FernUniversität in Hagen
Fakultät für Mathematik und Informatik

Artificial Intelligence Group

Evaluierung von Klassifikationsalgorithmen anhand eines Anwendungsfalls basierend auf einem abstrakten Argumentationsframework

Masterarbeit

zur Erlangung des Grades eines Master of Science (M.Sc.)
im Studiengang Praktische Informatik

vorgelegt von
Marvin Peters

Erstgutachter: Prof. Dr. Matthias Thimm
Artificial Intelligence Group

Betreuer: Lars Bengel
Artificial Intelligence Group

FernUniversität in Hagen

Selbstständigkeitserklärung

Name:	Marvin Feyyaz Peters
Matrikel-Nr.:	4569440
Fach:	Masterstudiengang Praktische Informatik
Modul:	Masterarbeit
Thema:	Evaluierung von Klassifikationsalgorithmen anhand eines Anwendungsfalls basierend auf einem abstrakten Argumentationsframework

Ich erkläre, dass ich die Masterarbeit selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für enthaltene Zeichnungen, Skizzen oder graphische Darstellungen. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Arbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate geprüft werden kann und ausschließlich für Prüfungszwecke gespeichert wird.

Datum: 18.11.2022 **Unterschrift:** Marvin Peters

Zusammenfassung

Argumentation gehört zu einem der wichtigsten Themen in der Wissenspräsentationsforschung, was unter anderem an dem starken Interesse von Multiagentensystemen (MAS) sowie weiteren Anwendungsfällen liegt [21]. Für diese stellt Argumentation eine grundlegende Basis für die Gestaltung von Kommunikationen und Verhandlungen dar. Die Argumente einer Argumentation werden mit Hilfe eines Argumentationsframeworks dargestellt [6]. In einem solchen Argumentationsframework können die einzelnen Argumente sich gegenseitig verteidigen oder angreifen, um so eine Diskussion zu führen und final eine Schlussfolgerung zu einem bestimmten Thema zu finden [21]. Eine interessante Frage, die sich hier nach gemeinsamer Erstanalyse mit dem Lehrgebiet stellt ist, inwieweit der Ausgang einer Diskussion vorhersagbar ist oder nicht. Mit Klassifikationsalgorithmen, die als Datengrundlage ein abstraktes Argumentationsframework verwenden, soll genau dieser Ansatz analysiert und evaluiert werden. Die zentrale Forschungsfrage dieser Masterarbeit lautet, ist das Label eines Zielarguments für ein gegebenes Argumentationsframework und einer Teilmenge gelabelter Argumente durch Klassifikationsalgorithmen vorhersagbar oder nicht ?

Abstract

Argumentation is one of the most important topics in knowledge representation research, which is partly due to the strong interest in multi-agent systems (MAS) and other use cases. For them, argumentation is a fundamental basis for the design of communications and negotiations. The arguments of an argumentation are visualized with the help of argumentation frameworks. In such an argumentation framework, the individual arguments can defend or attack each other in order to lead a discussion and finally find a conclusion on a certain topic. An interesting question that arises here after a first analysis (together with the academic chair) is to what extent the outcome of a discussion can be predicted or not. With classification algorithms, which use an abstract argumentation framework as a data basis, exactly this approach is to be analyzed and evaluated. Focus of this master thesis is the following research question, is the label of a target argument for a given argumentation framework and a subset of labeled arguments by classification algorithms predictable or not ?

Inhaltsverzeichnis

1 Einführung

Im ersten Abschnitt dieser Masterarbeit wird kurz auf die Motivation dieser Arbeit eingegangen, anschließend wird das zentrale Szenario inklusive Forschungsfrage vorgestellt.

1.1 Motivation

Die Argumentation ist eine der wichtigsten Formen des Commonsense Reasoning (menschliche Schlussfolgerung trotz unvollständiger Informationen). So ist eine Argumentation zu einem Thema möglich, selbst wenn nicht alle Hintergrundinformationen bekannt sind [21]. Sie ermöglicht somit nichtmonotone Inferenzen, was eine Anfechtbarkeit von Schlussfolgerungen bedeutet und in diesem Zusammenhang z.B. für Multiagentensysteme sehr wichtig ist [21]. Eine der bekanntesten Formalismen zur Darstellung von Argumenten ist hier das abstrakte Argumentationsframework von Dung [9]. Ein Argumentationsframework wird als gerichteter Graph dargestellt [21], wobei die Knoten die Argumente repräsentieren und Angriffe als Kanten dargestellt werden.

Beispiel 1 *In Abbildung 1 sind die sich gegenseitig attackierenden Argumente **A** und **B** zu sehen. Der Angriff (**A**,**B**) wird hierbei als Kante **A** $\rightarrow$ **B** abgebildet (der Angriff kann natürlich auch umgekehrt erfolgen). Es ist klar, dass beide Argumente sich gegenseitig widersprechen, da elektrische Autos keinen Treibstoff benötigen und somit emissionsfreies Autofahren ermöglichen. Des Weiteren ist ersichtlich, dass das Argument **A** das Argument **C** verteidigt.*

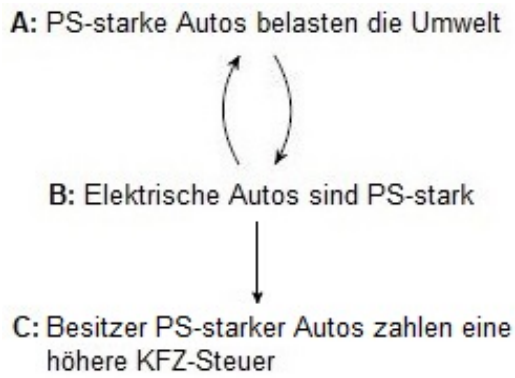

Abbildung 1: Hier ein beispielhaftes Argumentationsszenario bezüglich Autos.

Grundlegend bei einem solchen Argumentationsframework ist es herauszufinden, welche Argumente akzeptabel sind (und welche nicht). Die Mengen von akzeptablen Argumenten werden Extensionen genannt [21]. Hier gibt es unterschiedliche

Klassen von Extensionen, bei denen die Menge von Argumenten nach unterschiedlichen Kriterien bestimmt werden. Diese Klassen (Definitionen in Abschnitt 2.1) der jeweiligen Extensionen werden als Semantik eines abstrakten Argumentationsframeworks bezeichnet. Fokus für das Szenario sowie der Forschungsfrage werden die zulässige und vollständige Semantik sein. Neben dem Ansatz mit Extensionen gibt es auch Labelingfunktionen [6]. Der Labeling-Ansatz hat einen höheren Informationsgehalt als Extensionen, da jedes Argument innerhalb einer Semantik mit einem Label versehen wird. Die Extensionen können somit einfach abgelesen werden. Folgende Label gibt es [6]:

- Undec – Unsicher beim Einschätzen des Labels.
- In – Akzeptiertes Argument.
- Out – Nicht akzeptiertes Argument.

Ein zentrales Beispiel dieser Masterarbeit soll folgendes abstraktes Argumentationsframework **AF** darstellen (siehe Abbildung 2). Es repräsentiert eine Diskussion zum in Deutschland beschlossenen Ausstieg aus der Atomenergie, wobei hier das Argument **F** die Kernaussage darstellt (abgewandelt übernommen aus [21]).

Beispiel 2 *Hier das Argumentationsframework **AF** mit seinen Argumenten und Attacken (siehe Abbildung 2) :*

- *({**A, B, C, D, E, F, G**},*
- *{(**B,A**), (**B,C**), (**C,B**), (**B,F**), (**E,F**), (**D,E**), (**E,G**), (**G,E**)})*

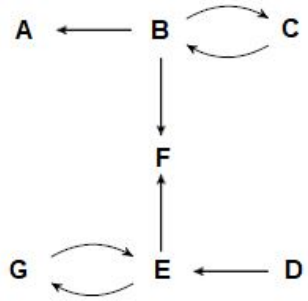

Abbildung 2: Das exemplarische Argumentationsframework **AF** (abgewandelt übernommen aus [21]).

