On the Impact of Data Selection when Applying Machine Learning in Abstract Argumentation

Isabelle KUHLMANN, Thorsten WUJEK and Matthias THIMM
Artificial Intelligence Group, University of Hagen, Germany

Abstract. We examine the impact of both training and test data selection in machine learning applications for abstract argumentation, in terms of prediction accuracy and generalizability. For that, we first review previous studies from a data-centric perspective and conduct some experiments to back up our analysis. We further present a novel algorithm to generate particularly challenging argumentation frameworks wrt. the task of deciding skeptical acceptability under preferred semantics. Moreover, we investigate graph-theoretical aspects of the existing datasets and perform some experiments which show that some simple properties (such as in-degree and out-degree of an argument) are already quite strong indicators of whether or not an argument is skeptically accepted under preferred semantics.

Keywords. Abstract Argumentation, Approximate Reasoning, Artificial Neural Networks, Graph Neural Networks, Machine Learning

1. Introduction

Formal argumentation is a modern approach for non-monotonic reasoning within the general area of *Artificial Intelligence*. In particular, an *(abstract) argumentation framework* [1] consists of a set of arguments and a relation describing attacks between such arguments. Semantics are expressed in form of *extensions*, i.e., sets of arguments that meet certain prerequisites and are thus considered mutually acceptable. Typical computational problems in abstract argumentation are deciding whether an argument is included in some extension of a given semantics (or in all such extensions), or enumerating one (or all) extensions of a given semantics [2,3,4]. Most algorithmic approaches for solving such problems are sound and complete methods; see [4] for a recent overview. However, the high complexity of these problems—for instance, the problem of deciding whether an argument is included in all preferred extensions is Π_2^P-complete [2]—prohibits a scalable behavior of such approaches in the worst case. Thus, with the rise of *Deep Learning* approaches for numerous fields of application in recent years, a few authors suggested to use artificial neural networks for this task [5,6,7]. The advantage of a neural network is that, once it is trained properly, it can solve problems in time linear to the input. Nevertheless, this comes at the cost of exactness: the result is not guaranteed to be correct.

In this work, we review the existing literature on the topic of deep learning approaches for abstract argumentation with emphasis on the data used in the individual works. Although there is some overlap, none of the works use the same dataset, neither for training nor for testing purposes. This is (at least partly) due to the reason that for abstract argumentation, there exists no dedicated “standard” data set suitable for Machine Learning (ML) purposes, as opposed to, e.g., image processing, where datasets such as MNIST[1] or CIFAR[2] are well-known and publicly available. This paper takes a data-centric perspective and examines which properties a dataset in our field should possess. We perform an experimental analysis in which we explore the impact of different training and test sets on two different neural network architectures known from the literature. We also propose an algorithm to generate particularly challenging argumentation frameworks for the task of deciding skeptical acceptability under preferred semantics.

Furthermore, we investigate graph-theoretical properties of the datasets at hand. We show that some simple properties can be quite strong indicators of an argument’s acceptability status. To illustrate this, we conduct some experiments in which we use a selection of “classical” ML methods which are only given the arguments’ in-degrees and out-degrees as features. While the accuracy of these approaches is still lower than the accuracy of the deep learning methods, it is surprisingly high and raises doubt about the requirement to use complex deep learning methods for this purpose.

The remainder of this paper is structured as follows. We begin by giving an overview on abstract argumentation as well as related neural network techniques in Section 2. In Section 3, we take a closer look at the data used in the context of existing deep learning solutions for abstract argumentation and propose a new dataset which is particularly challenging for the task of deciding skeptical acceptability under preferred semantics. We conduct some experimental analysis on this new dataset as well as existing ones in Section 4. Section 5 provides a discussion and deeper analysis, in particular regarding graph-theoretical properties, and Section 6 concludes this work.

2. Preliminaries

We provide the basics of abstract argumentation in Section 2.1 and give an overview on existing works using artificial neural networks for abstract argumentation in Section 2.2.

2.1. Abstract Argumentation

An abstract argumentation framework (AF) [1] is a tuple $F = (\mathsf{Args}, R)$, where Args is the set of arguments and $R \subseteq \mathsf{Args} \times \mathsf{Args}$ is the attack relation. An argument $a \in \mathsf{Args}$ is said to *attack* another argument $b \in \mathsf{Args}$ if $(a,b) \in R$. We abbreviate

$$a_F^- = \{b \in \mathsf{Args} \mid (b,a) \in R\} \qquad\qquad a_F^+ = \{b \in \mathsf{Args} \mid (a,b) \in R\}$$

and analogously E_F^- and E_F^+ for a set $E \subseteq \mathsf{Args}$. An argument $a \in \mathsf{Args}$ is *defended by* a set of arguments $E \subseteq \mathsf{Args}$ if $a_F^- \subseteq E^+$. A set $E \subseteq \mathsf{Args}$ is *conflict-free* if $E \cap E^+ = \emptyset$. A set $E \subseteq \mathsf{Args}$ is called *admissible* (we also say $E \in \mathsf{ad}(F)$) if E is conflict-free and each $a \in E$ is defended by E within a given AF F. E is a *preferred* extension (i.e., $E \in \mathsf{pr}(F)$) if $E \in \mathsf{ad}(F)$ and for every $E' \in \mathsf{ad}(F)$, $E \not\subset E'$. An argument $a \in \mathsf{Args}$ is *skeptically accepted wrt. preferred semantics* (abbreviated pa) iff a is contained in

