procedure of arguing and how we integrate it with conventional supervised learners to form a classification procedure. We conclude with a discussion of related work in section 3.3 and a summary of this chapter's contribution in section 3.4.

3.1. Why Join Argumentation & Computational Linguistics?

We believe there are numerous ways in which computational linguists and argumentation researchers may benefit from considering each other's work. Here we point out three particular advantages that illustrate the potential of exploring synergies between the two fields.

3.1.1. Knowledge consolidation

To classify text we need to identify characteristics that, at a minimum, hold true for a certain class of text more often than they would by chance. The more such characteristics we can identify, and the more reliable they are, the greater our chances to classify instances correctly. Using multiple means of representation broadens our options of how we may consolidate knowledge from different sources.

Say we want to classify the polarity of a Tweet *tw*. We may have a generic sentiment classification model available, e.g. by using a tool such as *Stanford CoreNLP* [128]. Let us assume that we additionally have access to some domain-specific knowledge, e.g. a set of rules as shown in table 3.1. These rules provide insights valuable to identifying sentiment in our domain of interest, but integrating them as features in our classification model may prove cumbersome. We would need to come up with a mapping of these rules into features and would then need to retrain the classification model. Using argumentation we can reformulate the rules as arguments of the format

tw is negative *because tw* contains the word *scandal*

where we treat the *IF* part of the rule as the premise and the *THEN* part as the conclusion. Thus, if we had a way to integrate arguments with classifiers, the mapping from the information represented by the rules format to an improved classification method is simplified and there is no need to retrain the model.

3.1.2. Accessibility for non-experts

By using argumentation we can develop rich representations of knowledge on a particular subject that hand us intuitively understandable justificatory

Table 3.1.: Examples of rules as they may be used to identify sentiment polarity of a Tweet *tw*

IF		**THEN**
tw contains the word *scandal*	→	*tw* is negative
tw contains the word *awful*	→	*tw* is negative
tw contains the word *love*	→	*tw* is positive
tw contains the word *amazing*	→	*tw* is positive

powers that are generally not available when using purely statistical methods. Supervised classifiers often work as *black boxes* that provide class labels with limited insight into how a particular class label was chosen over another. Though with such algorithms there are certain ways of gaining insights into *why* something is classified the way it is, such as variable importance testing in e.g. [131], argumentation more readily offers us insights into the *why*.

Say we are trying to determine whether a sentence is positive or negative. In our approach, described below, we use arguments to help us determine the answer to this question. An argument stating that a sentence should be classified one way or the other not only does this, but it also provides a justification in the form of one or more premises that make up the argument, together with the conclusion. For our example an argument may conclude that a sentence is positive, based on the premises that the sentence contains the words *lovely* and *great*. Conventionally the occurrence of positive (or negative) keywords would be reflected by one of a number of features. Determining how this particular feature impacts a classification decision may then be a difficult task and require expert knowledge of classifiers and the models they produce. The argument, on the other hand, naturally provides this information.

3.1.3. Extensibility

Supervised classifiers tend to be used as batch-processing methods, where we annotate a corpus, choose a set of features and then train a model. Should we want to incorporate new features we can only do so by re-training the model. Depending on the complexity and size of our corpus this may require significant computational resources. Integrating knowledge as arguments of-

fers an alternative to re-training models as new knowledge becomes available. Here we can simply add arguments as desired to the existing ones, without necessitating any changes to the classification model, itself.

3.2. Classification procedure

To classify an instance, e.g. a sentence whose sentiment is unknown, we first apply a conventional classification model, trained on an annotated corpus, which provides us with a class label, e.g. *Positive/Negative*. We then argue about whether the suggested class label is appropriate by identifying, and evaluating the strength of, arguments that apply to the instance in question. These arguments may then strengthen the class label suggestion, they may weaken it, or they may strengthen/weaken other class labels. If the suggested class label is sufficiently weakened, or another class label suggestion strengthened, we overrule the suggestion made by the classifier and base the classification on what the arguments tell us. The classification procedure for a given instance I is comprised of the following four steps:

1. Classify instance I and assign confidence scores to possible class labels, using a given trained classifier
2. Identify arguments, and *attack/support* relations amongst them, *applicable* to instance I within a given *argument base* AB
3. Calculate dialectical strength of class labels according to argumentation semantics of choice, e.g. ESAA or DF-QuAD, taking into account applicable arguments and relations determined in step (2)
4. Make final classification decision, based on dialectical strength calculated in step (3)

We describe each step in detail below.

3.2.1. Step (1): Instance classification

To make an initial classification decision for a particular instance I we take as a starting point the output of a classifier C for I. C is a supervised classifier, trained on a corpus of instances of text that have been assigned to one of a predefined set of classes. An instance may be a Tweet in a

collection of Tweets, a pair of sentences in a collection of sentence pairs, etc. Broadly, the resulting model can then be used to classify instances, whose classification is unknown, according to patterns that were identified to occur more often in one class than any of the other classes. In argumentation mining, for example, this may mean building solutions based on corpora where instances are labeled to be either arguments or non-arguments, as in e.g. [142]. In all evaluations reported throughout the remainder of this thesis we limit ourselves to the use of three popular supervised classifiers, which, compared to others we have tested (e.g. Multilayer Perceptrons [186] and *C4.5* decision trees [174]), have produced better results on the majority of our experiments:

- Random Forests [28]
- Support Vector Machines [52]
- Naïve Bayes classifiers [102]

We provide brief descriptions of each classifier in appendix D. Note that one may execute step (1) of CleAr using any other supervised classifier, also. For CleAr we thus assume that the output of classifier C is the set of all possible class labels of interest for the classification problem at hand. Generically we assume that this set of *class labels* is

$$A = \{a_1, ..., a_n\},\ where\ n \geq 2.$$

We also assume that each a_i in A is assigned a *score*, where P_{a_i} is the score of label a_i for instance I and

$$\sum_{i=1}^{n} P_{a_i} = 1.$$

If C is a probabilistic classifier, such as Naïve Bayes, we can use the confidence provided by the classifier as scores. Though classifiers such as Support Vector Machines and Random Forests are non-probabilistic, ways of assigning confidence measures have been proposed, e.g. [67, 79, 168]. As a simple, somewhat ad-hoc alternative we may use F_1 scores[1] obtained during

[1]Precision denotes the fraction of instances assigned a class label that have the correct label, while recall denotes the fraction of instances have been retrieved from each

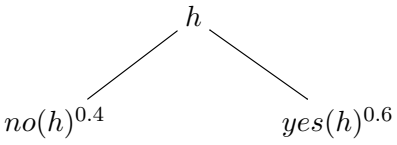


Figure 3.1.: Tree view of the output of step (1) for the hotel example, where to root node is h and the second level of the tree denotes the possible class labels, with their scores shown as superscripts.

the training of the classifier to inform our choice of scores. In this case we assign $P_{a_i} = F_1$ to the a_i returned by the classifier and

$$P_{a_j} = \frac{1 - F_1}{n - 1}$$

to each a_j not selected by the classifier. The output of step (1) is thus the set of all class labels, each of which has a score assigned.

To exemplify, imagine we are considering whether or not to book a room at a hotel h. We then classify instance $I = h$, where the possible class labels are

$$A = \{yes, no\}.$$

Say our classifier determines that we should book h, and it does so with a posterior probability of 0.6. We would then assign scores to our possible labels as $P_{yes(h)} = 0.6$ and $P_{no(h)} = 0.4$. The output of step (1) can be viewed as a tree, as shown in figure 3.1, where the root node is h and the second level of the tree denotes the possible class labels. We adopt a convention to show base scores of the arguments and class labels in superscripts and the dialectical strength as subscripts to distinguish them. Here we use $yes(h)$ and $no(h)$ to indicate that labels yes/no are under consideration for instance $I = h$

class. For example, if we have a corpus of 100 positive Tweets and we label 90 of them as positive, the recall for the positive class is 0.9. If we have wrongly labeled 30 negative Tweets as positive, i.e. a total of 120 Tweets are labeled as positive, the precision of our classifier is 90/120 = .75. The F_1 score, a standard measure of classification performance that jointly considers precision and recall, is calculated as $F_1 = 2 \times \frac{precision \times recall}{precision + recall}$.

3.2.2. Step (2): Argument identification

In step (2) we identify arguments to deliberate on whether or not class labels suggested by the supervised classifier in step (1) are correct. We assume as given an *argument base* AB, i.e. a collection of arguments and relations, where each argument formalises some sort of knowledge conducive to solving the problem in question. We refer to all arguments in AB as $Args(AB)$. The arguments in AB are of the general form

$$id(I, I') : Premise(I, I') \Rightarrow Conclusion(I, I')$$

where I is the instance that is to be classified, I' refers to optional additional information, $id(I, I')$ is an identifier univocally identifying the argument, $Premise(I, I')$ formalises some domain knowledge about I and I' and $Conclusion(I, I')$ may be one of the possible class labels in A or a statement attacking or supporting another argument's premise. When an argument does not rely on any additional information we set $I' = nil$. As shorthand, below we use $id(I, I')$ to refer to the argument with that id. In addition to its premise and conclusion, each argument is defined by three metrics:

(i) A base score $S^0_{id(I,I')} \in [0, 1]$

(ii) An argument type $T_{id(I,I')} \in \{Pro, Con\}$

(iii) A relation $R_{id(I,I')} \subseteq \{id(I, I')\} \times Args(AB) \cup A$

We consider an argument defined this way to be *applicable* to an instance I, given additional information I', if any, whenever its premise holds for I and I'. If $I' = nil$, then I' plays no role in determining the applicability of an argument. To continue our hotel example, say AB is concerned with hotels, and includes arguments

$$\alpha(I, nil) : \ I \text{ has star rating of } 1 \Rightarrow I \text{ is labeled as no}$$
$$\beta(I, nil) : \ I \text{ is in a good neighbourhood} \Rightarrow I \text{ is labeled as yes}$$
$$\gamma(I, SR) : \ \text{Expert gave star rating } SR \text{ of } I \ \Rightarrow I \text{ has star rating } SR$$

Assume also that the argument metrics have the below values

$$S^0_{\alpha(I,nil)} = 0.6;\ T_{\alpha(I,nil)} = Pro,;\ R_{\alpha(I,nil)} = \{(\alpha(I, nil), no)\}$$
$$S^0_{\beta(I,nil)} = 0.3;\ T_{\beta(I,nil)} = Pro;\ R_{\beta(I,nil)} = \{(\beta(I, nil), yes)\}$$
$$S^0_{\gamma(I,SR)} = 0.3;\ T_{\gamma(I,1)} = Pro;\ T_{\gamma(I,SR)} = Con\ if\ SR \neq 1;$$
$$R_{\gamma(I,SR)} = \{(\gamma(I, SR), \alpha(I, nil))\}$$

where *no/yes* are shorthand for *I is labeled as no/yes*, respectively. Argument α is concerned with a low star rating, e.g. on a review website, from which the conclusion *no* is drawn with respect to making a booking at the hotel. Argument β draws the opposite conclusion, based on a hotel being in a good neighbourhood. Finally, argument γ either supports or attacks argument α, based on an expert's rating of the hotel. The scores S^0 here may, for example, reflect that it is more important to us that a hotel is highly rated than it being well located. The general problem of choosing scores for arguments is an on-going research question, which we discuss in section 7.1.1. SR in γ modulates the argument type T based on the additional information provided. To determine the arguments' applicability to $I = h$ we need to match their premises to information about h. If h indeed has a one star rating, but we know nothing about its neighbourhood, nor do we know who provided the rating, we select and instantiate only $\alpha(I)$ from AB. $R_{\alpha(h)}$ tells us that $\alpha(h)$ is related to the label *no*. $T_\alpha : Pro$, in turn, tells us that the relation to the label is one of support. The result of step (2) can be viewed graphically in tree form, as shown in figure 3.2; this is before calculating the impact of the arguments on the scores of the class labels, which is done in step (3). Note that the root node denotes the instance in question, the second layer represents possible class labels, each assigned their score, and subsequent nodes denote arguments and their base scores. The output of step (2) is a thus a QuAD framework

$$< Ans, Pro, Con, R, S^0 >$$

where $Ans = A$, i.e. all possible class labels, and Pro/Con is the set of all applicable arguments in $Args(AB)$ with Pro/Con, respectively, the applicable type in $T_{id(I,I')}$. R is the set of all pairs (X, Y) such that

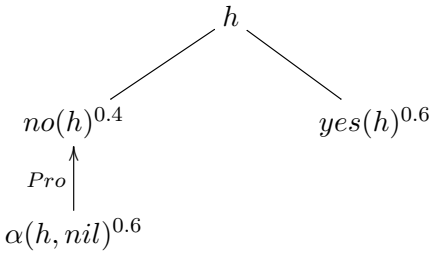


Figure 3.2.: Tree view of the output of step (2) for the hotel example

$$X, Y \in Pro \cup Con \cup Ans$$

and (X, Y) is an instance, with respect to I, in some $R_{id}(I, I')$. The example depicted in figure 3.2 can be seen as a QuAD framework as defined above with $Ans = \{yes, no\}$, $Pro = \{\alpha(h)\}$, $Con = \emptyset$, $R =$ and $S^0 =$. For $X \in Pro \cup Con$ let S^0 be defined as $S^0(X) = S^0_{id(I,I')}$, where $id(I, I')$ is the unique identifier of argument X and $S^0(X) = P_X$ if $X \in Ans$ and P is the confidence value returned by the classifier. This QuAD framework can be represented alternatively as an ESAA framework

$$< Args, Attack, Support, S^0 >$$

where S^0 is defined as above and the relations of applicable arguments to answers and other arguments are mapped as described in section 2.1.2. Independently of how we represent it, the output of step (2) is thus an AF for evaluating how to classify the instance in question.

3.2.3. Step (3): Answer strength calculation

In this step we calculate the dialectical strength of each class label, based on the output of step (2). Figure 3.3 shows the result of calculating this strength for our example using the ESAA algorithm, and thus treating the output of step (2) as an ESAA framework, but note that we may use any other algorithm to execute step (3), as long as there is a possible mapping of the relevant metrics as we have given from QuAD to ESAA in the previous section. Once we have calculated the strength of a class label, as shown in figure 3.3, we assign strength values $S_{no(h)}$ and $S_{yes(h)}$, replacing

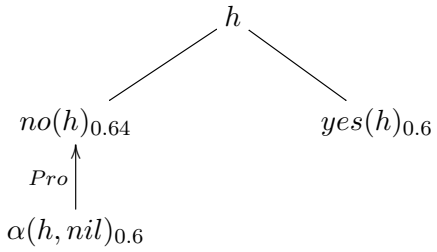


Figure 3.3.: Tree view of the output after executing step (3) for the hotel example, where only α is applicable to h

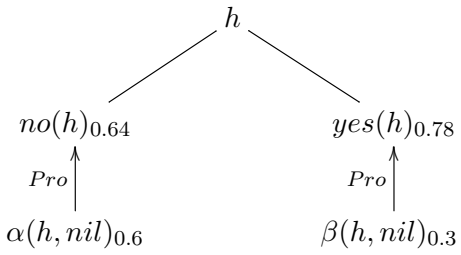


Figure 3.4.: Tree view of the output after executing step (3) for the hotel example, where both α and β are applicable to h

probabilities $P_{no(h)}$ and $P_{yes(h)}$, as is shown in figure 3.2. We use these probabilities as base scores, as discussed in section 3.2.1, for the calculation of strengths. Note that, in general, $\sum_{i=1}^{n} S_{a_i} \neq 1$, and thus strengths cannot be interpreted as probabilities.

To continue our illustration, say we also knew that the hotel was in a good neighbourhood. The resulting calculation would be as shown in figure 3.4, where we have added the appropriate support $S_{\beta(h)}$ to $S_{yes(h)}$. We may also encounter arguments that are applicable to other arguments, instead of class labels, such as $\gamma(I, SR)$. If, for our example, we knew $SR = 1$ was provided by an expert, $\gamma(h, 1)$ would also apply and we would obtain the tree shown in figure 3.5. Any further applicable arguments may be incorporated in the same manner.

3.2.4. Step (4): Final classification decision

To obtain our final classification we only need to consider the dialectical strength of all class labels in A after executing step (3)

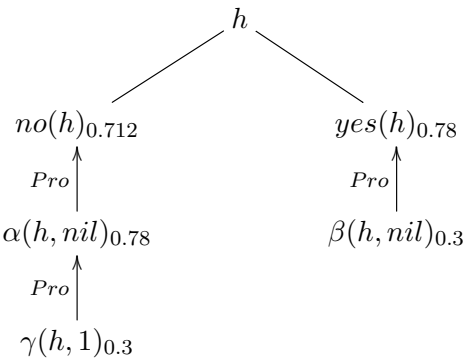


Figure 3.5.: Tree view of the output after executing step (3) for the hotel example, where both α, β and γ are applicable to h

$$S = \{S_{a_1}, ..., S_{a_n}\}$$

where S_{a_i} is the strength of a_i resulting from step (3). For the instantiations of CleAr described in chapters 4 and 6 the final classification decision is made by choosing the maximal element of S, but any other criteria, e.g. by setting thresholds, may be employed to choose the final class label. Should two or more class labels have the same strength after step (3) we use the label suggested by the classifier in step (1) as a tie breaker. In our example we would hence choose *no* if arguments were applicable as in figure 3.3 and *yes* if the situation were as shown in figures 3.4 and 3.5. Note that, for brevity's sake, we omit the instantiation of I' to *nil* in all examples throughout the remaining chapters.

3.3. Related work

Gao and colleagues [74] work on integrating argumentation with *Reinforcement Learning (RL)* through *Argumentation Accelerated Reinforcement Learning (AARL)*, while Mozina and colleagues focus on improving supervised learning with *Argument-based Machine Learning (ABML)* [141, 145, 146, 147, 244]. AARL employs *Value-Based Argumentation* [16] to improve the learning of agents' strategies in the Multi-Agent game of *RoboCup Keepaway-Takeaway (KATA)*. KATA is a game in which two groups of agents compete, where one group passes a ball between its members while the other attempts

to steal the ball. The agents need to learn strategies to keep the ball within their group and to steal the ball, respectively. Courses of action are modelled using *Markov Decision Processes (MDP)* [14]. Heuristics on the quality of actions using arguments are used to assign higher priority to more promising actions in certain states.

ABML combines supervised methods, such as the *CN2* algorithm [49], with argumentation to enhance the algorithm's performance. In ABML, some of the training examples have associated with them one or more arguments, explaining the reasoning for why an example is classified the way it is. The CN2 algorithm takes as input pairs $E = (a, c)$, where a is an attribute-value vector and c is the class the example belongs to. Arguments are acquired from experts who label the original training data to build an extended corpus of which a subset has the form

$$AE = (a, c, Arguments)$$

where $Arguments$ is a set of arguments $Arg_1, Arg_2, ..., Arg_n$. Each Arg_i provides insights of the form *c because Reasons* or *c despite Reasons*.

Our work is similar in spirit to both AARL and ABML. As in ABML we integrate argumentation with supervised learning, but additionally make use of the dialectical power of argumentation. ABML does not create argumentation frameworks, but uses a shallow structure, where single arguments are attached to instances and do not relate to each other. AARL does build upon dialectical relations between arguments, but instead integrates arguments with Reinforcement Learning. Also, arguments are treated as heuristics to speed up convergence in a learning problem that deals with time series, instead of attempting to improve overall classification as we have. In contrast to both AARL and ABML we also use a numerical representation of argument strength. Also, differently from BAFs, our arguments in an argument base have a structure. Moreover, rather than being actual arguments, they are templates for arguments, to be instantiated when I, and possibly I', are known.

In their work Amgoud and Serrurier [6, 7] use training examples, as conventionally used to build classification models, to construct arguments for assigning instances to classes, instead. Arguments are understood to provide reasons for assigning instances $a \in A$ to one of the possible classes $c \in C$.

Here a is an attribute-value vector as in ABML and A is the set of all possible instantiations of the vector. Arguments for assigning a to c are based either on the pair (a, c) being part of the training data, i.e. an example being present, or because there exists a *hypothesis* h that assigns a to c. A hypothesis is a mapping from A to C, meaning that it provides a model, e.g. a decision tree, that maps each possible $a \in A$ to one unique c. Preference relations over arguments are introduced by considering those whose premise is taken directly from examples preferable over the more generic arguments hypotheses. Instead of constructing arguments from knowledge external to the training data then, as we do in our work, Amgoud and Serrurier focus on constructing arguments directly from training data. Furthermore they do not use numerical representations of argument strength, but instead focus on preference relations based on where an argument's support stems from.

Other examples that have more generally addressed the integration of reasoning-, logic- or rule-based solutions with Machine Learning include, but are not limited to, Villena and colleagues [222] who combine rule bases and supervised learners to perform text categorization, as well as *Inductive Logic Programming*, e.g. [51, 148, 149], where examples, e.g. attribute-value vectors, background knowledge and hypotheses are represented using logic programming. Gomez and Chesnevar [78] use argumentation to overcome an issue faced by certain clustering technique *Fuzzy Adaptive Resonance Theory (ART) Neural Networks* [37]. Clusters produced by this technique may overlap and, conventionally, any data points that fall into overlapping regions are randomly assigned to one of the possible classes. Gomez and Chesnevar instead use preference-based argumentation to resolve the assignment to classes of such data points.

Ontanon and colleagues [154] describe a logical model of *Inductive Concept Learning (ICL)*, i.e. the induction of general rules from a given set of examples. Based on this logical model they define *multi-agent ICL*, where "argumentation is used to model the communication between agents, and ICL models their internal learning processes" [154]. To solve the ICL problem each agent has a set of examples and they share rules induced from the examples amongst each other. Argumentation is used to identify attack relations between shared rules and to thus identify inconsistencies.

3.4. Summary & Discussion

In this chapter we have introduced CleAr, a novel classification methodology for supervised learning problems. In our methodology we integrate argumentation with conventional supervised classifiers, illustrating how argumentation may help us move towards a classification regimen that joins data-driven classification with more knowledge-heavy approaches. We believe that this is in line with recent trends that have recognised this as a need to progress in certain areas of artificial intelligence [91, 93]. In chapters 4 and 6 we show how we instantiate CleAr to perform sentiment analysis and argumentation mining, respectively.

4. CleAr for binary cross-domain Sentiment Polarity Classification

One of the tasks for which we have developed an implementation of CleAr is sentiment analysis, which is broadly concerned with (1) distinguishing factual text from opinionated text, see e.g. [12, 27, 88, 104, 182, 233, 245], and (2) determining sentiment polarity, see e.g. [55, 162, 206, 217, 243]. The identification of sentiment may take place at word-level, across phrases or sentences, or larger texts.

We focus on challenge (2), where we classify sentences from movie reviews, as well as *Tweets*, which we know to be opinionated, as either positive or negative. Furthermore we are interested in *cross-domain classification*, whose potential to lessen the burden of domain dependence is of general interest across the computational linguistics community [21, 89, 159, 228, 241], as we discuss in detail in section 2.2.3. To achieve the classification of sentiment polarity across domains we train models on an annotated corpus and then classify instances from a separate corpus, using CleAr, as described in chapter 3. We show empirically that the argument base lessens the impact of domain-dependence, formalising knowledge that is more generally applicable than a classification model is likely to be.

In section 4.1 we present the instantiation of CleAr for sentiment polarity classification. We then describe the corpora we have used to train and test CleAr in section 4.2, as well as its evaluation in section 4.3. We discuss related work in section 4.4 and conclude the chapter with a summary of this chapter's contribution in section 4.5.

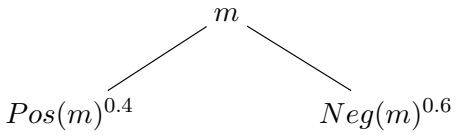


Figure 4.1.: Tree view of the output of step (1) for binary polarity classification for example sentence m

4.1. Applying CleAr to binary Polarity Classification

To perform cross-domain sentiment polarity classification we have developed an instantiation of CleAr, as described in section 3.2. This entailed training classification models, as well as creating an argument base for binary polarity classification. Below we reiterate each of the four steps as instantiated for this particular problem.

4.1.1. Step (1): Instance classification

Instance I is either Tweet or a sentence, such as the following sentence m:

He was not capable of resolving disputes within his party.

For binary polarity classification the class labels are

$$A = \{Neg, Pos\}$$

where Pos stands for *Positive* and Neg stands for *Negative*. Instance I classified as Neg/Pos is written as $I \in Neg/Pos$. Depending on the classifier used in step (1), the scores $P_{Pos(I)}$ and $P_{Neg(I)}$ (based on which we calculate S in step 3) are assigned based either on posterior probabilities provided by the classifier or based on F_1 scores, as described in general in section 3.2.1. E.g., by using a Naïve Bayes classifier, we may obtain posterior probabilities for the above sentence of $P_{Pos(m)} = 0.4$ and $P_{Neg(m)} = 0.6$, which would constitute the result of step (1), which, viewed as a tree, is shown in figure 4.1.

4.1.2. Step (2): Argument identification

Once we have classified an instance we extend the tree produced in step (1) by identifying arguments and relations amongst them from a given argument base that are applicable to the instance in question, as described in section 3.2.2. The argument base AB_{SA} we have built for binary polarity classification is comprised of three types of arguments:

1. Keyword (6,815 arguments)
2. Negation (6,815 argument)
3. Rhetorical Structure (2 arguments)

Table 4.1 gives examples from each category, indicating the various components and metrics of arguments as separate columns. These arguments can be naturally represented in the format described in section 3.2.2. For example, the topmost argument in table 4.1 may be viewed as follows:

$$\kappa_{hate}(I) : hate\ occurs\ in\ I \rightarrow I \in Neg$$

with metrics

(i) $S^0_{\kappa_{hate}(I)} = 0.5$

(ii) $T = Pro$

(iii) $R = \kappa_{hate}(I) \times I \in Neg.$

Note that, as part of our experiments, we have investigated the use of these textual characteristics both as arguments and as features to be used by the classifier, directly. We illustrate how using them as arguments, rather than as features, helps us with classification in section 5.6.3. For each keyword argument we have one negation argument that applies by attacking the according keyword argument whenever the keyword is negated in the instance. Note that, while negation arguments are assigned a score of 0.6, all arguments supporting positivity are assigned the same score, 0.5, while all those supporting negativity are assigned a score of 0.3. We assign different scores to the two groups of keyword arguments because we have a total of 2,014 arguments supporting positivity in AB_{SA}, compared to 4,801

arguments supporting negativity. The rhetorical structure arguments have a score of 0.4. We are investigating ways of assigning scores to individual arguments or more fine-grained groups of arguments, but at this time we consider all arguments in those broad categories to be equal. This is not because we believe that they should be, but rather a choice owed to the current lack of ways to make informed choices for scores. We have determined the scores currently assigned to arguments empirically through trial and error, maximising classification performance on a subset of the experiments described in section 4.3. We discuss possible ways of rendering score assignment more flexible in section 7.1.1.

To construct keyword arguments we have used existing resources, namely word lists of positive and negative words. Each word in such a word list is used to create an argument, where the word itself is considered the premise, and its polarity is treated as the conclusion. Table 4.1, top, shows example keyword arguments. To create keyword arguments we have experimented with lists from the *General Inquirer* collection[1], but have eventually opted to use the list compiled by Hu and Liu [96], as it is the larger list. This list is comprised of a total of $6,815$ keywords, $2,014$ whose class is *Positive* and $4,801$ whose class is *Negative*, while the General Inquirer collection is comprised of $1,915$ positive and $2,291$ negative words, a total of $3,106$ keywords.

In addition to using arguments that are based on keywords, we include a single argument per keyword to deal with negations preceding such keywords. These arguments are also shown in table 4.1. A keywords's sentiment will broadly either be neutralised (e.g. *not awful*) or reversed (e.g. *not good*) when negated. To reflect this, the presence of negations is formalised as an attack on applicable keyword arguments. To illustrate, consider again sentence m from above. The following three arguments from AB_{SA} would be applicable to this sentence, one of them dealing with negation:

- $\kappa_{dispute}(m)$: *dispute* occurs in m
- $\kappa_{capable}(m)$: *capable* occurs in m
- $\nu_{capable}(m)$: *capable* occurs negated in m

[1] `www.wjh.harvard.edu/~inquirer/homecat.htm`

Table 4.1.: Example arguments taken from the sentiment analysis argument base

Category	**id**	**Premise**	**Conclusion**	S^0	**T**	**R**
keyword	$\kappa_{hate}(I)$	*hate* occurs in I	$I \in Neg$	0.5	Pro	$\{(\kappa_{hate}(I), I \in Neg)\}$
	$\kappa_{dispute}(I)$	*dispute* occurs in I	$I \in Neg$	0.5	Pro	$\{(\kappa_{dispute}(I), I \in Neg)\}$
	$\kappa_{succeed}(I)$	*succeed* occurs in I	$I \in Pos$	0.3	Pro	$\{(\kappa_{succeed}(I), I \in Pos)\}$
	$\kappa_{capable}(I)$	*capable* occurs in I	$I \in Pos$	0.3	Pro	$\{(\kappa_{capable}(I), I \in Pos)\}$
	$\kappa_{good}(I)$	*good* occurs in I	$I \in Pos$	0.3	Pro	$\{(\kappa_{good}(I), I \in Pos)\}$
	$\kappa_{abysmal}(I)$	*abysmal* occurs in I	$I \in Neg$	0.5	Pro	$\{(\kappa_{abysmal}(I), I \in Neg)\}$
negation	$\nu_{hate}(I)$	*hate* occurs negated in I	$Premise(\kappa_{hate}(I))$	0.6	Con	$\{(\nu_{hate}(I), \kappa_{hate}(I))\}$
	$\nu_{succeed}(I)$	*succeed* occurs negated in I	$Premise(\kappa_{succeed}(I))$	0.6	Con	$\{(\nu_{succeed}(I), \kappa_{succeed}(I))\}$
	$\nu_{capable}(I)$	*capable* occurs negated in I	$Premise(\kappa_{capable}(I))$	0.6	Con	$\{(\nu_{capable}(I), \kappa_{capable}(I))\}$
RST	$\rho_{hate}(I)$	*hate* in I is in nucleus	$Premise(\kappa_{hate}(I))$	0.4	Pro	$\{(\rho_{hate}(I), \kappa_{hate}(I)\}$
	$\rho_{succeed}(I)$	*succeed* in I is in satellite	$Premise(\kappa_{succeed}(I))$	0.4	Con	$\{(\rho_{succeed}(I), \kappa_{succeed}(I))\}$
	$\rho_{abysmal}(I)$	*abysmal* in I is in nucleus	$Premise(\kappa_{abysmal}(I))$	0.4	Con	$\{(\rho_{abysmal}(I), \kappa_{abysmal}(I))\}$
	$\rho_{good}(I)$	*good* in I is in satellite	$Premise(\kappa_{good}(I))$	0.4	Con	$\{(\rho_{good}(I), \kappa_{good}(I))\}$

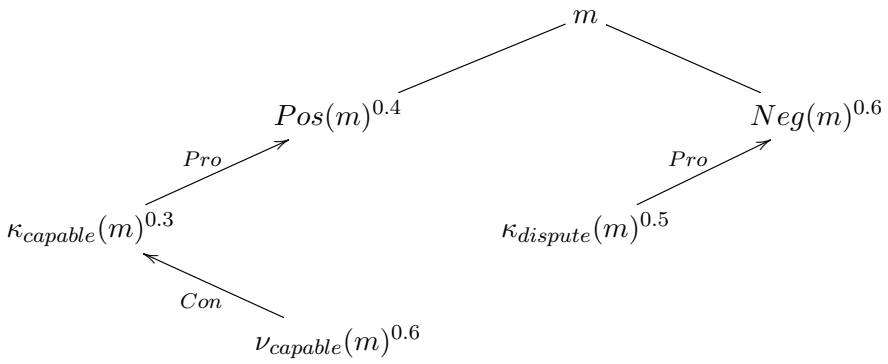


Figure 4.2.: Tree view of the output of step (2) for arguments and relations applicable to example sentence m

We can represent these arguments and how they relate to the possible class labels and to each other as a tree, as shown in figure 4.2. We see that the negation argument attacks, and thus lowers the strength of the argument purporting the keyword *capable* to be a support for *positive* polarity. Though such keywords already influence the classification returned by the classifier we find that additionally formalising them as arguments helps classification performance, as shown in the evaluations described in section 4.3. We did not perform any detailed analysis of why this is the case. One possibility may be that, through their interaction with other arguments, the keywords' dialectical strength, and thus their impact on classification, provides additional information on how important a keyword's occurrence in a particular instance is.