*In Abbildung 2 sind die einzelnen Argumente sowie die Attacken zu sehen. Es ist zum Beispiel gut ersichtlich, dass das Argument **D** nie angegriffen wird und somit immer zu akzeptieren ist. Im Umkehrschluss folgt daraus, dass das Argument **E** nie akzeptiert wird.*

*Für das Argumentationsframework **AF** werden zur Veranschaulichung Pro- und Kontra-Argumente erstellt. Im Detail wird das Argumentationsframework mit den Argumenten **A** bis **G** wie folgt interpretiert:*

- ***A**: Die bestehenden Atomkraftwerke produzieren günstigen Strom.*
- ***B**: Betreiber der Atomkraftwerke bestimmen den Strompreis (Dominanz auf Grund Energiehoheit).*
- ***C**: Stromerzeugung aus erneuerbaren Energien wird günstiger und wirtschafticher.*
- ***D**: Keine grüne Energie aufgrund ungeklärter Frage der Atommüll-Endlager.*
- ***E**: Atomenergie ist eine grüne und sichere Energie.*
- ***F**: Atomausstieg in Deutschland war sinnvoll.*
- ***G**: Gefahr von Atomunfällen durch Naturkatatstrophen oder Anschlägen.*

*Das Argument **F** ist relevanter als die anderen Argumente, da die Kernaussage, der Atomausstieg in Deutschland war sinnvoll, gleichzeitig den Ausgang der Diskussion bestimmt.*

In Beispiel 2 sind alle Argumente des Argumentationsframeworks vorhanden, das Label von Argument **F** zu bestimmen ist einfach. Bei umfangreicheren Argumentationssystemen ist dies jedoch nicht ohne weiteres möglich. Hier muss mit der Unterstützung eines Programms das Label eines bestimmten Zielarguments ermittelt werden. Genau hier beginnt die Motivation dieser Masterarbeit. Mit einem Data-Mining-Tool sollen Klassifikationsalgorithmen, die als Datengrundlage die zulässigen oder vollständigen Labelings eines Argumentationsframework verwenden, hinsichtlich der Vorhersagbarkeit eines Arguments analysiert und evaluiert werden. Sie sollen zeigen, wie gut Sie das Label eines Zielarguments vorhersagen können. Ferner wird untersucht, wie es sich mit der Vorhersagbarkeit bei abnehmender Vollständigkeit (siehe Abschnitt 4) der Argumente verhält. Diese bezieht sich hierbei auf die Testdaten (Anwendungsphase) der ungesehenen Teil-Labelings, was mehrere Labels ohne Werte zur Folge hat.

1.2 Szenario

Zur Erläuterung der Zielstellung und des Szenarios ist das Label des Arguments **F** als das Ergebnis einer Diskussion (Beispiel 2) zu interpretieren. Das Label In bedeutet somit die Diskussion ist gewonnen, Out sie ist verloren oder die Diskussion ist auf Grund der aktuell vorliegenden Argumente nicht klar entscheidbar (Undec). Für die Beschreibung des formalen Szenarios ist **Arg** eine Menge von Argumenten eines abstrakten Argumentationsframeworks **AF**, wo $\rightarrow \subseteq$ **Arg** $\times$ **Arg** eine binäre Angriffsrelation zwischen zwei Argumenten genannt wird ([9], siehe Abschitt 2.1).

Das Szenario ist unterteilt in eine Trainings- sowie Anwendungsphase.

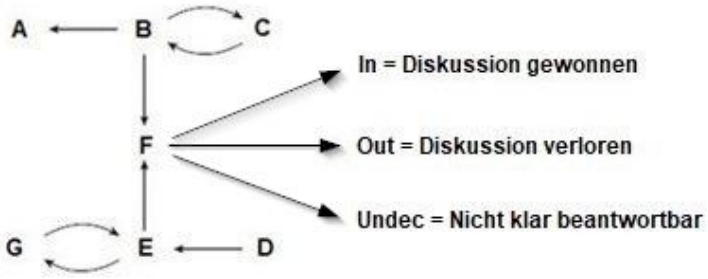

Abbildung 3: Der Wert des Labels von Argument **F** entscheidet über den Ausgang einer Diskussion.

1.2.1 Trainingsphase

Eingabe: $AF\,(Arg, \rightarrow)$, $eine\,Menge\,Labelings\,L_1, \ldots, L_n$, $A \in Arg$, $lab_A \in \{in, out, undec\}$

Ausgabe: Ist ein trainierter Klassifikator k, dass ein Labeling L anhand des Labels lab_A klassifiziert.

Methodik: Der Klassifikationsalgorithmus ist auf Basis der zulässigen Labelings zu trainieren.

1.2.2 Anwendungsphase

Eingabe: $lab_B1, \ldots., lab_Bn \in \{in, out, undec\}$

Ausgabe: $lab_A \in \{in, out, undec\}$, so dass es ein zulässiges Labelling L gibt mit $L(B1) = lab_B1, \ldots, L(Bn) = lab_Bn$, $L(A) = lab_A$

Methodik: Das zuvor genannte Klassifikationsproblem soll gelöst werden, indem der trainierte Klassifikator k anhand ungesehener Teil-Labelings die Labels des Zielarguments vorhersagt.

1.3 Zielstellung

Zentrale Forschungsfrage

Ist das Label eines Arguments für ein gegebenes Argumentationsframework und einer Teilmenge gelabelter Argumente durch Klassifikationsalgorithmen vorhersagbar ?

Hierbei ist wie folgend beschrieben vorzugehen. Es sind mehrere Argumentationsframeworks **AF** zu erstellen oder aus dem ICCMA'19 Datensatz [4] auszusu-

chen. Diese sollten genug zulässige oder vollständige Labelings haben, damit eine ausreichende Datengrundlage für das Trainieren der Klassifikationsalgorithmen vorhanden ist. Bei jedem Argumentationsframework ist ein Zielargument zu bestimmen (wie in Abbildung 3).

In der Trainingsphase werden die jeweiligen Klassifikatoren (siehe Abschnitt 2.2) mit den Labelings trainiert. Hierbei gilt es zu berücksichtigen, dass ein kleiner Prozentsatz der Labelings für die Anwendungsphase zurückzulegen ist und somit nicht zum Trainieren des Klassifikators verwendet werden darf.

In der Anwendungsphase soll anhand der zurückgelegten (ungesehenen) Labelings das Label des Zielarguments vorausgesagt werden, welches gleichzeitig als Klasse verwendet wird. Es wird somit drei mögliche Klassen geben (**Undec**, **In**, **Out** - siehe auch Abbildung 3). Die Klassifikationsgüte ist anhand der in Abschnitt 2.3 genannten Maße zu bestimmen. In der Anwendungsphase soll außerdem untersucht werden, wie es sich mit der Vorhersagbarkeit bei einer abnehmenden Vollständigkeit der Argumente (und der zugehörigen Labels) verhält (siehe Abschnitt 4). Somit soll u.a. evaluiert werden, ab wie viel Prozent vorhandener Argumente sich valide Ergebnisse vorhersagen lassen.

Beispiel 3 *Beispielhaft könnte man die Forschungsfrage in Abbildung 3 auf das Argument **F** in Beispiel 2 anwenden. Dies bedeutet, es wird versucht den Ausgang der Diskussion vorherzusehen, ob der Atomausstieg in Deutschland sinnvoll war oder nicht.*

Es wird eine Labelingfunktion ausgewählt, deren zulässige Labelings die Datengrundlage bilden. Die Labelings sind in einen Trainings- (ca. 70 %) sowie in einen Testdatensatz (ca. 30 %) aufzuteilen. Anschließend sind die verschiedenen Klassifikationsalgorithmen mit dem Trainingssatz zu trainieren. Im Anschluss wird der Testdatensatz auf den trainierten Klassifikationsalgorithmus angewendet.

2 Wissenschaftlicher Hintergrund

Im zweiten Abschnitt werden alle grundlegenden Begriffe, die für den weiteren Verlauf der Masterarbeit wichtig sind, definiert. Neben Begriffen hinsichtlich der abstrakten Argumentationsframeworks sind auch die zu evaluierenden Klassifikationsalgorithmen beschrieben. Zum Schluss wird auf die Kennzahlen zur Bestimmung der Vorhersagbarkeit, den Klassifikationsgüten, eingegangen.

2.1 Abstrakte Argumentationsframeworks, Extensionen und Labelings

Definition 1 *Dung [9] definiert ein abstraktes Argumentationsframework (**AF**) als ein Paar **(Arg,→)**:*

- ***Arg** ist eine Menge von Argumenten.*

- $\rightarrow \subseteq$ ***Arg*** $\times$ ***Arg*** *ist eine binäre Relation zwischen Argumenten, welche Angriffsrelation genannt wird.*

Wie bereits in Abschnitt 1.1 geschildert, wird ein Argumentationsframework als gerichteter Graph dargestellt, wobei die Knoten die Argumente repräsentieren und Angriffe als Kanten dargestellt werden.

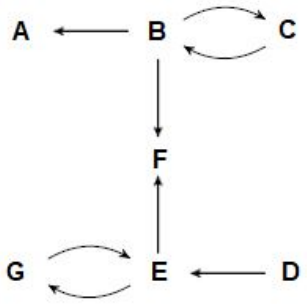

Abbildung 4: Zur besseren Übersichtlichkeit nochmals das Argumentationsframework **AF** aus Beispiel 2.