[1]http://yann.lecun.com/exdb/mnist/
[2]https://www.cs.toronto.edu/~kriz/cifar.html

every preferred extension. Let $\mathsf{pa}(F)$ denote the set of all pa arguments in F and define $\mathsf{pna}(F) = \mathsf{Args} \setminus \mathsf{pa}(F)$. We denote the computational problem of deciding whether an argument is skeptically accepted wrt. preferred semantics as $\mathsf{DS}_{\mathsf{pr}}$. An argument $a \in \mathsf{Args}$ is *credulously accepted wrt. preferred semantics* iff a is contained in some preferred extension.

We say that $E \in \mathsf{co}(F)$ (E is *complete*) if $E \in \mathsf{ad}(F)$ and for each $a \in \mathsf{Args}$ defended by E in F, it holds that $a \in E$. We define that E is a *grounded* extension if $E \in \mathsf{co}(F)$ and for every $E' \in \mathsf{co}(F)$, $E' \not\subset E$. Moreover, E is an *ideal* extension if $E \in \mathsf{ad}(F)$, for every $E' \in \mathsf{pr}(F)$, $E \subseteq E'$, and E is maximal (wrt. set inclusion) with these two properties. Note that every AF F has a uniquely defined grounded extension E_{gr} [1] and a uniquely defined ideal extension E_{id} [8] and that $E_{\mathsf{gr}} \subseteq E_{\mathsf{id}} \subseteq \mathsf{pa}(F)$. An argument $a \in \mathsf{Args}$ is *accepted wrt. grounded semantics* (abbreviated ga), if a is contained in the grounded extension E_{gr}. Let $\mathsf{ga}(F)$ denote the set of all ga arguments and let $\mathsf{ia}(F)$ be the corresponding notion for ideal semantics.

2.2. Artificial Neural Networks for Abstract Argumentation

The purpose of an ML approach, such as an artificial neural network, is to "learn" from given data (i.e., the *training set*), in order to subsequently apply the acquired "knowledge" to previously unknown data (e.g., the *test set* during evaluation). The term *validation set* refers to a dataset that is essentially used as a trial test set. It is used during the training process of a neural network to check how well it would perform on unknown data at certain training stages. A typical problem for the application of ML is that of classification. In abstract argumentation, our objective is to decide whether an argument is acceptable or not under a given semantics and wrt. a given reasoning mode (credulous or skeptical). Therefore, we aim to classify an argument as *acceptable* or *not acceptable*. More precisely, we can train, e.g., a neural network using a set of AFs with labels (*accepted/not accepted*) for each argument.

There are a few works about the application of neural networks to decide the acceptability status of arguments [5,6,7]. A first work [5] makes use of so-called Graph Convolutional Networks (GCNs) as proposed by Kipf and Welling [9]. The authors consider the problem of credulous acceptance under preferred semantics. Malmqvist et al. [7] improve the GCN approach used by [5] by proposing a randomized training regime as well as a scheme to dynamically balance the training data. They consider both credulous and skeptical acceptance under preferred semantics. Craandijk and Bex [6] introduce an Argumentation Graph Neural Network (AGNN) which learns a message-passing algorithm. The authors apply their approach on the tasks of deciding credulous and skeptical acceptance under all four classical semantics (i.e., *complete*, *grounded*, *preferred*, and *stable* semantics). The latter work is the most promising one so far—the authors report almost perfect predictions in their evaluation.

3. Selection of Data for Abstract Argumentation

As this paper takes a data-centric perspective, we now investigate how the problem of data selection was approached in the previously mentioned works (Section 2.2) and we discuss these design choices. In this paper we consider only the problem $\mathsf{DS}_{\mathsf{pr}}$, but note that some of the works mentioned above also consider other problems.

Kuhlmann and Thimm [5] generate training sets of different sizes using the *probo Benchmark Suite*[3] [10] as well as *AFBenchGen*[4] [11], which include a total of six different graph generators. The authors create datasets of different sizes which contain between 30 and 600 AFs, each consisting of 100–400 arguments. A test set of 120 AFs is generated in the same manner. In addition, they use some benchmark data from the *International Competition on Computational Models of Argumentation*[5] (ICCMA) 2017 for testing purposes. During training, a fraction of the training set is used as a validation set. Malmqvist et al. [7] used a dataset of 900 AFs from ICCMA 2017, divided into training set (90%) and test set (10%). Again, part of the training data is used as a validation set. Craandijk and Bex [6] use the same generators as [5], i.e., those included in the Probo Benchmark Suite and AFBenchGen, and generate a test set consisting of 1000 AFs with exactly 25 arguments each. A fixed validation set is generated analogously. As a training set they create a total of one million AFs with $5 \leq |\mathsf{Args}| \leq 25$.