The third category of arguments is concerned with the rhetorical structure of the text being analysed. As described in section 2.3 in RST we split texts into nuclei and satellites. We describe ways of achieving this split in section 4.3.4. A nucleus generally contains the more central information conveyed by a text, while a satellite's content is less important. Hogenboom and colleagues [93], as well as Taboada and colleagues [206, 208] use this gradation of importance of parts of a text to improve keyword-based sentiment analysis. Whenever a text in question contains multiple sentiment-laden words, they identify each word's position with regards to it being part of a satellite or nucleus. Those words that are part of a nucleus are assigned a higher im-

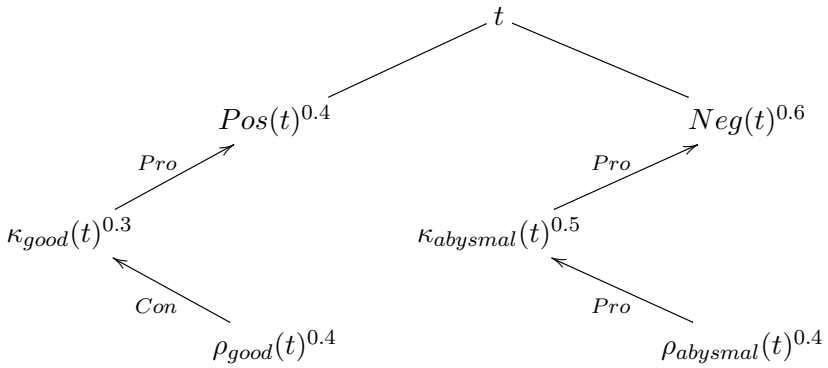


Figure 4.3.: Tree view of the output of step (2) for arguments and relations applicable to example sentence t

portance than those that appear in satellites. We include two arguments in AB_{SA} to reflect the notion that sentiment-laden words appearing in nuclei should be considered more important than those appearing in satellites. The two arguments are shown in table 4.1, at the bottom. There are different options for how we may identify nuclei and satellites. We discuss these in detail in section 4.3.4.

To exemplify, consider again the example from section 2.3, where $[_s...]$ frames the satellite and $[_n...]$ does the same for the nucleus:

$t = [_s$Though he did a *good* job at his latest public appearance$]$
$[_n$ his overall popularity ratings are *abysmal*$]$

Both the nucleus and the satellite contain a word that is covered by our keyword arguments. Applying all arguments in table 4.1 to this example would then yield the tree shown in figure 4.3, including the following arguments to form the tree:

- $\kappa_{good}(t)$: *good* occurs in t
- $\kappa_{abysmal}(t)$: *abysmal* occurs in t
- $\rho_{abysmal}(t)$: The word *abysmal* in t is in a nucleus
- $\rho_{good}(t)$: The word *good* in t is in a satellite

In addition to AB_{SA}, described above, we create a separate argument base AB_{SAext} to investigate the following question:

> If an argument supports one conclusion, should it also attack arguments with a conflicting conclusion, and, conversely, if an argument attacks one conclusion, should it also support arguments with a conflicting conclusion?

For example, consider argument $\kappa_{good}(t)$ in figure 4.3. The question we pose is whether it should be a Con argument against $Neg(t)$, rather than, or in addition to, supporting $Pos(t)$. To move towards finding an answer to this question we conduct a series of experiments, described in section 4.3, where we take every argument in AB_{SA} and create a duplicate. In this duplicate we change T to its opposite value. We also change R so that the duplicate argument relates to the opposing conclusion. To exemplify consider argument $\kappa_{hate(I)}$ in table 4.1. We add the following $\kappa_{hate_a(I)}$ to AB_{SAext}:

$$\kappa_{hate_a(I)} : \ hate\ occurs\ in\ I \Rightarrow I \in Pos$$

with

$$S^0_{\kappa_{hate_a(I)}} = 0.4,\ T_{\kappa_{hate_a(I)}} = Con,\ R_{\kappa_{hate_a(I)}} = \{(\kappa_{hate_a}(I), Pos)\}$$

We add arguments analogously for every keyword argument in AB_{SA}. Retrieving applicable arguments from the AB_{SAext} for our example in figure 4.2 would then yield the tree shown in figure 4.4. We describe the effect this extension has as part of our evaluation in section 4.3.

4.1.3. Step (3): Answer strength calculation

In the third step of CleAr we determine the strength of each possible class label, taking into account the applicable arguments and relations identified in step (2), as described in section 3.2.3. For our experiments described in section 4.3 we compare the ESAA algorithm, used in our example in section 3.2, with the DF-QuAD algorithm. The output of step (3) is an AF where the dialectical strength has been calculated for all nodes, as shown in

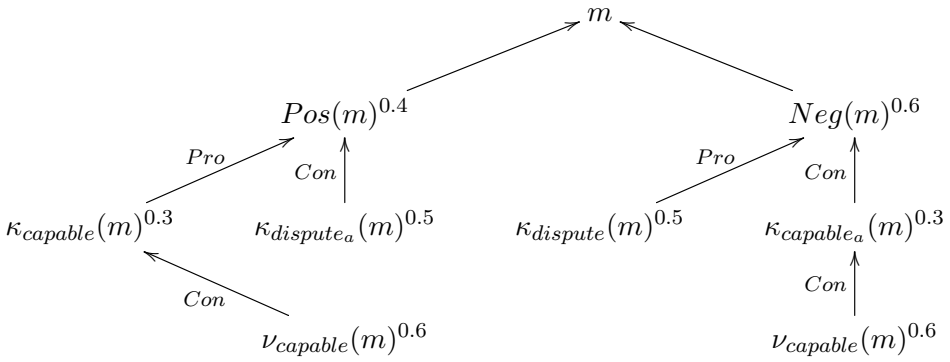


Figure 4.4.: Tree view of the output of step (2) for arguments and relations applicable to example sentence m (when considering the extended argument base)

figure 4.5, where we have taken the example output from step (2) shown in figure 4.2 and have applied the ESAA semantics to calculate the dialectical strength of all answers and arguments.

4.1.4. Step (4): Final classification decision

During the final step of CleAr we need to make a classification decision based on the strength of the labels that we have calculated in step (3). For example, we can determine the larger of the strengths

$$S_{Pos(I)}, S_{Neg(I)}$$

and when $S_{Pos(I)}$ is larger than $S_{Neg(I)}$ we label the sentence in question as *Positive*, if $S_{Neg(I)}$ is larger we label the sentence as *Negative*. Whenever $S_{Pos(I)} = S_{Neg(I)}$ we use the original classification proposed in step (1).

4.2. Corpora

For our cross-domain classification experiments we have chosen three corpora to work with. Table 4.2 shows an overview of the corpora, all of which are annotated for polarity, with each instance in a corpus labeled as either *Positive* or *Negative*. The *Sanders corpus* [190] and the *STS corpus* [188] are comprised of Tweets, while the third corpus [161] is made up of positive

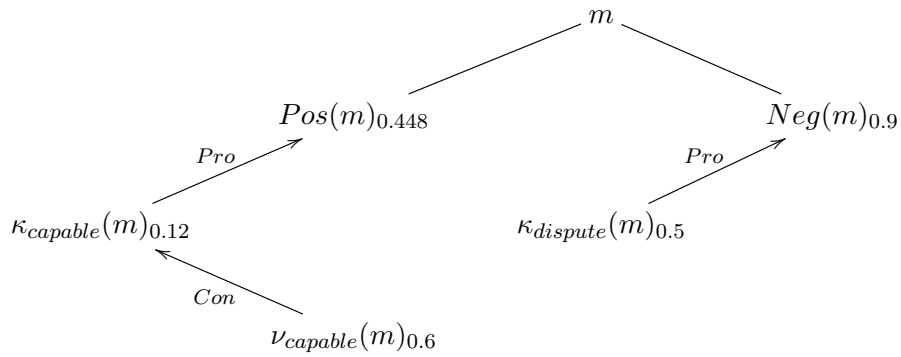


Figure 4.5.: Tree view of the output of step (3) for arguments and relations applicable to example sentence m

and negative sentences, taken from movie reviews. Using these three corpora gives us the opportunity to investigate classification performance on corpora that differ to lesser or stronger degrees. The Twitter corpora share the style of writing that is particular to the Twitter platform, yet the topics which the Tweets are on differ from one corpus to another; their contents are from different *domains*. While the Sanders corpus contains Tweets that address certain companies, e.g. Google or Microsoft, the STS corpus contains Tweets on a broader range of topics, including celebrities and health related topics. The movie review corpus, on the other hand, shares neither writing style nor topic with the Twitter corpora; its contents are from a different domain *as well as* a different *source*. We distinguish the degree to which corpora differ not only based on domain differences, but additionally according to whether or not they are collected from the same source. Whether a corpus is collected from a more or less similar source appears to be an important factor in cross-domain classification. Our evaluation of using CleAr to classify polarity across domains, described in section 4.3, corroborates this.

Throughout the remainder of this section we discuss each corpus in detail. Apart from the three corpora described in detail here we have experimented with other corpora that we did not choose for our evaluation; these are the following:

- Multi Purpose Question Answering (MPQA) corpus [234]
- Sentiment 140 corpus [77]

Table 4.2.: Corpora overview with counts for positive, negative and total instances

Corpus	**# Positive**	**# Negative**	**# Total**
Sanders Twitter	570	654	1,058
STS Twitter	588	1,211	1,799
movie reviews	5,331	5,331	10,662

We have opted not to use these corpora for our evaluation for different reasons. The MPQA corpus was conceived with annotating opinions and *other private states*[2] in text, which meant that only a limited subset of annotations is concerned with opinions. As for choosing Twitter corpora we opted to limit ourselves to two of the available corpora. We selected the Sanders and STS corpora over the Sentiment 140 corpus because they were manually annotated, rather than using heuristics based on the occurrence of emoticons, as is the case for the Sentiment 140 corpus[3].

4.2.1. Sanders Twitter corpus

One of the two Twitter corpora we have used to conduct our experiments on is the *Sanders* corpus described in [190]. It is comprised of $5,513$ manually annotated Tweets, each belonging to one of four categories:

1. Apple
2. Google
3. Microsoft
4. Twitter

Table 4.3 shows a detailed breakdown of the corpus' distribution across categories and classes and table 4.4 shows descriptions of the four classes each category is further broken down into. For our purposes (binary polarity classification) we use a subset of the corpus, namely all Tweets that are labeled as positive or negative. Some example Tweets selected from the corpus are shown in table 4.5.

[2] `mpqa.cs.pitt.edu/corpora/mpqa_corpus/mpqa_corpus_3_0/mpqa_3_0_readme.txt`
[3] `help.sentiment140.com/for-students/`

Table 4.3.: Class and category distribution of the Sanders Twittter corpus

Topic	**# Positive**	**# Neutral**	**# Negative**	**# Irrelevant**
Apple	191	581	377	164
Google	218	604	61	498
Microsoft	93	671	138	513
Twitter	68	647	78	611

Table 4.4.: Description of the Sanders Twitter corpus classes

Class	**Description**
Pos	- Positive indicator or topic
Neutral	- Neither positive nor negative indicators - Mixed positive and negative indicators - On topic, but indicator indeterminable - Simple factual statements - Questions with no strong emotions indicated
Neg	- Negative indicator on topic
Irrelevant	- Not English language - Not on topic (e.g. spam)

Table 4.5.: Example Tweets taken from the Sanders Twitter corpus

Class	**Tweet**
Neg	- Why is #Siri always down @apple - yo @apple this update is a disaster - I hate #Microsoft PowerPoint! - #Microsoft licensing process is annoying !!! - @apple why is my iPhone battery so crappy #fail - Life was easier when @apple @BlackBerry were just fruits!
Pos	- #Google + #Samsung = Perfect #Icecream sandwich - @Apple: Siri is amazing!!! Im in love! - Great up close & personal event @Apple in Regent St store! - #iCloud set up was flawless and works like a champ! - I keep forgettin how much I really like #Twitter lol - #Microsoft store here I come to spend my hard earned cash

Table 4.6.: Description of the STS corpus categories and the most frequent entities for each category

Concept	**Most frequent**	**2nd**	**3rd**	**4th**
Person	Taylor Swift	Obama	Oprah	Lebron
Company	Facebook	Youtube	Starbucks	McDonalds
City	London	Vegas	Sydney	Seattle
Country	England	US	Brazil	Scotland
Organisation	Lakers	Cavs	Nasa	UN
Technology	iPhone	iPod	Xbox	PSP
HealthCondition	Headache	Flu	Cancer	Fever

Table 4.7.: Example Tweets taken from the STS Twitter corpus

Class	**Tweet**
Neg	- Hayfever time not good! - I'm doing my homework. Its gosh darn hard!! - this week is not going as I had hoped - I'm so tired of work...i need a life.... - I don't understand... I really don't
Pos	- Nice my contract was extended for another month - just got home from soccer. Mcdonalds is sooo good - Momz just made it back from Vegas yayyyyy! - Just saw Sunshine Cleaning. I love Amy Adams - had the bestest day in london with dominic

4.2.2. STS Twitter corpus

The second Twitter corpus we use is taken from the *STS Twitter corpus* [188]. The STS corpus has been developed with a focus on annotating entities in Tweets alongside their sentiment. It is comprised of $2,034$ Tweets, each of which contains a mention of one of the entities shown in table 4.6.

To ensure that the corpus only contains Tweets on topics that are different from those that make up the Sanders corpus we removed all Tweets from the *Technology* category, giving us a total of $1,799$ Tweets. Of those Tweets $1,211$ are negative and the remaining 588 are positive. We show examples in table 4.7.

Table 4.8.: Example sentences taken from the movie review corpus

Class	Sentence
Neg	- simplistic, silly and tedious. - doesn't add up to much. - constantly slips from the grasp of its maker. - it's mildly amusing, but I certainly can't recommend it.
Pos	- highly engaging. - the entire movie establishes a wonderfully creepy mood. - the movie has several strong performances. - it's a satisfying summer blockbuster and worth a look. - a model of what movies like this should be like. - clever, brutal and strangely soulful movie.

4.2.3. Movie review corpus

The third corpus we use, which is also the only one not made up of Tweets, is comprised of sentences taken from movie reviews. The corpus was built by Pang and colleagues [160, 161, 162] as part of a larger sentiment analysis task of assigning star ratings to movie reviews. It is comprised of $10,662$ sentences, with $5,331$ sentences labeled as positive and the other $5,331$ labeled as negative. All text is taken from the rotten tomatoes website[4]. We show some example sentences in table 4.8.

4.3. Evaluation

We discuss three sets of experiments with which we have evaluated the performance of CleAr for binary polarity classification. In each we have used one of the corpora described in section 4.2 to train a classification model and another to test the classifier on. For our experiments we were provided with a natural split between training and test data: we have simply trained our models on one corpus and classified *all* instances from another corpus. For each of the three experiment runs we have trained and compared three different models:

1. Random Forest (see appendix D.1)
2. Support Vector Machines (see appendix D.2)

[4] `www.rottentomatoes.com`

3. Naïve Bayes (see appendix D.3)

After initial experiments to gain a gauge of performance we opted to train the Support Vector Machines using *polynomial* kernels, rather than, for example, *radial basis function* or *linear* kernels. For Random Forests Oshiro and colleagues [156] point out that little gain in classification performance seems to be made when using any more than 128 trees in a forest. We have found, however, that increasing the tree count from 128 to 256 helps performance for some corpus combinations and we have hence opted to report results from building Random Forests with 256 trees.

For each combination of classifiers and corpora we compare three scenarios. Our baseline in each run is provided by running the classifier on its own, without applying CleAr. In the second and third scenario we apply CleAr, using the original and extended argument bases, respectively, described in section 4.1.2. For these experiments we remove the rhetorical structure arguments from the argument base. We also compare the performance when using different argumentation semantics, either ESAA or DF-QuAD, on each argument base. In a separate experiment we then added the RST arguments to identify their impact on performance. We did this to illustrate how our methodology is modular: whenever we acquire some knowledge, such as information about the text's rhetorical structure, we can simply add it to the argument base and see whether performance improves, without having to expand and retrain the classifier. In this last experiment we take the best performing combination of classifier and argumentation semantics and try to improve upon the results by adding the rhetorical structure arguments. We describe this additional experiment in section 4.3.4. We report the confusion matrices[5] for each classifier, as well as the following standard measures:

- Accuracy[6]
- Precision & Recall

[5] A confusion matrix shows the labellings obtained from a classifier for all instances in a test set in comparison to the *True* labels, i.e. the labels provided with annotated instances. In each matrix the columns show the class labels returned by the classifier while the rows show the true class labels. Thus the diagonal from the top left to the bottom right in a matrix shows the correct classifications, while from the remaining fields we can read the classification errors that have been committed.

[6] Accuracy denotes the overall proportion of correct classifications made by a classifier, e.g. when we correctly classify 160 instances from a corpus that is comprised of 200 instances, in total, the accuracy is $160/200 = .80$.

- F_1 score

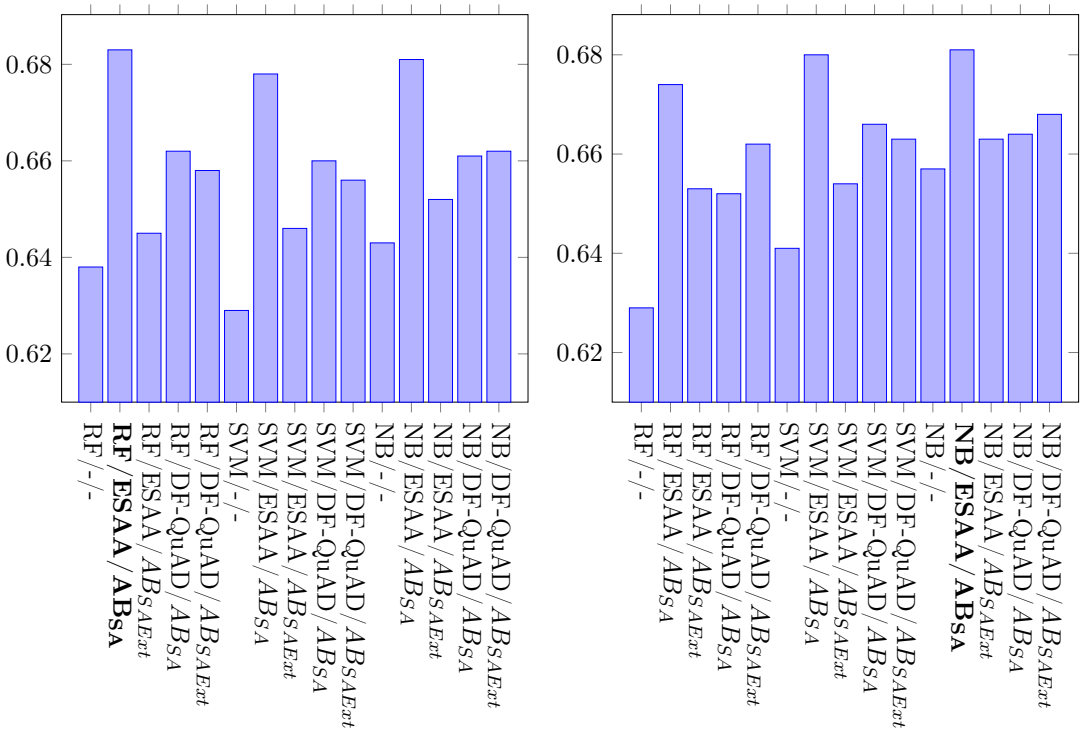
Additionally we report on significance of performance improvements, using standard tests. This yields a total of 120 experiments, 6 (corpus combinations) $\times 3$ (models) $\times 3$ (baseline vs. two argument bases with CleAr) $\times 2$ (argumentation semantics) + 12 (rhetorical structure arguments addition: 2 (corpus combinations) $\times$ 3 (models) $\times$ 2 (argumentation semantics)).

4.3.1. Sanders vs. STS

In the first iteration of our experiments we have used the Sanders Twitter corpus to train the classification models, which we then tested on the STS corpus, and vice versa. This run constituted the only one in which we compared corpora whose contents have been collected from the same source, i.e. Twitter. The difference between the two corpora, then, is not their source, and thus their style, but rather the topic the instances are concerned with. Accordingly, we consider this scenario to fall under *near-domain* classification.

As discussed in sections 4.2.1 and 4.2.2, while the Tweets that make up the Sanders corpus are concerned with certain technological entities, the STS corpus is made up of Tweets that mention persons, locations, etc. We avoided any overlap in topical domain by manually pruning Tweets from the STS corpus that deal with the same topics that make up the Sanders corpus. Table 4.9 and figure 4.6[7] show the results of using each of the corpora for training once and the other for testing, while tables 4.10 and 4.11 show the confusion matrices for each combination of corpora and classifiers. In table 4.9, as well as in all subsequent such tables, each row signifies the performance of a certain combination of classifiers, argumentation semantics and argument base. For example, row 4 in table 4.9 tells us that using a Random Forest classifier together with the DF-QuAD semantics and AB_{SA} produces $Accuracy = 0.66$, $Precision = 0.664$, $Recall = 0.66$ and $F_1 = .662$. The confusion matrix in table 4.10, as well as all subsequent such matrices, tells us that, using the Random Forest baseline, the classifier correctly labels 445 instances as positive (top left cell in the matrix) and 293 instances as negative (bottom right cell), while it mistakenly classifies 290 negative instances

[7]We do not show accuracy scores in any of the graphical representation of the results because in the vast majority of cases it is equal to recall



(a) *Trained on STS*; *tested on Sanders*

(b) *Trained on Sanders*; *tested on STS*

Figure 4.6.: F_1 scores for the Sanders vs. STS experiment, where the x-axis labels combinations of $Classifier/Semantics/AB$. The best performing combination is shown in bold.

as positive (bottom left cell) and 138 positive instances as negative (top right cell).

4.3.2. Sanders vs. movie reviews

The second run of experiments pitted the Sanders Twitter corpus against the movie reviews corpus. Again, on the one hand we trained a classifier using the Sanders Twitter corpus. Here, however, we classified instances from the movie reviews corpus. Secondly, we trained a model using the movie reviews corpus and classified instances from the Sanders Twitter corpus. Results for both runs are shown in table 4.12 and figure 4.7, while tables 4.13 and 4.14 show confusion matrices for different corpus and classifier combinations. Here we thus couple two corpora that are not only topically different, as was the case in the previous experiments, but, in addition, have

Table 4.9.: Classification results for the Sanders vs. STS experiment, where *Acc.* stands for *Accuracy*, *Prec.* for *Precision* and *Rec.* for *Recall*. The best performing combinations of semantics and *AB* for each classifier are shown in bold.

Classifier	**Semantics**	**AB**	**Acc.**	**Prec.**	**Rec.**	**F_1**
RF	-	-	0.633	0.643	0.633	0.638
	ESAA	AB_{SA}	0.683	0.683	0.683	**0.683**
		AB_{SAext}	0.645	0.645	0.645	0.645
	DF-QuAD	AB_{SA}	0.66	0.664	0.66	**0.662**
		AB_{SAext}	0.657	0.658	0.657	0.658
SVM	-	-	0.629	0.629	0.629	0.629
	ESAA	AB_{SA}	0.676	0.681	0.676	**0.678**
		AB_{SAext}	0.642	0.651	0.642	0.646
	DF-QuAD	AB_{SA}	0.66	0.66	0.66	**0.66**
		AB_{SAext}	0.654	0.657	0.654	0.656
NB	-	-	0.642	0.645	0.642	0.643
	ESAA	AB_{SA}	0.678	0.685	0.678	**0.681**
		AB_{SAext}	0.646	0.658	0.646	0.652
	DF-QuAD	AB_{SA}	0.66	0.661	0.66	0.661
		AB_{SAext}	0.66	0.664	0.66	**0.662**

(a) *Trained on the Sanders corpus* and *tested on the STS corpus*

Classifier	**Semantics**	**AB**	**Acc.**	**Prec.**	**Rec.**	**F_1**
RF	-	-	0.629	0.629	0.629	0.629
	ESAA	AB_{SA}	0.672	0.675	0.672	**0.674**
		AB_{SAext}	0.65	0.658	0.649	0.653
	DF-QuAD	AB_{SA}	0.652	0.652	0.652	0.652
		AB_{SAext}	0.661	0.662	0.661	**0.662**
SVM	-	-	0.639	0.642	0.639	0.641
	ESAA	AB_{SA}	0.68	0.68	0.68	**0.68**
		AB_{SAext}	0.653	0.654	0.653	0.654
	DF-QuAD	AB_{SA}	0.665	0.666	0.665	**0.666**
		AB_{SAext}	0.663	0.663	0.663	0.663
NB	-	-	0.657	0.658	0.657	0.657
	ESAA	AB_{SA}	0.68	0.681	0.68	**0.681**
		AB_{SAext}	0.661	0.666	0.661	0.663
	DF-QuAD	AB_{SA}	0.664	0.664	0.664	0.664
		AB_{SAext}	0.668	0.669	0.668	**0.668**

(b) *Trained on the STS corpus* and *tested on the Sanders corpus*

Table 4.10.: Confusion matrices for classification of classifiers *trained on the Sanders corpus* and *tested on the STS corpus*

		Classified as: Pos	Classified as: Neg
True	Pos	445	138
	Neg	290	293

(a) RF-Baseline

		Classified as: Pos	Classified as: Neg
True	Pos	352	231
	Neg	202	381

(b) SVM-Baseline

		Classified as: Pos	Classified as: Neg
True	Pos	331	252
	Neg	166	417

(c) NB-Baseline

		Classified as: Pos	Classified as: Neg
True	Pos	410	173
	Neg	197	386

(d) RF-ESAA-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	345	238
	Neg	140	443

(e) SVM-ESAA-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	336	247
	Neg	129	454

(f) NB-ESAA-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	368	215
	Neg	199	384

(g) RF-ESAA-Extended

		Classified as: Pos	Classified as: Neg
True	Pos	301	282
	Neg	136	447

(h) SVM-ESAA-Extended

		Classified as: Pos	Classified as: Neg
True	Pos	294	289
	Neg	124	459

(i) NB-ESAA-Extended

		Classified as: Pos	Classified as: Neg
True	Pos	428	155
	Neg	241	342

(j) RF-DF-QuAD-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	369	214
	Neg	183	400

(k) SVM-DF-QuAD-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	364	219
	Neg	177	406

(l) NB-DF-QuAD-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	410	173
	Neg	227	356

(m) RF-DF-QuAD-Extended

		Classified as: Pos	Classified as: Neg
True	Pos	343	240
	Neg	163	420

(n) SVM-DF-QuAD-Extended

		Classified as: Pos	Classified as: Neg
True	Pos	334	249
	Neg	148	435

(o) NB-DF-QuAD-Extended

Table 4.11.: Confusion matrices for classification of classifiers *trained on the STS corpus* and *tested on the Sanders corpus*

		Classified as: Pos	Classified as: Neg
True	Pos	302	194
	Neg	176	325

(a) RF-Baseline

		Classified as: Pos	Classified as: Neg
True	Pos	350	146
	Neg	214	287

(b) SVM-Baseline

		Classified as: Pos	Classified as: Neg
True	Pos	306	190
	Neg	152	349

(c) NB-Baseline

		Classified as: Pos	Classified as: Neg
True	Pos	301	195
	Neg	132	369

(d) RF-ESAA-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	334	162
	Neg	157	344

(e) SVM-ESAA-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	314	182
	Neg	137	364

(f) NB-ESAA-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	267	229
	Neg	120	381

(g) RF-ESAA-Extended

		Classified as: Pos	Classified as: Neg
True	Pos	298	198
	Neg	148	353

(h) SVM-ESAA-Extended

		Classified as: Pos	Classified as: Neg
True	Pos	282	214
	Neg	124	377

(i) NB-ESAA-Extended

		Classified as: Pos	Classified as: Neg
True	Pos	316	180
	Neg	167	334

(j) RF-DF-QuAD-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	351	145
	Neg	189	312

(k) SVM-DF-QuAD-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	334	162
	Neg	173	328

(l) NB-DF-QuAD-Basic

		Classified as: Pos	Classified as: Neg
True	Pos	303	193
	Neg	145	356

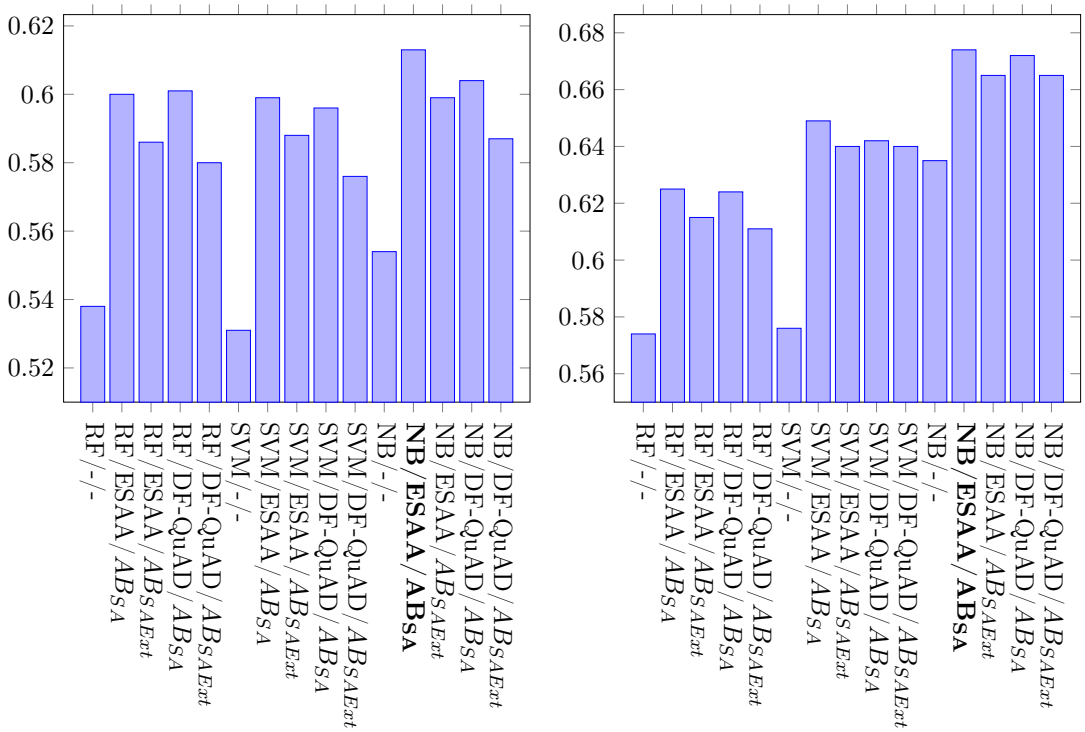
(m) RF-DF-QuAD-Extended

		Classified as: Pos	Classified as: Neg
True	Pos	334	162
	Neg	174	327

(n) SVM-DF-QuAD-Extended

		Classified as: Pos	Classified as: Neg
True	Pos	315	181
	Neg	150	351

(o) NB-DF-QuAD-Extended



(a) *Trained on Sanders*; *tested on movie reviews* (b) *Trained on movie reviews*; *tested on Sanders*

Figure 4.7.: F_1 scores for the Sanders vs. movie reviews experiment, where the x-axis labels combinations of $Classifier/Semantics/AB$. The best performing combination is shown in bold.

also been collected from different sources. Hence we consider this scenario as *out-of-domain classification*, where most, or all, aspects of the corpora in question differ.

4.3.3. STS vs. movie reviews

For our third run of experiments we used the STS Twitter corpus and the movie reviews corpus. Table 4.15 and figure 4.8 show the results of training models on the STS Twitter corpus and classifying instances from the movie reviews corpus, and vice versa, while tables 4.16 and 4.17 show confusion matrices for different corpus and classifier combinations. The setting in this experiment was thus very similar to that described in section 4.3.2, where both topic and source of the two corpora are different and we hence deal with *out-of-domain classification*, rather than near-domain.

Table 4.12.: Classification performance for the Sanders vs. movie reviews experiment, where *Acc.* stands for *Accuracy*, *Prec.* for *Precision* and *Rec.* for *Recall*. The best performing combinations of semantics and AB for each classifier are shown in bold.