2.1.1 Extensionsbasierte Semantiken

Die Mengen von akzeptablen Argumenten, welche Extensionen genannt werden [21], lassen sich in unterschiedliche Klassen zusammenfassen. Diese Klassen der jeweiligen Extensionen werden als Semantik eines abstrakten Argumentationsframeworks bezeichnet. Eine grundlegende Forderung für alle Extensionen ist die Konfliktfreiheit. Konfliktfreiheit besteht, wenn die Argumente einer Extension keine Konflikte untereinander haben [9]. Darauf basieren alle Semantiken.

Definition 2 *AF = (Arg,*$\rightarrow$*) ist als Argumentationsframework definiert. Eine Menge von Argumenten* $W \subseteq Arg$ *wird* ***konfliktfrei*** *genannt, wenn für alle Argumente* $a, b \in W$: $(a, b) \notin \rightarrow$ *gilt [9].*

Beispiel 4 *In Abbildung 4 wäre dies z.B. die Extension {**A**,**C**}. Das Argument **C** greift Argument **B** an, welches wiederum das Argument **A** angreift. Die beiden Argumente **A** und **C** stehen somit in keinem Konflikt zueinander. Folglich wäre die Extension {**A**,**B**,**C**} nicht konfliktfrei, da **B** die beiden anderen Argumente angreift.*

Ein weiterer Begriff im Zusammenhang mit abstrakten Argumentationsframeworks ist die Verteidigung.

Definition 3 *Eine Menge von Argumenten* $W \subseteq Arg$ ***verteidigt*** *ein Argument* $a \in Arg$ *genau dann, wenn jedes Element* $b \in Arg$*, das a angreift, von W angegriffen wird [9].*

Beispiel 5 *In Abbildung 4 verteidigt die Menge W = {**C**,**D**} das Argument **F**.*

Im Rahmen dieser Masterarbeit sind die folgend genannten Semantiken inklusive Definitionen von Interesse [9]. Bei allen Definitionen wird auf das Argumentationsframework AF = (Arg,→) zurückgegriffen.

Definition 4 *Eine Menge von Argumenten $W \subseteq Arg$ wird **zulässig (admissible)** genannt, wenn W konfliktfrei ist und W jedes Argument $a \in W$ verteidigt [9].*

Dies bedeutet, dass jedes Argument einer zulässigen Extension sich selbst verteidigen kann oder von einem anderen Argument (der zulässigen Extension) verteidigt wird.

Beispiel 6 *In Abbildung 4 wären dies z.B. die Extension {**A,C**}. **A** wird hier zwar von **B** angegriffen, kann sich jedoch durch den Angriff **C** → **B** verteidigen. Folglich kann sich **C** selbst gegen **B** verteidigen. Folgende weitere zulässige Extensionen (Auszug) aus Abbildung 4 wären möglich:*

- *{**B**}, {**C**}, {**A,C,D**}, {**B,G**}, {**C,G**}, {**D,G**}, {**A,C,D,F,G**}.*

Definition 5 *Eine Menge von Argumenten $W \subseteq Arg$ wird **vollständig (complete)** genannt, wenn W zulässig ist und jedes Argument $a \in Arg$, dass von W verteidigt wird, auch in W ist [9].*

Dies bedeutet, es müssen alle Argumente (Gesamtmenge) in der Extension enthalten sein, die verteidigt werden.

Beispiel 7 *In Abbildung 4 wäre dies z.B. die Extension {**A,C,D,F,G**}. Jedes einzelnes Argument dieser Extension wird verteidigt, ferner werden alle weiteren zulässigen Argumente dieser Extension genannt. Es gibt somit folgende vollständige Extensionen:*

- *{**A,C,D,F,G**}, {**D,G**}, {**B,D,G**}*

Definition 6 *Eine Menge von Argumenten $W \subseteq Arg$ wird **bevorzugt (preferred)** genannt, wenn W vollständig ist und W die maximal mögliche Menge (bezogen auf die Teilmengenbeziehung) an Argumenten besitzt [9].*

Wie in der Definition beschrieben, muss eine bevorzugte Extension die größtmögliche Menge an Argumenten beinhalten.

Beispiel 8 *Die vollständige Extension {**A,C,D,F,G**} ist zugleich eine bevorzugte Extension, da Sie die maximal mögliche Menge an Argumenten besitzt. In Abbildung 4 gibt es somit folgende bevorzugte Extensionen:*

- *{**A,C,D,F,G**}, {**B,D,G**}.*

Definition 7 *Eine Menge von Argumenten $W \subseteq Arg$ wird **stabil (stable)** genannt, wenn W vollständig ist und gleichzeitig gilt, dass jedes Argument $a \in Arg \setminus W$ von W angegriffen wird [9].*

Sie setzt somit voraus, dass alle Argumente die nicht Bestandteil der stabilen Extension sind von dieser auch angegriffen werden. Es ist durchaus möglich, dass ein Argumentationsframework keine stabile Extension besitzt.

Beispiel 9 *Die vollständige Extension {**A,C,D,F,G**} (Abbildung 4) ist gleichzeitig auch eine stabile Extension. In Abbildung 4 gibt es somit folgende stabile Extensionen:*

- *{**A,C,D,F,G**}, {**B,D,G**}.*

2.1.2 Labelingbasierte Semantiken

Bei dem Ansatz mit Extensionen ist der Informationsgehalt sehr gering, es ist nicht ersichtlich warum z.B. ein Argument abgelehnt wurde. Hierzu wurden die Labelingfunktionen in die Argumentationstheorie eingeführt [6]. Hier nochmals die bereits eingeführten Labels:

- Undec – Unsicher beim Einschätzen des Labels.
- In – Akzeptiertes Argument.
- Out – Nicht akzeptiertes Argument.

Ein Vorteil der Labelings gegenüber Extensionen ist der zusätzliche Status **Undec**. Er ermöglicht es, nicht akzeptierte Argumente in zwei Gruppen aufzuteilen. Zum einen in das Label Undec, welcher theoretisch noch akzeptierbar mit den anderen Argumenten ist, zum anderen in das Label Out, das eine aktive Ablehnung bedeutet.

Definition 8 *Unter Verwendung des Argumentationsframeworks AF = (Arg,→) ist eine Labelingfunktion definiert als eine Funktion $l : Arg \rightarrow \{in, out, undec\}$ [21].*

Zu einem Labeling l existiert somit immer genaue eine Extension, die über die Funktion In(l) dargestellt werden kann. Da in der Masterarbeit die zulässigen bzw. die vollständigen Labelings die Datengrundlage bilden, wird nun auf die formale Definition eingegangen. Vorab die Definition der Konfliktfreiheit bezogen auf Labelings.

Definition 9 *Unter Verwendung des Argumentationsframeworks AF = (Arg,→) gelten für ein konfliktfreies Labeling l für alle $a \in Arg$ folgende Bedingungen [6]:*

- *Wenn a das Label In hat, wird a von keinem Argument mit dem Label In angegriffen.*
- *Wenn a das Label Out hat, wird a von einem Argument mit dem Label In angegriffen.*
- *Ansonsten hat a das Label Undec.*

Ein zulässiges Labeling, welches konfliktfrei sein muss, ist wie folgt definiert.

Definition 10 *Unter Verwendung des Argumentationsframeworks AF = (Arg,→) gelten für ein zulässiges Labeling l für alle $a \in Arg$ folgende Bedingungen [6]:*

- *Wenn a das Label In hat, haben alle attackierende Argumente (die a attackieren) das Label Out.*
- *Wenn a das Label Out hat, wird a von einem Argument mit dem Label In angegriffen.*
- *Ansonsten hat a den Status Undec.*

In Beispiel 10 werden die zulässigen Labelings $l1$ sowie $l2$ angezeigt.

Beispiel 10 *Die in Beispiel 6 genannten zulässige Extension {**A,C,D,F,G**} sowie {**A,C,D**} lassen sich exemplarisch wie folgt als Labeling-Funktionen tabellarisch darstellen (Tabelle 1):*

	A	**B**	**C**	**D**	**E**	**F**	**G**	**In(l)**	**Out(l)**	**Undec(l)**
l_1	In	Out	In	In	Out	In	In	{A,C,D,F,G}	{B,E}	$\emptyset$
l_2	In	Out	In	In	Out	Undec	Undec	{A,C,D}	{B,E}	{F,G}

Tabelle 1: Teilauszug der in Beispiel 6 dargestellen zulässigen Extension als Labeling l_1 und l_2.

Die Labelingfunktionen (siehe Tabelle 1) geben somit schnell Auskunft darüber, welches Argument akzeptiert wurde und welches nicht.

Aus Tabelle 1 ist somit schnell ersichtlich, dass die Argumente **B** und **E** nicht akzeptiert wurden. Ein typisches Szenario, wo beide Argumente Undec sein können, sind die Argumente **F** und **G**.