All works above use different datasets, which makes their results difficult to compare. Further, none of the papers poses the question what a suitable dataset should look like. To begin with, a test set should contain instances with various properties (different sizes, complexity, etc.), and at least some of them should be considered "challenging"—after all, the purpose of using a deep learning approach for abstract argumentation is to solve problems faster than with an exact approach. While at least some instances of the ICCMA dataset can be considered "challenging" (as they are competition benchmarks), the data generated by the Probo Benchmark Suite and AFBenchGen is a different matter. Craandijk and Bex [6] use a test set consisting of AFs that are made up of 25 elements each by using the aforementioned generators. Deciding whether an argument in such an AF is pa merely takes 0.0013 s on average using, e.g., the μ-toksia solver[6] [12]. In comparison, an AGNN takes 0.00003 s on average[7] for the same task[8]. Consequently, there is a reduction in computation time when using the neural network approach, nevertheless this advantage must be weighed against the disadvantage that exactness can never be guaranteed with a deep learning method. In the case of such small AFs, it might be more practical to use an exact approach, since the *absolute* difference in runtimes (0,00127 s on average absolute difference between the correct system μ-toksia and the approximate system AGNN) is very likely to be negligible for real-world applications. However, in the case of more complex AFs, the fast solving time of a neural network may be the more suitable solution in practice. This highlights that researchers should select their (test) data in a way that it represents those cases, in which a deep learning approach would grant an actual advantage in practical applications.

Furthermore, the ICCMA instances exhibit a property that makes them somewhat less "challenging" as well, as most arguments that are pa are also ga (see the $|\mathsf{ga}|$ and $|\mathsf{pa}|$ values of iccma-test in Table 1). Recall that the computational complexity of deciding $\mathsf{DS}_{\mathsf{pr}}$ is Π_P^2-complete [13] in general. However, there are certain easy cases where $\mathsf{DS}_{\mathsf{pr}}$ can be decided with much less effort. For example, arguments in the grounded ex-

[3]https://sourceforge.net/projects/probo/

[4]https://sourceforge.net/projects/afbenchgen/

[5]http://argumentationcompetition.org/

[6]System properties: 32 GB RAM, AMD® Ryzen 7 pro 5850u

[7]Note that we measured the time the model took to process the entire test set and afterwards divided it by the number of arguments in the test set (i.e., 25,000).

[8]System and training setup as described in Section 4.1

Table 1. Overview of all datasets used in the experiments. Let F be an arbirtrary AF. All columns including mean values additionally include the standard deviation.

Dataset	# AFs	# arguments ($= n$)	Mean # arguments per AF	Mean # attacks per AF	Mean $\|\mathsf{pa}(F)\|$	Mean $\|\mathsf{ia}(F)\|$	Mean $\|\mathsf{ga}(F)\|$
pbbg-train	100000	1898327	18.98 ±6.07	55.34 ±48.62	5.96 ±4.50	5.94 ±4.51	5.33 ±4.84
pbbg-test	1000	25000	25.00 ±0.0	84.97 ±60.96	7.07 ±5.70	7.02 ±5.74	6.14 ±6.15
kwt-train	1000	151000	151.00 ±0.0	6523.54 ±1728.29	69.44 ±23.47	69.32 ±23.82	16.31 ±34.45
kwt-test	1000	151000	151.00 ±0.0	6567.07 ±1734.52	69.12 ±23.63	68.99 ±23.99	15.90 ±34.16
iccma-test	450	292221	649.38 ±1398.99	52734.64 ±175454.90	70.23 ±333.51	64.233 ±324.47	59.08 ±313.15

tension are always pa (and the grounded extension can be computed in polynomial time) as well as arguments in the ideal extension [14] (where related problems are "only" Θ_2^P-complete), and arguments attacked by some admissible set are never pa (and deciding this is a problem in NP). We developed a graph generator (to which we refer by KWT in the following) that is tailored towards generating abstract argumentation frameworks that are particularly hard for tasks related to $\mathsf{DS}_{\mathsf{pr}}$ by avoiding (as much as possible) these easy cases. The graph generator KWT takes as parameters (among others) the total number of arguments, the number of arguments to be pa, the number of arguments to be contained in at least one preferred extension, and the number of preferred extensions. Arguments are associated to the different preferred extensions (at random, using some further parameters for randomisation) and each argument of each extension is attacked by at least one other argument (so we will have a small grounded extension in most cases). We performed some simple experiments to verify that these graphs are indeed relatively hard for problems related to $\mathsf{DS}_{\mathsf{pr}}$, but a careful analysis of this is part of ongoing work (and also not relevant for the work reported here, due to our results below). In our experiments we generated argumentation frameworks with 151 arguments, 60–90 pa arguments, 15–60 further arguments that are in at least one extension, and 100–200 preferred extensions. The graph generator[9] and the script[10] we used can be found online.