Classifier	**Semantics**	**AB**	**Acc.**	**Prec.**	**Rec.**	**F_1**
RF	-	-	0.538	0.538	0.538	0.538
	ESAA	AB_{SA}	0.599	0.6	0.599	**0.6**
		AB_{SAext}	0.583	0.59	0.583	0.586
	DF-QuAD	AB_{SA}	0.601	0.602	0.601	**0.601**
		AB_{SAext}	0.58	0.581	0.58	0.58
SVM	-	-	0.527	0.534	0.527	0.531
	ESAA	AB_{SA}	0.594	0.603	0.594	**0.599**
		AB_{SAext}	0.579	0.598	0.579	0.588
	DF-QuAD	AB_{SA}	0.596	0.596	0.596	**0.596**
		AB_{SAext}	0.573	0.579	0.573	0.576
NB	-	-	0.549	0.56	0.549	0.554
	ESAA	AB_{SA}	0.608	0.618	0.608	**0.613**
		AB_{SAext}	0.589	0.608	0.589	0.599
	DF-QuAD	AB_{SA}	0.604	0.604	0.604	**0.604**
		AB_{SAext}	0.584	0.59	0.584	0.587

(a) *Trained on the Sanders corpus* and *tested on the movie reviews corpus*

Classifier	**Semantics**	**AB**	**Acc.**	**Prec.**	**Rec.**	**F_1**
RF	-	-	0.565	0.584	0.564	0.574
	ESAA	AB_{SA}	0.617	0.635	0.616	**0.625**
		AB_{SAext}	0.604	0.629	0.603	0.615
	DF-QuAD	AB_{SA}	0.622	0.627	0.621	**0.624**
		AB_{SAext}	0.604	0.62	0.603	0.611
SVM	-	-	0.575	0.579	0.574	0.576
	ESAA	AB_{SA}	0.645	0.654	0.644	**0.649**
		AB_{SAext}	0.633	0.649	0.632	0.64
	DF-QuAD	AB_{SA}	0.641	0.643	0.641	**0.642**
		AB_{SAext}	0.637	0.644	0.636	0.64
NB	-	-	0.634	0.636	0.634	0.635
	ESAA	AB_{SA}	0.673	0.675	0.673	**0.674**
		AB_{SAext}	0.662	0.669	0.662	0.665
	DF-QuAD	AB_{SA}	0.672	0.672	0.672	**0.672**
		AB_{SAext}	0.664	0.666	0.664	0.665

(b) *Trained on the movie reviews corpus* and *tested on the Sanders corpus*

Table 4.13.: Confusion matrices for classification of classifiers *trained on the Sanders corpus* and *tested on the movie reviews corpus*

		Classified as	
		Pos	Neg
True	Pos	2,582	2,743
	Neg	2,176	3,146

(a) RF-Baseline

		Classified as	
		Pos	Neg
True	Pos	1,553	3,772
	Neg	1,267	4,055

(b) SVM-Baseline

		Classified as	
		Pos	Neg
True	Pos	1,781	3,544
	Neg	1,261	4,061

(c) NB-Baseline

		Classified as	
		Pos	Neg
True	Pos	2,836	2,489
	Neg	1,784	3,538

(d) RF-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	2,374	2,951
	Neg	1,370	3,952

(e) SVM-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	2,475	2,850
	Neg	1,319	4,003

(f) NB-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	2,408	2,917
	Neg	1,519	3,808

(g) RF-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	1,917	3,408
	Neg	1,076	4,246

(h) SVM-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	2,030	3,295
	Neg	1,076	4,246

(i) NB-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	3,381	1,944
	Neg	2,303	3,019

(j) RF-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	2,045	2,280
	Neg	2,019	3,303

(k) SVM-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	3,118	2,207
	Neg	2,011	3,311

(l) NB-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	2,848	2,477
	Neg	1,996	3,326

(m) RF-DF-QuAD-Extended

		Classified as	
		Pos	Neg
True	Pos	2,292	3,033
	Neg	1,516	3,806

(n) SVM-DF-QuAD-Extended

		Classified as	
		Pos	Neg
True	Pos	2,414	2,911
	Neg	1,523	3,799

(o) NB-DF-QuAD-Extended

Table 4.14.: Confusion matrices for classification of classifiers *trained on the movie reviews corpus* and *tested on the Sanders corpus*

		Classified as	
		Pos	Neg
True	Pos	157	339
	Neg	95	406

(a) RF-Baseline

		Classified as	
		Pos	Neg
True	Pos	224	272
	Neg	152	349

(b) SVM-Baseline

		Classified as	
		Pos	Neg
True	Pos	280	216
	Neg	149	352

(c) NB-Baseline

		Classified as	
		Pos	Neg
True	Pos	213	283
	Neg	99	402

(d) RF-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	259	237
	Neg	117	384

(e) SVM-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	305	191
	Neg	135	366

(f) NB-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	188	308
	Neg	87	414

(g) RF-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	231	265
	Neg	101	400

(h) SVM-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	278	218
	Neg	119	382

(i) NB-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	256	240
	Neg	137	364

(j) RF-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	286	210
	Neg	148	353

(k) SVM-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	336	160
	Neg	167	334

(l) NB-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	206	290
	Neg	105	396

(m) RF-DF-QuAD-Extended

		Classified as	
		Pos	Neg
True	Pos	257	239
	Neg	123	378

(n) SVM-DF-QuAD-Extended

		Classified as	
		Pos	Neg
True	Pos	302	194
	Neg	141	360

(o) NB-DF-QuAD-Extended

Table 4.15.: Classification performance for the STS vs. movie reviews experiment, where *Acc.* stands for *Accuracy*, *Prec.* for *Precision* and *Rec.* for *Recall*. The best performing combinations of semantics and *AB* for each classifier are shown in bold.

Classifier	**Semantics**	**AB**	**Acc.**	**Prec.**	**Rec.**	**F_1**
RF	-	-	0.57	0.57	0.57	0.57
	ESAA	AB_{SA}	0.625	0.625	0.625	**0.625**
		AB_{SAext}	0.608	0.613	0.608	0.611
	DF-QuAD	AB_{SA}	0.619	0.621	0.619	**0.62**
		AB_{SAext}	0.61	0.61	0.61	0.61
SVM	-	-	0.557	0.559	0.557	0.558
	ESAA	AB_{SA}	0.62	0.62	0.62	**0.62**
		AB_{SAext}	0.603	0.604	0.603	0.604
	DF-QuAD	AB_{SA}	0.616	0.62	0.616	**0.618**
		AB_{SAext}	0.606	0.606	0.606	0.606
NB	-	-	0.568	0.568	0.568	0.568
	ESAA	AB_{SA}	0.622	0.622	0.622	**0.622**
		AB_{SAext}	0.606	0.609	0.606	0.607
	DF-QuAD	AB_{SA}	0.618	0.62	0.618	**0.619**
		AB_{SAext}	0.608	0.608	0.608	0.608

(a) *Trained on the STS corpus* and *tested on movie reviews corpus*

Classifier	**Semantics**	**AB**	**Acc.**	**Prec.**	**Rec.**	**F_1**
RF	-	-	0.579	0.608	0.579	0.593
	ESAA	AB_{SA}	0.65	0.682	0.65	**0.667**
		AB_{SAext}	0.617	0.658	0.617	0.637
	DF-QuAD	AB_{SA}	0.642	0.653	0.642	**0.648**
		AB_{SAext}	0.626	0.654	0.626	0.64
SVM	-	-	0.601	0.613	0.601	0.607
	ESAA	AB_{SA}	0.672	0.688	0.672	**0.68**
		AB_{SAext}	0.66	0.687	0.66	0.673
	DF-QuAD	AB_{SA}	0.669	0.674	0.669	**0.676**
		AB_{SAext}	0.664	0.678	0.664	0.671
NB	-	-	0.6	0.61	0.6	0.605
	ESAA	AB_{SA}	0.664	0.673	0.664	**0.668**
		AB_{SAext}	0.648	0.665	0.648	0.656
	DF-QuAD	AB_{SA}	0.663	0.665	0.663	**0.664**
		AB_{SAext}	0.65	0.658	0.65	0.654

(b) *Trained on the movie reviews corpus* and *tested on the STS corpus*

Table 4.16.: Confusion matrices for classification of classifiers *trained on the STS corpus* and *tested on movie reviews corpus*

		Classified as	
		Pos	Neg
True	Pos	2954	2371
	Neg	2211	3111

(a) RF-Baseline

		Classified as	
		Pos	Neg
True	Pos	3395	1930
	Neg	2782	2540

(b) SVM-Baseline

		Classified as	
		Pos	Neg
True	Pos	3186	2139
	Neg	2461	2861

(c) NB-Baseline

		Classified as	
		Pos	Neg
True	Pos	3214	2111
	Neg	1885	3437

(d) RF-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	3465	1860
	Neg	2187	3135

(e) SVM-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	3321	2004
	Neg	2017	3305

(f) NB-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	2710	2615
	Neg	1556	3766

(g) RF-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	2922	2403
	Neg	1825	3497

(h) SVM-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	2816	2509
	Neg	1686	3636

(i) NB-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	3616	1709
	Neg	2350	2972

(j) RF-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	3776	1549
	Neg	2544	2778

(k) SVM-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	3696	1629
	Neg	2442	2880

(l) NB-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	3131	2194
	Neg	1962	3360

(m) RF-DF-QuAD-Extended

		Classified as	
		Pos	Neg
True	Pos	3379	1946
	Neg	2249	3073

(n) SVM-DF-QuAD-Extended

		Classified as	
		Pos	Neg
True	Pos	3259	2066
	Neg	2106	3216

(o) NB-DF-QuAD-Extended

Table 4.17.: Confusion matrices for classification of classifiers *trained on the movie reviews corpus* and *tested on the STS corpus*

		Classified as	
		Pos	Neg
True	Pos	187	396
	Neg	95	488

(a) RF-Baseline

		Classified as	
		Pos	Neg
True	Pos	256	327
	Neg	138	445

(b) SVM-Baseline

		Classified as	
		Pos	Neg
True	Pos	263	320
	Neg	146	437

(c) NB-Baseline

		Classified as	
		Pos	Neg
True	Pos	257	326
	Neg	82	501

(d) RF-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	307	276
	Neg	106	477

(e) SVM-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	319	264
	Neg	128	455

(f) NB-ESAA-Basic

		Classified as	
		Pos	Neg
True	Pos	212	371
	Neg	75	508

(g) RF-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	272	311
	Neg	86	497

(h) SVM-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	282	301
	Neg	110	473

(i) NB-ESAA-Extended

		Classified as	
		Pos	Neg
True	Pos	297	286
	Neg	131	452

(j) RF-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	339	244
	Neg	142	441

(k) SVM-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	357	226
	Neg	167	416

(l) NB-DF-QuAD-Basic

		Classified as	
		Pos	Neg
True	Pos	240	343
	Neg	93	490

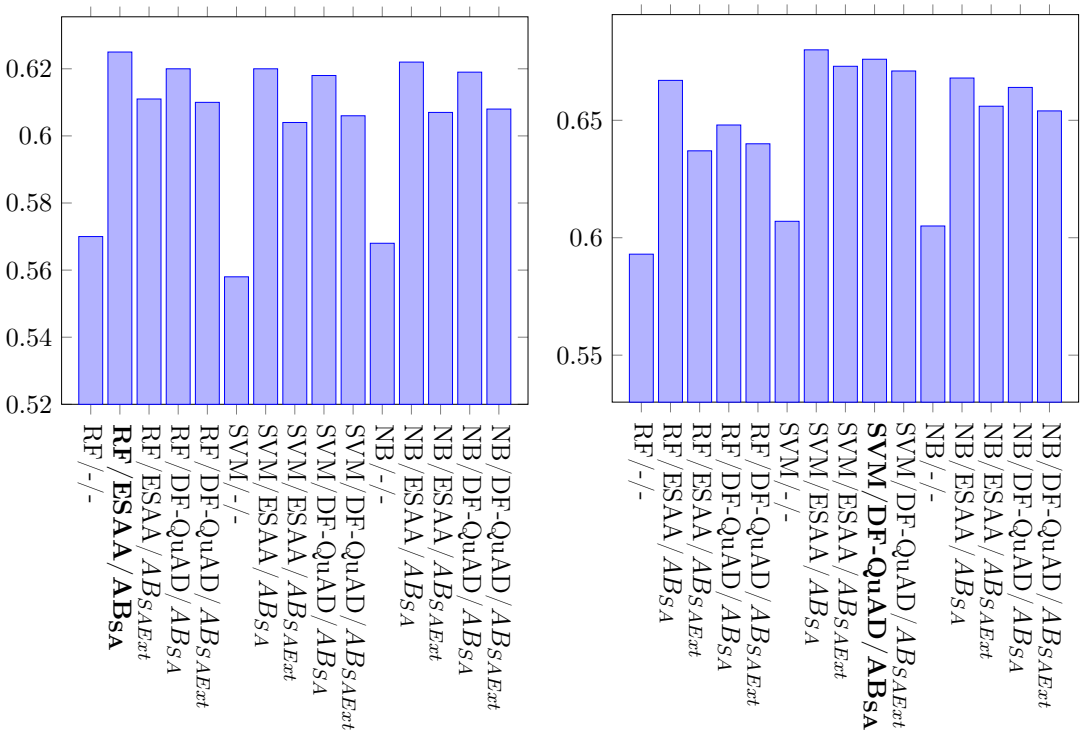
(m) RF-DF-QuAD-Extended

		Classified as	
		Pos	Neg
True	Pos	306	277
	Neg	115	468

(n) SVM-DF-QuAD-Extended

		Classified as	
		Pos	Neg
True	Pos	314	269
	Neg	139	444

(o) NB-DF-QuAD-Extended



(a) *Trained on STS*; *tested on movie reviews*

(b) *Trained on movie reviews*; *tested on STS*

Figure 4.8.: F_1 scores for the STS vs. movie reviews experiment, where the x-axis labels combinations of $Classifier/Semantics/AB$. The best performing combination is shown in bold.

4.3.4. Adding rhetorical structure arguments

To illustrate the modular nature of our approach we have held out the arguments pertaining to the rhetorical structure of sentences from the above experiments. In a separate experiment we compared the effect of applying CleAr with an argument base that is comprised of keyword and negation arguments, only, to applying CleAr with an argument base that additionally contains the two rhetorical structure arguments. Here we use AB_{SA} together with the the rhetorical structure arguments; we ignore AB_{Ext} because, as the above evaluations show, with one exception the use of AB_{SA} yields better results throughout. This illustrates how we may extend or change the scope of CleAr by adding arguments to the existing argument base, and how this may affect classification.

To be able to test whether including rhetorical structure arguments has an

effect on classification performance we need instances that are labeled with nucleus and satellite spans. We have two options to obtain these labelings. On the one hand we may use existing RST parsers [61, 70, 71, 90, 202], as described in section 2.3, as part of the classification procedure to automatically label instances with nucleus and satellite spans. On the other hand we may classify instances that have already been labeled with the spans, e.g. via manual annotation. Though the second option is more labour-intensive, as we need manual annotations, and arguably does not translate as well into real-life applications, we have nevertheless opted for this approach. Here our intuition was that we wanted to test the impact of actual rhetorical structure relations. If we were to use the output provided by a parser, we would be determining whether the parser's output, rather than rhetorical structures proper, helps us in improving classification. These parsers are prone to err in their classifications, e.g. Heilman and Sagae [90] report an F_1 score of 0.681 for identifying nuclei in text, and any statement on the impact of actual rhetorical relations would thus be less informed if it were based on the output of a parser.

To run this experiment we took a subset of 500 instances from the movie reviews corpus, split evenly among classes, and manually annotated these instances for nucleus and satellite spans. We decided to test the movie corpus, only, because Tweets are arguably rather poor subjects for an analysis of rhetorical structure. For simplicity's sake we chose a subset of instances that were unambiguously comprised of one nucleus and one satellite, giving us a small corpus of 110 negative and 101 positive instances. We then ran a further 12 experiments on this corpus, 2 (corpus combinations) $\times$ 3 (models) $\times$ 2 (argumentation semantics). An overview of the results is shown in table 4.18, as well as figures 4.9 and 4.10.

4.3.5. Evaluation Discussion

Throughout our experiments we observe improved classification when using CleAr, compared to using classifiers, only. We achieve improvements on F_1 scores of up to 7.4 points, with an average improvement of 4.3 points across all experiments. Table 4.19 and figure 4.11 show the classification results averaged over all experiments, while figure 4.12 shows the baselines for each combination of classifiers and corpora, in comparison to the results achieved when applying CleAr with ESAA and AB_{SA}.

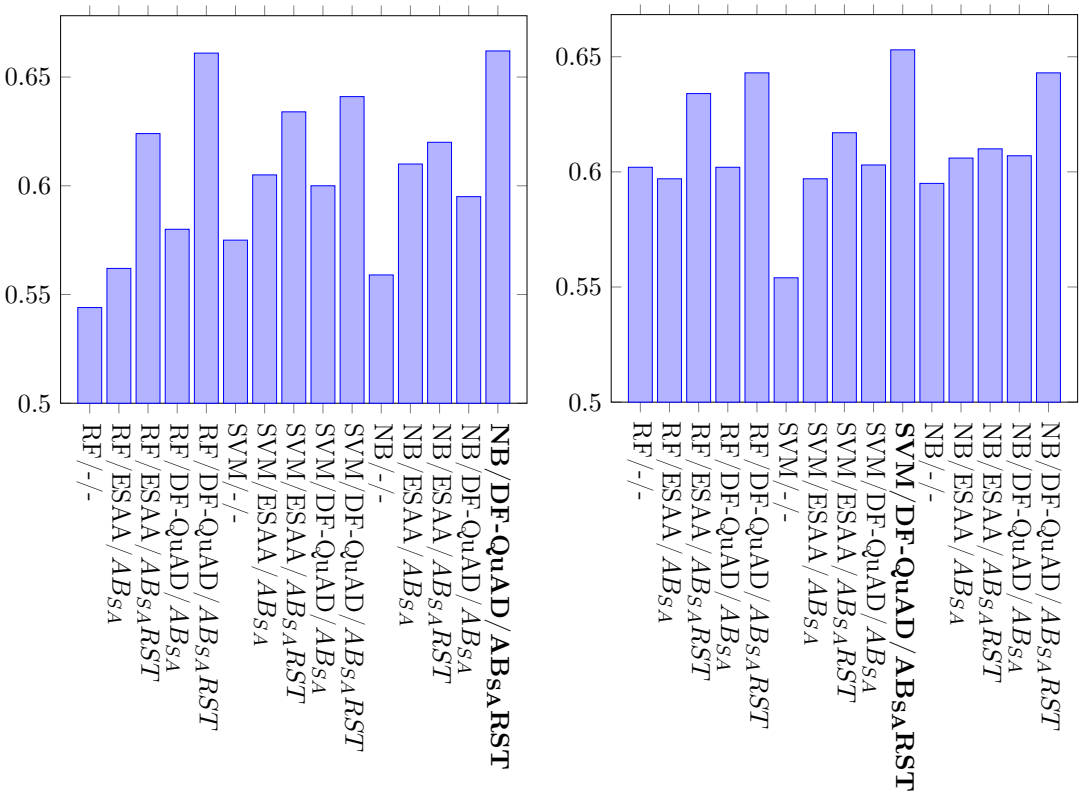
Table 4.18.: Classification performance for the *RST annotated movie corpus*, where *Acc.* stands for *Accuracy*, *Prec.* for *Precision* and *Rec.* for *Recall*. The best performing combinations of semantics and *AB* for each classifier are shown in bold

Classifier	**Semantics**	**Arguments**	**Acc.**	**Prec.**	**Rec.**	**F_1**
RF	-	-	0.545	0.544	0.544	0.544
	ESAA	AB_{SA}	0.564	0.562	0.562	0.562
		AB_{SA}+RST	0.626	0.625	0.624	**0.624**
	DF-QuAD	AB_{SA}	0.583	0.582	0.579	0.58
		AB_{SA}+RST	0.659	0.67	0.652	**0.661**
SVM	-	-	0.578	0.578	0.573	0.575
	ESAA	AB_{SA}	0.607	0.607	0.602	0.605
		AB_{SA}+RST	0.635	0.636	0.631	**0.634**
	DF-QuAD	AB_{SA}	0.602	0.604	0.596	0.6
		AB_{SA}+RST	0.64	0.65	0.633	**0.641**
NB	-	-	0.559	0.559	0.559	0.559
	ESAA	AB_{SA}	0.611	0.610	0.609	0.61
		AB_{SA}+RST	0.621	0.62	0.62	**0.62**
	DF-QuAD	AB_{SA}	0.597	0.596	0.594	0.595
		AB_{SA}+RST	0.659	0.672	0.652	**0.662**

(a) Trained on the STS corpus

Classifier	**Semantics**	**Arguments**	**Acc.**	**Prec.**	**Rec.**	**F_1**
RF	-	-	0.592	0.605	0.598	0.602
	ESAA	AB_{SA}	0.592	0.599	0.596	0.597
		AB_{SA}+RST	0.63	0.635	0.633	**0.634**
	DF-QuAD	AB_{SA}	0.602	0.601	0.602	0.602
		AB_{SA}+RST	0.645	0.646	0.641	**0.643**
SVM	-	-	0.54	0.558	0.549	0.554
	ESAA	AB_{SA}	0.597	0.607	0.587	0.597
		AB_{SA}+RST	0.611	0.618	0.615	**0.617**
	DF-QuAD	AB_{SA}	0.602	0.603	0.603	0.603
		AB_{SA}+RST	0.654	0.655	0.65	**0.653**
NB	-	-	0.573	0.607	0.584	0.595
	ESAA	AB_{SA}	0.597	0.609	0.603	0.606
		AB_{SA}+RST	0.602	0.612	0.607	**0.61**
	DF-QuAD	AB_{SA}	0.607	0.607	0.607	0.607
		AB_{SA}+RST	0.645	0.645	0.641	**0.643**

(b) Trained on the Sanders corpus



(a) *Trained on STS* (b) *Trained on Sanders*

Figure 4.9.: F_1 scores for the *RST annotated movie reviews corpus*, where the x-axis labels combinations of $Classifier/Semantics/AB$. The best performing combination is shown in bold.

We observe significant differences ($p < 0.01$[8,9]) between baselines and the application of CleAr in all cases shown in figure 4.12.

Apart from overall performance improvement we are interested in determining changes in performance along choices of three parameter settings:

1. Classification model

[8]To determine whether we observe a statistically significant difference between the performance we test the *null hypothesis*, which states that there is *no* difference between the classification performance. The p-value represents the probability of observing the same, or a more extreme, result than the one obtained. A small p-value, e.g. $p < 0.05$, thus indicates that the null hypothesis should be rejected, based on which we judge the difference to be statistically significant.

[9]p was calculated using a *one-way analysis of variance (ANOVA)*, see `www.mathworks.com/help/stats/anova1.html` for details

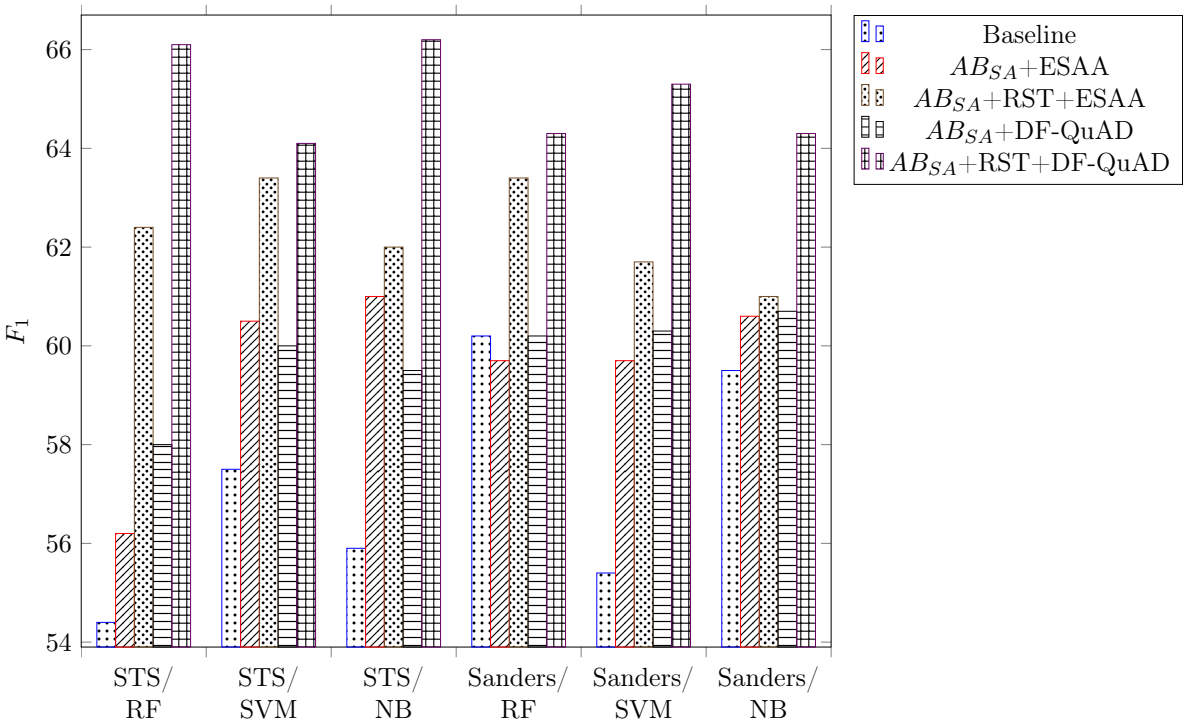


Figure 4.10.: Summary of evaluation results of applying CleAr to cross-domain sentiment polarity classification, where we compare the F_1 scores of applying models trained on either the STS or Sanders corpus to the RST annotated Movie reviews corpus, where the x-axis labels *Training corpus/Classifier* combinations, using three classifiers, RF/SVM/NB

2. Argumentation semantics

3. Argument base

As for (1) which classifiers we should choose for the task described in this chapter, there appears to be no clear-cut answer. While in three of the six experiment runs Naïve Bayes classifiers yield the highest F_1 scores, in two runs Random Forests produce the best results and in one Support Vector Machines do so. The average F_1 scores across all experiments for Random Forests, Support Vector Machines and Naïve Bayes classifiers lie at 0.624, 0.631 and 0.639, respectively. As for the other two parameters the picture appears to be more clear. In all six runs, using the ESAA semantics in combination with AB_{SA} produced the best results. The average F_1 score across classifiers where we use the ESAA Semantics and AB_{SA} at 0.651 is

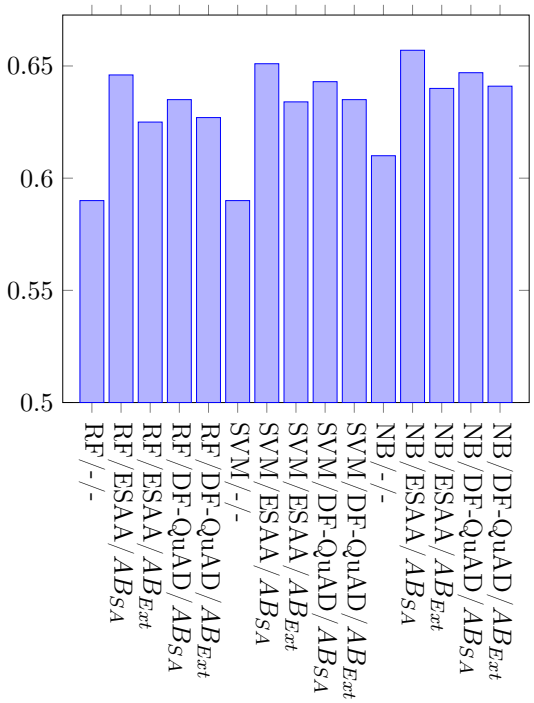


Figure 4.11.: Average classification performance over all corpus & argument base combinations

significantly better than other combinations ($p < 0.01$[10]), which produce an average F_1 score of 0.636.

While the use of AB_{Ext}, on average, produced inferior classification in comparison to the use of AB_{SA}, the addition of rhetorical structure arguments appears to be more successful.. As figure 4.10 illustrates introducing the rhetorical structure arguments improves classification across the board, where, on average, F_1 score improves by 6.53 points compared to the baseline and by 3.98 points compared to the application of CleAr without the rhetorical structure arguments. The overall difference between the F_1 scores shown in figure 4.10 is statistically significant ($p < 0.01$[9]). Apart from using AB_{SA} with DF-QuAD we achieve significant improvements upon the baseline in all cases. However, we need to be careful to compare these results with the rest of our evaluation, as we use only a small subset of one of

[10] p was calculated using a *paired t-test*, see `http://www.mathworks.com/help/stats/ttest.html` for details. We use a t-test instead of an *ANOVA* because we compare two groups of F_1 scores. We use the ANOVA when we have more than two groups of F_1 scores to compare

Table 4.19.: Average classification performance over all corpus & argumentation semantics combinations

Classifier	Semantics	AB	Acc.	Prec.	Rec.	F_1
RF	-	-	0.586	0.595	0.586	0.59
	ESAA	AB_{SA}	0.641	0.65	0.641	0.646
		AB_{SAext}	0.618	0.632	0.618	0.625
	DF-QuAD	AB_{SA}	0.633	0.637	0.633	0.635
		AB_{SAext}	0.623	0.631	0.623	0.627
SVM	-	-	0.588	0.593	0.588	0.59
	ESAA	AB_{SA}	0.648	0.654	0.648	0.651
		AB_{SAext}	0.628	0.641	0.628	0.634
	DF-QuAD	AB_{SA}	0.641	0.643	0.641	0.643
		AB_{SAext}	0.633	0.638	0.649	0.635
NB	-	-	0.608	0.613	0.608	0.61
	ESAA	AB_{SA}	0.654	0.659	0.654	0.657
		AB_{SAext}	0.635	0.646	0.635	0.64
	DF-QuAD	AB_{SA}	0.647	0.648	0.647	0.647
		AB_{SAext}	0.639	0.643	0.639	0.641

the three corpora to test this scenario. The evidence that the addition of rhetorical structure arguments improves classification should accordingly be treated as anecdotal, and demands more in-depth evaluation in the future.

One may expect to observe better classification results when classifying Twitter data with a model trained on other Twitter data when compared to training models on the movie review corpus and classifying Twitter data, or the other way around. This is indeed the case when we consider the average F_1 scores shown in table 4.20. The performance in scenarios where both

Table 4.20.: Average F_1 scores, split according to whether both training and test data were Twitter corpora or one of the two was the movie review corpus

Semantics	AB	Twitter/Twitter	Twitter/Movies
Baseline	-	0.64	0.576
ESAA	AB_{SA}	0.68	0.637
	AB_{SAext}	0.661	0.628
DF-QuAD	AB_{SA}	0.652	0.627
	AB_{SAext}	0.662	0.621
All	-	0.659	0.618

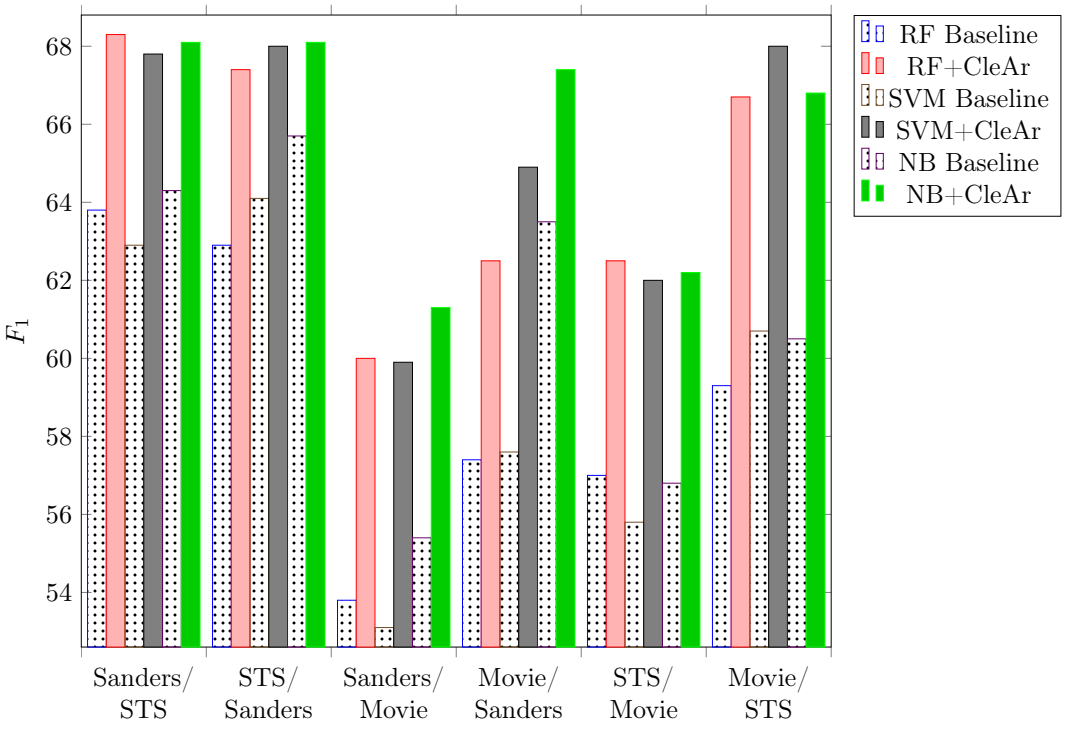


Figure 4.12.: Summary of evaluation results of applying CleAr to cross-domain sentiment polarity classification, where we compare the F_1 scores across all three baseline classifiers for each corpus combination with the results of applying ESAA together with the basic argument base, where the x-axis labels corpus combinations as *Training corpus/Test corpus*. The baselines are shown as dotted bars while the results of applying CleAr are shown in solid bars.

training and test data are provided by the Twitter corpora is, on average, 4.1 points higher than in those scenario where the Movie reviews corpus is used as either training or test data (significant with $p < 0.01$[10]). Based on these results we believe that the choice of training corpora for performing cross-domain classification is an important factor in achieving useful performance. In our case, the fact that the corpora are drawn from the same source helps our classification performance, even though the domains discussed are considerably different. This may hold lessons for choosing future training data when considering cross-domain classification. There may be corpora that are *nearer* to the problem at hand than others, whether that be in terms of content, style or both. We summarise the best performing

Table 4.21.: Summary of best performing combinations of corpora, classifiers, argumentation semantics and argument bases, according to F_1 scores

Train	**Test**	**Classifier**	**Semantics**	**AB**	$\mathbf{F}_1$
Sanders	STS	RF	ESAA	AB_{SA}	0.683
STS	Sanders	NB	ESAA	AB_{SA}	0.681
Sanders	Movies	NB	ESAA	AB_{SA}	0.613
Movies	Sanders	NB	ESAA	AB_{SA}	0.674
STS	Movies	RF	ESAA	AB_{SA}	0.625
Movies	STS	SVM	ESAA	AB_{SA}	0.68
STS	Movies subset	NB	DF-QuAD	AB_{SA}+RST	0.662
Sanders	Movies subset	SVM	DF-QuAD	AB_{SA}+RST	0.653

combinations of corpora, classifiers, argumentation semantics and argument bases from the experiments reported above in table 4.21.