Ein vollständiges Labeling, welches zulässig sein muss, ist wie folgt definiert.

Definition 11 *Unter Verwendung des Argumentationsframeworks AF = (Arg,→) gelten für ein vollständiges Labeling l für alle $a \in Arg$ folgende Bedingungen [6]:*

- *Wenn a das Label In hat, haben alle attackierende Argumente (die a attackieren) das Label Out.*
- *Wenn a das Label Out hat, wird a von einem Argument mit dem Label In angegriffen.*
- *Wenn a das Label Undec hat, wird a von mindestens einem Argument mit dem Label Undec angegriffen. Trifft dies zu, darf es außerdem kein Argument mit dem Label In geben, das a angreift.*

Das zulässige Labeling $l1$ aus dem Beispiel 10 ist ebenfalls ein vollständiges Labeling.

2.2 Klassifikationsalgorithmen

Im Folgenden werden einige für die Masterarbeit relevante Klassifikationsalgorithmen kurz vorgestellt.

2.2.1 Überwachtes Lernverfahren und Klassifikation

Das in Beispiel 3 genannte Vorgehen bzgl. Aufteilung in einen Trainings- sowie Testsatz wird Überwachtes Lernverfahren (supervised learning) genannt, da bei den zu Grunde liegenden Daten die korrekten Klassenzuordnungen bereits bekannt sind [11]. "Die Klassifikation selbst ist somit ein überwachtes Lernverfahren, das markierte Daten verwendet um Objekte zu Klassen zuzuordnen [17]". Bei allen in dieser Masterarbeit vorkommenden Datensätzen ist die Klassenzuordnung bekannt und es wird eine Aufteilung in einen Trainings- sowie Testdatensatz geben.

Neben dieser Möglichkeit der Aufteilung gibt es auch die Option der Kreuzvalidierung. Hier wird ein vorliegender Datensatz mit bekannter Klassifizierung in gleichgroße Teilmengen k aufgeteilt [11] (wird deshalb auch K-fache Kreuzvalidierung genannt). Genau eine Teilmenge wird zum Validieren verwendet, die restlichen Teilmengen zum Trainieren. Dies wird k-Runden durchgeführt, wobei hier jede Teilmenge nur einmal zum Validieren verwendet werden darf.

In dieser Masterarbeit wird ein Klassifikator K als eine folgende Funktion definiert:

Definition 12 *In der u.s. Funktion (1) ist $\mathbb{R}^n$ ein Merkmalsvektor im mehrdimensionalen Merkmalsraum mit n Attributen, das genau einer bekannten Klasse C (aus der Menge aller Klassen) zugeordnet ist. t steht für die verschiedenen Klassifikatoren (Naive Bayes, K-Nächste Nachbarn, etc.).*

$$K_t = \mathbb{R}^n \rightarrow \mathbb{C} \tag{1}$$

2.2.2 Naive Bayes

Dieser Klassifikator basiert auf dem Satz von Bayes, mit welchem die Berechnung bedingter Wahrscheinlichkeiten (wird auch a-posteriori-Wahrscheinlichkeit genannt) möglich ist [3].

Definition 13 *Die Variablen A und B sind Ereignisse. Der Satz von Bayes berechnet die Wahrscheinlichkeit für ein Ereignis A, wenn ein Ereignis B (als Bedingung) zuvor eingetreten ist [17]:*

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

- $P(A|B)$ = *Die bedingte Wahrscheinlichkeit des Ereignisses A, vorausgesetzt B ist eingetreten.*
- $P(B|A)$ = *Die bedingte Wahrscheinlichkeit des Ereignisses B, vorausgesetzt A ist eingetreten.*
- $P(A)$ = *Die Wahrscheinlichkeit des Ereignisses A.*
- $P(B)$ = *Die Wahrscheinlichkeit des Ereignisses B.*

Mit dem Satz von Bayes ist die Berechnung einer Wahrscheinlichkeit einer Klassenzugehörigkeit A_i für ein Ereignis B möglich, wobei hier die Klassifizierung anhand der berechneten Wahrscheinlichkeit erfolgt. Folglich ist die a-posteriori-Wahrscheinlichkeit einer Klassenzugehörigkeit A_i für einen Datensatz B (mit den Attributen $b_1, ..., b_n$) möglich:

$$P(A_i|B) = \frac{P(B|A_i) * P(A_i)}{P(B)}$$

Beispiel 11 *Hier ein Beispiel zur Anwendung des Satzes von Bayes:*

- $P(B)$ = *Draußen ist es trocken = 70 %.*
- $P(A)$ = *Es scheint die Sonne = 40 %.*
- $P(B|A)$ = *Wahrscheinlichkeit, dass es bei Sonnenschein draußen trocken ist = 95 %.*
- $P(A|B)$ = *Die Wahrscheinlichkeit, dass bei einer trockenen Witterung gleichzeitig die Sonne scheint.*

Gesucht wird somit $P(A|B)$, die sich laut Satz von Bayes wie folgt berechnet:

$$P(A|B) = \frac{0.95 * 0.4}{0.7} = 0.54 = Eine\, Wahrscheinlichkeit\, von\, 54\,\%.$$

Der **naive Bayes´sche Klassifikator** sucht die höchste a-posteriori-Wahrscheinlichkeit zu einer Klassenzugehörigkeit A_i [3]. Er klassifiziert ein Ereignis B zu einer Klasse A_i genau dann, wenn $P(A_i|B)$ maximal ist. Dies kann bei Datensätzen mit vielen Attributen sehr aufwendig werden. Um dies zu vereinfachen, wird die naive Annahme getroffen, dass die einzelnen Attribute bei gegebener Klasse unabhängig voneinander sind und es keine Abhängigkeiten gibt. Die a-posteriori-Wahrscheinlichkeit von $P(B|A_i)$ kann somit wie folgt berechnet werden [3]:

$$P(B|A_i) = \prod_{k=1}^{n} P(b_k|A_i)$$

Der **naive Bayes´sche Klassifikator** $K_{NaiveBayes}$ (siehe Definition 1) besitzt somit folgende Formel:

$$K_{NaiveBayes}(v) = \underset{A_i}{\operatorname{argmax}}\, P(A_i) * \prod_{k=1}^{n} P(b_k|A_i) \tag{2}$$

2.2.3 K-Nächste Nachbarn (KNN)

Beim Klassifikator *K-Nächste Nachbarn* (KNN) werden Daten im mehrdimensionalen Merkmalsraum auf Grund Ihrer Nachbarschaftsbeziehungen zu den benachbarten Objekten klassifiziert [17].

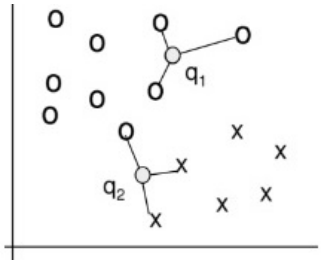

Abbildung 5: Beispiel einer Nächster-Nachbar-Klassifikation im zweidimensionalen Merkmalsraum mit zwei Klassen (übernommen aus [8]).

Eine zu klassifizierende Instanz wird durch einen Punkt repräsentiert, anschließend zählt man die bereits klassifizierten Nachbar-Instanzen (Anzahl = k) und fällt eine Mehrheitsentscheidung bezüglich der zu klassifizierenden Instanz [13].

Beispiel 12 *Das Beispiel in Abbildung 5 zeigt exemplarisch für $k = 3$ auf, dass die Instanz q_1 zur Klasse O sowie die Instanz q_2 zur Klasse X klassifiziert werden.*

Da die Nachbar-Instanzen nicht immer so trivial wie in Abbildung 5 ermittelbar sind, wird zur Bestimmung der nächsten Nachbarn Distanzmaße verwendet. Für die weitere Ausarbeitung der Masterarbeit wird hierbei die euklidische Distanz fokussiert. Bei der euklidischen Distanz wird der direkte Abstand d zwischen zwei Instanzen, der zu klassifizierenden Instanz x_i sowie der bereits klassifizierten Nachbar-Instanz x_j, berechnet [13]. Dies kann für beliebig viele Dimensionen p berechnet werden. Die euklidische Distanz besitzt somit folgende Formel:

Definition 14 *x_i und x_j sind Vektoren der Dimension p. Das euklidische Distanzmaß d ist über folgende Formel definiert [13]:*

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2}$$

Der **K-Nächste-Nachbarn-Klassifikator** lässt sich in Anlehnung an die Definition in 1 somit wie folgt verwenden:

$$K_{EuklidischeDistanz} = \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2} \tag{3}$$

Neben der euklidischen Distanz ist für die späteren Testreihen noch die **Manhattan-Distanz** [13] (zum Vergleich der Distanzmaße) von Interesse:

$$d(x_i, x_j) = \sum_{k=1}^{p}|x_{ik} - x_{jk}| \tag{4}$$

Bei der Manhattan-Distanz wird das Distanzmaß d zwischen zwei Punkten als die Summe der absoluten Differenzen ihrer Einzelkoordinaten definiert [13] - siehe Formel 4.