4. Experimental Analysis

We conduct two experiments. First we examine different test sets for the same trained model (an AGNN as presented by Craandijk and Bex [6]), and, as follow-up work, we use an alternative training set in order to achieve more accurate results for one of the more challenging test sets. We repeat these experiments with a different neural network model which was proposed in [5].

4.1. Experimental Setup

We generate AFs as in [6]. We create a test set and a validation set containing 1000 AFs each, with each AF consisting of exactly 25 arguments (i.e., $|\mathsf{Args}| = 25$), as well as a training set consisting of AFs with $5 \leq |\mathsf{Args}| \leq 25$. We denote these sets pbbg-test, pbbg-val, and pbbg-train, respectively. However, we only create 100,000 AFs for the training set, as opposed to Craandijk and Bex, who used 1 million AFs, as the results of a training with 100,000 AFs are expressive enough (see Section 4.2). See Table 1 for

[9]http://tweetyproject.org/r/?r=kwt_gen
[10]http://tweetyproject.org/r/?r=kwt_gen_ex

a statistical overview of the training and test set we generated. To actually generate the AFs and corresponding solutions, we use the AGNN framework[11], which also offers the option to train and evaluate other neural network models.

As a second dataset, we generate a total of 2200 KWT instances[12] as described in Section 3. We split this data into a training set (kwt-train) consisting of 1000 AFs, a test set (kwt-test), also consisting of 1000 instances, and a validation set (kwt-val) of 200 instances. Moreover, we use part of the ICCMA 2017 data as an additional test set[13] (iccma-test[14]). In total, we use 450 AFs[15] from groups A, B, and C, spanning all difficulty levels. Details on all these sets are displayed in Table 1 The corresponding solutions were generated using the solvers *Pyglaf* [15] and *μ-toksia* [12].

In our first experiment, we train[16] an AGNN as in [6] (the only difference being the smaller training set). Hence, for training we use pbbg-train, for validation pbbg-val, and for testing pbbg-test. We then test the trained model on the other two test sets (kwt-test and iccma-test) to examine whether it performs similarly well. As a follow-up experiment, we train an AGNN on kwt-train, in order to inspect if it is able to perform better on kwt-test. Again, we use all three test sets. Afterwards, we conduct the same two experiments on an implementation of the FM2 model [5] which is also included in the AGNN framework. In each experiment, the network is trained for 300 epochs, i.e., the training set is passed through the model 300 times.

In order to quantify our results, we use the Matthews Correlation Coefficient (MCC), as well as accuracy, True Positive Rate (TPR) and True Negative Rate (TNR). Let TP/FP and TN/FN denote *true/false positives* and *true/false negatives*, respectively, where the positive class are the pa arguments. Then we can define $\mathrm{MCC} = \frac{\mathsf{TP}\cdot\mathsf{TN}-\mathsf{FP}\cdot\mathsf{FN}}{\sqrt{(\mathsf{TP}+\mathsf{FP})\cdot(\mathsf{TP}+\mathsf{FN})\cdot(\mathsf{TN}+\mathsf{FP})\cdot(\mathsf{TN}+\mathsf{FN})}}$. Moreover, $\mathrm{TPR} = \mathsf{TP}/(\mathsf{TP}+\mathsf{FN})$, and $\mathrm{TNR} = \mathsf{TN}/(\mathsf{TN}+\mathsf{FP})$. Accuracy is defined as $\frac{\mathsf{TP}+\mathsf{TN}}{\mathsf{TP}+\mathsf{TN}+\mathsf{FP}+\mathsf{FN}}$, and precision as $\mathsf{TP}/\mathsf{TP}+\mathsf{FP}$.

4.2. *Results*

Reproducing the training procedure described by Craandijk and Bex [6] yields an MCC of 0.962. Although this value is slightly lower than the one provided in the original paper (0.997), it is still quite high. Besides, the small discrepancy can be explained by the fact that we used a smaller training set, and could be compensated by the use of more message passing steps during the testing phase. However, the MCC values regarding kwt-test and iccma-test are significantly lower (0.631 and 0.507, respectively). As Table 2 reveals, this is mostly due to a low TPR, meaning that arguments that are pa are not recognized as such in many cases. Training an AGNN with kwt-train results in an increased MCC for kwt-test (0.927). This shows that AGNNs are actually able to learn KWT data quite well if they are exposed to such data during training. The MCCs regarding the other two

[11] https://github.com/DennisCraandijk/DL-abstract-argumentation

[12] https://fernuni-hagen.sciebo.de/s/ZEmipULEN05FxxC

[13] Due to hardware limitations (in particular concerning the GPU memory), we could not use the ICCMA data for training.

[14] https://fernuni-hagen.sciebo.de/s/qjbSqMETOjb2JSY

[15] https://fernuni-hagen.sciebo.de/s/ZvNyWJ4af4ehEaB

[16] All computations were conducted on a computer equipped with an NVIDIA GeForce GTX 980 Ti GPU (6144 MB internal memory), an AMD® Ryzen 5 2600 processor, and 16 GB RAM.