4.4. Related work

Below we provide a brief summary of related work in sentiment analysis. We describe, in turn, the two primary challenges of sentiment analysis, (1) the identification of sentimental text and (2) the classification of text's polarity. Though we separate the two challenges here they have also been addressed as a single, three-class problem (positive, negative, neutral), as in e.g. [106, 122]. Indeed, Agarwal and colleagues [1] analyse Tweets both in a three-class and two-class setting, where, for the latter, they preselect opinionated Tweets and classify them as positive or negative, as we do.

Both supervised and unsupervised classifiers, e.g. [97, 158, 217], have been employed to perform sentiment analysis, as well as combinations of the two, e.g. [124]; we focus here on supervised classification, the approach we have taken in our work. For a more thorough treatment of sentiment analysis, refer to e.g. [34, 120].

As for challenge (1), a popular approach towards determining sentiment on sentence-level has been to identify *opinion-bearing* words in sentences and using them as a proxy for deciding a sentence's opinionatedness, as in e.g. [27, 104]. In section 7.1.2 we discuss ways of how we may integrate this approach with our use of opinion-bearing words as part of arguments to apply CleAr to the problem of sentiment detection. Other approaches to

challenge (1) have included the use of context in which text appears, e.g. as in [15], as well as rule-based classifiers that identify sentiment according to certain patterns appearing in text, see e.g. [232]. In the future such rule bases may prove to be useful guides in further extending our argument bases for sentiment analysis.

Challenge (2), polarity classification, has been our focus in this chapter. Whereas we classify the polarity of sentences or Tweets, the identification of polarity on a coarser level has generated much interest, also. Dave and colleagues [55] label whole reviews from *C|Net* and *Amazon* as positive or negative, while Pang and colleagues [162] classify movie reviews. They use various features to represent the reviews, including review metadata and n-gram occurrences to train the classifiers. Regardless of the level of granularity, certain classifiers have proven popular in polarity classification. These include those we have used in our work, e.g. Support Vector Machines in [116, 160] and Naïve Bayes in [60, 229]. Other classifiers commonly used include, but are not limited to, *Maximum Entropy* classifiers, e.g. [55], and various forms of *Neural Nets*, e.g. [209, 246].

While we focus on binary polarity classification, i.e. positivity vs. negativity, more fine-grained polarity scales have been proposed, also. In their research Pang and colleagues [160, 161, 162] address not only determining the polarity of text, but a broader rating-inference problem in which they attempt to infer star-ratings on a scale from one to five. Though we focus on a binary decision problem here, it may be worthwhile applying CleAr to more fine-grained classification problems such as the inference of ratings.

We have mentioned that there is a broad interest in developing solutions for cross-domain classification in computational linguistics. This appears to be particularly true for sentiment analysis, illustrated by the wealth of work on this particular topic, e.g. [21, 22, 89, 114, 159, 215, 230, 231, 236, 240]. For example, Bollegala and colleagues [21, 22] construct a *sentiment sensitive* lexicon, whose contents are used to expand feature sets that represent text. Tsai and colleagues [215] build a concept-level sentiment lexicon based entirely on common sense knowledge. We may use such lexicons to try and expand on argument bases and thus incorporate more knowledge in our classification methodology. Xia and colleagues [240] propose the *feature ensemble plus sample selection (SS-FE)* method; they also provide an overview of domain adaptation work in sentiment analysis. They create ensembles of

features based on how domain-dependent they appear to be, determining features whose sentiment carries across domains and those whose sentiment changes across domains. Sample selection refers to the process of determining a useful subset of available annotated instances, based on which feature ensembles are then built. Weichselbraun and colleagues [230] propose *contextualised sentiment analysis*, identifying and resolving ambiguous terms according to the domain in which they are used. They take domain-specific corpora and extract features from them that translate well across domains. Again, both SS-FE and contextualised sentiment analysis may guide future extensions of argument bases.

4.5. Summary & Discussion

In this chapter we have described our work on binary sentiment polarity classification. In particular, we have focused on classifying sentiment across domains, i.e. training models on a certain corpus and classifying instances from a separate, marginally related or entirely unrelated one. This is relevant because domain dependence restricts the applicability of many solutions in terms of what type of text they work well on. Here, the domain specificity of the trained model limits its transferability to sentences, or Tweets, from other domains. We lessen the impact of this by building an argument base that is comprised of arguments that are more generically applicable than the model itself. It consists of arguments whose premises are mostly unambiguously positive or negative words and the conclusion is the according sentiment. Additionally we have formalised arguments that reflect the impact of negations on such keywords, as well as arguments that reflect where, in terms of a sentence's rhetorical structure, keywords appear. Through the application of CleAr we have been able to improve commonly poor cross-domain classification to more promising levels, showing what may prove to be an alternative to the often costly construction of annotated resources.

To conclude, we believe that, depending on the problem at hand, it may be prudent to *infuse* a more or less domain specific model with more generally applicable knowledge in the way we did, as opposed to manually building new corpora for every new domain of interest. It may also prove useful to pursue a combination of these two strategies, flexibly integrating new knowledge either as features or arguments, depending on format, require-

ments, etc. Apart from identifying how to best utilise new knowledge we will also need to identify useful sources from which to obtain such knowledge, in the first place. As we have illustrated as part of the discussion of related work in section 4.4, much work is already in existence that may offer us new ways of extending our argument base and/or combining CleAr with existing solutions.

5. Relation-based Argumentation Mining

Argumentation mining is a fairly young subfield of corpus-based computational linguistics where we broadly aim to identify (1) natural language arguments from text and (2) relations between them. An argument may be related to one or more other arguments by attacking/supporting them or by being attacked/supported, itself. However, the separation of identifying arguments and their relations to each other into two independent steps raises a number of issues, such as the disregard of context in which a text appears. With ***R**elation-**b**ased **A**rgumentation **M**ining (RbAM)* we offer a novel approach to identifying argumentative text. Instead of attempting to identify individual arguments, and then relate them, we classify *sentence pairs* according to the relation that holds between the two sentences. We classify a directed relation from one sentence, the *Child*, to another, the *Parent*. Whenever this relation is classified to be an attack or support we consider *both* sentences to be *argumentative*, irrespective of their individual argumentative quality.

In section 5.1 we first motivate placing our focus on argumentative relations, rather than arguments, themselves. We then give a detailed description of RbAM in section 5.2. Throughout our work it has been important to consider ways of matching sentences to build sentence pairs. To do so we currently consider two options, which we describe in section 5.3. We can take sentence pairs extracted from text and connect them further to build directed trees of arguments. We discuss the process of doing so in section 5.4. We then describe the corpora we have worked with in section 5.5. We describe and discuss the evaluation of our work on argumentation mining in section 5.6. To conclude the chapter we discuss related work in section 5.7 and give a brief summary of this chapter's contribution in section 5.8.

5.1. Why *Relation-based* Argumentation Mining?

The separation of identifying arguments and the relations between them raises a number of problems. Here we focus on two issues we believe to be particularly relevant:

1. The disregard of a sentence's context in identifying its argumentativeness (section 5.1.1)
2. The added workload that potentially results from splitting the problem into a two-step process (section 5.1.2)

We discuss each issue in detail below. Apart from at least partially addressing these problems, considering relations between texts is necessarily part of any attempt at fully fledged argumentation mining. Our approach, as described in detail in this chapter, may thus prove useful not only on its own, as we purport in this thesis, but also in unison with argumentation mining solutions that aim to identify arguments in isolation. Should we be able to identify arguments, matching them to form pairs and classifying them may still be a viable approach towards establishing argumentative structures and relations in text. In this setting, the classification of individual sentences as argumentative or non-argumentative would function as a pruning step that precedes the identification of argumentative relation.

5.1.1. This is a fact - why does it attack this other sentence?

The context in which a sentence appears can change its meaning significantly, and with it a sentence's *argumentativeness*. Consider the following statement:

> (1) Nigel Farage[1] has attended private school and used to work as a banker in the City.

This is a simple enough fact and, on its own, conveys no particular attitude towards Nigel Farage, his education, or his professional past. If, however, we consider the above sentence in relation to the sentence below, the situation changes:

[1] Nigel Farage is the leader of the UK Independence Party (UKIP), see `www.ukip.org`

(2) Nigel Farage understands the common folks; he is the face of UKIP, the people's army!

It now becomes quite possible that sentence (1) is meant to be an attack on sentence (2) and the notion of Nigel Farage being the leader of a *people's army*. After all, how could someone who attended private school and has a history as a banker possibly understand the common people? This conclusion is not stated explicitly, but one may easily infer it. Trying to identify arguments in isolation may lead us to discard factual sentences such as sentence (1), even though, when considered in context with sentence (2), they should arguably considered to be argumentative.

5.1.2. Relating found arguments is a three-class problem

Let us consider again the task of identifying a sentence as argumentative or non-argumentative. Say we have built a model that provides us with a good split between the two classes, so that we can reliably discard non-argumentative sentences (though, as we contend in section 5.1.1, this concept may be questionable in itself). We now need to find relations, e.g. attacks and supports, between the sentences that have been classified as argumentative. In spite of our knowledge of all sentences in question being arguments, we are still faced with a three-class problem, as three scenarios need to be accounted for. A sentence may attack another, it may supports another, or, lastly, both sentences may be argumentative, but otherwise unrelated. Additionally, to fully cover the possible set of argumentative relations we may also need to classify every pair of arguments twice, once to classify whether an argument α attacks or supports an argument β and a second time to determine the relation from β to α. Thus, while we do reduce the dimensionality of our hypothesis space for the construction of an AF by discarding non-argumentative sentences, the problem itself remains a three-class problem.

5.2. RbAM overview

To address the issues described above, and to offer a way of identifying argumentative relations, we have set out to offer an alternative, *relation-based*,

view on argumentation mining (RbAM) [42]. Instead of identifying sentences, or other text, as (non-)argumentative we classify *pairs* of sentences, or sentence-like structures, according to the relation one sentence exhibits to another. To do so we create sentence pairs either directly from a text or by using some provided structure. For example, when working with text taken from debating platforms such as *Quaestio-it* [65], described in section 5.5.4, where each post is aimed at another post through an attack or support relation, we are naturally dealing with pairs of sentences. The resulting pairs are comprised of a *Parent* and a *Child*, where we want to determine whether the Child attacks or supports the Parent, or whether it does neither. When extracting sentence pairs from text the Parent is the sentence appearing earlier in the text. When taking pairs from a platform such as Quaestio-it, the earlier post is considered the Parent and the later post, which is in reply to it, is considered the Child. Whenever a Child is classified to *Attack* or *Support* its Parent we consider *both* to be argumentative, irrespective of their stand-alone argumentativeness. We assign sentence pairs to one of three classes

$$A = \{Att, Sup, N\}$$

where *Att* stands for *Attack*, *Sup* stands for *Support* and N stands for $Neither$. The Neither category may include both sentence pairs where the sentences are unrelated and those where a relation is present, but one that is not argumentative. Say, for example, two sentences describe the same entity, but do not address each other's description of it, as is the case for the following two sentences:

> Mr. Farage and his Eurosceptic UKIP party clearly pander to unreasonable fears of foreigners.

> Nigel Farage, leader of the Eurosceptic UKIP party, survived a plane crash in 2010.

Though these two sentences are related insofar as discussing Nigel Farage, they stand in no argumentative relation, and would hence need to be assigned to the Neither class.

5.3. Sentence pair construction & representation

To construct sentence pairs we have considered two strategies:

1. Compare a text to a single given sentence: The sentence serves as a fixed *parent* and the text is analysed in search of possible *children*.
2. Construct sentence pairs directly from a text: The text is analysed in search of both possible *parents* and possible *children*.

For strategy (1) we use a single sentence to serve as the *Parent* to each sentence in a text. Take, for example, the following sentence α:

> As the population across Western European countries ages quickly, refugees will eventually be a boon to the societies that welcome them.

We may want to compare this argument to other arguments from relevant text, e.g. a collection of news articles. To do so we take text and split it into a set of sentences $\{\beta, \gamma, ...\}$. Using our above argument α we then construct sentence pairs

$$(\alpha, \beta), (\alpha, \gamma), ...$$

where each needs to be assigned a class label (Att, Sup, N). Say sentences β and γ, retrieved from a news article, respectively go as follows:

> It will be a tall order to integrate all the refugees currently streaming into Europe to be successful members of society.

> Studies show that immigrants are often young and overall pay far more into the state's coffers than they claim in benefits, and refugees will be immigrants should we allow them.

We would then want to label the pair (α, β) with Att and (α, γ) with Sup since, arguably, β expresses doubts on the possibility of refugees being a boon to society, while γ points to studies supporting just that claim, and proceed likewise with any additional sentences. This provides us with an overview of arguments for and against our original argument, without having to sift through entire articles.

For strategy (2) we want to create sentence pairs directly from text, instead of relating sentences to an external sentence serving as the Parent. If the text is not very large and/or we have the computing power available we may simply create sentence pairs by matching every sentence with every other sentence appearing in the article, thus generating multiple instances of scenario (1). This, however, means that we have to classify a quadratically growing number of pairs as we consider larger texts. It may hence be prudent to preselect pairs.

To this end we match sentences to construct pairs based on them containing one or more of the same nouns. If two sentences in a text contain the same noun we create a pair, where we consider the sentence appearing *earlier* in the text as the Parent and the one appearing *later* as the Child. Both the focus on nouns and the consideration of directionality based on where sentences appear are simplifying assumptions to narrow the search space. We discuss alternatives in section 7.1.3. As our intention is to construct trees of arguments we want to minimise the number of isolated sentence pairs. We hence only consider nouns that appear at least three times within a text. Say the above sentences, α, β and γ appeared in a text, in that same order. Since they all share common nouns we would construct the following sentence pairs:

$$(\alpha, \beta), (\alpha, \gamma), (\beta, \gamma)$$

Once we have constructed sentence pairs, by either strategy (1) or strategy (2), we need to represent them as vectors. This then enables us to train classification models and classify new instances. We considered two of the options described in section 2.2.2:

1. Binary bag-of-words representation
2. Feature-based representation

The bag-of-words representation is handled by the *Weka* toolbox [87] and there is no need to process the text beyond the preprocessing we describe in section 2.2.1. For the feature-based representation we use a collection of semantic and syntactic features. The feature engineering conducted during this development also formed the foundation of what was to later shape our argument base for RbAM, which we describe in detail in section 6.2. More

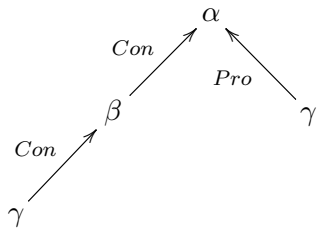


Figure 5.1.: Example argument tree created from sentence pairs that share common arguments, using strategy (2).

specifically we represent each sentence pair with the features shown in table 5.1.

5.4. Argument tree construction & visualisation

Once we have constructed sentence pairs we may link them further, forming trees of related arguments. In our work we construct trees of linked arguments, based on them sharing either a Parent or a Child. We have chosen to build trees, rather than more general directed graphs, because this allows us to use quantitative argumentation semantics like ESAA and DF-QuAD to evaluate the strength of the arguments that make up a tree.

Consider again our example sentences α, β and γ from the previous section, when using strategy (2). When passing through a text that contains these three sentences, in this order, we would construct the pairs (α, β), (α, γ) and (β, γ) and represent their relations to each other as a tree structure, as shown in figure 5.1.

When we opt for strategy (1) and compare a text to a single sentence the result is a simple tree with two levels, where the provided sentence forms the root node and all sentences identified to be attacking or supporting it form the second level of the tree. When we apply argumentation semantics here the strength calculation provides an aggregate of the amount of attacks and supports on the root node in a single measure. Say, for example, we assign the root node a default strength of 0.5. Whenever there are more arguments attacking the root node than there are ones supporting it, applying DF-QuAD will yield a score that is lower than its base score of 0.5. The larger the count of attacking arguments in comparison to supporting arguments

Table 5.1.: Overview of features used to represent sentence pairs

Type	Name	Details
Similarity	word2vec	Measuring of similarity based on word vectors
	WordNet	Measure of semantic similarity based on word distances in WordNet
	Edit Distance	Count of *replace* and *delete* operations to turn the Child into the Parent
	Word overlap	Count of words appearing both in Parent and Child
Sentence metrics	Punctuation	Sentence ending punctuation (Full stop, question mark, etc.)
	Sentence length	Word count of the Child and Parent
	Average word length	Average word length across Child and Parent
Word lists	Discourse markers	Three counts of discourse marker occurrences, one each for Attacking, Supporting and ambiguous markers
	Sentiment words	Two counts of sentiment word occurrences, one each for positive and negative words
Sentiment	Stanford	Two sentiment scores obtained by running the Stanford CoreNLP sentiment library, one each for the Child and Parent
	SentiWordNet	Average sentiment scores of words, three each (Positivity, Negativity, Neutrality) for the Parent and Child

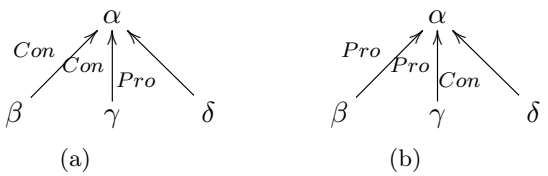


Figure 5.2.: Example argument trees resulting from employing strategy (1).

is the further the root node's score will move towards *zero*, away from its base score. Conversely, it will move towards *one* as the count of supporting arguments grows in comparison to attacking arguments. For example, a root node with a base score of 0.5, two attacks and one support, as shown in figure 5.2a, would have a score of 0.375, while one with two supports and one attack, as shown in figure 5.2b, would have a score of 0.625.

When we choose strategy (2) and create sentence pairs directly from text, we generate tree structures whose depth and breadth are limited only by the size of the text and the amount of argumentatively related sentences in it. This means that visualising the argumentative structure may help us in understanding and interpreting the arguments and their relations to one another. The analysis of a text may yield multiple trees, one for each group of sentences that share common nouns and are related through attacks or supports. Say, for example, the argumentative structure of a piece of text is as in figure 5.1 and of a separate set of sentences is as shown in figure 5.3. Once we have calculated the dialectical strength of the arguments we can store such trees in a format that allows their combined visualisation as subtrees of a single tree. We generate a file that can be opened in a browser to view the tree structure as shown in figure 5.4. We borrow the visualisation from Evripidou and Toni [65], where it is used to visualise trees of connected arguments in the online debating platform *Quaestio-it*.

Quaestio-it provides an interactive way of leading debates, which are then evaluated using the ESAA argumentation semantics. In the visualisation the calculated strength is highlighted by rendering stronger nodes visibly larger. This provides a quick insight about the dominant (strongest) nodes within a tree. Hovering over each node displays additional information for each comment or answer, including its text and calculated strength. The edges, directed upwards in the tree, connect the nodes to indicate the re-

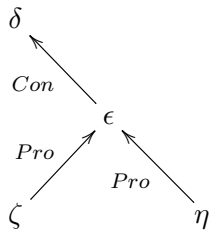


Figure 5.3.: Example argument tree resulting from employing strategy (2).

lations between question, answers and arguments. Dashed edges indicate direct answers to the question, while solid, red (-) or green (+), edges show attacking or supporting arguments on the answers or on other arguments. This visualisation is well aligned with our needs: the root nodes of the sub-trees are akin to answers in Quaestio-it, while the *question* in Quaestio-it is replaced with a *dummy* node that allows us to visualise individual trees as a single one.

5.5. Corpora

The computational linguistics community has produced ample resources when it comes to publicly available corpora. For our research we have explored numerous such corpora and their applicability to our aims, while also developing our own corpora. Below we describe the following corpora, each of which we have used to build classification models:

1. Internet Argument Corpus (IAC) [225]
2. AIFdb corpus [109, 110]
3. News articles corpus [41]

While corpora (1) and (2) are based on third party corpora, we have purpose built corpus (3) as part of our developments. Corpora (2) and (3) consist of pairs of sentences or sentence-like structures. The constituents of corpus (1) may be comprised of multiple sentences. While corpora (1) and (2) are comprised of instances from the *Att* and *Sup* classes, only, corpus (3) contains instances assigned to the *N* class. We describe each corpus in detail below. We have explored the use of a number of other corpora

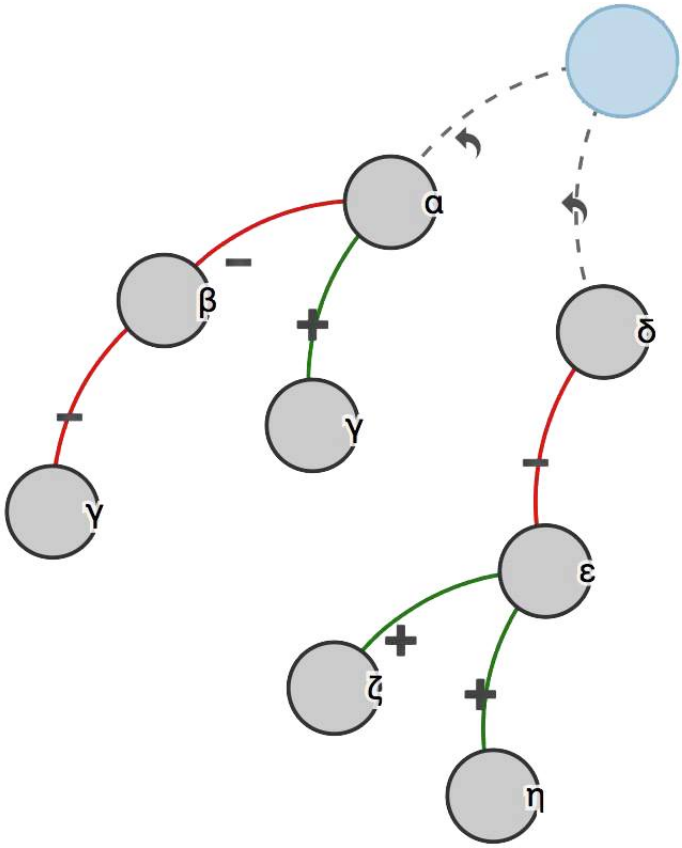


Figure 5.4.: Schematic view of Quaestio-it visualisation, joining example argument subtrees from figures 5.1 and 5.3

that we ultimately chose not to use for our developments, e.g. due to them being too small or their structure not being amenable to our analysis. We have compiled these corpora from the online debating platform Quaestio-it, as well as the *Amazon*[2] and *IMDB*[3] web presences. Additionally we have used the Arguana corpus [224], a collection of annotated *Trip Advisor*[4] hotel reviews, which formed a review corpus together with data taken from Amazon. We summarise these corpora in section 5.5.4. A summary of corpus sizes and class distributions for all five corpora is shown in table 5.2.

[2]We have used reviews from the UK website `www.amazon.co.uk`
[3]`www.IMDB.com`
[4]`www.tripadvisor.com`

Table 5.2.: Corpora overview with counts split by class labels

Corpus	Att Count	Sup Count	N Count	total Count
IAC	318	637	0	955
AIFdb	1,014	6,518	0	13,995
News articles	413	456	1,385	2,254
Quaestio-it	74	75	0	149
Reviews	198	500	0	698

5.5.1. Internet Argument Corpus

The Internet Argument Corpus (IAC) [225] is comprised of text extracted from the debating website *4forums*[5]. As part of their posts 4forums users have the option of quoting other users and replying to the quoted text. From posts that contain quotes Walker and colleagues [225] have created annotated pairs of quotes and responses to the quotes, *Q-R pairs*, as well as chains of related posts. As these text passages are clearly marked in the forum, they are easily identifiable. The format of this corpus, however, is problematic for our purposes, due to the fact that quotes and responses need not be limited to single sentences. To illustrate consider the Q-R pair below, taken from a discussion on the topic of evolution:

> Q: How can you say such things? The Bible says that God CREATED over and OVER and OVER again! And you reject that and say that everything came about by evolution? If you reject the literal account of the Creation in Genesis, you are saying that God is a liar! If you cannot trust God's Word from the first verse, how can you know that the rest of it can be trusted?
>
> R: It's not a literal account unless you interpret it that way.

The structure of the Q-R pairs is rather similar to that needed in RbAM, as we may consider the quote of the pair to be the Parent and the response to be the Child. However, the quote in this pair is substantially longer than one sentence. This holds true for quotes, as well as responses, in other pairs, also. Despite this we use the corpus to experiment with the range of text from which we may extract argumentative relations.

[5] `www.4forums.com`

Table 5.3.: Example pairs from the IAC

Parent	**Child**	**Label**
most pro lifers do not care about women hence why they are pro life	it is idiotic thinking like this that makes me respect you less and less	Att
the author seems to think there is no extra-biblical evidence of a young earth.	all of those so called evidence have been debunked by real scientists with real degrees and have true understanding of science.	Att
yeah free to do whatever you want without worrying about the consequences the joy of a life without believing in god	that is right I do not want another parental figure to tell me what to do when to do it and how to do it	Sup
did anyone else expect anything less these evil fundie christians can have affairs or even marriages yet gay people are a threat to marriage by wanting to get married	you hear that cry allowing gays to marry will cause the downfall of civilization but you never hear how or why	Sup

The IAC is split into a number of categories, each of which is a *contentious topic*, such as the acceptance or dismissal of evolution. The annotations provided by the IAC were obtained using the *Mechanical Turk* service from Amazon. Here, human annotators complete *Human Intelligence Tasks (HITs)*, which, in this case, were the annotation of Q-R pairs according to a number of features. The annotators are then paid for each HIT, providing the HIT is approved according to some predefined quality measure. Using this web service the developers of the IAC have obtained a total of $10,003$ annotated Q-R pairs. The annotators executing the HITs, the *Turkers*, followed an *annotation scheme* to provide judgements on certain qualities of the Q-R pairs. The annotation scheme asks questions such as the following:

> Agree/Disagree: Does the respondent agree or disagree with the prior post?
>
> Fact/Emotion: Is the respondent attempting to make a fact based argument or appealing to feelings and emotions?

Such judgements were either scalar or binary. In some cases the Turkers

thus needed to provide a numerical value, e.g. on *how factual* a response was. In other cases they needed to provide *true/false* judgements, e.g. whether statements were sarcastic or not. We constructed our corpus based on annotations where the Turkers judge Q-R pairs on a scale from -5 (attack) to $+5$ (support). We chose a subset of Q-R pairs based on three criteria:

1. The score for a Q-R pair, averaged across Turkers, needed to be below or equal to -2 for attacks and above or equals to 2 for supports
2. The word count of members of a Q-R pair needed to be above three
3. The word count of members of a Q-R pair needed to be below 50

We applied criterion 1 to avoid the use of examples where the Turkers either assigned differing scores or agreed on a middling score, pointing to the Q-R pair being ambiguous or not argumentatively related, respectively. It may have been possible to assign those Q-R pairs with a middling score to the Neither class, but we decided against doing so. We did so because from the averages assigned it was not possible to determine whether Turkers had actually assigned middling scores or whether they simply disagreed, in which case the score would not allow us to conclude that the response is actually neither an attack on nor a support for the Quote. We applied criteria 2 and 3 to avoid overly short and long Q-R pairs. Example pairs are shown in table 5.3.

5.5.2. Argument Interchange Format corpus

Based on the Argument Interchange Format, described in section 2.1.4, Lawrence and colleagues [109, 110] have built the *Argument Interchange Format corpus*, or *AIFdb*[6], an online repository of corpora annotated according to AIF specifications. It is comprised of 54 sub-corpora concerned with a variety of topics, ranging from discussions that have taken place on the *BBC Moral Maze*[7] to argumentative micro-texts. Each corpus is subdivided into a collection of *argument maps*, graph structures like the one shown in figure 5.5. This visualisation is offered in addition to a variety of file formats (*svg, png, json, rdf*, etc.) in which one may download either

[6] `www.arg.dundee.ac.uk/aif-corpora/`
[7] `www.bbc.co.uk/programmes/b006qk11`

Table 5.4.: Class distribution of AIFdb corpora according to S-node types (see section 2.1.4 for more details)

S-Node type	**Count**	**Class label**
CA	1,014	Att
MA	595	N/A
PA	32	N/A
RA	6,518	Sup
TA	6,497	N/A
YA	14,187	N/A

individual such argument maps or whole corpora. Of the 54 corpora we have chosen a subset of 18 and downloaded them in json format. We used this subset because the remaining 36 corpora were either small (below 20 argument maps) or the content was in a language other than English. We also decided not to use the subset of corpora on the *Language of Opposition* [76], because argument maps were generally comprised of single pairs of *I-nodes* and thus exhibited little argumentative structure.

To build a corpus of sentence pairs, or pairs of sentence-like structures, from the 18 corpora, we extracted all pairs of Information nodes (I-nodes) from the argument maps that are connected through a single Scheme node (S-node). We then grouped the resulting pairs according to the type of the S-node connecting them. Throughout the 18 corpora, *six* types of S-nodes are used, those that are part of the original AIF definition described in section 2.1.4, as well as three types introduced as part of AIF^+ [178, 179]: MA, YA and TA-nodes. The overall class distribution is shown in table 5.4. For details on the node types refer to section 2.1.4. Table 5.5 shows example pairs from each class, apart from PA-nodes, which we discarded due to their low number of occurrence.

We created two corpora from the AIFdb, one binary and one with five classes. To create a *Two-class* corpus we treated CA-nodes as attacks, while RA-nodes were considered supports. For the *Five-class* corpus, on the other hand, we simply retained the original S-node types and used them as class labels. To ensure that the sentence pairs were comprised of sentences or sentence-like structures we removed all instances where either the Parent or the Child were shorter than two words.