2.2.4 Entscheidungsbaum

Hier wird aus einer Datenmenge eine hierarchische Struktur (ähnlich eines Baums) aufgebaut, an deren Ende (Blätter) die jeweils zuzuordnende Klasse ableitbar ist und somit die Klassifizierung vorgenommen werden kann [17].

Ein Entscheidungsbaum verfolgt einen Top-Down-Ansatz, von der Wurzel bis zum Blatt. Ein Knoten entspricht hierbei einer Entscheidung anhand eines Attributwerts. Die Kanten repräsentieren die jeweiligen Übergänge zwischen den Knoten, basierend auf der zuvor getätigten Entscheidung am Knoten.

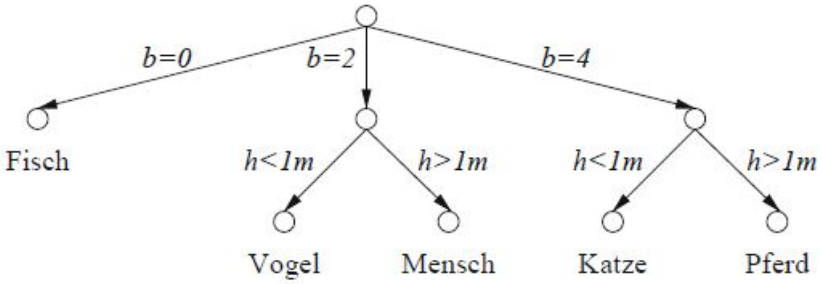

Abbildung 6: Ein exemplarischer Entscheidungsbaum zur Klassifizierung von Lebewesen, übernommen aus [17].

Beispiel 13 *In Abbildung 6 wird genau so ein Entscheidungsbaum zur Klassifizierung dargestellt. Der erste Knoten ist das Attribut Anzahl Beine (b), der nächste Knoten ist das Attribut Größe (h). Die Blätter (in Abhängigkeit der Attributswerte) sind die möglichen Klassen.*

Bei diesem hierarchischen Ansatz werden Attribute mit dem größten Informationsgehalt gesucht, was u.a. über die Entropie ermöglicht wird (siehe Definition 5). Mit ihr lässt sich der zu erwartende Informationsgehalt berechnen und wird in Bits angegeben. Ziel ist es, dass das Attribut mit dem größten Informationsgewinn (siehe Definition 15) gewählt wird [17]. Mit diesem Ansatz wird die Tiefe des Entscheidungsbaumes minimiert und der kleinstmögliche Baum ermöglicht.

Definition 15 *Folgende Variablen sind für die Berechnung der Entropie $E(t)$ wichtig [13]:*

- *k = Anzahl Klassen.*

- t = *Ein beliebiger Knoten.*
- $p_i(t)$ = *Wahrscheinlichkeit im Knoten t, dass eine Instanz in Klasse i fällt.*
- *Entropie* $E(t) = -\sum_{i=1}^{k} p_i(t) * log_2(p_i(t))$

Der Informationszuwachs (information gain) bezüglich eines herausgesuchten Attributs für die Festlegung eines Knotens ist wie folgt definiert: "Der Informationszuwachs ist die Differenz der Entropie des Elternknotens und der erwarteten gewichteten Entropie der möglichen Kindknoten [22]".

Definition 16 *Der Informationszuwachs lässt sich wie folgt berechnen:*

- *Attribut A teilt die Trainingsdaten an einem Knoten t in* T_n *Partitionierungen.*
- *Der Informationsgewinn durch das Attribut A berechnet sich wie folgt:*

$$Informationsgewinn(T, A) = E(T) - \sum_{i=1}^{n} \frac{|T_i|}{|T|} * E(T_i) \tag{5}$$

Tabelle 2 zeigt exemplarisch Datensätze zu Abbildung 6, anhand welcher für den ersten Knoten (Attribut Anzahl Beine b) der Informationszuwachs berechnet wird.

Index	1	2	3	4	5	6	7	8
Größe h	0.1	0.2	1.8	0.2	2.1	1.7	0.1	1.6
Beine b	0	2	2	4	4	2	4	2
Klasse	**Fisch**	**Vogel**	**Mensch**	**Katze**	**Pferd**	**Mensch**	**Katze**	**Mensch**

Tabelle 2: Exemplarische Datensätze inklusive Klassenzugehörigkeit bezüglich Abbildung 6.

Beispiel 14 *Für die in der Abbildung 6 vorgenommene Reihenfolge (Beine vor Größe) wird nun der Informationsgehalt für den Wurzelknoten, die Beine b, berechnet.*

- *Attribut Beine b teilt die Trainingsdaten in 3 Partitionierungen:*
- *E(T=Gesamt)= 2.1556 bit*
- *E(T(b=0)) = 0 bit, E(T(b=2)) = 0.8113 bit, E(T(b=4)) = 0.9183 bit*
- *Informationsgewinn(T, A=b) = 2.1556 -* $\frac{1}{8}$**0 -* $\frac{4}{8}$**0.8113 -* $\frac{3}{8}$**0.9183 = 1.4056 bit*

Wenn man analog am Wurzelknoten statt das Attribut Beine b nun das Attribut Größe h verwendet, ist der Informationsgewinn(T, A=h) = 1.0 bit. Da der Informationsgewinn für Attribut b höher ist (1.4056 bit > 1.0 bit), erhält man am Wurzelknoten für das Attribut b den optimaleren Entscheidungsbaum.

Der ID3-Algorithmus geht genau nach dem zuvor beschriebenen Berechnungsweg vor. Er berechnet rekursiv an jedem Knoten das Attribut mit dem maximalsten Informationsgewinn, um so den entropieoptimalen (und somit kleinsten) Entscheidungsbaum zu ermitteln [17].

2.2.5 Neuronale Netze (NN)

Neuronale Netze bilden Neuronenstrukturen ab, die dem menschlichen Gehirn nachempfunden sind [17]. Sie bestehen aus Neuronen, die sich in drei Schichten aufteilen:

- Einer Eingabeschicht: Hier werden Daten empfangen, aufbereitet und weitergeleitet.
- Beliebig vielen Zwischenschichten (auch verdeckte Schichten genannt): Informationen werden hier verarbeitet, gewichtet und an weitere Zwischenschichten geleitet.
- Ausgabeschicht: Neuronen geben die verarbeiteten Informationen als Ergebnis aus. Dies könnte z.B. die Wahrscheinlichkeit eines Datenobjekts zu einer Klasse sein.

Abbildung 7 stellt exemplarisch ein neuronales Netz mit 8 Neuronen dar, wo es um eine Klassifikationsfragestellung geht. Die gewichteten Verbindungen (Gewichtungen w_i) haben je nach Ausprägung stärkeren Einfluss auf das nachgelagerte Neuron.

Folgende Funktionen/Werte sind für für die Ein- und Ausgabe eines neuronalen Netzes wichtig [17]:

- Jedes Neuron i hat einen Ausgangswert $O_i \in \mathbb{R}$.
- Ein Kantengewicht $w_{ij} \in \mathbb{R}$ (von Neuron i zu Neuron j).
- Der Eingangswert I_i jedes Neurons i welcher sich als gewichtete Summe wie folgt berechnet: $I_i = \sum_j w_{ji} * O_j$.
  Sie wird auch Propagierungsfunktion genannt, da I_i aus den Ausgaben der Neuronen O_j mit den jeweiligen Kantengewichten w_{ji} berechnet wird [18].
- Eine Aktivierungsfunktion $s : \mathbb{R} \rightarrow \mathbb{R}$, die den Ausgangswert des Neurons i berechnet: $O_i = s(I_i)$

Eine gebräuchliche Aktivierungsfunktion ist die Sigmoid-Funktion [17], die eine Beschränkung auf einen bestimmten Wertebereich für den Ausgangswert O_i (Näherung an 0 oder 1) ermöglicht.

$$s(x) = \frac{1}{1 + e^{-x}} \in (0, 1) \tag{6}$$

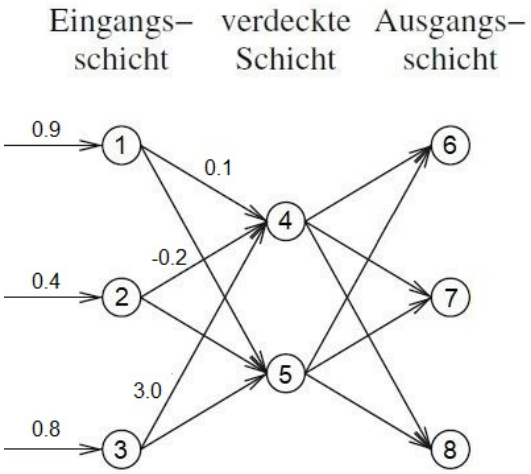

Abbildung 7: Ein exemplarisches neuronales Netz mit einer Klassifikationsfragestellung (abgewandelt übernommen aus [17]).