Table 2. Overview of the experimental results.

Training Set	Test set	MCC	Accuracy	TPR	TNR	Precision
Model: AGNN						
pbbg-train	pbbg-test	0.962	0.985	0.964	0.993	0.981
	kwt-test	0.631	0.801	0.588	0.980	0.962
	iccma-test	0.507	0.847	0.446	0.974	0.845
kwt-train	pbbg-test	0.693	0.880	0.666	0.965	0.882
	kwt-test	0.927	0.962	1.000	0.930	0.924
	iccma-test	0.250	0.780	0.312	0.928	0.577
Model: FM2						
pbbg-train	pbbg-test	0.558	0.822	0.670	0.882	0.692
	kwt-test	0.359	0.642	0.220	0.998	0.989
	iccma-test	0.211	0.763	0.315	0.905	0.512
kwt-train	pbbg-test	0.078	0.719	0.030	0.991	0.568
	kwt-test	0.364	0.643	0.220	1.000	1.000
	iccma-test	0.055	0.761	0.016	0.996	0.583

test sets are lower than in the previous experiment (see Table 2). Thus, a training on KWT data alone is not practical either, and a future standard training set should probably contain instances of both KWT data and other datasets, such as pbbg-train. But these results suggest that the AGNN does no actually learn the concept (or an approximate concept) of skeptical acceptance wrt. preferred semantics, but rather particular properties of the benchmarks that correlate with it.

The results of our experiments with the FM2 network point in the same direction as the previously described ones. Training the network with pbbg-train results in a lower MCC wrt. kwt-test and iccma-test than wrt. pbbg-test (see the bottom half of Table 2). However, the FM2 model is clearly less accurate (in particular wrt. accepted arguments) and does not generalize very well. This problem becomes even more apparent when the network is trained with kwt-train. The resulting model does not significantly recognize accepted arguments from pbbg-test or iccma-test. Nevertheless, one should note that the problem of FM2 with imbalanced data has been recognized in the literature [5,7] and that other parameters could potentially improve the model's accuracy. Also, the fact that FM2 does not learn KWT data very well supports the hypothesis that the KWT instances are rather challenging—which was our goal when we developed the generator.

5. Discussion and Further Analysis

In the following, we look a little bit deeper into the actual approach that ML methods take to solve $\mathsf{DS}_{\mathsf{pr}}$, and how the data may bias learning. In particular, we have a look at the different graph types and how the accuracy of the existing approaches varies over these. Furthermore, we show that classical ML approaches perform already quite well when just considering very simple graph-theoretic features, suggesting that the complex deep learning approaches tend to simply learn these features as well.

Table 3. AGNN results on different subsets of iccma-test. Rows containing sets of AFs generated using methods which are *not* included in the generation process of pbbg-train and pbbg-test are marked in gray.

	Training Set									
	pbbg-train					kwt-train				
Test Set	MCC	Accuracy	TPR	TNR	Precision	MCC	Accuracy	TPR	TNR	Precision
ABA2AF	0.804	0.989	0.692	0.999	0.947	0.680	0.978	0.764	0.985	0.626
admbuster	−0.110	0.486	0.003	0.968	0.086	−0.454	0.329	0.000	0.657	0.000
BA	0.946	0.973	0.977	0.970	0.963	0.645	0.812	0.917	0.729	0.725
ER	−0.001	0.999	0.000	0.999	0.000	0.009	0.998	0.012	0.999	0.028
grd	0.690	0.970	0.600	0.992	0.827	0.265	0.935	0.226	0.979	0.391
Planning2AF	0.660	0.914	0.759	0.938	0.655	0.227	0.693	0.552	0.715	0.231
scc	0.135	0.998	0.127	0.999	0.151	0.016	0.989	0.049	0.990	0.008
sembuster	–	0.687	–	0.687	–	–	0.804	–	0.804	–
stb	0.463	0.926	0.329	0.988	0.742	0.172	0.904	0.097	0.987	0.438
traffic	0.789	0.931	0.723	0.989	0.947	0.541	0.858	0.499	0.958	0.768
WS	0.262	0.992	0.233	0.997	0.302	0.067	0.986	0.091	0.991	0.062

5.1. *Analysis of Graph Types*

As iccma-test overall seems to be hard to predict, we divide the dataset into multiple subsets. Each subset contains data from one distinct generator or source[17]. The sets grd, scc, and stb contain those instances produced by the generators of the *Probo Benchmark Suite*, i.e., the *GroundedGenerator*, *SccGenerator*, and *StableGenerator*, respectively. The sets BA, ER, and WS contain the instances generated using *AFBenchGen2*, i.e., they correspond to the *Barabasi-Albert*, *Erdős-Rényi*, and *Watts-Strogatz* approach, respectively. Consequently, these six graph "types" are also included in pbbg-train (and pbbg-test)[18]. The other five subsets, namely ABA2AF, admbuster, Planning2AF, sembuster, and traffic, conform to the remaining five generators (for a detailed explanation, see the official competition benchmark report [16]).