We additionally considered creating a Three-class corpus, where MA-

Figure 5.5.: Screenshot of visualisation provided by AIFdb [109, 110]

Table 5.5.: Example pairs from the AIFdb corpus, where the RST labels are given, with class labels (Att,Sup,N) in brackets

Parent	**Child**	**Label**
the fireman has used too much water to quench the fire	these judgments are very difficult to make	CA (Att)
With regard to the second premise it is far from obvious that the Bible teaches eternal punishment for minor sins.	The Bible teaches that God will eternally punish minor sins	CA (Att)
Therese should tell him what she does want, what she would like Paul to do for her.	Therese has to tell Paul what she would like.	MA
Men and women just think differently.	part of it has to do with the way men and women think.	MA
families have low income.	And the family cannot afford a number of food and clothing items.	RA (Sup)
Strawberries are good for your ticker, too.	Strawberries are the best choice for Your breakfast meal!	RA (Sup)
Family lives in overcrowded housing.	there are 5 million people on the waiting list for social housing.	TA (Sup)
No parent in the family is in work.	we have a huge problem with unemployment.	TA (Sup)
What should they do in order to look after their children?	they should do xxx in order to look after their children.	YA
Wife: 'I don't think that would uh I wouldn't agree to that!'	Wife wouldn't agree to that.	YA

nodes would have been labeled as N. We eventually opted against creating such a corpus, however, as there was no clear link between MA-nodes and the N class. Reframing someone else's statement may have a variety of purposes, such as reinforcing someone's statement; this may arguably make the restatement a support of the original statement. It thus seemed like somewhat of a leap to consider MA-nodes as neither attacks nor supports.

For both the Two- and the Five-class corpora the class distribution is skewed heavily towards instances labeled as *Sup*, i.e. RA and TA, with $13,015$ instances belonging to these classes. To train unbiased classification models we hence balanced the corpora, where we subsample from each class to obtain an even split among class labels. This gave us a total of $2,028$ instances for the Two-class corpus and $2,975$ instances for the Five-class corpus.

5.5.3. News articles corpus

To build a corpus of sentence pairs concerned with political news we have used data dumps extracted from *The Juicer*, a web application developed by the BBC News labs[8]. The Juicer provides a database of news articles from various sources that are tagged for entities such as people, places, events and other concepts. In addition to providing such metadata the Juicer also allows users to filter articles based on keywords and categories. We have used the filter and keyword search facilities to create a corpus comprised only of sentence pairs that are concerned with the *UK Independence Party (UKIP)*. In the News articles corpus each pair is assigned to one of the three classes Att, Sup, N. In a first phase we annotated a preliminary set of sentence pairs constructed in the following manner: We extracted 84 single sentence comments concerned with UKIP from the comments sections of articles on the Guardian website[9]. We then selected 15 sentences from the UKIP article data set and created $1,260$ $(84 * 15)$ sentence pairs where each of the sentences taken from the Guardian website serves as a Parent node to each of the sentences taken from the UKIP article data set. The strategy to match sentences taken from news articles with arguments from comment sections is based on the aim to try and model strategy (1) for sentence pair construction, as described in section 5.3. Here we put a single argument,

[8] `www.bbc.co.uk/partnersandsuppliers/connectedstudio`
[9] `www.theguardian.com/uk`

such as the one below, in relation with full texts, such as news articles:

> I think Farage makes a much better leader than any of the other parties because he's not a PR man like the current leaders of the 3 main parties.

Since the amount of sentence pairs that fall into class N far outweighs the pairs belonging to the first two classes, balancing the data set resulted in an initial set of 363 annotated sentence pairs (121 sentence pairs from each class). We created this initial corpus as an aide to developing first RbAM solutions and to allow us to run the entire pipeline of extracting pairs and classifying them based on a real model. Since this corpus is neither large enough nor created in a principled manner (we did, for example, not test for inter-annotator agreement), in the second phase we developed an annotation scheme based on which we have obtained a larger corpus.

After initial attempts at crowd-sourcing the annotation of this larger corpus, described below, we have opted to obtain the corpus annotations *offline*, after all. The instructions given to the annotators are shown in appendix C. Following the instructions we gave three example sentence pairs, one for each possible class in (Att, Sup, N). Additionally we clarified the task by explaining the role of implicit information in text, as well as providing definitions for arguments, premises and conclusions. The corpus resulting from the second run of offline annotation is comprised of $2,274$ instances. The split between classes in the corpus is as follows:

- $Att = 413$
- $Sup = 456$
- $N = 1,385$

We selected a random subset of 200 instances to be labeled by three annotators to test for inter-annotator agreement. We calculated agreement using *Fleiss' kappa*, which provides an agreement measure that takes the probability of chance agreement into consideration. Agreement on the 200 instances lay at $\kappa = 0.4287$ (Standard Error = 0.0705, 95% Confidence Interval = 0.2905 to 0.5670). Though the significance of kappa values varies from one problem to the next, a popular rule of thumb, proposed by Landis and Koch

[108] states that a kappa larger than 0.4 may be considered to signal moderate agreement between annotators, where a kappa larger than 0.6 would mean substantial agreement. We believe that, given that the annotation of argument relations is a rather ambiguous task, agreement at this level is encouraging. Other complex tasks have yielded similar levels of agreement, as shown for example in [187, 196]. Furthermore, the reported kappa is based on just one iteration of annotations. With more resources we may have run multiple iterations of annotation, revising the guidelines at each pass and hopefully achieving higher agreement.

As with the AIFdb corpora we created two corpora, a Two-class (Att, Sup) corpus and a Three-class (Att, Sup, N) corpus and balanced the class distribution in each, retaining 413 instances per class, a total 826 for the Two-class corpus and $1,239$ for the Three-class corpus. Example sentence pairs, taken from the final corpus, are shown in table 5.6.

Crowdsourcing experiments

To build the News articles corpus we initially considered using one of the Crowdsourcing platforms *Crowdflower*[10] or *Amazon Mechanical Turk*[11]. We decided against using Mechanical Turk early on because one needs a US-based IP address in order to use it. To determine the viability of using Crowdflower we conducted an *annotation test run*. This test run yielded rather discouraging results, both in terms of inter-annotator agreement and usefulness of feedback provided by the annotators. Accordingly, we opted to run annotations *offline* with a smaller group of annotators, in a more controlled environment. The experiments on Crowdflower did, however, provide some anecdotal insights towards understanding why the identification of arguments may be a complicated task, even for humans. To illustrate these we summarise the results of the initial experiment on Crowdflower below.

During the experiment each instance was labeled by three separate annotators. Though the instructions were generally judged to be clear and the participants considered the task to be complex, but achievable, we achieved rather poor agreement between the annotators. *Fleiss' Kappa* for 3 raters lay at $\kappa = 0.1532$ (Standard Error = 0.0438, 95% Confidence Interval = 0.0674 to 0.2390) and agreement was particularly low on determining attack

[10] `www.crowdflower.com`
[11] `www.mturk.com/mturk/welcome`

Table 5.6.: Example pairs from the News articles corpus

Parent	**Child**	**Label**
I think Farage makes a much better leader than any of the other parties, he's full of enthusiasm and is not a PR man like the current leaders of the 3 main parties.	I was delighted to cast my vote for [UKIP leader] Nigel Farage's team in Sussex, where I live, said Lynam.	Sup
UKIP do not bear scrutiny either as a coherent policy offer or a group of potential ministers , they are an insurgent party of protest with one message.	Mr Cameron dismissed UKIP's showing as a protest by voters and promised to stay true to his principles.	Sup
The sooner we are out of the EU, the sooner we can start making this country great again.	Isolating Britain and damaging Britain's reputation is not the right history to write, he said.	Att
A forensic examination of UKIP's policies would nail once and for all the Establishment lie that they're a bunch of swivel-eyed racists.	Asked whether some UKIP members were closet racists, he added that the truth is some UKIP members are.	Att
A forensic examination of UKIP's policies would nail once and for all the Establishment lie that they're a bunch of swivel-eyed racists.	The question now, for Farage's nervous rivals in the big three parties, is what next?	N
UKIP are a joke outfit full of small-minded Neanderthals who think the 1950s are the future.	On Friday, Mr Farage accepted the Lib Dem leader's offer of a debate, saying he could hardly wait.	N

and support relations. Only 1 percent of instances were labeled as attacks by all annotators while 5 percent were labeled as supports. We believe that a variety of reasons may have played their part in yielding faulty annotations and poor agreement between annotators.

First, instances often seemed to be labeled according to the *sentiment* of the Child, rather than *the relation between the sentences*. This meant that pairs were labeled as an attack when the Child appeared to be negative and as a support when the Child seemed positive. The below examples illustrate this:

> *Parent:* UKIP are an independent party and have shown that their appeal is broad, making big dents in staunchly Labour voting areas as well as the recent bi-election successes.
>
> *Child:* He was accused of sexism by fellow panelist Claire Gerada, chairman of the Royal College of GPs.

This sentence pair was labeled as an attack, despite the two sentences being clearly unrelated. It may be that the annotators conflated the negativity of the Child with its argumentative quality in relation to the Parent, leading them to consider it to be an attack. In a similar example, the Child's positivity may have had the same effect:

> *Parent:* The people and media that are criticising UKIP are highlighting the genuine concerns about UKIP and their policies towards immigrants.
>
> *Child:* In Barnsley Central last month, UKIP came an impressive second to Labour - on the back of a locally focused campaign, which argued that money going to Europe should be spent on the town instead.

This sentence was labeled as a support, again seemingly based on the sentiment of the Child sentence. One may concoct a justification for labelling this instance as an attack, saying that an impressive second place counteracts the notion of concern about UKIP in the media. Alternatively one may judge the Child to neither attack nor support the Parent. There is, however, hardly a way to reasonably claim that the Child sentence supports the Parent.

Another issue that appeared to cause problems was the use of discourse markers in the Child sentence, such as *also* or *but*. In many cases, sentence

pairs that were unrelated were labeled as attack or support, seemingly because the Child sentence started with an apparent discourse marker. This is arguably caused by the way we construct sentences, not generally preserving adjacency of sentences in their original context. In the future it may be valuable to incorporate some sort of consideration of adjacency, for example by preferring sentence pair candidates that appear in text right after one another. For example, the below sentence pairs were labeled as support:

> *Parent:* It is because of UKIP that we are finally discussing the European question and immigration, and thank goodness for that.
>
> *Child:* It was also reported, by The Times, that UKIP MEP Godfrey Bloom told, at the same event, he and party leader Nigel Farage wanted an invitation to a bunga bunga party hosted by former Italian prime minister Silvio Berlusconi.

> *Parent:* It is because of UKIP that we are finally discussing the European question and about immigration and thank goodness for that.
>
> *Child:* And we used to be the natural governing party of this country when I joined it.

while the following sentence pair was labeled as an attack:

> *Parent:* UKIP's strong performance in traditional working-class constituencies could see it snatch a number of Labour seats at this year's general election.
>
> *Child:* But there are questions about how effective this strategy will be, and there is still perhaps a large cultural gulf between a typical UKIP member and a typical Lib Dem.

We also found some evidence that the use of colloquial language appeared to cause issues. The below sentence pair was labeled as a support, though, if considered to be related at all, the pair should arguably be labeled as an attack. The description of UKIP as a *local pub bore* may have caused confusion among non-native speakers attempting the task as to what the sentiment of this term is.

Parent: UKIP defied the odds again at last year's European elections, beating Labour into third place, much to the dismay of those, on the left and right, who had written the party off as a busted flush.

Child: UKIP are the political manifestation of *the local pub bore*.

5.5.4. Other corpora

Throughout our work on argumentation mining we have worked with a range of corpora, while also developing our own corpora. Below we provide a brief description of two corpora we eventually opted not to use for our experiments, that have nonetheless guided the developments presented in this thesis. Other corpora recently developed to aid Argumentation Mining include, but are not limited to, [2, 103, 198].

Quaestio-it corpus

Quaestio-it [65], described in section 5.4, is an online debating platform where users can create topics, ask questions, post answers, comment and vote for good answers. It provides an interactive way of engaging in conversation, regarding any question within the platform. In order to obtain the relations between arguments within a debate, each user, when posting an argument, has to explicitly state the nature of the comment (i.e. attacking or supporting argument). Accordingly, Quaestio-it provides us with data in the exact format needed. Pairs of text, usually sentences, are linked via attack and support relations, provided by the users of the platform. This obviates any need for manual annotation on our side. From Quaestio-it we can thus directly extract tuples of related Child and Parent sentences. Note that, due to the nature of the platform, this corpus does not contain any sentence pairs labeled as N. Each argument has a maximum length of 160 characters, but its contents are otherwise unrestricted. This means that each Child or Parent need not be proper sentence, but their structure generally resembles some sort of sentence structure. The exception to this have been very short posts, where, for example, users may have posted attacks by simply posting *No* or *Not true!*. From the data retrieved from Quaestio-it we have hence removed any tuple where either the Child or Parent are shorter than four

Table 5.7.: Example pairs from the Quaestio-it corpus

Parent	**Child**	**Label**
nuclear energy is the most efficient once all potential hazards are eliminated	zero co emissions and can now be produced much safer than previous generations	Sup
friends best comedy series ever nothing comes close to that	friends is my favourite television series it has a great sense of humor and very entertaining	Sup
samsung produces some great tvs since they have fantastic image quality and are full of features	I have had some bad experience with the reliability of samsung products	Att
marihuana is the worst smelling thing in the world legalising it would kill my sense of smell	I do not like the smell of fish that does not mean it should be illegal	Att

words. Examples from the final corpus are shown in table 5.7. The reason we decided not to use this corpus as part of our evaluations was its small size of less than 200 instances.

Customer reviews

Apart from using third party corpora we have built a corpus of sentence pairs, comprised of hotel reviews taken from *TripAdvisor* and reviews posted on the UK presence of *Amazon*. Like the Quaestio-it corpus this corpus is comprised of instances of two classes *Att*/*Sup*. To create the corpus we extracted sentence pairs from reviews in the following manner:

- Crawl website for all reviews on a single product (Amazon only)
- Concatenate individual reviews into one larger text
- Create sentence pairs from sentences mentioning the same entities
- Extract resulting pairs together with metadata provided by reviews (e.g. star rating of the review in which each sentence appears)

We did not have to crawl the web for TripAdvisor reviews since we used an existing corpus of hotel reviews [224]. Once we had created sentence pairs

Table 5.8.: Example pairs from the Customer reviews corpus

Category	Parent	Child	Label
TripAdvisor	there were no restaurants or shops nearby, and driving back to the airport was also a bit of a challenge.	the hotel is very close to the airport, about 2 miles from university and about 6 miles from the car rental.	Att
	the bed was hard as a rock.	the bed was comfy and there was a fridge and micro.	Att
	the restaurant is fantastic as well.	I ordered food from the restaurant and it was good.	Sup
	the manager that we spoke to was efficient, helpful, and super concerned with the quality of our stay.	we particularly booked here for the golf and were not disappointed, mr gaffney the manager and the golfing support staff were so professional in all their roles.	Sup
Amazon	there were no restaurants or shops nearby, and driving back to the airport was also a bit of a challenge.	the hotel is very close to the airport, about 2 miles from university and about 6 miles from the car rental.	Att
	the bed was hard as a rock.	the bed was comfy and there was a fridge and micro.	Att
	the restaurant is fantastic as well.	I ordered food from the restaurant and it was good.	Sup
	the manager that we spoke to was efficient, helpful, and super concerned with the quality of our stay.	we particularly booked here for the golf and were not disappointed, mr gaffney the manager and the golfing support staff were so professional in all their roles.	Sup

in the above manner we manually annotated each pair. Example sentence pairs, taken from the final corpus, are shown in table 5.8.

5.6. Evaluation

To test the performance of RbAM and gain insights into its general feasibility we have built a number of classification models for each corpus discussed in sections 5.5.1 to 5.5.3. Below we describe the results of training models using the classifiers described in appendix D:

1. Random Forest (see appendix D.1)
2. Support Vector Machines (see appendix D.2)
3. Naïve Bayes (see appendix D.3)

We have experimented with both modes of representation described in section 5.3. We exemplify this in section 5.6.3, but otherwise focus on reporting results obtained using the bag-of-words representation. We have chosen to focus on this representation as it is considerably simpler in terms of preprocessing and computation needed to represent instances numerically, and choosing the more complex representation actually produced marginally inferior results. For each corpus we choose the most successful classification models yielded by the below evaluation to serve as our baselines in developing CleAr for RbAM, which we describe in chapter 6. Again, we report the confusion matrices for each classifier, as well as the standard measures, accuracy, precision, recall and F_1 scores.

5.6.1. Internet Argument Corpus

Table 5.9 shows the classification performance of classifiers trained on the IAC. Random Forests yield the best performance, with an F_1 score of 0.676. This is, however, not significantly higher than the F_1 score achieved by the Naïve Bayes classifiers, which lies at 0.675. Table 5.10 shows the confusion matrices for each classifier. All three classifiers exhibit similar distributions, with no apparent bias towards either of the two classes.

Table 5.9.: Classification performance for the *IAC*, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold.

Classifier	**Accuracy**	**Precision**	**Recall**	**F_1**
Random Forest	0.676	0.676	0.676	**0.676**
Support Vector Machine	0.583	0.583	0.583	0.583
Naïve Bayes classifiers	0.675	0.675	0.675	0.675

Table 5.10.: Confusion matrices for classification of the *IAC*

		Classified as	
		Att	Sup
True label	Att	213	105
	Sup	102	218

(a) Random Forest

		Classified as	
		Att	Sup
True label	Att	187	131
	Sup	134	184

(b) Support Vector Machines

		Classified as	
		Att	Sup
True label	Att	217	101
	Sup	106	212

(c) Naïve Bayes classifiers

5.6.2. AIFdb

For AIFdb we report results on the two versions described in section 5.5.2. Results for the Two-class corpus are shown in tables 5.11 and 5.12, while tables 5.13 and 5.14 show results for the Five-class corpus. For the Two-class corpus it is again Random Forests that achieve the highest performance, with an F_1 score of 0.868, while for the Five-class corpus Support Vector Machines produce the best performing classifier, with an F_1 score of 0.642. For the Two-class corpus both the Random Forest and the Support Vector Machines appear to exhibit a slight bias towards classifying instances as attacks, while for the Naïve Bayes classifiers the opposite seems to be the case.

It appears that the *MA* class in the Five-class corpus is the one most reliably identified. This seems reasonable, as this class is comprised of sentence pairs where the Parent *reframes* the Child, as illustrated in table 5.5. These pairs are thus generally comprised of a Parent and Child that are both semantically and syntactically very similar. Perhaps surprisingly, this does not quite hold true for the *YA* class in the Five-class corpus, which addresses connections explaining illocutionary force, despite most sentence

Table 5.11.: Classification performance for the *Two-class AIFdb corpus*, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold.

Classifier	**Accuracy**	**Precision**	**Recall**	**F_1**
Random Forest	0.836	0.836	0.836	**0.836**
Support Vector Machine	0.762	0.766	0.762	0.764
Naïve Bayes classifiers	0.669	0.676	0.669	0.673

pairs in this class being comprised of two similar sentences, as well.

Table 5.12.: Confusion matrices for classification of the *Two-class AIFdb corpus*

		Classified as	
		Att	Sup
True label	Att	843	171
	Sup	161	853

(a) Random Forest

		Classified as	
		Att	Sup
True label	Att	835	179
	Sup	304	710

(b) Support Vector Machines

		Classified as	
		Att	Sup
True label	Att	781	233
	Sup	438	576

(c) Naïve Bayes classifiers

Table 5.13.: Classification performance for the *Five-class AIFdb corpus*, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold.

Classifier	**Accuracy**	**Precision**	**Recall**	**F_1**
Random Forest	0.645	0.645	0.645	0.642
Support Vector Machine	0.668	0.666	0.668	**0.665**
Naïve Bayes classifiers	0.596	0.619	0.596	0.588

5.6.3. News articles corpus

Finally, we report the results on building classifiers for the News articles corpora. Again we show them in order of class count, starting with the Two-class corpus, followed by the Three-class corpus. Table 5.15 shows the classification performance for the Two-class corpus and table 5.17 provides

Table 5.14.: Confusion matrices for classification of the *Five-class AIFdb corpus*

		Classified as				
		CA	MA	RA	TA	YA
True label	CA	365	33	100	39	58
	MA	15	537	10	14	19
	RA	67	24	319	61	124
	TA	46	29	84	309	127
	YA	39	32	60	78	388

(a) Random Forest

		Classified as				
		CA	MA	RA	TA	YA
True label	CA	394	54	101	21	25
	MA	27	536	11	9	12
	RA	149	49	319	38	40
	TA	43	30	52	388	82
	YA	51	39	63	91	351

(b) Support Vector Machines

		Classified as				
		CA	MA	RA	TA	YA
True label	CA	272	102	83	120	18
	MA	60	460	26	48	1
	RA	88	68	279	116	44
	TA	23	42	30	487	13
	YA	29	46	53	193	274

(c) Naïve Bayes classifiers

Table 5.15.: Classification performance for the *Two-class News articles corpus*, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold.

Classifier	**Accuracy**	**Precision**	**Recall**	**F_1**
Random Forest	0.695	0.695	0.695	**0.695**
Support Vector Machine	0.691	0.692	0.691	0.691
Naïve Bayes classifiers	0.663	0.663	0.663	0.663

Table 5.16.: Classification performance for the *Two-class News articles corpus* when representing instances with features instead of a bag-of-words, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold.

Classifier	**Accuracy**	**Precision**	**Recall**	$\mathbf{F_1}$
Random Forest	0.684	0.684	0.684	**0.684**
Support Vector Machine	0.57	0.57	0.57	0.57
Naïve Bayes classifiers	0.561	0.562	0.562	0.562

the confusion matrices for each classifier. The same is shown for the Three-class corpus in tables 5.18 and 5.19, respectively. To illustrate why we have decided to represent instances with a bag-of-words, rather than with the set of features described in section 5.3, in table 5.16 we show the results of training classifiers for the Two-class corpus where instances are represented by the features summarised in table 5.1. Note that all three classifiers perform poorer than they do when we represent instances with a bag-of-words, as shown in table 5.16. This is the case not just for the results shown for this particular corpus; rather, for all corpora the feature-based representation of instances yields performance that is either on-par with or inferior to the bag-of-words representation. This may point to us having chosen features that do not represent the true class distribution amongst the corpora. However, we construct our argument base, which we use to instantiate CleAr for RbAM (see chapter 6), upon these very features and, in the form of arguments, they do help us improve classification, as we show in chapter 6.

For the Two-class corpus, Random Forests again yield the highest performance, with an F_1 score of 0.695. For the Three-class corpus, on the other hand, Support Vector Machines yield the best results, with an F_1 score of 0.608, while the Naïve Bayes classifier produces the poorest performance with $F_1 = 0.583$. It appears that distinguishing Attacks from Supports is more challenging than distinguishing between the presence or absence of an argumentative relations. Indeed, with the exception of Random Forests, all classifiers commit the largest amount of misclassifications between Attack and Supports.

Table 5.17.: Confusion matrices for classification of the *Two-class News articles corpus*

		Classified as: Att	Classified as: Sup
True label	Att	281	131
	Sup	121	293

(a) Random Forest

		Classified as: At	Classified as: Sup
True label	Att	295	118
	Sup	137	276

(b) Support Vector Machines

		Classified as: Att	Classified as: Sup
True label	Att	272	141
	Sup	137	276

(c) Naïve Bayes classifiers

Table 5.18.: Classification performance for the *Three-class News articles corpus*, obtained using ten-fold cross-validation. The highest F_1 score is shown in bold.

Classifier	**Accuracy**	**Precision**	**Recall**	**F_1**
Random Forest	0.588	0.591	0.588	0.589
Support Vector Machine	0.605	0.609	0.605	**0.607**
Naïve Bayes classifiers	0.584	0.583	0.584	0.583

Table 5.19.: Confusion matrices for classification of the *Three-class News articles corpus*

		Classified as: Att	Classified as: Sup	Classified as: N
True label	Att	206	92	115
	Sup	85	217	111
	N	49	58	306

(a) Random Forest

		Classified as: Att	Classified as: Sup	Classified as: N
True label	Att	268	89	56
	Sup	127	220	66
	N	89	63	261

(b) Support Vector Machines

		Classified as: Att	Classified as: Sup	Classified as: N
True label	Att	220	113	80
	Sup	104	229	80
	N	68	70	275

(c) Naïve Bayes classifiers

5.7. Related work

There have been two basic interpretations of how systems may support, or indeed automate, the task of argumentation mining. One is found in programs that support the manual construction of arguments and AFs by human experts. The largest group of applications that fall in this category are debating websites such as *Debatepedia*[12] or *Quaestio-it*, described in section 5.4. Beyond such websites, Wyner and colleagues [193, 238, 239] have built annotation interfaces, e.g. based on the GATE framework [53], highlighting various syntactical and lexical features in text that help identify its argumentative parts more easily. The actual argumentation framework is then constructed by a human expert who is guided through the text's content by the annotations. Another support tool, *Araucaria* [177], was designed as an interface that lets its users create diagrams of arguments, which are then stored in the *Argument Markup Language (AML)*. While our focus has been the automatic identification of arguments, as discussed below, we have used data that such systems have proposed to build solutions. For example, as discussed in section 5.5.2, we have used argument maps produced through Araucaria to build classification models. There is thus an important link between semi-automatic systems such as the ones described and fully autonomous solutions, such as we have developed.

The second stream of research is concerned with the automatic identification of arguments and relations in text, which has been our focus, as well. Much work here has focused on the identification of individual arguments in text, e.g. [24, 163], as opposed to considering relations the way we have. *AraucariaDB* [177], which constitutes a significant chunk of AIFdb described in section 5.5.2, has been a popular target for argumentative analysis. Rooney and colleagues [185] train Support Vector Machines on AraucariaDB, based on words, word lemmata and part-of-speech tags. Another development on AraucariaDB is presented by Moens and colleagues [142], who focus on identifying argumentative sentences using *multinomial Naïve Bayes* classifiers. For each sentence, eleven features are extracted upon which the classifier is trained. Among those features we find key words, punctuation, verb and adverb occurrences, etc., a similar collection of features to the one we have used. Palau and colleagues [157] compare the performance of

[12] `debatepedia.idebate.org/`

Maximum Entropy models for the classification of text as argumentative or non-argumentative with Support Vector Machines and Naïve Bayes classifiers, noting that for the investigated setting Maximum Entropy models and Naïve Bayes classifiers outperform Support Vector Machines. In comparison, in our experiments we have found that Support Vector Machines outperform Naïve Bayes classifiers on all corpora. We believe that much work on identifying individual arguments is complimentary to our work on identifying argumentative relations. While we propose that the focus on individual arguments may entail certain problems, as discussed in section 3.1, it may help us preselect argument candidates where necessary (i.e. when working with large texts). In similar spirit to our work, Lawrence and Reed [111] combine more than one approach to identifying arguments to improve overall performance. In particular, they use discourse indicators, topical similarity and argumentation scheme structures in text in unison to identify argumentative structures in text based on AIFdb.

Some research has also addressed the challenges of trying to identify relations between arguments [105, 163], as well as within arguments, such as the detection of evidence supporting claims [183]. Boltuzic and Šnajder [23] use *Textual Entailment (TE)* [54] to identify support relations between posts in discussion fora. Similarly, Cabrio and Villata [32] use TE to extract arguments and their relations from debates, *Depatepedia*[13], in particular. Based on the extractions provided by TE they then apply abstract argumentation to determine sets of acceptable arguments prevalent in an online discussion. In our work we have decided not to use TE as a proxy for argumentative relations, as initial experiments we conducted yielded rather poor performance in trying to classify sentence pairs. One reason for this is the complexity of the TE task, itself; existing TE parsers tend to produce F_1 scores in the mid-sixties, as, for example, in [125]. When applied to argumentation mining we expect this performance to be lower, still, as TE relations do not perfectly translate to argumentative relations. Apart from the performance we achieve on the Three-class news articles corpus, this would mean lower performance than what we have achieved. Indeed, the highest (macro)-F_1 score reported in [23] is 0.511.

In addition to their work with conventional supervised classifiers, Palau and colleagues [136, 137] also offer an alternative route to training classifiers

[13] `http://www.debatepedia.org`

Table 5.20.: Summary of best performing RbAM classifiers to be used as baselines in chapter 6

Corpus	Classifier	Class labels
IAC	Random Forest	Att,Sup
Two-class AIFdb corpus	Random Forest	Att,Sup
Five-class AIFdb corpus	SVM	CA,MA,RA,TA,YA
Two-class News articles corpus	Random Forest	Att,Sup
Three-class News articles corpus	SVM	Att,Sup,N

to find arguments. They instead build a *Context free grammar (CFG)* to represent argumentative structures in case law texts. As a base for their developments they use a corpus of cases from the *European Court of Human Rights (ECHR)*[14]. Similarly, Amgoud and colleagues [3] define a logical language with which to represent arguments that have been extracted from natural language text. For our own work such knowledge-heavy approaches may provide resources upon which to extend our argument base (see chapter 6) and thus incorporate the insights gained by such developments with other, data-driven argumentation mining solutions.

More application domains for argumentation mining have included other legal texts, e.g. [95, 137], biomedical texts, e.g. [66, 82, 83, 94], persuasive essays, e.g. [152, 153, 201, 203, 205], and social media [81, 155, 191]. Despite these, and other, developments, only few attempts, such as [204], appear to have been made at fully automating the process of building AFs from text. For more in-depth surveys of the field refer, for example, to [117, 119, 136, 157, 166, 221].

5.8. Summary & Discussion

In this chapter we have introduced RbAM, a relation-based interpretation of argumentation mining, where we identify argumentative relations between sentences, rather than considering individual text spans. We believe that this circumvents a number of issues caused by classifying sentences, such as the impact of context on a sentence's argumentativeness. We have considered multiple ways of constructing and representing sentence pairs. To build useful classifiers we have used both existing corpora and have developed a

[14] `http://www.echr.coe.int/Pages/home.aspx?p=caselaw/analysis&c=`

novel corpus from scratch. This has allowed us to test RbAM on a variety of types of text. To the best of our knowledge this is the first attempt to identify arguments by considering pairs of sentences as single instances.

RbAM offers a novel way of approaching argumentation mining, placing argumentative relations at the centre of analysis. We provide a first step towards building a fully fledged argumentation mining solution that not only identifies arguments, but, crucially, the relations between arguments. RbAM may prove to be successful on its own, but we believe that combining it with conventional argumentation mining approaches holds promise, as well. We describe how such consolidation with other argumentation mining solutions may be brought about in section 7.1.3. We use the best performing classifiers identified in section 5.6 as our baselines for developing an instantiation of CleAr for RbAM, described in detail in the following chapter. The classifiers we choose, based on the achieved F_1 scores, are shown in table 5.20.

6. CleAr for Relation-based Argumentation Mining

As with polarity classification we have developed an instantiation of CleAr, as described in chapter 3, for RbAM. Again, our focus was to build a variety of classification models and an informative argument base that helps us improve upon the baseline provided by the classifiers. We describe each step of CleAr, as instantiated to perform RbAM, in sections 6.1 to 6.4. To evaluate the impact the application of CleAr has on RbAM we use the baselines established in chapter 5 and attempt to improve upon them. We describe the evaluation and its results in section 6.5. We conclude with a summary of this chapter's contribution in section 6.6.

6.1. Step (1): Instance classification

The first step amounts to classifying a given instance I, using a model trained on one of the corpora described in section 5.5, where an instance is a sentence pair such as pair *pc* below:

> *Parent*: With her cautious attitude Mrs. M. will struggle as party leader!
>
> *Child*: And we can all agree that her reserved character will not please voters, either!

In the evaluation described in section 6.5 we use a *binary bag-of-words (BOW)* [189] representation, instead of the more complex, feature-based, representation, described in section 5.3. Indeed, we established as part of our evaluation of classification models in section 5.6 that the simpler bag-of-words representation actually outperforms the more complex feature-based representation. For Three-class RbAM the class labels are

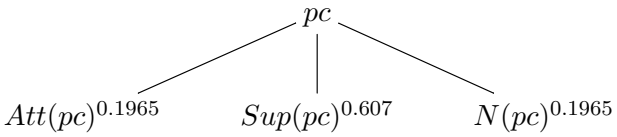


Figure 6.1.: Tree view of output of step (1) for RbAM for example pair *pc*

$$A = \{Att, Sup, N\}$$

while for Two-class RbAM $A = \{Att, Sup\}$. To illustrate, say that we have used a Support Vector Machine from section 5.6.3, trained on the News articles corpus, whose F_1 score is 0.607 to label the above example, *pc*, and it has returned the class label *Sup*. The result of Step (1) for this situation, viewed as a tree, is shown in figure 6.1, where the scores have been calculated as described in section 3.2.1.