In Abhängigkeit der zuvor genannten Ein- bzw. Ausgangswerte sowie der Aktivierungsfunktion s könnte z.B. in Abbildung 7 die Berechnung der Klassifikation (Neuron 6, 7 oder 8 der Ausgangsschicht) vorgenommen werden.

Beispiel 15 *Als Beispiel soll aus Abbildung 7 der Ausgangswert O_4 des vierten Neurons (verdeckten Schicht) als ersten Teilschritt berechnet werden:*

- *Der Eingangswert (inkl. Kantengewichten) I_4 = 0.9*0.1 + 0.4*(-0.2) + 0.8*3.0 = 2.41*
- *Der Ausgangswert (unter Anwendung $s(x)$) $O_4 = s(2.41) = \frac{1}{1+e^{-2.41}}$ = 0.92*

Um die Abweichungen zwischen den berechneten Werten der Ausgangsschicht und denen der Trainingsdaten (in unserem Fall Klassenzugehörigkeiten) zu optimieren, werden in einem iterativen Verfahren die Gewichte w_i so lange optimiert, bis die prognostizierten Werte der Ausgangsschicht auch tatsächlich den Klassifizierungen der Trainingsdaten entsprechen [13].

Dieses iterative Verfahren ist mittels des Backpropagation-Algorithmus definiert [17]. Im Wesentlichen bilden die folgenden drei Schritte den Kern des Algorithmus [13]:

- Schritt 1 - Feed-forward-step: Nach einer ersten zufälligen Initialisierung der Gewichte w_i werden die Ausgangswerte der Ausgangsschicht berechnet (siehe Beispiel 15).
- Schritt 2 - Abweichung berechnen: Am Ende von Schritt 1 wird über eine Fehlerfunktion E die Abweichung ermittelt und der durchschnittliche Fehlerwert F ermittelt.

- Schritt 3 - Backpropagation: Nun werden die Gewichte von der Ausgangsschicht startend zur Eingangsschicht rückwärts im Netz modifiziert, so dass F immer kleiner wird. Dieser iterative Schritt wird nach einer vorgegebenen Anzahl von Runden (auch Epochen genannt) terminiert.

Convolutional Neural Networks (CNN) sind eine Sonderform von Neuralen Netzwerken, die sich besonders gut für die Mustererkennung eignen [11] und in der Studie in Abschnitt 3.3 Anwendung finden. CNN legen Ihren Fokus auf Sprach- und Bildklassifizierungsaufgaben, so können Sie z.B. statt Zahlenwerte in der Eingangsschicht eine mehrdimensionale Matrix verarbeiten. Dies ist vor allem bei Bildern sehr hiflreich, da hier die Angabe meist in Pixeln erfolgt (Breite, Höhe, Farbwert) [23].

CNNs arbeiten mit Filtern in den Zwischenschichten (verdeckten Schichten), die in der Lage sind Strukturen wie Linien und Farben zu erkennen und an die nächste Zwischenschicht weiterzuleiten [23]. Mit jeder Schicht können somit komplexere Details identifiziert werden, so lange bis die Ergebnisse in der Ausgangsschichtt zu einer Klasse eingeordnet werden können.

2.2.6 Supportvektormaschine (SVM)

Mit Supportvektormaschinen (SVM) lassen sich Daten im mehrdimensionalen Merkmalsraum bestmöglich in die zu bestimmenenden Klassen trennen [17]. Durch eine Ebene, im Zusammenhang mit SVM Hyperebene genannt, wird im mehrdimensionalen Raum versucht eine Trennfläche zu implementieren, welche die Trainingsdaten (mit bekannter Klassifizierung) in die jeweilige Klasse teilt [13]. Hierbei wird mathematisch versucht den Abstand der Datenpunkte, die am nächsten zur Hyperebene liegen, zu maximieren (siehe Abbildung 8). Dieser datenfreie Streifen wird Margin genannt [13]. Für eine Klassifikation mittels SVM werden die Trainingsdaten im Raum d als Vektoren in der Form $x_i \in \mathbb{R}^d$ mit bekannter Klassenzugehörigkeit $y_i \in \{1, -1\}$ benötigt [7]. Die Trainingsdaten T sind somit wie folgt formal definiert:

$$T = \{(x_i, y_i) \in \mathbb{R}^d, y_i \in (1, -1)\} \tag{7}$$

Das Lernen des Klassifikationsalgorithmus besteht darin, die Hyperebene H (mit der folgenden Formel) gleich Null zu finden [7]:

$$H = \langle w, x_i \rangle + b = 0 \tag{8}$$

Folgende Parameter sind hierbei wichtig [7]:

- w ist der Normalenvektor (welcher senkrecht zur Hyberebene steht)
- Der Bias b verschiebt die Hyperebene in Abhängigkeit von w in eine Richtung

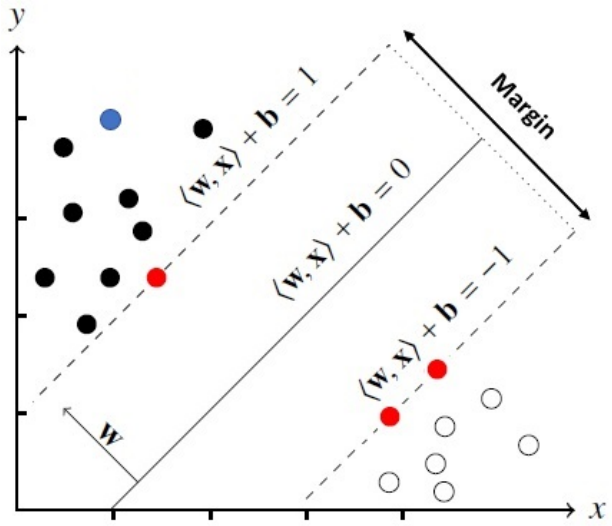

Abbildung 8: Linear trennbare Daten im $\mathbb{R}^2$. Die beiden Klassen (weiße Punkte, schwarze Punkte) werden durch die bestmögliche Hyperebene (H=0) voneinander abgegrenzt (abgewandelt übernommen aus [7]).

Die bestmöglichste Hyberebene ist die mit der breitesten Margin. Die äußeren Grenzen der Margin liegen bei H = 1 sowie H = -1, die Hyberebene liegt dann hier genau mittig dazwischen (H= 0). Der Normalenvektor w wird so lange optimiert, bis die Trainingsdatenpunkte auf dem Rand oder außerhalb der Margin liegen.

Die rot markierten Punkte auf dem Rand (siehe Abbildung 8) werden Stützvektoren (Support-Vectors) genannt und liegen genau auf der jeweiligen Außengrenze der Margin [13]. Da die Klassenzugehörigkeit durch $y_i = \pm 1$ ausgedrückt wird, kann (8) in die folgende Gleichung überführt werden:

$$y_i = sgn(\langle w, x_i \rangle + b) \tag{9}$$

Beispiel 16 *In Abbildung 8 ist der blaue Datenpunkt* $d_1 = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$ *zu sehen. Angenommen es wäre ein Normalvektor* w_1 *sowie der Bias* b_1 *für die optimale Hyperebene (siehe 8) gefunden, so könnte die Klassenzugehörigkeit für* d_1 *wie folgt berechnet werden:*

- $y_i = sgn(\langle w_1, \begin{pmatrix} 1 \\ 4 \end{pmatrix} \rangle + b_1)$ = *1*
- *Der Datenpunkt* d_1 *gehört somit zur Klasse y=1*

Wie links in der Abbildung 9 zu sehen ist, sind die Klassen nicht immer linear bestimmbar.

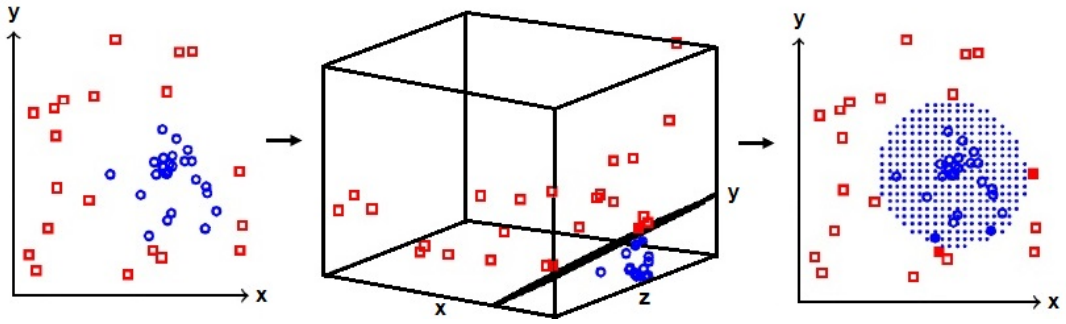

Abbildung 9: Abbildung des Kernel-Tricks. Links in der Abbildung sind die nicht linear trennbaren Daten im $\mathbb{R}^2$, die in einen Raum höherer Dimensionalität $\mathbb{R}^3$ überführt wurden (siehe Mitte). Dort kann dann wieder eine lineare Unterteilung erfolgen (abgewandelt übernommen aus [2]).