We considered the AGNN models trained on pbbg-train and kwt-train which we described in Section 4.2 and tested them on all individual subsets of iccma-test. Table 3 shows that the results vary widely. For instance wrt. pbbg-train, the BA instances are predicted quite accurately, with an MCC of 0.946. On the other hand, wrt. the ER subset, the model essentially just learned to classify all arguments as pna, which results in a TNR of 0.999, but a TPR of 0.000 (and an MCC of −0.001). Further, we can observe that both models tend to perform similarly. Although the model trained on pbbg-train overall performs superior to the one trained on kwt-train, they both perform best (wrt. MCC) on ABA2AF and BA, and worst on admbuster and ER. Moreover, it should be highlighted once again that the model trained on pbbg-train performs very poorly on ER—although the training set itself contained AFs generated using the Erdős-Rényi algorithm. Another interesting observation is that the AGNN performs very poorly on the admbuster set [17]. This is surprising because the admbuster graphs are acyclic and pa(F) coincides with the grounded extension in all its instances F. So, although "being in the grounded extension" is a sufficient condition for pa and easy to compute for classical algorithms, it is not learned by the deep learning approach.

5.2. *Impact of In-degree and Out-degree*

An advantage of deep learning approaches to ML is that the tedious task of feature engineering is taken over by the learning approach. The approaches [5,6,7] all learn the

[17]Overview of the AFs belonging to each subset: https://fernuni-hagen.sciebo.de/s/pySMMAfE7zEzn6i

[18]Note, however, that different parameters were used.

features used for classification implicitly when presented with the raw input data. While this is beneficial from the point of view of the engineer, it also makes the model hard to explain as it is not clear, how exactly recommendations are drawn from the input. This can also lead to models that make predictions on correlations rather than on causal relationships. A famous example of such a misbehavior comes from image recognition [18]. There, an ML approach was given a set of images classified either as wolf or dog (husky) and a classifier was trained to predict these two classes. But rather than identifying some intrinsic feature to distinguish wolves from dogs, the ML approach learned the feature of "snow in the background". In both the training and test set, most images of wolves had snow in the background, and this feature actually could be used very well for distinguishing wolves from dogs. However, it is clear that "snow in the background" is not a good feature to describe what makes a wolf a wolf.

While the above example is an extreme case of an ML approach focussing on correlation, we would like to analyse whether something similar happens in the case of predicting $\mathsf{DS}_{\mathsf{pr}}$ in abstract argumentation frameworks. For that we analysed the distributions of several graph-theoretic features, such as degrees, diameter, clustering coefficient, etc., of the graphs in our datasets, with respect to the differences between pa and pna arguments. We found out that the two features of in- and out-degree (i.e., the number of incoming and outgoing attacks per argument) already distinguish these two classes very well. Table 4, which comprises the mean degrees of pas and pnas wrt. all datasets used in this work, shows that both mean in-degree and mean out-degree are in all cases lower wrt. pa compared to pna. Regarding the out-degree, this is only a tendency, and there exist exceptions—for instance, we examined a dataset consisting of 1000 AFs (25 arguments each) generated by Probo's *GroundedGenerator*, where the mean out-degree per pa (14.83) was higher than the mean out-degree per pna (10.75). Regarding the in-degree, it is also possible to construct cases where the mean pa in-degree is higher than the mean pna in-degree. However, the clear tendency of lower in-degrees of pa is clearly visible. This is no coincidence in the data, but an intrinsic property of $\mathsf{DS}_{\mathsf{pr}}$: the number of the incident attacks in $\mathsf{pna}(F)$ is necessarily at least as large as the number of incident attacks in $\mathsf{pa}(F)$ (this is actually true for all conflict-free semantics).

Proposition 1. *For any* AF F, $|\mathsf{pa}(F)^{+}_{F}| + |\mathsf{pa}(F)^{-}_{F}| \leq |\mathsf{pna}(F)^{+}_{F}| + |\mathsf{pna}(F)^{-}_{F}|$.

Proof. Let $(a,b) \in R$. Since $\mathsf{pa}(F)$ is conflict-free, it follows that either

1. $a \in \mathsf{pa}(F)$, $b \in \mathsf{pna}(F)$,
2. $a \in \mathsf{pna}(F)$, $b \in \mathsf{pa}(F)$, or
3. $a \in \mathsf{pna}(F)$, $b \in \mathsf{pna}(F)$.

In the first two cases, both sets $\mathsf{pa}(F)$ and $\mathsf{pna}(F)$ are incident to the attack (a,b), while in case three, only $\mathsf{pna}(F)$ is incident to the attack (with both end points). Summing over all attacks, the claim follows. $\square$

The above observation becomes important, when we recall what "exact" problem the classifiers are learned upon. The actual problem is that, given an argumentation framework, predict pa/pna *simultaneously* for all arguments of the framework. As already discussed in Section 3, for many arguments the problem of deciding whether they are pa/pna is actually quite easy. With the additional information that low in-degrees are a

Table 4. Comparison of the mean number (and standard deviation) of outgoing and incoming attacks (i.e., in-degree and out-degree) wrt. the set of skeptically accepted arguments and the set of unaccepted arguments under preferred semantics.