6.2. Step (2): Argument identification

Once we have classified I we identify arguments and relations that are applicable to the instance in question, as described in section 3.2.2. We have built an *argument base* AB_{RbAM} that currently consists of 171 arguments and contains arguments from 5 categories:

1. Keyword (73 arguments)
2. First word (10 arguments)
3. Sentiment (3 arguments)
4. Similarity (12 arguments)
5. Negation (73 arguments)

We show examples from each category in table 6.1, the full list can be found in appendix A. Here, $I \in Att/Sup$ stands for instance I belonging to Att/Sup. Arguments may support or attack either the class labels Att, Sup or other arguments from AB_{RbAM}. When arguments support/attack other arguments they do so by supporting/attacking the other arguments' premise.

Note that at present there are no arguments in AB_{RbAM} attacking or supporting the N class label.

Table 6.1.: Example arguments taken from AB_{RbAM}, where Sim stands for Similarity. I is a pair $(Parent, Child)$

Category	**id**	**Premise**	**Conclusion**	S^0	**T**	**R**
Keyword	$\kappa_{disagree}(I)$	Child in I occurs in *disagree*	$I \in Att$	0.5	Pro	$\{(\kappa_{disagree}(I), Att)\}$
	$\kappa_{and}(I)$	Child in I occurs in *and*	$I \in Sup$	0.5	Pro	$\{(\kappa_{and}(I), Sup)\}$
FirstWord	$\phi_{disagree}(I)$	Child in I starts with *disagree*	$I \in Att$	0.5	Pro	$\{(\phi_{disagree}(I), \kappa_{disagree}(I))\}$
	$\phi_{and}(I)$	Child in I starts with *and*	$I \in Sup$	0.5	Pro	$\{(\phi_{and}(I), \kappa_{and}(I))\}$
Sentiment	$\sigma_1(I)$	Child in I has neg. sentiment	$I \in Att$	0.5	Pro	$\{(\sigma_1(I), Att)\}$
	$\sigma_2(I)$	Child in I has pos. sentiment	$I \in Sup$	0.5	Pro	$\{(\sigma_2(I), Sup)\}$
	$\sigma_3(I)$	Parent in I has neg. sentiment	$Premise(\sigma_1(I))$	0.5	Con	$\{(\sigma_3(I), \sigma_1(I))\}$
Similarity	$\theta_1(I)$	Sim of Parent and Child in I is $>.1$	$I \in Att$	0.5	Pro	$\{(\theta_1(I), Att)\}$
	$\theta_3(I)$	Sim of Parent and Child in I is $=<.1$	$I \in Att$	0.5	Con	$\{(\theta_2(I), Att)\}$
	$\theta_5(I)$	Sim of Parent and Child in I is $=<.1$	$Premise(\sigma_1(I))$	0.5	Con	$\{(\theta_3(I), \sigma_1(I))\}$
	$\theta_6(I)$	Sim of Parent and Child in I mainly stopwords	$Premise(\theta_1(I))$	0.5	Con	$\{(\theta_4(I), \theta_1(I))\}$
	$\theta_7(I)$	Sim of Parent and Child in I mainly nonStopwords	$Premise(\theta_1(I))$	0.5	Con	$\{(\theta_5(I), \theta_1(I))\}$
Negation	$\nu_{disagree}(I)$	Keyword *disagree* in I negated	$Premise(\kappa_1(I))$	0.5	Con	$\{(\nu_{disagree}(I), \kappa_1(I)\}$

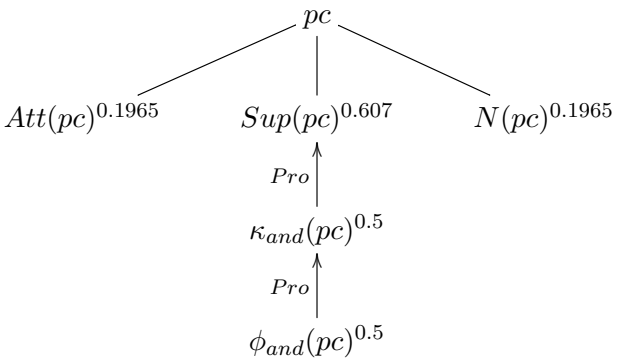


Figure 6.2.: Tree view of output of step (2) for RbAM for example pair *pc*

Arguments from the *Keyword* category are concerned with words appearing in either the Parent or the Child that are indicative of discourse, i.e. discourse markers. We use the discourse connectives proposed as part of the PDTB [172]. Additionally we check whether certain discourse markers are the first word of the Child sentence, further supporting the notion of a relation being present. This is reflected by the *FirstWord* category, whose applicable arguments support the relevant *Keyword* argument. Consider again our example sentence pair, where the keyword *And* appears as the first word in the Child. This renders two arguments applicable, namely $\kappa_{and}(I)$ and $\phi_{and}(I)$, illustrated as a tree view in figure 6.2.

For the *Sentiment* category we use the Stanford CoreNLP sentiment library [199] to classify the individual sentiment of the Child and the Parent. We have found that the sentiment of the Child is a good indicator of the argumentative relation to another sentence, if one is present, which may be mediated by the Parent's sentiment. The *Similarity* category contains various arguments reflecting different types of relatedness between the Parent an the Child. The degree of similarity is measured using Wu and Palmer's wordNet similarity measure [237] and we argue for or against relatedness depending on whether this similarity is above or below a threshold of 0.1, respectively. We have determined this threshold empirically by determining the average relatedness measure for a held out collection of sentence pairs, split by class. Whenever the similarity surpasses the determined threshold arguments apply that support both the labels *Att* and *Sup*, indicating that some relation is present. When the similarity is below the threshold, on the

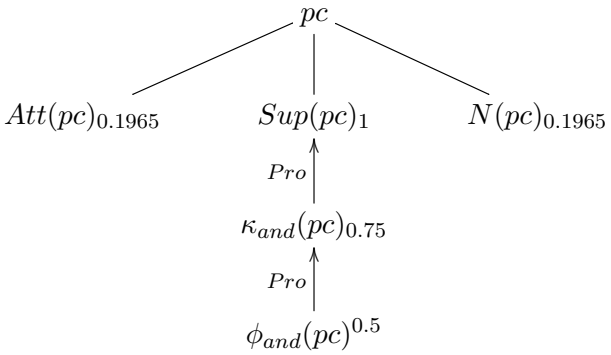


Figure 6.3.: Tree view of output of step (3) for RbAM for example pair *pc*

other hand, arguments that attack the same class labels apply. We further argue about where the similarity between Parent and Child stem from, e.g. from stop words or non-stop words. Whenever the majority of words that contribute to a high degree of relatedness are stop words an argument applies that attacks the similarity arguments. $\theta_3(I)$ in table 6.1 attacks both $\theta_1(I)$ and $\theta_2(I)$ whenever applicable. Finally we have arguments dealing with negation, $\nu_{Keyword}(I)$ in table 6.1, analogous to those used for sentiment analysis.

6.3. Step (3): Answer strength calculation

In the third step of CleAr we determine the strength of each possible class label by including the applicable arguments identified in step (2), as described in section 3.2.3. As with our experiments for polarity classification, described in section 4.3, we compare two ways of calculating answer strength for RbAM, as well. Again, on the one hand we use ESAA semantics, while, on the other hand we use DF-QuAD semantics, described in section 2.1.2. We compare results of using the different schemes as part of our evaluation in section 6.5. The output of step (3) for our running example is shown in figure 6.3, where the dialectical strength has been calculated for all nodes by applying the ESAA semantics. Since $Sup(pc)$ is the only node in the tree that has other nodes pointing towards it, it is also the only node whose dialectical strength is different from its initial score.

6.4. Step (4): Final classification decision

The classifier needs to make a choice based on the final scores of the labels that we have calculated in step (3). For example, we can determine the maximum of the scores

$$S_{Sup(I)}, S_{Att(I)}, S_{N(I)}.$$

Whichever score is the highest determines the final class label. For our example, we would thus choose $Sup(pc)$. As before, if two or more of the scores are the same we revert to the class label originally chosen in step (1).

6.5. Evaluation

To evaluate the performance of applying CleAr to RbAM we chose the best performing classifiers identified in section 5.6 for each corpus described in section 5.5 and tried to improve upon their performance. All classifiers were trained using the *Weka* toolbox [75, 87]. We did not use the 5-class AIFdb corpus in this context as this corpus' class labels do not align with AB_{RbAM}, which argues about instance labels (Att, Sup). We chose the classifiers based on F_1 scores, resulting in the following combinations of corpora and classifiers:

- Internet Argument Corpus & Random Forest
- Two-class AIFdb corpus & Random Forest
- Two-class News articles corpus & Random Forest
- Three-class News articles corpus & Support Vector Machines

We compare the performance of using classifiers, only, with that of using CleAr, applying either ESAA or DF-QuAD for class label and argument strength calculation. We conduct a total of 12 experiments, 4 (corpora) $\times$1 (model per corpus) $\times$3 (baseline vs. two argumentation semantics with CleAr).

Table 6.2.: Classification performance for the *IAC*, comparing the Random Forest baseline from section 5.6 with results of applying CleAr. The highest F_1 score is shown in bold.

Classifier	**Accuracy**	**Precision**	**Recall**	**F_1**
RF baseline	0.675	0.675	0.675	0.675
RF baseline & ESAA	0.675	0.675	0.675	0.675
RF baseline & DF-QuAD	0.692	0.692	0.692	**0.692**

6.5.1. Internet Argument corpus

For the first run of experiments we took the best performing model trained on the IAC, Random Forest, and used this model as part of CleAr. The aim was then to identify whether, by using ESAA or DF-QuAD, we may improve upon the baseline provided by the Random Forest. Table 6.2 shows the baseline classification performance in comparison to the two applications of CleAr. We found that applying CleAr with ESAA yielded no improvements on the baseline. Using DF-QuAD, on the other hand, gave us an improvement on the baseline F_1 score of 1.7 points. The confusion matrices in table 6.3 show that the improvements are similarly spread across the two classes, with *six* more Attacks and *five* more Supports classified correctly.

Table 6.3.: Confusion matrices for classification of the *IAC*, comparing the Random Forest baseline from section 5.6 with results of applying CleAr

		Classified as: Att	Classified as: Sup
True label	Att	213	105
	Sup	102	216

(a) RF baseline

		Classified as: Att	Classified as: Sup
True label	Att	213	105
	Sup	102	216

(b) RF baseline & ESAA

		Classified as: Att	Classified as: Sup
True label	Att	219	99
	Sup	97	221

(c) RF baseline & DF-QuAD

Table 6.4.: Classification performance for the *Two-class AIFdb corpus*, comparing the Random Forest baseline from section 5.6 with results of applying CleAr. The highest F_1 score is shown in bold.

Classifier	**Accuracy**	**Precision**	**Recall**	**F_1**
RF baseline	0.836	0.836	0.836	**0.836**
RF baseline & ESAA	0.836	0.836	0.836	**0.836**
RF baseline & DF-QuAD	0.843	0.843	0.844	**0.843**

6.5.2. AIFdb corpus

For the application of CleAr to the AIFdb corpora we limited ourselves to the two-class version. This is due to the fact that our argument base is concerned with determining attack and support relations, whereas in the 5-class AIFdb corpus instances are labeled according to the RST node types connecting the I-nodes. Table 6.4 shows the results of applying CleAr in comparison to the baseline. Again, applying CleAr with ESAA produced no performance improvement. Applying CleAr with DF-QuAD did yield an improvement in F_1 score by 0.6 points. Table 6.5 shows the confusion matrices for all three runs. Here we see that the only improvement made is by bettering the classification of Supports, while the classification of Attacks has actually worsened, though only by a single instance.

Table 6.5.: Confusion matrices for classification of the *Two-class AIFdb corpus*, comparing the Random Forest baseline from section 5.6 with results of applying CleAr

		Classified as: Att	Classified as: Sup
True label	Att	843	171
	Sup	161	853

(a) RF baseline

		Classified as: Att	Classified as: Sup
True label	Att	843	171
	Sup	161	853

(b) RF baseline & ESAA

		Classified as: Att	Classified as: Sup
True label	Att	844	170
	Sup	148	866

(c) RF baseline & DF-QuAD

Table 6.6.: Classification performance for the *Two-class News articles corpus*, comparing the Random Forest baseline from section 5.6 with results of applying CleAr. The highest F_1 score is shown in bold.

Classifier	**Accuracy**	**Precision**	**Recall**	$\mathbf{F_1}$
RF baseline	0.695	0.695	0.695	0.695
RF baseline & ESAA	0.695	0.695	0.695	0.695
RF baseline & DF-QuAD	0.717	0.717	0.717	**0.717**

6.5.3. News articles corpus

For the final run of experiments we compared the baseline from section 5.6.3, obtained from training Random Forests and Support Vector Machines on the Two- and Three-class News articles corpora, respectively, described in section 5.5.3. Tables 6.6 and 6.7 show the results for the Two-class corpus, while tables 6.8 and 6.9 show the results for the Three-class corpus. Again, in both cases, applying CleAr with ESAA had no impact on classification, while applying it using DF-QuAD yielded an improvement in both cases. For the Two-class corpus the F_1 score was improved by 2.2 points, for the Three-class corpus we achieved an improvement of 1.4 points. For both corpora we saw the largest improvement in the *Att* class, while for the Three-class corpus there is no improvement for the N class. This was to be expected, as we currently do not use any arguments that apply to N.

Table 6.7.: Confusion matrices for classification of the *Two-class News articles corpus*, comparing the Support Vector Machine baseline from section 5.6 with results of applying CleAr

		Classified as	
		Att	Sup
True label	Att	281	131
	Sup	121	293

(a) RF baseline

		Classified as	
		At	Sup
True label	Att	281	131
	Sup	121	293

(b) RF baseline & ESAA

		Classified as	
		Att	Sup
True label	Att	298	114
	Sup	120	294

(c) RF baseline & DF-QuAD

Table 6.8.: Classification performance for the *Three-class News articles corpus*, comparing the Support Vector Machine baseline from section 5.6 with results of applying CleAr. The highest F_1 score is shown in bold.

Classifier	**Accuracy**	**Precision**	**Recall**	$\mathbf{F_1}$
SVM baseline	0.605	0.609	0.605	0.607
SVM baseline & ESAA	0.605	0.609	0.605	0.607
SVM baseline & DF-QuAD	0.618	0.624	0.618	**0.621**

Table 6.9.: Confusion matrices for classification of the *Three-class News articles corpus*, comparing the Support Vector Machine baseline from section 5.6 with results of applying CleAr

		Classified as		
		Att	Sup	N
True label	Att	268	89	56
	Sup	127	220	66
	N	89	63	261

(a) SVM baseline

		Classified as		
		Att	Sup	N
True label	Att	268	89	56
	Sup	127	220	66
	N	89	63	261

(b) SVM baseline & ESAA

		Classified as		
		Att	Sup	N
True label	Att	279	88	46
	Sup	124	226	63
	N	86	66	261

(c) SVM baseline & DF-QuAD

Table 6.10.: Summary of best performing combinations of corpora, classifiers and argumentation semantics, according to F_1 scores

Corpus	**Classifier**	**Semantics**	$\mathbf{F}_1$
IAC	RF	DF-QuAD	0.692
Two-class AIFdb	RF	DF-QuAD	0.831
Two-class News articles corpus	RF	DF-QuAD	0.717
Three-class News articles corpus	SVM	DF-QuAD	0.621

6.5.4. Evaluation Discussion

To evaluate the application of CleAr to RbAM we have conducted experiments on four of the corpora described in section 5.5. For each of the corpora we have compared the baseline established in chapter 5 to the application of CleAr using either ESAA or DF-QuAD semantics. Figure 6.4 and table 6.10 show an overview of the results, where we compare F_1 scores of each combination of corpora and argumentation semantics. Note that the application of ESAA produces the exact same results as the baseline, while the application of DF-QuAD yields improvements across the board. It appears that the difference in impact of the two different frameworks stems from the way arguments affect comparatively low and high values of the answers/arguments they attack/support. In ESAA the same argument will have a larger impact on a high-scoring argument than it will on a low-scoring argument. Say we have three arguments, α, β, γ, where

$$(\alpha, \beta) \in \mathit{Support}$$
$$(\alpha, \gamma) \in \mathit{Support}$$

so that α supports both β and γ and say their initial scores are $S^0_\alpha = 0.3$, $S^0_\beta = 0.6$ and $S^0_\gamma = 0.2$. In this case we would get $S_\beta = 0.78$ and $S_\gamma = 0.26$, a change of 0.18 and 0.06, respectively. Note that the change in score is higher for S_β, which had a relatively high score to begin with. This means that, unless a fairly large number of arguments were applicable to one of the alternative arguments, any increase in the score of the label suggested by the classifier will render it more or less *out of reach* of the other possible class label, whose S^0 is considerably lower. Analogously, for class labels not

chosen by the classifier to overtake the scores of the chosen class label, we need to identify a considerable amount of arguments. When we compare this result to those obtained as part of our sentiment analysis evaluation, we see the impact that the size of the argument base appears to have. For the polarity classification problem we have used a considerably larger argument base, and, on average, we thus identify more applicable arguments for each instance. This means that the application of ESAA ought to have a larger impact on changing class label scores, which indeed appears to be the case.

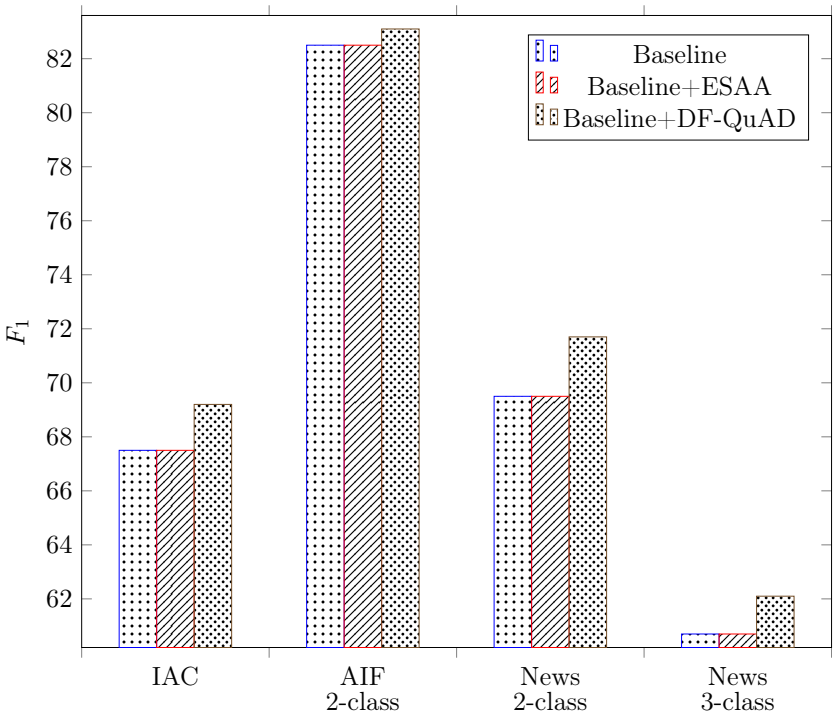


Figure 6.4.: Summary of evaluation results of applying CleAR to RbAM, where we compare the F_1 score of the baseline with the application of CleAr using ESAA or DF-QuAD for each corpus

With the application of CleAr with DF-QuAD we improve on all four baselines. We do not, however, achieve statistically significant differences ($p = 0.686$[1]). Note, however, that the statistical power of the significance tests is low, due to the small number of experiments we compare. Yeh [242] also points out that standard tests for statistical significance often under-

[1] p is calculated using a *paired-sample t-test*, see `www.mathworks.com/help/stats/ttest.html` for details

estimate the significance of differences in classification for natural language problems. We believe that more tests are needed to confirm or disprove the results provided here, but we see it as an encouraging sign that, using DF-QuAD semantics, we do obtain improvements on all four corpora.

6.6. Summary & Discussion

In this chapter we have presented our instantiation of CleAr for RbAM. We have built an argument base that is comprised of a combination of five different types of arguments, with a total of 171 arguments. We have formalised arguments based on both individual qualities of sentences, such as sentiment, and qualities two sentences exhibit in relation to each other, such as similarity measures. With this argument base we were able to improve upon the performance of the best classifiers, as chosen in chapter 5, for all four corpora tested in this chapter. Though these improvements were not statistically significant we do believe the outcome to be encouraging, in so far as having consistently made small improvements across very disparate corpora, on a linguistically challenging problem.

7. Future work & Conclusion

In this thesis we have described our work at the intersection of argumentation and supervised learning to help us in solving computational linguistics challenges. Specifically, in chapter 3 we have introduced *Classification enhanced with Argumentation (CleAr)*, a novel classification methodology that integrates knowledge in the form of arguments with supervised learning for computational linguistics. With CleAr we classify instances using supervised classifiers and subsequently argue about the validity of the class label proposed. For this we use quantitative argumentation semantics, specifically ESAA [65] and DF-QuAD [175]. Whenever we identify sufficient arguments, either arguing against the proposed class label or for another, we overturn the classifier's decision and follow what the arguments propose, instead. We have developed two instantiations of CleAr to address specific computational linguistics tasks:

1. *Cross-domain sentiment polarity classification*
2. *Relation-based Argumentation Mining (RbAM)*

RbAM is a novel approach to argumentation mining that capitalises on argumentative relations between sentences. While for sentiment polarity classification we have mostly used existing resources, i.e. corpora used to train classifiers, we have expended considerable effort towards developing resources for performing RbAM. This has included the construction of a tailor-made corpus of annotated sentence pairs. For both tasks we have conducted extensive evaluations, described in sections 4.3 and 5.6/6.5, respectively, showing that the application of CleAr improves classification throughout.

Throughout the remainder of chapter we describe challenges we have encountered throughout our work, but that remain to be addressed in detail, and we conclude the thesis with a brief reflection on our main contributions.

7.1. Future work

Many avenues at the intersection of argumentation and computational linguistics remain to be explored and to conclude the thesis we highlight some challenges that we believe to be particularly relevant. We distinguish between general challenges in section 7.1.1 and those specific to the two fields we have worked on, sentiment analysis in section 7.1.2 and argumentation mining in section 7.1.3. Additionally, in section 7.1.4 we discuss how sentiment analysis and argumentation mining challenges both overlap and diverge, and how we may capitalise on their commonalities.

7.1.1. General challenges

While certain challenges we face apply only to one of the tasks we have addressed in this thesis, others apply to any instantiation of CleAr. One we have only marginally addressed both for argumentation mining and sentiment polarity classification is the assignment of base scores to arguments. The choice of base scores is important because through them we determine an argument's impact on others, as well as its initial, non-dialectical strength. In table 4.1 we have shown example arguments, each of which is assigned a base score. We have noted that, at this juncture, we only create rough groups of arguments, e.g. by grouping arguments of the same type. Within each group all arguments are then assigned the same score, whose value we have determined empirically. By changing the scores for all arguments in a group we can change the overall impact this group has on reconsidering the class label suggested by the classifier. When we increase the argument scores we need fewer arguments to overturn a suggested class label; when we decrease the scores we need to find more applicable arguments, instead. It may, however, be useful to assign different scores to smaller groups of arguments, or indeed individual ones. This would allow us to tailor the impact an argument has to how reliably its conclusion holds. Recall the example argument from section 4.1.2:

$$\kappa_{hate(I)} : \ hate\ occurs\ in\ I \Rightarrow I \in Neg$$

If, for instance, we found that instances containing the word hate are almost exclusively negative, while those containing other keywords are not

as reliably classified, we may want to increase the score of this argument, and others that are similarly reliable, but leave the score of other arguments unchanged. To develop a more tailored way of assigning scores to arguments we will need to identify ways of identifying the desired impact of arguments that go beyond cumbersome empirical approaches.

Secondly, we believe it will be interesting to identify new application areas for CleAr. We have defined our methodology in chapter 3 in such a way that it may be applied to any supervised learning problem, using any combination of (supervised) classifiers and (quantitative) argumentation semantics. This means that we need not limit ourselves to applying CleAr to the tasks presented in this thesis. We believe that the ability to flexibly combine various sources of knowledge and to equip oneself with justificatory powers over classification outcomes may prove beneficial to many other classification problems, as well. Villena and colleagues [222], for example, combine rule bases and supervised classifiers to perform text categorization. To apply CleAr to this problem we would need to translate the rule base into an argument base, akin to the process we describe in section 3.1. Uzunu and colleagues [218], on the other hand, compare two approaches to assertion classification in patient medical records, one rule-based and the other using machine learning. In assertion classification one attempts to determine from text whether a certain (medical) problem is present or absent (in a patient), or whether it is uncertain. In situations such as this, where both rule-based and machine learning classifiers are used to address the same problem, CleAr may offer ways of consolidating the different approaches, where we translate the rules into arguments and subsequently chain the different approaches into a single solution. Additionally, with CleAr we can express and resolve conflicts between arguments, as they are naturally integrated through attack relations.

Apart from applying CleAr to novel problems we may also investigate ways of making CleAr more successful at solving the problems we have described in this thesis. We have seen in section 6.5 that, while DF-QuAD does help us to improve results, the way ESAA calculates the dialectical strength of arguments appeared to nullify any possible impact of arguments on the answers/arguments they attack or support. Other, qualitative se-

mantics, such as those described in section 2.1.3, e.g. [58, 139], may prove to incorporate arguments more successfully in the context of CleAr.

There are also a number of avenues which we may explore to identify new arguments to help us with classification. For example, the rhetorical structure of text [127], which we have used to help us with polarity classification in chapter 4, may prove conducive towards identifying argumentative structures, as well. RST not only categorises texts as nuclei or satellites, it also defines a set of relations that hold between text spans, such as those shown in figure **??**. Should we be able to determine that two sentences stand in a contrast relation, for example, this may point towards them being in an attacking relation rather than in a supporting one. Motivation and Enablement relations, on the other hand, may point towards the presence of a support relation. Peldszus and Stede [166] provide an outline of how we may interpret argumentation mining as the task of representing text through rhetorical structure. If we were able to distinguish between argumentative and non-argumentative rhetorical relations, we may be able to exploit available RST parsers, such as [71, 90], to label sentences as attacking or supporting others. Indeed Azar [10] proposes such a categorisation, where he considers five of the defined RST relations to be argumentative, and the grammar developed by Mochales and colleagues [136] to represent argumentative structures is at least inspired by RST, as well.

At the same time the use of nucleus and satellite labelings does not seem to translate as neatly to our understanding of argumentation mining as rhetorical relations might. The distinction of nuclei and satellites occurs on sentence level and it may thus be more suited to distinguishing between premises and conclusions. Consider again our example from section 2.3:

> [$_s$Though he did a *good* job at his latest public appearance] [$_n$ his overall popularity ratings are *abysmal*]

Here the nucleus forms the conclusion, while the satellite contains the premise. We currently do not consider the internal structure of arguments and this aspect of RST is not applicable, accordingly. However, as we mention in section 5.1, it may be interesting to join our approach with others, where the internal structure of arguments is considered. In this case, identifying satellites and nuclei in sentences may prove a potent mechanism towards solving this problem.

7.1.2. Sentiment Analysis challenges

The most pressing shortcoming of our work on sentiment analysis is arguably the fact that we have limited ourselves to polarity classification. As we have focused on investigating the value of applying CleAr to sentiment analysis, this limitation was prudent, but fully fledged sentiment analysis solutions must address the initial detection of sentiment, as well. We believe that our approach has applicability to sentiment detection in the same manner in which we have applied it to polarity classification. To develop a sentiment detection solution that uses CleAr, we will need to define either a new argument base or an altered and extended version of the one we have worked with thus far. It may be feasible to jointly consider the arguments for negativity and positivity as arguments for the presence of sentiment. If we were able to translate the arguments in this manner we would need to extend our existing argument base with arguments that reason either for an instance being *non-sentimental* or against being *sentimental*.

In our experiments in section 4.3 we have distinguished between *near-domain* and *out-of-domain* classification. It seems as though the degree to which domains differ has a substantial impact on the ability to transfer classification performance across domains. The differences between corpora may stem from their being concerned with dissimilar topics, e.g. movie reviews vs. politics, as well as their having different writing styles, e.g. Tweets vs. sentences from movie reviews. We have noted in section 4.3 that the performance of classification across domains drops markably when both the topics and writing styles differ across corpora, as opposed to corpus combinations where only one of the two differs. We believe that systematically identifying nearness of domains and content sources may help us identify corpora more amenable to performing cross-domain classification. This may then help us to identify existing resources and corpora to be used for classification problems more successfully.

7.1.3. Argumentation Mining challenges

Argumentation research has produced numerous formalisms to represent arguments and how they relate, as described in section 2.1. These, however, have not necessarily been conceived with natural language in mind, but

have often instead been based on more formal interpretations of arguments, e.g. as in [17, 43, 138]. We thus need to consider whether and how certain representations of arguments are amenable to representing arguments encountered in natural language. For our research we have mostly deferred this problem by interpreting language akin to the understanding of arguments proposed in abstract argumentation [57]. By considering relations, primarily, we do not need to translate arguments themselves into any particular logical form. If, however, we were to deal with the internal structure of arguments, linking representations conceived for argumentation to natural language may prove more difficult. Take, for example, Lippi and Torroni [118], who aim to identify claims in text, thus partially addressing the identification of fully fledged arguments. They find claims by identifying partial parse trees more commonly found in argumentative claims than in other types of text. It is not clear how we may consolidate more fine-grained representation of natural language arguments such as this with logical representations in argumentation and we will need to investigate novel ways of doing so. In [3], Amgoud and colleagues extrapolate a formal representation of arguments, as described in section 2.1.3, and their internal structure as part of a framework to represent arguments that have been mined from natural language text. This constitutes a first attempt at intertwining argumentation mining with formal representations of arguments. This perhaps indicates a path through which we may bridge the apparent gap between arguments as they are used in natural language and arguments as they are used in formal representations of argumentation.

In section 5.3 we have described three ways of building sentence pairs. First, we may use a brute force approach and generate every possible sentence pair from the text we are trying to extract arguments from. This, however, is generally only feasible when we analyse small text passages. Secondly we may generate pairs by relating a text to a single sentence, building pairs in which the separate sentence always functions as the Parent and the sentences from the text are considered Children of this sentence. This approach is of use only when we are interested in how a text relates to a single external argument. Thirdly we may create sentence pairs directly from text, but, unlike in the first approach, we only consider those sentences to be candidates that contain the same nouns. We believe that the third op-

tion has the most general applicability, as we can ensure its computational feasibility, while also being able to analyse a text in its own right. Our current approach, however, is defined based on the simple heuristics that sentences containing the same nouns may be more prone to being related and that argumentative directionality is based on the sentences' order. We need to reconsider our choices and possibly redesign the selection process used to construct sentence pairs. One option to render the generation of candidate sentence pairs more sophisticated may be to consider not only the presence of common nouns/entities, but to also take into account the proximity of sentences. For this we may assign a loss factor to potential sentence pairs, where, the further removed sentences are from each other, the larger the factor and the less the likelihood of them being related. In addition to identifying more sophisticated ways of building sentence pairs we may also need to re-examine the choice of working at sentence-level granularity. An alternative to considering sentence may, for example, be the use of *Argumentative Discourse Units (ADU)* [166], where not only sentence boundaries are considered boundaries, but splits are also made at sub-sentence level. This tends to split text into its phrases, but an ADU may be also be comprised of multiple phrases, should multiple phrases have a single argumentative purpose.

Having illustrated the usefulness of RbAM theoretically we now need to identify concrete use cases to show how it may be used in practise. We have defined one such use case together with the BBC News labs, where the aim is to integrate RbAM functionalities with the Juicer, described in section 5.5.3. Concretely, the Juicer currently offers a range of search capabilities for a large database of news publications, based on both search filters and annotations of entities. The aim is to add a search option, based on strategy (1) described in section 5.3, where we compare sentences from one or more texts to a single external argument. This option would be provided in the form of a free text field, added to a search window, where a user could provide the external argument and compare it to a (possibly filtered) collection of news articles. This way the Juicer would provide a simple way for a user to search a large amount of e.g. news articles for arguments supporting or attacking the argument provided. One way of integrating this functionality with the Juicer may be the development of an API, which, if implemented in

a sufficiently generic manner, may enable the integration of argumentation mining capabilities in other applications, as well.

7.1.4. Argumentation Mining and Sentiment Analysis

Argumentation mining and sentiment analysis share common problems, yet they also differ decidedly in complexity and scope. In sentiment analysis we are faced with a classification problem in which we need to assign a word, a sentence, or any larger piece of text to one of at least three classes, *neutral*, *positively opinionated* or *negatively opinionated*. These classes may be further subdivided to achieve a more fine grained classification. To successfully tackle argumentation mining not only do we need to perform a similar task, i.e. distinguishing arguments from non-arguments, but we also need to identify relations between arguments once we have found them. As such, argumentation mining is somewhat more challenging than sentiment analysis, since the range of problems it needs to address is broader.