Mit Hilfe des Kernel-Tricks ist es möglich, die Daten in einen Raum mit höherer Dimensionalität abzubilden. Nach diesem Schritt sind die Daten wieder linear klassifizierbar, siehe Abbildung 9. Folgende Kernel-Funktionen sind mit die am häufigsten verwendeten Funktionen, um eine nicht lineare Funktion in einen höherdimensionalen Raum abzubilden (für die hinzugenommenen Variablen gilt: $a, b, c, \sigma \in \mathbb{R}, n \in \mathbb{N}$) [2]:

- Radiale Basisfunktion des Gaußschen Kernels:

$$exp(-\frac{\|x-y\|^2}{2\sigma^2})$$

- Polynomieller Kern:

$$(\langle x, y\rangle + a)^n$$

- Sigmoidkern:

$$tanh(b\langle x, y\rangle + c)$$

Alle drei zuvor genannten Kernel werden im Data-Mining-Tool Weka (siehe Abschnitt 4.2.3) unterstützt.

2.3 Klassifikationsgüten

Die folgend kurz vorgestellten Maße bestimmen die Performanz einer Klassifizierung. Alle Maße basieren auf der Konfusionsmatrix (siehe Tabelle 3). Dies ist eine 4-Felder-Matrix und visualisiert binär (wahr oder falsch), in wie weit die klassifizierten Instanzen einer Klasse richtig oder falsch zugeordnet wurden ([5] & [17]).

		Actual Class	
		Positive	Negative
Predicted Class	Positive	**TP True Positve**	**FP False Positive**
	Negative	**FN False Negative**	**TN True Negative**

Tabelle 3: Dargestellt ist die Konfusionsmatrix mit Ihren Feldern (leicht abgewandelt aus [5]). Die Spalten repräsentieren die tatsächliche Klassenzuordnung, die Zeilen die vorhergesagten Klassifizierungen.

- **TP-Rate (Recall):** Auch Trefferquote genannt, berechnet ob alle tatsächlichen Instanzen gefunden wuren:
$$\frac{TP}{TP + FN}$$
- **FP-Rate:** Berechnet den Anteil der als fälschlicherweise positiv klassifizierten Instanzen im Verhältnis zu allen tatsächlichen Nicht-Instanzen:
$$\frac{FP}{FP + TN}$$
- **Präzision (Precision):** Berechnet, wie genau die Instanzen der Klasse vorhergesagt wurden:
$$\frac{TP}{TP + FP}$$
- **Genauigkeit (Accuracy):** Anteil der korrekt vorhergesagten Instanzen im Verhältnis zu allen vorausgesagten Instanzen:
$$\frac{TP + TN}{TP + TN + FP + FN}$$
- **F-Wert (F-Measure):** Gewichteter Durchschnitt von Präzision und Trefferquote:
$$2 * \frac{Precision * Recall}{Precision + Recall}$$

3 Verwandte Arbeiten

Im Folgenden werden Arbeiten vorgestellt, die sich im weitesten Sinne mit abstrakter Argumentation sowie Machine Learning bzw. Klassifikation beschäftigen. Der Fokus liegt hier beim Vergleich der unterschiedlichen Ansätze. Besonders interessant ist die in Abschnitt 3.3 genannte Studie, da hier ein ähnlicher Ansatz wie die in Abschnitt 1.3 genannte Forschungsfrage verfolgt wird.

3.1 Argumentationsframework vs. Maschinelles Lernen zur Entdeckung von Alzheimer [1]

In der Studie von Achilleos et al. [1] wird anhand von Magnetresonanztomographien (kurz MRT) des Gehirns analysiert, ob eine Erkrankung von Alzheimer (oder ein Vorstadium) bei einem untersuchten Patienten vorliegt. Achilleos verwendet Daten von der Alzheimer's Disease Neuroimaging Initiative (ADNI [1]), die wie folgt klassifiziert sind:

- Kein Alzheimer (NC - normal controls)
- Alzheimer (AD - Alzheimer´s Disease)

Basierend auf den MRT-Bildern besitzen die verwendeten Datensätze u.a. Attribute des Hippocampus (Teil des Gehirns), auf welche Achilleos seinen Fokus für die weitere Analyse legt. Er trainiert im Anschluss die beiden Entscheidungsbaum-Klassifikatoren Decision Trees sowie Random Forests anhand der Datensätze und wendet hierbei eine 10-fache Kreuzvalidierung an. Die Ergebnisse der Klassifkation sind in Tabelle 4 anhand der Genauigkeit zu sehen. Basierend auf diesen Ergebnissen (Klassifikation mittels der beiden Entscheidungsbäumen) werden mit Hilfe verschiedener Data-Mining-Plattformen Entscheidungsregeln extrahiert. Um die extrahierten Regeln anzuwenden, greift Achilleos auf das Argumentationsframework von Gorgias zurück [19]. Mit der Open-Source-Plattform Gorgias können innerhalb eines Argumentationsframeworks Regel-Sets implementiert werden, die vereinfacht ausgedrückt die Akzeptanz der jeweiligen Argumente mittels Regeln definieren. Mit Gorgias und den extrahierten Regeln erstellt Achilleos ein Argumentationsframework und verwendet hierbei ebenfalls die Datensätze der ADNI. Hier exemplarisch die verwendete Regel r1, die einen Patienten ab einem gewissen Volumen des Hippocampus als nicht alzheimerkrank klassifiziert.

$$(r1) : \text{If } hipVolume >= 2890mm^3\text{: return subject in NC}$$

Tabelle 4 zeigt eine hohe Genauigkeit von 91 % bei Anwendung des regelbasierten Argumentationsframeworks auf die ADNI-Datensätze.

Klassifizierer	**Genauigkeit**
Decision Trees	77 %
Random Forests	74 %
Regelbasiertes Argumentationsframework	91 %

Tabelle 4: In der Tabelle von Achilleos (Teilausschnitt aus [1]) werden die beiden Entscheidungsbaum-Klassifikatoren mit dem regelbasierten Argumentationsframework hinsichtlich Genauigkeit miteinander verglichen.

Achilleos zeigt in dieser Studie, dass der kombinierte Ansatz aus Anwendung von

verschiedenen Machine-Learning-Algorithmen mit anschließender Erstellung eines regelbasierten Argumentationframeworks zu einem vielversprechenden Ansatz für die Klassifikation von Daten ist.

3.2 Argumentation zur Unterstützung von Entscheidungen im Gesundheitssektor [15]

In der Studie von Longo [15] wird die Rückfallquote (Brustkrebs) von Frauen nach einer erfolgten Brustkrebsbehandlung analysiert. Die hier von einer Klinik verwendeten und veröffentlichten Datensätze (Patientendaten) besitzen die folgende Klassifizierungen:

- Wiederauftreten Brustkrebs (reappearance of cancer after 5 years)
- Kein Wiederauftreten Brustkrebs (no reappearance of cancer after 5 years)

In dieser Studie erstellen Experten aus der Brustkrebsforschung [15] ein Argumentationsframework, in welches ausschließlich medizinisches Fachwissen einfließt. Anschließend wurde auf Basis der bevorzugten Semantik die Klassifikation durchgeführt. Parallel dazu führt Longo auf den selben Patientendaten mittels unterschied-

Classifier	**70 % split**
decision tables	73,25
bayesian network	68,60
best-first decision tree	62,79
regression	66,26
multilayer perceptron	58,13
alternating decision tree	65,11
Preferred semantic (AT)	73,42

Tabelle 5: In der Tabelle von Longo (Teilausschnitt [15]) wird anhand veschiedener Aufteilungen in Trainings- und Testdaten (in diesem Fall 70 % Trainingsdaten, 30 % Testdaten) die Genauigkeit von Klassifkationsalgorithmen mit einem Argumentationsframework verglichen (Angabe in Prozent).

licher Klassifikationsalgorithmen eine Klassifikation durch. Er bewertet beide Ansätze anhand der Klassifikationsgüte Genauigkeit (Kennzahlen siehe Abschnitt 2.3). Das Ergebnis wird in Tabelle 5 dargestellt.