Dataset	Mean out-degree per pa		Mean out-degree per pna		Mean in-degree per pa		Mean in-degree per pna	
pbbg-train	2.15	±2.02	2.71	±1.93	0.82	±0.98	3.29	±1.86
pbbg-test	2.06	±2.09	3.41	±2.38	0.84	±1.04	3.90	±2.25
kwt-train	15.92	±12.83	66.45	±12.40	13.64	±10.77	69.17	±12.01
kwt-test	15.97	±12.78	66.62	±12.33	13.88	±10.75	69.27	±11.92
iccma-test	20.69	±69.57	56.71	±133.83	12.11	±48.15	57.07	±134.16

Table 5. Results of a naive classifier which uses the constraint "$a_{\mathsf{in}} < \mathsf{mean}^{\mathsf{pa}}_{\mathsf{in}} \vee a_{\mathsf{out}} < \mathsf{mean}^{\mathsf{pa}}_{\mathsf{out}}$?" to classify arguments as pa or pna.

Training Set	Test Set	MCC	Accuracy	TPR	TNR	Precision
pbbg-train	pbbg-test	0.396	0.674	0.825	0.615	0.458
pbbg-train	iccma-test	0.364	0.737	0.827	0.726	0.268
kwt-train	kwt-test	0.739	0.868	0.768	0.951	0.930

very good indicator for having a pa argument, we can already classify many arguments correctly.

In the following, we will describe the results of some additional experiments to explore the impact of in-degree and out-degree in classification tasks. For that, we simplified the instances of our datasets to a tabular format, in which each row only contains an argument ID, the argument's in-degree, out-degree, and label (pa or pna). So, the actual argumentation frameworks are not given to the learning algorithm as input! We then train several ML algorithms on this data and measure the resulting MCC, accuracy, TPR and TNR. To begin with, we define a naive classifier whose "training" consists of calculating the mean pa in-degree ($\mathsf{mean}^{\mathsf{pa}}_{\mathsf{in}}$) and out-degree ($\mathsf{mean}^{\mathsf{pa}}_{\mathsf{out}}$) of the training set. The classification process then simply consists of checking whether the in-degree or the out-degree of a given argument a (a_{in} and a_{out}) is smaller than $\mathsf{mean}^{\mathsf{pa}}_{\mathsf{in}}$or $\mathsf{mean}^{\mathsf{pa}}_{\mathsf{out}}$, respectively: if one of them is indeed smaller, the argument is classified as pa, if this is not the case, it is classified as pna. Although the results are, as one would expect, far from perfect, Table 5 shows that in-degree and out-degree alone are quite good indicators of the acceptability status of an argument. In particular, accepted arguments are predicted surprisingly accurately, with a TPR between 0.768 and 0.825.

Furthermore, we train a number of "classical" ML models. Again, we use the formatted datasets which only contain the in-degree and out-degree of each node, but no further information about the graph structure. To be precise, we consider the following methods [19]: k-Nearest Neighbors (KNN) (with $k = 5$), Naive Bayes (NB), Decision Tree (DT), and Random Forest (RF)[19]. The results, displayed in Table 6, show that the previously described results yielded by the naive classifier can generally be improved by using a more sophisticated approach. In particular, the results on kwt-test are quite accurate, with an MCC of 0.924 when using an RF. While DT and RF exhibit the best performance wrt. MCC, one should notice that NB offers the most "balanced" results wrt. TPR and TNR, in particular regarding pbbg-train. Besides, it is noticeable that DT and RF feature very similar, or even identical results in all cases. This is most likely due

[19]We used the implementations and default parameters provided by https://scikit-learn.org/stable/.

Table 6. Overview of test results wrt. multiple training and test sets as well as four different ML methods.

Training Set	Test Set	Approach	MCC	Accuracy	TPR	TNR	Precision
pbbg-train	pbbg-test	KNN	0.567	0.834	0.576	0.936	0.779
		NB	0.514	0.761	0.831	0.733	0.551
		DT	**0.630**	0.857	0.615	0.952	0.836
		RF	**0.630**	0.857	0.615	0.952	0.836
pbbg-train	iccma-test	KNN	0.191	0.832	0.309	0.895	0.264
		NB	0.389	0.775	0.793	0.773	0.298
		DT	**0.399**	0.911	0.200	0.997	0.895
		RF	**0.399**	0.911	0.200	0.997	0.895
kwt-train	kwt-test	KNN	0.884	0.942	0.952	0.934	0.924
		NB	0.868	0.934	0.935	0.934	0.923
		DT	0.923	0.960	0.998	0.929	0.922
		RF	**0.924**	0.961	0.998	0.929	0.922

to the fact that the datasets only possess two features, which might lead to the decision trees in the random forest to be similar (or identical) to a single decision tree.