Another difference we observe is the larger flexibility when it comes to analysing sentiment at varying granularities. We may try and identify the sentiment of a word, but we can do the same for phrases, sentences, paragraphs, etc. Argumentation mining, on the other hand, generally needs to consider individual arguments, and there is no clear-cut answer to the question of whether arguments should be searched for on sub-sentence level, sentence level or on combinations of sentences. Though most current work in argumentation mining analyses sentences, e.g. [142], we are unaware of anybody proposing a well-founded justification for doing so. The reason we have opted to work on sentence level, as well, is simply that sentence boundaries provide convenient delimitations to textual units. To identify arguments proper it may well be prudent to develop schemes that allow flexibility in the level of granularity at which one searches for arguments.

We believe one of the more relevant commonalities between the two fields to be the fact that in both cases we aim to identify subjective language of some sort. This may offer ways of cross-fertilisation between the two problems, one of which we have already used in RbAM, described in section 6.2. Here we use sentiment scores to help us identify attack and support relations between sentences. It seems intuitive that in tackling argumentation mining and sentiment analysis we may search for similar language, such as subjective words and phrases, structure, etc.

7.2. Conclusion

We believe that our developments illustrate a novel and promising application area for argumentation, namely the improvement of corpus-based classifiers for computational linguistics problems. We have shown that by applying the CleAr methodology we can improve performance both in (cross-domain) sentiment polarity classification and in (relation-based) argumentation mining. Additionally we have highlighted the importance of considering text in relation to what surrounds it to identify arguments and argumentative structures in text. By applying argumentation to computational linguistics problems and by using computational linguistics to identify argumentative text we have highlighted the reciprocal link between the two fields, illustrating how we stand to benefit from considering insights from both fields in unison.

Finally our approach to classification illustrates a modular way that may provide quick access to gains in classification performance, for example in an environment where the development of complex machine learning solutions needs to give way to pragmatic and easily implemented solutions. As Chiticariu and colleagues [48] succinctly point out in the paper titled *Rule-based Information Extraction is Dead! Long Live Rule-based Information Extraction Systems!*, there seems to be a divide between academia and industry when it comes to placing value in rule-based systems. Approaches such as ours may offer an avenue towards consolidating this divide and provide value for researchers and industry professionals alike.

Bibliography

[1] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, pages 30–38. ACL, 2011.

[2] E. Aharoni, A. Polnarov, T. Lavee, D. Hershcovich, R. Levy, R. Rinott, D. Gutfreund, and N. Slonim. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68, 2014.

[3] L. Amgoud, P. Besnard, and A. Hunter. Representing and reasoning about arguments mined from texts and dialogues. In *Proceedings of European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (LNCS) volume 9161*, pages 60–71. Springer, 2015.

[4] L. Amgoud, C. Cayrol, and M.-C. Lagasquie-Schiex. On the bipolarity in argumentation frameworks. In *NMR*, pages 1–9. Wiley Online Library, 2004.

[5] L. Amgoud, C. Cayrol, M.-C. Lagasquie-Schiex, and P. Livet. On bipolarity in argumentation frameworks. *International Journal of Intelligent Systems*, 23(10):1062–1093, 2008.

[6] L. Amgoud and M. Serrurier. Agents that argue and explain classifications. *Autonomous Agents and Multi-Agent Systems*, 16(2):187–209, 2008.

[7] L. Amgoud and M. Serrurier. Arguing and explaining classifications. In *Argumentation in Multi-Agent Systems*, pages 164–177. Springer, 2008.

[8] A. Andreevskaia and S. Bergler. When specialists and generalists work together: Overcoming domain dependence in sentiment tagging. In *Proceedings of The 46th Annual Meeting of the ACL (ACL)*, pages 290–298. ACL, 2008.

[9] A. Aue and M. Gamon. Customizing sentiment classifiers to new domains: A case study. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, volume 1, pages 2–9. John Benjamins Publishing Company, 2005.

[10] M. Azar. Argumentative text as rhetorical structure: An application of rhetorical structure theory. *Argumentation*, 13(1):97–114, 1999.

[11] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 3, pages 805–810. AAAI Press, 2003.

[12] M. Baroni and S. Vegnaduzzo. Identifying subjective adjectives through web-based mutual information. In *Proceedings of Konferenz zur Verarbeitung natuerlicher Sprache (KONVENS)*, volume 4, pages 17–24. Citeseer, 2004.

[13] P. Baroni, M. Romano, F. Toni, M. Aurisicchio, and G. Bertanza. Automatic evaluation of design alternatives with quantitative argumentation. *Argument & Computation*, 6(1):24–49, 2015.

[14] R. Bellman. A markovian decision process. Technical report, DTIC Document, 1957.

[15] F. Benamara, B. Chardon, Y. Y. Mathieu, V. Popescu, et al. Towards context-based subjectivity analysis. In *Proceedings of International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1180–1188. ACL, 2011.

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

[17] T. J. M. Bench-Capon. Value-based argumentation frameworks. In *9th International Workshop on Non-Monotonic Reasoning (NMR 2002), April 19-21, Toulouse, France, Proceedings*, pages 443–454, 2002.

[18] P. Besnard, A. Garcia, A. Hunter, S. Modgil, H. Prakken, G. Simari, and F. Toni. Introduction to structured argumentation. *Argument & Computation*, 5(1):1–4, 2014.

[19] F. Bex, S. Modgil, H. Prakken, and C. Reed. On logical specifications of the argument interchange format. *J. Log. Comput.*, 23(5):951–989, 2013.

[20] C. Bishop. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[21] D. Bollegala, D. Weir, and J. Carroll. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies (ACL-HLT)-Volume 1*, pages 132–141. ACL, 2011.

[22] D. Bollegala, D. Weir, and J. Carroll. Cross-domain sentiment classification using a sentiment sensitive thesaurus. *IEEE Transactions on Knowledge and Data Engineering*, 25(8):1719–1731, 2013.

[23] F. Boltuzic and J. Šnajder. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58. ACL, 2014.

[24] F. Boltuzic and J. Šnajder. Identifying prominent arguments in online debates using semantic textual similarity. *Proceedings of the Second Workshop on Argumentation Mining*, pages 110–115, 2015.

[25] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial intelligence*, 93(1):63–101, 1997.

[26] F. Bravo-Marquez, M. Mendoza, and B. Poblete. Combining strengths, emotions and polarities for boosting twitter sentiment analysis. In *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining, WISDOM*, pages 2:1–2:9, 2013.

[27] E. Breck, Y. Choi, and C. Cardie. Identifying expressions of opinion in context. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2683–2688, 2007.

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

[29] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*, pages 112–116. ACL, 1992.

[30] S. Bromuri, V. Urovi, M. Morge, K. Stathis, and F. Toni. A multi-agent system for service discovery, selection and negotiation. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)-Volume 2*, pages 1395–1396. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[31] M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. In *Proceedings of the National Academy of Sciences-volume 97*, page 262. National Acad Sciences, 2000.

[32] E. Cabrio and S. Villata. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the ACL: Short Papers-Volume 2*, pages 208–212. ACL, 2012.

[33] E. Cambria, D. Olsher, and D. Rajagopal. Senticnet 3: A common and common-sense knowledge base for cognition-driven sentiment analysis. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1515–1521, 2014.

[34] E. Cambria, B. Schuller, Y. Xia, and C. Havasi. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, pages 15–21, 2013.

[35] E. Cambria and B. White. Jumping nlp curves: a review of natural language processing research [review article]. *IEEE Computational Intelligence Magazine*, 9(2):48–57, 2014.

[36] L. Carlson, D. Marcu, and M. E. Okurowski. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue - Volume 16*, pages 1–10. ACL, 2001.

[37] G. A. Carpenter, S. Grossberg, and D. B. Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural networks*, 4(6):759–771, 1991.

[38] L. Carstens, X. Fan, Y. Gao, and F. Toni. An overview of argumentation frameworks for decision support. In *Graph Structures for Knowledge Representation and Reasoning - 4th International Workshop, GKR*, pages 32–49, 2015.

[39] L. Carstens and F. Toni. Enhancing sentiment extraction from text by means of arguments. In *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining, WISDOM*, pages 4:1–4:9. ACM, 2013.

[40] L. Carstens and F. Toni. Improving out-of-domain sentiment polarity classification using argumentation. In *Proceedings of the 5th ICDM Workshop on Sentiment Elicitation from Natural Text for Information Retrieval and Extraction (SENTIRE)*, pages 1294–1301. IEEE, 2015.

[41] L. Carstens and F. Toni. Towards relation based argumentation mining. In *Proceedings of the Second Workshop on Argumentation Mining*, pages 29–34. ACL, 2015.

[42] L. Carstens, F. Toni, and V. Evripidou. Argument mining and social debates. *Proceedings of Computational Models of Argument (COMMA)*, 2:3, 2014.

[43] C. Cayrol and M. Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *Proceedings of the European Conferences on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 378–389, 2005.

[44] C. Cayrol and M.-C. Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *Symbolic and quantita-*

tive approaches to reasoning with uncertainty, pages 378–389. Springer, 2005.

[45] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[46] G. Cheng and Y. Qu. Term dependence on the semantic web. In *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, pages 665–680, 2008.

[47] C. Chesñevar, J. McGinnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, and S. Willmott. Towards an argument interchange format. *Knowledge Engineering Review*, 21(4):293–316, 2006.

[48] L. Chiticariu, Y. Li, and F. R. Reiss. Rule-based information extraction is dead! long live rule-based information extraction systems! In *EMNLP*, pages 827–832. ACL, 2013.

[49] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine learning*, 3(4):261–283, 1989.

[50] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[51] D. Corapi, A. Russo, and E. Lupu. Inductive logic programming in answer set programming. In *Inductive Logic Programming*, pages 91–97. Springer, 2012.

[52] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[53] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. *GATE: an Architecture for Development of Robust HLT Applications*. University of Sheffield, 2002.

[54] I. Dagan, O. Glickman, and B. Magnini. The pascal recognising textual entailment challenge. *Machine Learning Challenges. Evaluating*

Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment, pages 177–190, 2006.

[55] K. Dave, S. Lawrence, and D. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international Conference on World Wide Web (WWW)*, pages 519–528. ACM, 2003.

[56] H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[57] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and< i> n</i>-person games. *Artificial intelligence*, 77(2):321–357, 1995.

[58] P. M. Dung, R. A. Kowalski, and F. Toni. Assumption-based argumentation. In *Argumentation in Artificial Intelligence*, pages 199–218. Springer, 2009.

[59] P. M. Dung and P. M. Thang. Towards (probabilistic) argumentation for jury-based dispute resolution. *Proceedings of Computational Models of Argument (COMMA)*, 216:171–182, 2010.

[60] K. T. Durant and M. D. Smith. Predicting the political sentiment of web log posts using supervised machine learning techniques coupled with feature selection. In *Advances in Web Mining and Web Usage Analysis*, pages 187–206. Springer, 2007.

[61] D. A. Duverle and H. Prendinger. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 665–673. ACL, 2009.

[62] G. Escudero, L. Marquez, and G. Rigau. An empirical study of the domain dependence of supervised word sense disambiguation systems. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and very large corpora-Volume 13*, pages 172–180. ACL, 2000.

[63] A. Esuli and F. Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of The International Conference on Language Resources and Evaluation (LREC)*, volume 6, pages 417–422. Citeseer, 2006.

[64] V. Evripidou, L. Carstens, F. Toni, and D. Cabanillas. Argumentation-based collaborative decisions for design. In *26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 805–809. IEEE, 2014.

[65] V. Evripidou and F. Toni. Quaestio-it. com: a social intelligent debating platform. *Journal of Decision Systems*, 23(3):333–349, 2014.

[66] S. I. Faiz and R. E. Mercer. Extracting higher order relations from biomedical text. *ACL 2014*, page 100, 2014.

[67] W. Fan, E. Greengrass, J. McCloskey, P. S. Yu, and K. Drummey. Effective estimation of posterior probabilities: Explaining the accuracy of randomized decision tree approaches. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM*, pages 154–161. IEEE, 2005.

[68] X. Fan, R. Craven, R. Singer, F. Toni, and M. Williams. Assumption-based argumentation for decision-making with preferences: A medical case study. In *Computational Logic in Multi-Agent Systems*, pages 374–390. Springer, 2013.

[69] X. Fan and F. Toni. Agent strategies for aba-based information-seeking and inquiry dialogues. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, pages 324–329. IOS Press, 2012.

[70] V. W. Feng and G. Hirst. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the ACL: Long Papers-Volume 1*, pages 60–68. ACL, 2012.

[71] V. W. Feng and G. Hirst. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*, pages 511–521. ACL, 2014.

[72] J. Fox, D. Glasspool, D. Grecu, S. Modgil, M. South, and V. Patkar. Argumentation-based inference and decision making–a medical perspective. *IEEE Intelligent Systems*, pages 34–41, 2007.

[73] T. Furey, N. Cristianini, N. Duffy, D. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906, 2000.

[74] Y. Gao and F. Toni. Argumentation accelerated reinforcement learning for cooperative multi-agent systems. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, volume 263, page 333. IOS Press, 2014.

[75] S. R. Garner et al. Weka: The waikato environment for knowledge analysis. In *Proceedings of the New Zealand computer science research students Conference*, pages 57–64. Citeseer, 1995.

[76] D. Ghosh, S. Muresan, N. Wacholder, M. Aakhus, and M. Mitsui. Analyzing argumentative discourse units in online interactions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 39–48. ACL, 2014.

[77] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12, 2009.

[78] S. A. Gómez and C. I. Chesnevar. A hybrid approach to pattern classification using neural networks and defeasible argumentation. In *FLAIRS Conference*, pages 393–398. AAAI, 2004.

[79] M. Gönen, A. G. Tanugur, and E. Alpaydin. Multiclass posterior probability support vector machines. *Neural Networks, IEEE Transactions on*, 19(1):130–139, 2008.

[80] N. Gorogiannis and A. Hunter. Instantiating abstract argumentation with classical logic arguments: Postulates and properties. *Artificial Intelligence*, 175(9):1479–1497, 2011.

[81] T. Goudas, C. Louizos, G. Petasis, and V. Karkaletsis. Argument extraction from news, blogs, and social media. In *Artificial Intelligence: Methods and Applications*, pages 287–299. Springer, 2014.

[82] N. L. Green. Towards creation of a corpus for argumentation mining the biomedical genetics research literature. *Proceedings of The 52nd Annual Meeting of the ACL (ACL)*, page 11, 2014.

[83] N. L. Green. Identifying argumentation schemes in genetics research articles. *NAACL HLT 2015*, page 12, 2015.

[84] B. B. Greene and G. M. Rubin. *Automatic grammatical tagging of English*. Department of Linguistics, Brown University, 1971.

[85] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pages 1–4. Citeseer, 2006.

[86] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422, 2002.

[87] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[88] V. Hatzivassiloglou and J. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th Conference on Computational linguistics-Volume 1*, pages 299–305. ACL, 2000.

[89] Y. He, C. Lin, and H. Alani. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies (ACL-HLT)-Volume 1*, pages 123–131. ACL, 2011.

[90] M. Heilman and K. Sagae. Fast rhetorical structure theory discourse parsing. *CoRR*, abs/1505.02425, 2015.

[91] K. M. Hermann, T. Kociský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015.

[92] G. Hirst and D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database*, 305:305–332, 1998.

[93] A. Hogenboom, F. Frasincar, F. de Jong, and U. Kaymak. Using rhetorical structure in sentiment analysis. *Communications of the ACM*, 58(7):69–77, 2015.

[94] H. Houngbo and R. E. Mercer. An automated method to build a corpus of rhetorically-classified sentences in biomedical texts. In *Proceedings of the First Workshop on Argumentation Mining*, pages 19–23. Citeseer, 2014.

[95] C. Houy, T. Niesen, P. Fettke, and P. Loos. Towards automated identification and analysis of argumentation structures in the decision corpus of the german federal constitutional court. In *7th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, pages 72–77. IEEE, 2013.

[96] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international Conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[97] X. Hu, J. Tang, H. Gao, and H. Liu. Unsupervised sentiment analysis with emotional signals. In *Proceedings of the 22nd international Conference on World Wide Web (WWW)*, pages 607–618. International World Wide Web Conferences Steering Committee, 2013.

[98] A. Hunter. Some foundations for probabilistic abstract argumentation. *Proceedings of Computational Models of Argument (COMMA)*, 245:117–128, 2012.

[99] A. Hunter. A probabilistic approach to modelling uncertain logical arguments. *International Journal of Approximate Reasoning*, 54(1):47–81, 2013.

[100] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, cmp-lg/9709008, 1997.

[101] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies (ACL-HLT)-Volume 1*, pages 151–160. ACL, 2011.

[102] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in artificial intelligence (UAI)*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[103] J. Kiesel, K. Al-Khatib, M. Hagen, and B. Stein. A shared task on argumentation mining in newspaper editorials. *NAACL HLT 2015*, page 35, 2015.

[104] S. Kim and E. Hovy. Automatic detection of opinion bearing words and sentences. In *Companion Volume to the Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*. ACL, 2005.

[105] C. Kirschner, J. Eckle-Kohler, and I. Gurevych. Linking the thoughts: Analysis of argumentation structures in scientific publications. *Proceedings of the Second Workshop on Argumentation Mining*, pages 1–11, 2015.

[106] E. Kouloumpis, T. Wilson, and J. Moore. Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the AAAI Conference on Web and Social Media (ICWSM)*, volume 11, pages 538–541. AAAI, 2011.

[107] R. A. Kowalski and F. Toni. Abstract argumentation. *Artificial intelligence and law*, 4(3-4):275–296, 1996.

[108] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[109] J. Lawrence, F. Bex, C. Reed, and M. Snaith. Aifdb: Infrastructure for the argument web. In *Proceedings of Computational Models of Argument (COMMA)*, pages 515–516. IOS Press, 2012.

[110] J. Lawrence and C. Reed. Aifdb corpora. *Proceedings of Computational Models of Argument (COMMA)*, 266:465, 2014.

[111] J. Lawrence and C. Reed. Combining argument mining techniques. *NAACL HLT 2015*, page 127, 2015.

[112] C. Leacock and M. Chodorow. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.

[113] H. Li, N. Oren, and T. J. Norman. Probabilistic argumentation frameworks. In *Theory and Applications of Formal Argumentation*, pages 1–16. Springer, 2012.

[114] T. Li, V. Sindhwani, C. Ding, and Y. Zhang. Knowledge transformation for cross-domain sentiment classification. In *Proceedings of the 32nd international ACM SIGIR Conference on Research and development in information retrieval*, pages 716–717. ACM, 2009.

[115] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 98, pages 296–304. ACM, 1998.

[116] Y. Lin, J. Zhang, X. Wang, and A. Zhou. An information theoretic approach to sentiment polarity classification. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*, pages 35–40. ACM, 2012.

[117] M. Lippi and P. Torroni. Argument mining: A machine learning perspective. In *Theory and Applications of Formal Argumentation*, pages 163–176. Springer, 2015.

[118] M. Lippi and P. Torroni. Context-independent claim detection for argument mining. In *Proceedings of the 24th International Conference on Artificial Intelligence (ICAI)*, pages 185–191. AAAI Press, 2015.

[119] M. Lippi and P. Torroni. Argumentation mining: State of the art and emerging trends. *ACM Trans. Internet Technol.*, 16(2), Mar. 2016.

[120] B. Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

[121] H. Liu, T. Christiansen, W. A. Baumgartner Jr, and K. Verspoor. Biolemmatizer: a lemmatization tool for morphological processing of biomedical text. *J. Biomedical Semantics*, 3(3):17, 2012.

[122] K. Liu, W. Li, and M. Guo. Emoticon smoothed language models for twitter sentiment analysis. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI, 2012.

[123] J. B. Lovins. *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory, 1968.

[124] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies (ACL-HLT)-Volume 1*, pages 142–150. ACL, 2011.

[125] B. Magnini, R. Zanoli, I. Dagan, K. Eichler, G. Neumann, T.-G. Noh, S. Pado, A. Stern, and O. Levy. The excitement open platform for textual inferences. In *ACL (System Demonstrations)*, pages 43–48, 2014.

[126] W. Mann, C. Matthiessen, and S. Thompson. Rhetorical structure theory and text analysis. *Discourse Description: Diverse linguistic analyses of a fund-raising text*, 16:39, 1992.

[127] W. Mann and S. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text & Talk*, 8(3):243–281, 1988.

[128] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of The 52nd Annual Meeting of the ACL: System Demonstrations*, pages 55–60. ACL, 2014.

[129] C. D. Manning. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189. Springer, 2011.

[130] D. McClosky, E. Charniak, and M. Johnson. Automatic domain adaptation for parsing. In *Proceedings of the Conference on Human Lan-*

guage Technologies: Conference of the North American Chapter of the ACL (HLT-NAACL), pages 28–36. ACL, 2010.

[131] N. Meinshausen. Hierarchical testing of variable importance. *Biometrika*, 95(2):265–278, 2008.

[132] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[133] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119. Citeseer, 2013.

[134] G. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[135] P. Mitchell, B. Santorini, M. Marcinkiewicz, and A. Taylor. Treebank-3. *Linguistic Data Consortium, University of Pennsylvania*, 1999.

[136] R. Mochales and M. Moens. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22, 2011.

[137] R. Mochales and M.-F. Moens. Study on the structure of argumentation in case law. *Proceedings of the 2008 Conference on Legal Knowledge and Information Systems (JURIX)*, pages 11–20, 2008.

[138] S. Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173(9):901–934, 2009.

[139] S. Modgil and H. Prakken. A general account of argumentation with preferences. *Artificial Intelligence*, 195:361–397, 2013.

[140] S. Modgil and H. Prakken. The aspic+ framework for structured argumentation: a tutorial. *Argument & Computation*, 5(1):31–62, 2014.

[141] S. Modgil, F. Toni, F. Bex, I. Bratko, C. I. Chesñevar, W. Dvořák, M. A. Falappa, X. Fan, S. A. Gaggl, A. J. García, et al. The added value of argumentation. In *Agreement Technologies*, pages 357–403. Springer, 2013.

[142] M. Moens, E. Boiy, R. Palau, and C. Reed. Automatic detection of arguments in legal texts. In *Proceedings of the 11th international Conference on Artificial intelligence and law (ICAIL)*, pages 225–230. ACM, 2007.

[143] A. Montejo-Ráez, E. Martínez-Cámara, M. T. Martín-Valdivia, and L. A. Ureña-López. Ranked wordnet graph for sentiment polarity classification in twitter. *Computer Speech & Language*, 28(1):93–107, 2014.

[144] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252. Citeseer, 2005.

[145] M. Mozina, P. Tolchinsky, and U. Cortes. Project N002307 ASPIC, 2004.

[146] M. Mozina, J. Zabkar, T. Bench-Capon, and I. Bratko. Argument based machine learning applied to law. *Artificial Intelligence and Law*, 13(1):53–73, 2005.

[147] M. Mozina, J. Zabkar, and I. Bratko. Argument based machine learning. *Artificial Intelligence*, 171(10-15):922–937, 2007.

[148] S. Muggleton. Inductive logic programming. *New generation computing*, 8(4):295–318, 1991.

[149] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

[150] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551, 2011.

[151] G. Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

[152] H. V. Nguyen and D. J. Litman. Extracting argument and domain words for identifying argument components in texts. *NAACL HLT 2015*, page 22, 2015.

[153] N. Ong, D. Litman, and A. Brusilovsky. Ontology-based argument mining and automatic essay scoring. *Proceedings of The 52nd Annual Meeting of the ACL*, page 24, 2014.

[154] S. Ontanon, P. Dellunde, L. Godo, and E. Plaza. A defeasible reasoning model of inductive concept learning from examples and communication. *Artificial intelligence*, 193:129–148, 2012.

[155] S. Oraby, L. Reed, R. Compton, E. Riloff, M. Walker, and S. Whittaker. And thatÕs a fact: Distinguishing factual and emotional argumentation in online dialogue. *NAACL HLT 2015*, page 116, 2015.

[156] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas. How many trees in a random forest? In *Proceedings of the International Conference on Machine Learning and Data Mining (MLDM)*, pages 154–168. Springer, 2012.

[157] R. Palau and M. Moens. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international Conference on Artificial Intelligence and Law (ICAIL)*, pages 98–107. ACM, 2009.

[158] G. Paltoglou and M. Thelwall. Twitter, myspace, digg: Unsupervised sentiment analysis in social media. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4):66, 2012.

[159] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international Conference on World Wide Web (WWW)*, pages 751–760. ACM, 2010.

[160] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on ACL (ACL)*, page 271. ACL, 2004.

[161] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on ACL (ACL)*, pages 115–124. ACL, 2005.

[162] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)-Volume 10*, pages 79–86. ACL, 2002.

[163] J. Park and C. Cardie. Identifying appropriate support for propositions in online user comments. *Proceedings of The 52nd Annual Meeting of the ACL*, page 29, 2014.

[164] S. Patwardhan. *Incorporating dictionary and corpus information into a context vector measure of semantic relatedness.* PhD thesis, University of Minnesota, Duluth, 2003.

[165] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at Human Language Technologies: Conference of the North American Chapter of the ACL (HLT-NAACL)*, pages 38–41. ACL, 2004.

[166] A. Peldszus and M. Stede. From argument diagrams to argumentation mining in texts: A survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31, 2013.

[167] C. Perelman. The new rhetoric. In *Pragmatics of natural languages*, pages 145–149. Springer, 1971.

[168] J. C. Platt. Probabilities for sv machines. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, March 1999.

[169] J. Plisson, N. Lavrac, D. Mladenic, et al. A rule based approach to word lemmatization. *Proceedings of IS-2004*, pages 83–86, 2004.

[170] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[171] H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1(2):93–124, 2010.

[172] R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. K. Joshi, and B. L. Webber. The penn discourse treebank 2.0. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC*. Citeseer, 2008.

[173] R. Prasad, E. Miltsakaki, N. Dinesh, A. Lee, A. Joshi, L. Robaldo, and B. L. Webber. *The penn discourse treebank 2.0 annotation manual*, 2007.

[174] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[175] A. Rago, F. Toni, M. Aurisicchio, and P. Baroni. Discontinuity-free decision support with quantitative argumentation debates. In *Proceedings of the 15th International Conference on Principles and Knowledge Representation and Reasoning (KR)*, pages 63–73. AAAI, 2016.

[176] I. Rahwan and C. Reed. The argument interchange format. In *Argumentation in artificial intelligence*, pages 383–402. Springer, 2009.

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

[178] C. Reed, S. Wells, K. Budzynska, and J. Devereux. Building arguments with argumentation: the role of illocutionary force in computational models of argument. In *Proceedings of Computational Models of Argument (COMMA)*, pages 415–426. IOS Press, 2010.

[179] C. Reed, S. Wells, J. Devereux, and G. Rowe. Aifˆ+: Dialogue in the argument interchange format. *Frontiers in Artificial Intelligence and Applications*, 172:311, 2008.

[180] R. Rehurek and P. Sojka. *Software framework for topic modelling with large corpora*. University of Malta, 2010.

[181] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI*, pages 448–453. Elsevier, 1995.

[182] E. Riloff, J. Wiebe, and T. Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh Conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 25–32. ACL, 2003.

[183] R. Rinott, L. Dankin, C. Alzate, M. M. Khapra, E. Aharoni, and N. Slonim. Show me your evidence–an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in NLP (EMNLP), Lisbon, Portugal*, pages 17–21. ACL, 2015.

[184] G. Rios and H. Zha. Exploring support vector machines and random forests for spam detection. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, pages 284–292, 2004.

[185] N. Rooney, H. Wang, and F. Browne. Applying kernel methods to argumentation mining. In *Proceedings of the 25th Florida Artificial Intelligence Research Society Conference (FLAIRS)*. AAAI, 2012.

[186] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[187] T. Ruotsalo, L. Aroyo, and G. Schreiber. Knowledge-based linguistic annotation of digital cultural heritage collections. *IEEE Intelligent Systems*, pages 64–75, 2009.

[188] H. Saif, M. Fernandez, Y. He, and H. Alani. Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold. In *1st Interantional Workshop on Emotion and Sentiment in Social and Expressive Media: Approaches and Perspectives from AI (ESSEM 2013)*, pages 9–21. AI*IA, 2013.

[189] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, Inc., 1986.

[190] N. J. Sanders. *Sanders-Twitter Sentiment Corpus*, 2011.

[191] C. Sardianos, I. M. Katakis, G. Petasis, and V. Karkaletsis. Argument extraction from news. *NAACL HLT 2015*, page 56, 2015.

[192] H. Schmid. Probabilistic part-of-speech tagging using decision trees, 1994.

[193] J. Schneider and A. Wyner. Identifying consumers' arguments in text. In *Proceedings of the Workshop on Semantic Web and Information Extraction (SWAIE)*, pages 31–42, 2012.

[194] S. Scott and S. Matwin. Feature engineering for text classification. In *ICML*, volume 99, pages 379–388. Citeseer, 1999.

[195] S. Sekine. The domain dependence of parsing. In *Proceedings of the fifth Conference on Applied Natural Language Processing (ANLP)*, pages 96–102. ACL, 1997.

[196] I. Shafran, M. Riley, and M. Mohri. Voice signatures. In *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*, pages 31–36. IEEE, 2003.

[197] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu. Open mind common sense: Knowledge acquisition from the general public. In *On the move to meaningful internet systems 2002: Coopis, doa, and odbase*, pages 1223–1237. Springer, 2002.

[198] P. Sobhani, D. Inkpen, and S. Matwin. From argumentation mining to stance classification. *NAACL HLT 2015*, page 67, 2015.

[199] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. ACL, 2013.

[200] A. Søgaard. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies: short papers (ACL-HLT)-Volume 2*, pages 48–52. ACL, 2011.

[201] Y. Song, M. Heilman, B. Beigman, and K. P. Deane. Applying argumentation schemes for essay scoring. *ACL 2014*, page 69, 2014.

[202] R. Soricut and D. Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the ACL on Human Language Technology-Volume 1*, pages 149–156. ACL, 2003.

[203] C. Stab and I. Gurevych. Annotating argument components and relations in persuasive essays. In *Proceedings of the 25th International*

Conference on Computational Linguistics (COLING), pages 1501–1510. Citeseer, 2014.

[204] C. Stab and I. Gurevych. Parsing argumentation structures in persuasive essays. *CoRR*, abs/1604.07370, 2016.

[205] C. Stab, C. Kirschner, J. Eckle-Kohler, and I. Gurevych. Argumentation mining in persuasive essays and scientific articles from the discourse structure perspective. *Frontiers and Connections between Argumentation Theory and Natural Language Processing*, 2014.

[206] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307, 2011.

[207] M. Taboada and W. Mann. Rhetorical structure theory: Looking back and moving ahead. *Discourse Studies*, 8(3):423–459, 2006.

[208] M. Taboada, K. Voll, and J. Brooke. Extracting sentiment as a function of discourse structure and topicality. Technical report, Simon Fraser Univeristy School of Computing Science Technical Report, 2008.

[209] D. Tang, F. Wei, B. Qin, T. Liu, and M. Zhou. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 208–212. ACL, 2014.

[210] M. Thimm. A probabilistic semantics for abstract argumentation. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, pages 750–755. IOS Press, 2012.

[211] F. Toni. A tutorial on assumption-based argumentation. *Argument & Computation*, 5(1):89–117, 2014.

[212] S. Toulmin. *The uses of argument*. Cambridge University Press, 2003.

[213] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference of the North American Chapter of the ACL on*

Human Language Technology (HLT-NAACL)-Volume 1, pages 173–180. ACL, 2003.

[214] K. Toutanova and C. D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and very large corpora-Volume 13*, pages 63–70. ACL, 2000.

[215] A. C.-R. Tsai, C.-E. Wu, R. T.-H. Tsai, and J. Y.-j. Hsu. Building a concept-level sentiment dictionary based on commonsense knowledge. *IEEE Intelligent Systems*, pages 22–30, 2013.

[216] Y. Tsuruoka, Y. Miyao, and J. Kazama. Learning with lookahead: can history-based models rival globally optimized models? In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL)*, pages 238–246. ACL, 2011.

[217] P. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on ACL (ACL)*, pages 417–424. ACL, 2002.

[218] Ö. Uzuner, X. Zhang, and T. Sibanda. Machine learning and rule-based approaches to assertion classification. *Journal of the American Medical Informatics Association*, 16(1):109–115, 2009.

[219] F. Van Eemeren and R. Grootendorst. *A systematic theory of argumentation: The pragma-dialectical approach*, volume 14. Cambridge University Press, 2003.

[220] C. J. Van Rijsbergen, S. E. Robertson, and M. F. Porter. *New models in probabilistic information retrieval.* Computer Laboratory, University of Cambridge, 1980.

[221] M. P. G. Villalba and P. Saint-Dizier. Some facets of argument mining for opinion analysis. In *Proceedings of Computational Models of Argument (COMMA)*, pages 23–34. IOS Press, 2012.

[222] J. Villena Román, S. Collada Pérez, S. Lana Serrano, and J. C. González Cristóbal. Hybrid approach combining machine learning and

a rule-based expert system for text categorization. In *Proceedings of the 24th Florida Artificial Intelligence Research Society Conference (FLAIRS)*. AAAI, 2011.

[223] G. A. Vreeswijk. Abstract argumentation systems. *Artificial intelligence*, 90(1):225–279, 1997.

[224] H. Wachsmuth, M. Trenkmann, B. Stein, G. Engels, and T. Palakarska. A review corpus for argumentation analysis. In *Computational Linguistics and Intelligent Text Processing*, pages 115–127. Springer, 2014.

[225] M. A. Walker, J. E. F. Tree, P. Anand, R. Abbott, and J. King. A corpus for research on deliberation and debate. In *Proceedings of the 8th international Conference on Language Resources and Evaluation (LREC)*, pages 812–817. Citeseer, 2012.

[226] D. Walton. *Argumentation schemes for presumptive reasoning*. Routledge, 2013.

[227] D. N. Walton. Enthymemes, common knowledge, and plausible inference. *Philosophy and rhetoric*, 34(2):93–112, 2001.

[228] P. Wang, C. Domeniconi, and J. Hu. Using wikipedia for co-clustering based cross-domain text classification. In *Eighth IEEE International Conference on Data Mining (ICDM)*, pages 1085–1090. IEEE, 2008.

[229] S. Wang and C. D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the ACL (ACL): Short Papers-Volume 2*, pages 90–94. ACL, 2012.

[230] A. Weichselbraun, S. Gindl, and A. Scharl. Enriching semantic knowledge bases for opinion mining in big data applications. *Knowledge-Based Systems*, 69:78–85, 2014.

[231] M. Whitehead and L. Yaeger. Building a general purpose cross-domain sentiment mining model. In *WRI World Congress on Computer Science and Information Engineering*, volume 4, pages 472–476. IEEE, 2009.

[232] J. Wiebe and E. Riloff. Creating subjective and objective sentence classifiers from unannotated texts. In *Computational Linguistics and Intelligent Text Processing*, pages 486–497. Springer, 2005.

[233] J. Wiebe, T. Wilson, R. Bruce, M. Bell, and M. Martin. Learning subjective language. *Computational linguistics*, 30(3):277–308, 2004.

[234] J. Wiebe, T. Wilson, and C. Cardie. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2):165–210, 2005.

[235] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 347–354. ACL, 2005.

[236] Q. Wu, S. Tan, H. Zhai, G. Zhang, M. Duan, and X. Cheng. Sentirank: Cross-domain graph ranking for sentiment classification. In *Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 309–314. IEEE Computer Society, 2009.

[237] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on ACL (ACL)*, pages 133–138. ACL, 1994.

[238] A. Wyner, W. Peters, and D. Price. Argument discovery and extraction with the argument workbench. *NAACL HLT 2015*, page 78, 2015.

[239] A. Wyner, J. Schneider, K. Atkinson, and T. Bench-Capon. Semi-automated argumentative analysis of online product reviews. In *Proceedings of the 4th International Conference on Computational Models of Argument (COMMA 2012)*, pages 43–50. IOS Press, 2012.

[240] R. Xia, C. Zong, X. Hu, and E. Cambria. Feature ensemble plus sample selection: domain adaptation for sentiment classification. *IEEE Intelligent Systems*, 28(3):10–18, 2013.

[241] G.-R. Xue, W. Dai, Q. Yang, and Y. Yu. Topic-bridged plsa for cross-domain text classification. In *Proceedings of the 31st annual interna-*

tional ACM SIGIR Conference on Research and development in information retrieval, pages 627–634. ACM, 2008.

[242] A. Yeh. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational linguistics-Volume 2*, pages 947–953. ACL, 2000.

[243] H. Yu and V. Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)-Volume 10*, pages 129–136. ACL, 2003.

[244] J. Zabkar, M. Mozina, J. Videcnik, and I. Bratko. Argument based machine learning in a medical domain. In *Proceeding of the 2006 Conference on Computational Models of Argument (COMMA)*, pages 59–70. IOS Press, 2006.

[245] M. Zhang and X. Ye. A generation model to unify topic relevance and lexicon-based sentiment for opinion retrieval. In *Proceedings of the 31st annual international ACM SIGIR Conference on Research and development in information retrieval*, pages 411–418. ACM, 2008.

[246] S. Zhou, Q. Chen, and X. Wang. Active deep networks for semi-supervised sentiment classification. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1515–1523. ACL, 2010.

Appendices

A. RbAM Argument base

Table A.1.: Arguments comprising the RbAM argument base (1/4)

Category	**id**	**Premise**	**Conclusion**	$\mathbf{S^0}$	**T**	**R**
Sentiment	$\sigma_1(I)$	Child in I has neg. sentiment	$I \in Att$	0.5	Pro	$\{(\sigma_1(I), Att)\}$
	$\sigma_2(I)$	Child in I has pos. sentiment	$I \in Sup$	0.5	Pro	$\{(\sigma_2(I), Sup)\}$
	$\sigma_3(I)$	Parent in I has neg. sentiment	$Premise(\sigma_1(I))$	0.5	Con	$\{(\sigma_3(I), \sigma_1(I))\}$
Similarity	$\theta_1(I)$	Similarity is $>.1$	$I \in Att$	0.5	Pro	$\{(\theta_1(I), Att)\}$
	$\theta_2(I)$	Similarity is $>.1$	$I \in Sup$	0.5	Pro	$\{(\theta_1(I), Sup)\}$
	$\theta_3(I)$	Similarity is $=<.1$	$I \in Att$	0.5	Con	$\{(\theta_2(I), Att)\}$
	$\theta_4(I)$	Similarity is $=<.1$	$I \in Sup$	0.5	Con	$\{(\theta_2(I), Sup)\}$
	$\theta_5(I)$	Similarity is $=<.1$	$Premise(\sigma_1(I))$	0.5	Con	$\{(\theta_3(I), \sigma_1(I))\}$
	$\theta_6(I)$	Similarity is $=<.1$	$Premise(\sigma_2(I))$	0.5	Con	$\{(\theta_3(I), \sigma_2(I))\}$
	$\theta_7(I)$	Similarity mainly stopwords	$Premise(\theta_1(I))$	0.5	Con	$\{(\theta_4(I), \theta_1(I))\}$
	$\theta_8(I)$	Similarity mainly stopwords	$Premise(\theta_2(I))$	0.5	Con	$\{(\theta_4(I), \theta_2(I))\}$
	$\theta_9(I)$	Similarity mainly nonStopwords	$Premise(\theta_1(I))$	0.5	Con	$\{(\theta_5(I), \theta_1(I))\}$
	$\theta_{10}(I)$	Similarity mainly nonStopwords	$Premise(\theta_2(I))$	0.5	Con	$\{(\theta_5(I), \theta_2(I))\}$
	$\theta_{11}(I)$	Similarity mainly entities	$Premise(\theta_9(I))$	0.5	Pro	$\{(\theta_{11}(I), \theta_9(I)\}$
	$\theta_{12}(I)$	Similarity mainly entities	$Premise(\theta_{10}(I))$	0.5	Pro	$\{(\theta_{11}(I), \theta_{10}(I)\}$
FirstWord	$\phi_{well}(I)$	Child in I starts with *well*	$I \in Att$	0.5	Pro	$\{(\phi_{well}(I), \kappa_{well}(I))\}$
	$\phi_{and}(I)$	Child in I starts with *and*	$I \in Sup$	0.5	Pro	$\{(\phi_{and}(I), \kappa_{and}(I))\}$
	$\phi_{likewise}(I)$	Child in I starts with *likewise*	$I \in Sup$	0.5	Pro	$\{(\phi_{likewise}(I), \kappa_{likewise}(I))\}$
	$\phi_{yes}(I)$	Child in I starts with *yes*	$I \in Sup$	0.5	Pro	$\{(\phi_{yes}(I), \kappa_{yes}(I))\}$
	$\phi_{i\ agree}(I)$	Child in I starts with *i agree*	$I \in Sup$	0.5	Pro	$\{(\phi_{i\ agree}(I), \kappa_{agree}(I))\}$
	$\phi_{instead}(I)$	Child in I starts with *instead*	$I \in Sup$	0.5	Pro	$\{(\phi_{instead}(I), \kappa_{instead}(I))\}$
	$\phi_{indeed}(I)$	Child in I starts with *indeed*	$I \in Sup$	0.5	Pro	$\{(\phi_{indeed}(I), \kappa_{indeed}(I))\}$
	$\phi_{but}(I)$	Child in I starts with *but*	$I \in Sup$	0.5	Pro	$\{(\phi_{but}(I), \kappa_{but}(I))\}$
	$\phi_{no}(I)$	Child in I starts with *no*	$I \in Sup$	0.5	Pro	$\{(\phi_{no}(I), \kappa_{no}(I))\}$
	$\phi_{actually}(I)$	Child in I starts with *actually*	$I \in Sup$	0.5	Pro	$\{(\phi_{actually}(I), \kappa_{actually}(I))\}$

Table A.2.: Arguments comprising the RbAM argument base (2/4)

Category	**id**	**Premise**	**Conclusion**	$\mathbf{S}^0$	**T**	**R**
Keyword	$\kappa_{disagree}(I)$	Child in I contains *disagree*	$I \in Att$	0.4	Pro	$\{(\kappa_{disagree}(I), Att)\}$
	$\kappa_{alternatively}(I)$	Child in I contains *alternatively*	$I \in Att$	0.4	Pro	$\{(\kappa_{alternatively}(I), Att)\}$
	$\kappa_{although}(I)$	Child in I contains *although*	$I \in Att$	0.4	Pro	$\{(\kappa_{although}(I), Att)\}$
	$\kappa_{as\ an\ alternative}(I)$	Child in I contains *as an alternative*	$I \in Att$	0.4	Pro	$\{(\kappa_{as\ an\ alternative}(I), Att)\}$
	$\kappa_{as\ though}(I)$	Child in I contains *as though*	$I \in Att$	0.4	Pro	$\{(\kappa_{as\ though}(I), Att)\}$
	$\kappa_{but}(I)$	Child in I contains *but*	$I \in Att$	0.4	Pro	$\{(\kappa_{but}(I), Att)\}$
	$\kappa_{by\ comparison}(I)$	Child in I contains *by comparison*	$I \in Att$	0.4	Pro	$\{(\kappa_{by\ comparison}(I), Att)\}$
	$\kappa_{by\ contrast}(I)$	Child in I contains *by contrast*	$I \in Att$	0.4	Pro	$\{(\kappa_{by\ contrast}(I), Att)\}$
	$\kappa_{conversely}(I)$	Child in I contains *conversely*	$I \in Att$	0.4	Pro	$\{(\kappa_{conversely}(I), Att)\}$
	$\kappa_{however}(I)$	Child in I contains *however*	$I \in Att$	0.4	Pro	$\{(\kappa_{however}(I), Att)\}$
	$\kappa_{else}(I)$	Child in I contains *else*	$I \in Att$	0.4	Pro	$\{(\kappa_{else}(I), Att)\}$
	$\kappa_{except}(I)$	Child in I contains *except*	$I \in Att$	0.4	Pro	$\{(\kappa_{except}(I), Att)\}$
	$\kappa_{in\ contrast}(I)$	Child in I contains *in contrast*	$I \in Att$	0.4	Pro	$\{(\kappa_{in\ contrast}(I), Att)\}$
	$\kappa_{instead}(I)$	Child in I contains *instead*	$I \in Att$	0.4	Pro	$\{(\kappa_{instead}(I), Att)\}$
	$\kappa_{lest}(I)$	Child in I contains *lest*	$I \in Att$	0.4	Pro	$\{(\kappa_{lest}(I), Att)\}$
	$\kappa_{much\ as}(I)$	Child in I contains *much as*	$I \in Att$	0.4	Pro	$\{(\kappa_{much\ as}(I), Att)\}$
	$\kappa_{nevertheless}(I)$	Child in I contains *nevertheless*	$I \in Att$	0.4	Pro	$\{(\kappa_{nevertheless}(I), Att)\}$
	$\kappa_{nonetheless}(I)$	Child in I contains *nonetheless*	$I \in Att$	0.4	Pro	$\{(\kappa_{nonetheless}(I), Att)\}$
	$\kappa_{nor}(I)$	Child in I contains *nor*	$I \in Att$	0.4	Pro	$\{(\kappa_{nor}(I), Att)\}$
	$\kappa_{on\ the\ contrary}(I)$	Child in I contains *on the contrary*	$I \in Att$	0.4	Pro	$\{(\kappa_{on\ the\ contrary}(I), Att)\}$
	$\kappa_{on\ the\ other\ hand}(I)$	Child in I contains *on the other hand*	$I \in Att$	0.4	Pro	$\{(\kappa_{on\ the\ other\ hand}(I), Att)\}$
	$\kappa_{otherwise}(I)$	Child in I contains *otherwise*	$I \in Att$	0.4	Pro	$\{(\kappa_{otherwise}(I), Att)\}$
	$\kappa_{rather}(I)$	Child in I contains *rather*	$I \in Att$	0.4	Pro	$\{(\kappa_{rather}(I), Att)\}$
	$\kappa_{regardless}(I)$	Child in I contains *regardless*	$I \in Att$	0.4	Pro	$\{(\kappa_{regardless}(I), Att)\}$
	$\kappa_{unless}(I)$	Child in I contains *unless*	$I \in Att$	0.4	Pro	$\{(\kappa_{unless}(I), Att)\}$
	$\kappa_{whereas}(I)$	Child in I contains *whereas*	$I \in Att$	0.4	Pro	$\{(\kappa_{whereas}(I), Att)\}$

Table A.3.: Arguments comprising the RbAM argument base (3/4)

Category	**id**	**Premise**	**Conclusion**	$\mathbf{S}^0$	**T**	**R**
Keyword	$\kappa_{yet}(I)$	Child in I contains *yet*	$I \in Att$	0.4	Pro	$\{(\kappa_{yet}(I), Att)\}$
	$\kappa_{no}(I)$	Child in I contains *no*	$I \in Att$	0.4	Pro	$\{(\kappa_{no}(I), Att)\}$
	$\kappa_{actually}(I)$	Child in I contains *actually*	$I \in Att$	0.4	Pro	$\{(\kappa_{actually}(I), Att)\}$
	$\kappa_{well}(I)$	Child in I contains *well*	$I \in Att$	0.4	Pro	$\{(\kappa_{well}(I), Att)\}$
	$\kappa_{i\ agree}(I)$	Child in I contains *i agree*	$I \in Sup$	0.5	Pro	$\{(\kappa_{i\ agree}(I), Sup)\}$
	$\kappa_{because}(I)$	Child in I contains *because*	$I \in Sup$	0.5	Pro	$\{(\kappa_{because}(I), Sup)\}$
	$\kappa_{also}(I)$	Child in I contains *also*	$I \in Sup$	0.5	Pro	$\{(\kappa_{also}(I), Sup)\}$
	$\kappa_{as\ a\ result}(I)$	Child in I contains *as a result*	$I \in Sup$	0.5	Pro	$\{(\kappa_{as\ a\ result}(I), Sup)\}$
	$\kappa_{as\ soon\ as}(I)$	Child in I contains *as soon as*	$I \in Sup$	0.5	Pro	$\{(\kappa_{as\ soon\ as}(I), Sup)\}$
	$\kappa_{as\ well}(I)$	Child in I contains *as well*	$I \in Sup$	0.5	Pro	$\{(\kappa_{as\ well}(I), Sup)\}$
	$\kappa_{besides}(I)$	Child in I contains *besides*	$I \in Sup$	0.5	Pro	$\{(\kappa_{besides}(I), Sup)\}$
	$\kappa_{consequently}(I)$	Child in I contains *consequently*	$I \in Sup$	0.5	Pro	$\{(\kappa_{consequently}(I), Sup)\}$
	$\kappa_{earlier}(I)$	Child in I contains *earlier*	$I \in Sup$	0.5	Pro	$\{(\kappa_{earlier}(I), Sup)\}$
	$\kappa_{finally}(I)$	Child in I contains *finally*	$I \in Sup$	0.5	Pro	$\{(\kappa_{finally}(I), Sup)\}$
	$\kappa_{and}(I)$	Child in I contains *and*	$I \in Sup$	0.5	Pro	$\{(\kappa_{and}(I), Sup)\}$
	$\kappa_{for\ example}(I)$	Child in I contains *for example*	$I \in Sup$	0.5	Pro	$\{(\kappa_{for\ example}(I), Sup)\}$
	$\kappa_{for\ instance}(I)$	Child in I contains *for instance*	$I \in Sup$	0.5	Pro	$\{(\kappa_{for\ instance}(I), Sup)\}$
	$\kappa_{further}(I)$	Child in I contains *further*	$I \in Sup$	0.5	Pro	$\{(\kappa_{further}(I), Sup)\}$
	$\kappa_{furthermore}(I)$	Child in I contains *furthermore*	$I \in Sup$	0.5	Pro	$\{(\kappa_{furthermore}(I), Sup)\}$
	$\kappa_{hence}(I)$	Child in I contains *hence*	$I \in Sup$	0.5	Pro	$\{(\kappa_{hence}(I), Sup)\}$
	$\kappa_{in\ addition}(I)$	Child in I contains *in addition*	$I \in Sup$	0.5	Pro	$\{(\kappa_{in\ addition}(I), Sup)\}$
	$\kappa_{in\ other\ words}(I)$	Child in I contains *in other words*	$I \in Sup$	0.5	Pro	$\{(\kappa_{in\ other\ words}(I), Sup)\}$
	$\kappa_{in\ particular}(I)$	Child in I contains *in particular*	$I \in Sup$	0.5	Pro	$\{(\kappa_{in\ particular}(I), Sup)\}$
	$\kappa_{in\ short}(I)$	Child in I contains *in short*	$I \in Sup$	0.5	Pro	$\{(\kappa_{in\ short}(I), Sup)\}$
	$\kappa_{yes}(I)$	Child in I contains *yes*	$I \in Sup$	0.5	Pro	$\{(\kappa_{yes}(I), Sup)\}$
	$\kappa_{in\ sum}(I)$	Child in I contains *in sum*	$I \in Sup$	0.5	Pro	$\{(\kappa_{in\ sum}(I), Sup)\}$

Table A.4.: Arguments comprising the RbAM argument base (4/4). Here we only show a generic negation argument $\nu_{Keyword}(I, I')$. However, AB_{RbAM} contains one negation argument for each keyword argument listed, as described in section 6.2.

Category	**id**	**Premise**	**Conclusion**	$\mathbf{S^0}$	**T**	**R**
Keyword	$\kappa_{in\ the\ end}(I)$	Child in I contains *in the end*	$I \in Sup$	0.5	Pro	$\{(\kappa_{in\ the\ end}(I), Sup)\}$
	$\kappa_{in\ turn}(I)$	Child in I contains *in turn*	$I \in Sup$	0.5	Pro	$\{(\kappa_{in\ turn}(I), Sup)\}$
	$\kappa_{indeed}(I)$	Child in I contains *indeed*	$I \in Sup$	0.5	Pro	$\{(\kappa_{indeed}(I), Sup)\}$
	$\kappa_{insofar\ as}(I)$	Child in I contains *insofar as*	$I \in Sup$	0.5	Pro	$\{(\kappa_{insofar\ as}(I), Sup)\}$
	$\kappa_{later}(I)$	Child in I contains *later*	$I \in Sup$	0.5	Pro	$\{(\kappa_{later}(I), Sup)\}$
	$\kappa_{likewise}(I)$	Child in I contains *likewise*	$I \in Sup$	0.5	Pro	$\{(\kappa_{likewise}(I), Sup)\}$
	$\kappa_{moreover}(I)$	Child in I contains *moreover*	$I \in Sup$	0.5	Pro	$\{(\kappa_{moreover}(I), Sup)\}$
	$\kappa_{next}(I)$	Child in I contains *next*	$I \in Sup$	0.5	Pro	$\{(\kappa_{next}(I), Sup)\}$
	$\kappa_{now\ that}(I)$	Child in I contains *now that*	$I \in Sup$	0.5	Pro	$\{(\kappa_{nowthat}(I), Sup)\}$
	$\kappa_{overall}(I)$	Child in I contains *overall*	$I \in Sup$	0.5	Pro	$\{(\kappa_{overall}(I), Sup)\}$
	$\kappa_{plus}(I)$	Child in I contains *plus*	$I \in Sup$	0.5	Pro	$\{(\kappa_{plus}(I), Sup)\}$
	$\kappa_{similarly}(I)$	Child in I contains *similarly*	$I \in Sup$	0.5	Pro	$\{(\kappa_{similarly}(I), Sup)\}$
	$\kappa_{simultaneously}(I)$	Child in I contains *simultaneously*	$I \in Sup$	0.5	Pro	$\{(\kappa_{simultaneously}(I), Sup)\}$
	$\kappa_{since}(I)$	Child in I contains *since*	$I \in Sup$	0.5	Pro	$\{(\kappa_{since}(I), Sup)\}$
	$\kappa_{so\ that}(I)$	Child in I contains *so that*	$I \in Sup$	0.5	Pro	$\{(\kappa_{so\ that}(I), Sup)\}$
	$\kappa_{specifically}(I)$	Child in I contains *specifically*	$I \in Sup$	0.5	Pro	$\{(\kappa_{specifically}(I), Sup)\}$
	$\kappa_{thereafter}(I)$	Child in I contains *thereafter*	$I \in Sup$	0.5	Pro	$\{(\kappa_{thereafter}(I), Sup)\}$
	$\kappa_{thereby}(I)$	Child in I contains *thereby*	$I \in Sup$	0.5	Pro	$\{(\kappa_{thereby}(I), Sup)\}$
	$\kappa_{therefore}(I)$	Child in I contains *therefore*	$I \in Sup$	0.5	Pro	$\{(\kappa_{therefore}(I), Sup)\}$
	$\kappa_{thus}(I)$	Child in I contains *thus*	$I \in Sup$	0.5	Pro	$\{(\kappa_{thus}(I), Sup)\}$
	$\kappa_{till}(I)$	Child in I contains *till*	$I \in Sup$	0.5	Pro	$\{(\kappa_{till}(I), Sup)\}$
	$\kappa_{ultimately}(I)$	Child in I contains *ultimately*	$I \in Sup$	0.5	Pro	$\{(\kappa_{ultimately}(I), Sup)\}$
	$\kappa_{accordingly}(I)$	Child in I contains *accordingly*	$I \in Sup$	0.5	Pro	$\{(\kappa_{accoringly}(I), Sup)\}$
	$\kappa_{additionally}(I)$	Child in I contains *additionally*	$I \in Sup$	0.5	Pro	$\{(\kappa_{additionally}(I), Sup)\}$
Negation	$\nu_{Keyword}(I, I')$	*Keyword* occurs negated in I	$Premise(\kappa_{I'}(I))$	0.5	Con	$\{(\sigma_{Keyword}(I), \kappa_{I'}(I)\}$

B. Discourse markers

Table B.1.: All *Explicit Discourse Connectives* that are listed in the *Penn Discourse Treebank (PDTB)* manual [173], as described in section 2.2.2, grouped by classes *Attacking*, *Supporting* and *Neither/Ambiguous* (1/2)

Attacking	alternatively	although	as an alternative
	as though	but	by comparison
	by contrast	conversely	else
	except	however	in contrast
	instead	lest	much as
	neither...nor	nevertheless	nonetheless
	nor	on the contrary	on the other hand
	otherwise	rather	regardless
	though	unless	whereas
	yet		
Supporting	accordingly	additionally	also
	and	as a result	as soon as
	as well	because	besides
	consequently	earlier	finally
	for example	for instance	further
	furthermore	hence	in addition
	in other words	in particular	in short
	in sum	in the end	in turn
	indeed	insofar as	later
	likewise	moreover	next
	now that	overall	plus
	previously	similarly	simultaneously
	since	so	so that
	specifically	thereafter	thereby
	therefore	thus	till
	ultimately		

Table B.2.: All *Explicit Discourse Connectives* that are listed in the *Penn Discourse Treebank (PDTB)* manual [173], as described in section 2.2.2, grouped by classes *Attacking*, *Supporting* and *Neither/Ambiguous* (2/2)

Neither/ Ambiguous	after	afterward	as
	as if	before	before and after
	by then	either...or	for
	if	if and when	if...then
	in fact	meantime	meanwhile
	once	or	separately
	still	then	until
	when	when and if	while

C. News articles corpus annotation instructions

Table C.1.: Instructions that were given to annotators to label sentence pairs as (Att, Sup, N) to create the corpus described in section 5.5.3 (1/2)

For this task you will need to decide how two sentences are related.
You will be presented with pairs of sentences, *sentence A* and *sentence B*,
and you will need to decide whether sentence B *attacks* or *supports* sentence A,
or whether it does *neither*.

Here are some examples:
- A: I like apples because they are so sweet.
  B: Most of the time apples are quite sour, actually.
  B Attacks A

- A: I like apples because they are so sweet.
  B: Apples are often red or green.
  B Neither attacks nor supports A

All sentences you will come across are, in one way or another,
about British politics (not apples!), more specifically
the UK Independence Party (UKIP), whose leader is named Nigel Farage.

Before you start this task a few definitions may be useful:
- An argument is a group of statements,
  comprised of one or more premises and one conclusion.
- A statement is either true or false, e.g. "The cat is on the mat."
- A premise (or 'claim') is a statement in an argument
  that provides reason or support for the conclusion.
- A conclusion is a statement in an argument that indicates of what the arguer
  is trying to convince the reader/listener. What is the argument trying to prove?

Table C.2.: Instructions that were given to annotators to label sentence pairs as (Att, Sup, N) to create the corpus described in section 5.5.3 (2/2)

In natural language parts of an argument are often left implicit. A sentence may thus be 'argumentative', despite not containing both premises and a conclusion. A sentence may neither attack nor support another sentence because - it is not aimed at a claim/opinion stated in the sentence it is being compared to or - the two sentences are simply not related.
Below please enter whether sentence B attacks (Att) or supports (Sup) sentence A or if whether does neither (N)

Sentence A	**Sentence B**	**Att/Supp/N**
This is an example sentence.	**Yes, indeed it explains how to do this task!**	**Sup**
The sooner we are out of the EU, the sooner we can start making this country great again.	The UK should quit the EU, with the referendum being moved forward.	
...	...	

D. Machine Learning

The tasks of both argumentation mining and sentiment analysis are ones of recognising patterns in text that reveal which class an instance ought to be assigned to. The algorithms we use in our work to do so are all supervised and thus rely on annotated corpora, consisting of a set of instances, each assigned a class label. We have experimented with a number of supervised algorithms, mostly using either the *Weka* tool [75] or *libSVM* [45]. The ones we have eventually focused on are the following:

- Random Forests
- Support Vector Machines
- Naïve Bayes classifiers

Below we describe each algorithm in turn.

D.1. Random Forests

A Random Forest [28] is an ensemble classifier in which we construct a number of decision trees and aggregate the results to obtain classifications. Each tree classifier is constructed in a two-step process:

1. Randomly draw a sample $\boldsymbol{d} = \{d_1, d_2, ..., d_n\}$ from training data, where each d_i is represented by a feature vector θ_i, comprised of M features
2. Grow a classification tree from $\boldsymbol{d}$ as follows: at each node randomly select a predefined number of $m < M$ features and split the node based on the selected features so that the best split between the classes in $\boldsymbol{d}$ is achieved

This process is repeated to grow a predefined number of decision trees. Formally, a Random Forest then consists of a collection of decision trees

$$\{h(\boldsymbol{x}, \theta_1), ..., h(\boldsymbol{x}, \theta_k)\}$$

where θ_j is an independent identically distributed (iid) random vector and $\boldsymbol{x}$ is the input to the decision tree. Each tree in the forest contributes a unitary vote to the most popular class at input $\boldsymbol{x}$. The final class label is then decided according to a majority vote. Two variables have to be set when using random forests:

1. The number of trees
2. The size of the subset of features selected from at each tree node

D.2. Support Vector Machines

Support Vector Machines (SVM), pioneered by Corinna Cortes and Vladimir Vapnik [52], have been one of the more popular Supervised Learning algorithms not just in sentiment analysis and argumentation mining, but various other fields, as well, such as spam filtering, e.g. [56, 184], or medical research, e.g. [31, 73, 86]. Originally an algorithm used to solve binary classification problems, SVM have been adapted to perform multi-class classification, as well. The description below has been adapted from [20].

Consider first a binary classification problem, such as determining a text as being either argumentative or non-argumentative. We can describe such a problem with a model of the form

$$y(x) = \mathbf{w}^T \phi(x) + b$$

where $\phi(x)$ is a feature-space transformation of the data x, $\mathbf{w}^T$ is a weight vector and b is a bias. Transforming data into feature space can yield linear separability of data that is not linearly separable in the original data space. A training data set consists of N input vectors $x_1, ..., x_N$, all of which have a class label $C \in \{1, -1\}$ and new data is classified according to the sign of $y(x)$. If the training data is linearly separable in feature space, we can perfectly split the data according to their class with one hyperplane in feature space. When this is not the case, so called slack variables may be introduced to allow some of the training instances to be misclassified. SVM find the hyperplane that not only separates the data but that maximises the margin,

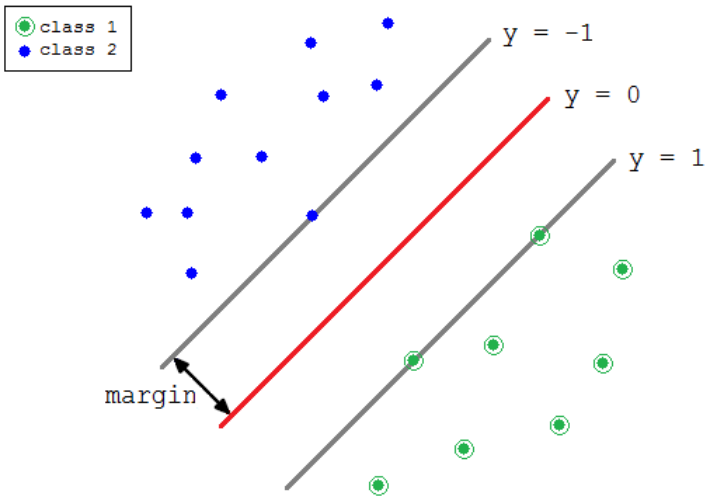


Figure D.1.: Illustration of support vectors. "The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points" [20]. The support vectors are those data points that lie on the decision boundaries.

i.e. the distance between the hyperplane and the support vectors, as well. This means, the separating hyperplane is placed in such a manner that the distance between the separating plane and the nearest points of both classes is maximised. This is illustrated in figure D.1. One of the crucial benefits of SVM is the fact that, in order to classify new data, we only need to retain the support vectors, i.e. those data points that define the maximum margin planes that separate the classes.

To apply SVM to multi-class problems we use either *one-versus-one* or *one-versus-all* SVM. For *one-versus-all* we construct K binary SVM "in which the k^{th} model $y_k(x)$ is trained using the data from class C_k as the positive examples and the data from the remaining $K-1$ classes as the negative examples" [20]. For the *one-versus-one* SVM we also use multiple binary classifiers, one for each possible combination of classes. A test point is then classified through a majority vote.

D.3. Naïve Bayes classifiers.

We adopt the below explanation of Naïve Bayes Classifiers from [162]. A Naïve Bayes Classifier assigns an instance s the class $c^* = \arg\max_c P(c|s)$ where, according to Bayes rule,

$$P(c|s) = \frac{P(c)P(s|c)}{P(s)}$$

in which $P(s)$ is a normalising factor. The probability of an instance belonging to one of the possible classes, e.g. argumentative/non-argumentative, is decided upon on the basis of a set of m possibly occurring features, $\{f_1, f_2, ..., f_m\}$. Assuming conditional independence between all f_i, the above equation may be rewritten as

$$P(c|s) = \frac{P(c)(\prod_{i=1}^{m} P(f_i|c)^{n_i(s)})}{P(s)}$$