Overall, our results suggest that classical ML techniques trained only on the features *in-degree* and *out-degree* might not be accurate enough for practical applications, yet they also show that these two simple features are still very strong indicators for the acceptance status of an argument. Therefore, the question arises whether (and if so, in which manner) such a dominant feature influences a neural network's training process.

6. Conclusion

The objective of this work was to shed some light on the importance of data selection—a common problem in ML research, but less common in knowledge representation and reasoning. To successfully bridge the gap between both fields, we need to take different perspectives. Our experiments showed that a training set consisting of only rather small AFs (5–25 arguments) offers little room for variability. Thus, a neural network cannot solve more complex cases as accurately. However, we also saw that a neural network (in particular the AGNN architecture [6]) is in fact able to learn more complex features if it is exposed to them during training. We also discussed the question whether the application of ML techniques is even useful for small and simple AFs. Such cases could be solved by an exact approach in a reasonable amount of time without any losses in terms of accuracy. Further, there are other "simple" cases which should be taken into consideration. For example, when regarding $\mathsf{DS}_{\mathsf{pr}}$, a dataset should contain a significant number of arguments that are pa but not ga, since the grounded extension can be computed in polynomial time.

Furthermore, as a subject of future work, we need to take a closer look at graph-theoretical aspects. For instance, a dataset could benefit from AFs which include accepted arguments that have similar properties (such as in-/out-degree) as arguments that are not accepted. Therefore, a neural network would need to learn less superficial features to learn the acceptability status of arguments. Another topic of future work is to examine other semantics and tasks more closely, as we only focused on $\mathsf{DS}_{\mathsf{pr}}$. Yet another aspect that could be explored is to use ML techniques to guide sound and complete solvers, in order to accelerate their execution times. Moreover, the compilation of a standard dataset that includes all of the above perspectives and aspects could facilitate future research in this area.

Acknowledgements

We would like to thank Lars Malmqvist for providing a large sum of solutions for the ICCMA 2017 benchmark instances. We would also like to thank Mark Mints for his active support concerning various GPU issues. The research reported here was supported by the Deutsche Forschungsgemeinschaft under grant 375588274.

References

[1] Dung PM. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artificial Intelligence. 1995;77(2):321-58.

[2] Dvorák W, Dunne PE. Computational problems in formal argumentation and their complexity. Handbook of formal argumentation. 2018:631-87.

[3] Cerutti F, Gaggl SA, Thimm M, Wallner JP. Foundations of Implementations for Formal Argumentation. In: Handbook of Formal Argumentation. College Publications; 2018. p. 2623-705.

[4] Lagniez JM, Lonca E, Mailly JG, Rossit J. Design and Results of ICCMA 2021. arXiv preprint arXiv:210908884. 2021.

[5] Kuhlmann I, Thimm M. Using graph convolutional networks for approximate reasoning with abstract argumentation frameworks: A feasibility study. In: International Conference on Scalable Uncertainty Management. Springer; 2019. p. 24-37.

[6] Craandijk D, Bex F. Deep learning for abstract argumentation semantics. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence; 2020. p. 1667-73.

[7] Malmqvist L, Yuan T, Nightingale P, Manandhar S. Determining the Acceptability of Abstract Arguments with Graph Convolutional Networks. In: SAFA@ COMMA; 2020. p. 47-56.

[8] Dung PM, Mancarella P, Toni F. Computing ideal sceptical argumentation. Artificial Intelligence. 2007;171(10-15):642-74.

[9] Kipf TN, Welling M. Semi-Supervised Classification with Graph Convolutional Networks. In: International Conference on Learning Representations (ICLR); 2017. .

[10] Cerutti F, Oren N, Strass H, Thimm M, Vallati M. A Benchmark Framework for a Computational Argumentation Competition. In: COMMA; 2014. p. 459-60.

[11] Cerutti F, Giacomin M, Vallati M. Generating Challenging Benchmark AFs. COMMA. 2014;14:457-8.

[12] Niskanen A, Järvisalo M. μ-toksia: An efficient abstract argumentation reasoner. In: Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning; 2020. p. 800-4.

[13] Dunne PE, Bench-Capon TJM. Coherence in finite argument systems. Artificial Intelligence. 2002;141(1/2):187-203.

[14] Dunne PE. The computational complexity of ideal semantics. Artificial Intelligence. 2009 December;173(18):1559-91.

[15] Alviano M. The pyglaf argumentation reasoner. In: Technical Communications of the 33rd International Conference on Logic Programming; 2018. p. 2:1-2:3.

[16] Gaggl SA, Linsbichler T, Maratea M, Woltran S. Design and results of the Second International Competition on Computational Models of Argumentation. Artificial Intelligence. 2020;279:103193.

[17] Caminada M, Podlaszewski M. AdmBuster: a benchmark example for (strong) admissibility; 2017. The Second International Competition on Computational Models of Argumentation (ICCMA'17).

[18] Ribeiro MT, Singh S, Guestrin C. " Why should i trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining; 2016. p. 1135-44.

[19] Alpaydin E. Introduction to machine learning. MIT press; 2020.