Es ist ersichtlich, dass ein durch Experten erstelltes abstraktes Argumentationsframework (Zeile Prefered semantic (AT) = 73,42 %) in Ihrer Genauigkeit durchaus mit den trainierten Klassifikationsalgorithmen konkurrieren kann.

Longo ist es in dieser Studie gelungen, in einem realen Anwendungsszenario ein Argumentationsframework anzuwenden. Er zeigt exemplarisch, dass dieser wissensbasierte Ansatz mindestens genauso gute Ergebnisse liefert wie die Anwendung

von Klassifikationsalgorithmen und hier sehr viel Potenzial für weitere Studien sowie Forschungen sieht.

3.3 Neuronale Netze zur Vorhersagbarkeit von bevorzugten Extensionen [14]

Die Studie von Kuhlmann [14] hat einen ähnlichen Ansatz wie die in Abschnitt 1.3 formulierte Forschungsfrage. Hier wird ein Graph Convolutional Network (GCN) als Datengrundlage verwendet, das ausschließlich den Fokus auf die Verarbeitung von Graphiken als neuronales Netz setzt und somit eine Form der Convolutional Neural Networks (siehe Abschnitt 2.2.5) darstellt [24]. Die Studie verwendet Daten eines abstrakten Argumentationsframework und versucht mit Hilfe der Graph Convolutional Networks vorauszusagen, ob ein vorherzusehendes Argument in den bevorzugten Extensionen vorhanden ist oder nicht. Das vorherzusehende Argument ist wie folgt klassifiziert:

- **Yes** (enthalten)
- **No** (nicht enthalten)

Kuhlmann verwendet das Graph Convolutional Network von Kipf [14], das sich auf die Bestimmung von Labels für Knoten (Neuronen) innerhalb einer Graphik spezialisiert hat. Bevor dieses jedoch anwendbar ist, werden mit verschiedenen Programmen (welche unterschiedliche Semantiken im Fokus haben) Graphen erstellt. Anschließend werden die jeweiligen Extensionen ausgegeben, welche die Grundlage für das Trainieren bilden. Da die Größe der Graphen einen starken Einfluss auf die Ergebnisse hat, wurde folgende übergreifende Einteilung der Trainingsdaten in sechs Trainings-Sets (aufsteigend nach der Größe) getroffen. Hier ein Auszug der Trainings-Sets:

- Set 1 (Minimale Anzahl Knoten): 30 Graphen, Insgesamt 5.461 Knoten
- Set 6 (Maximale Anzahl Knoten): 600 Graphen, Insgesamt 149.130 Knoten

Kuhlmann untersucht in Abhängigkeit weiterer Parameter (abgesehen von der Größe des Trainings-Sets), wie gut sich die GCN´s trainieren lassen (hier ein Auszug):

- Aufbau der Matrix
- Anzahl Epochen (siehe Schritt 3 Backpropagation - Abschnit 2.2.5)
- Laufzeitverhalten

Tabelle 6 zeigt die Ergebnisse eines Trainingsvorgangs anhand der Kennzahl Genauigkeit (pro Set).

Es ist ersichtlich, dass die Genauigkeit mit zunehmender Testgröße sich leicht verbessert. Laut Kuhlmann hat sich das Trainieren eines GCN für die Vorhersage, ob

Trainings-Set	Genauigkeit
Set 1	77,01
Set 2	79,72
Set 3	80,20
Set 4	80,43
Set 5	80,41
Set 6	80,44

Tabelle 6: In der Tabelle von Kuhlmann (abgewandelter Teilausschnitt [14]) sind die Ergebnisse eines Trainingsvorgangs mit den zuvor genannten Trainings-Sets (Set 1 bis 6) abgebildet (Angabe in Prozent).

ein ausgewähltes Argument Bestandteil einer bevorzugten Extension ist, als moderat erwiesen. Tabelle 6 sowie weitere Auswertungen der Studie mit (unterschiedlichen Parameter-Einstellungen) haben eine maximale Genauigkeit von ca. 80.5 % gezeigt. Es wird jedoch darauf hingewiesen, dass die o.g. Parameter noch viel mehr Möglichkeiten zur Justierung der Test-Sets ermöglichen und evtl. sogar die Hinzunahme von weiteren Parametern Sinn machen kann.

4 Vorgehen

Die folgenden Teilabschnitte beschreiben das Vorgehen in der Masterarbeit und geben nochmals einen Gesamtüberblick.

4.1 Datengrundlage abstraktes Argumentationsframework

Es sind mit einem Software-Tool sowie mit den zur Verfügung gestellten Beispielen der International Competition on Computational Models of Argumentation (kurz ICCMA) [4] mehrere Argumentationsframeworks zu erstellen bzw. auszuwählen. Für die Erstellung eines Argumentationsframeworks mittels eines Software-Tools ist die Java-Bibliothek TweetyProject zu verwenden [20]. Diese wird online zum Download bereitgestellt und ist eine Sammlung von Java-Bibliotheken, welche ausführbare Implementierungen verschiedenster Forschungsbereiche der künstlichen Intelligenz abbildet. So bietet das TweetyProject u.a. eine Implementierung von abstrakten Argumentationsframeworks nach Dung an. Dabei ist es möglich, sich zufällige Argumentationsframeworks sowie darauf basierend die Labelings nach einer festgelegten Semantik generieren zu lassen.

Die online frei zur Verfügung gestellten Argumentationsframeworks der ICCMA enthalten zahlreiche Beispiele, die sich ebenfalls als Datengrundlage für die Erstellung der Labelings eignen. Die Labelings selbst können, nach vorherigem Upload des gewählten ICCMA-Argumentationsframeworks, ebenfalls über das TweetyPro-

ject erstellt werden. In der Masterarbeit werden hierbei der ICCMA'19 Datensatz (von 2019, [4]) näher analysiert.

4.2 Klassifizierung

Im vorherigen Teilabschnitt wurden die beiden Wege zur Auswahl/Erstellung der Argumentationsframeworks sowie Erstellung der Labelings eingeführt, nun ist ein Data-Mining-Tool auszuwählen, mit welchem die folgenden Schritte durchzuführen sind:

4.2.1 Labelings exportieren

Die in Abschnitt 4.1 erzeugten Labelings müssen maschinenlesbar aufbereitet (z.B. als CSV-File) und exportiert werden.

4.2.2 Labelings in Teilmengen aufteilen

Bei den exportierten Labelings (pro Argumentationsframework) ist eine Aufteilung in einen Trainings- sowie Testdatensatz vorzunehmen (70 % bzw. 30 %).

4.2.3 Labelings in das Data-Mining-Tool Weka importieren

Als Data-Mining-Tool wird Weka [12] verwendet, das eine Open-Source-Software ist und unter die GNU General Public License fällt. Mit ihr ist es möglich auf alle in Abschnitt 2.2 genannten Klassifikationsalgorithmen zuzugreifen. In Weka können die gängigsten Datenformate (u.a. CSV- und TXT-Dateien) gelesen werden, es wird jedoch eine Aufbereitung der exportierten Labelings (CSV-Files) in das Weka-eigene Dateiformat ARFF (Attribute-Relation File Format) angestrebt. Die Aufbereitung kann hierbei in Weka durchgeführt werden. Vorteil eines ARFF-Files ist, dass es einen Kopfabschnitt mit Informationen über die Argumente enthält. Tabelle 7 zeigt exemplarisch den Aufbau eines ARFF-Files für das Argumentationsframework **AF** aus Beispiel 2 mit den zulässigen Labelings l_1 sowie l_2 aus Tabelle 1. Ein ARFF-File ist immer in 3 Abschnitte gegliedert:

- **@RELATION** steht für den Namen des Argumentationsframeworks.
- In Weka werden die Argumente als Attribute (**@ATTRIBUTE**) interpretiert und deklariert (inkl. der vorkommenden Attributsausprägungen).
- Im **@DATA**-Abschnitt werden die Labelings angezeigt.

Im **@ATTRIBUTE**-Abschnitt ist somit schnell zu sehen, welche Labels pro Argument in allen Labelings überhaupt vorkommen. Somit kann u.a. die Kennzahl **durchschnittliche Anzahl der Labels** pro Argumentationsframework schnell berechnet werden (siehe Abschnitt 5.1).

@RELATION Beispiel

@ATTRIBUTE attribute_A {IN,OUT,UNDEC}
@ATTRIBUTE attribute_B {IN,OUT,UNDEC}
@ATTRIBUTE attribute_C {IN,OUT,UNDEC}
@ATTRIBUTE attribute_D {IN,UNDEC}
@ATTRIBUTE attribute_E {OUT,UNDEC}
@ATTRIBUTE attribute_F {IN,OUT,UNDEC}
@ATTRIBUTE attribute_G {IN,UNDEC}

@DATA

IN,OUT,IN,IN,OUT,IN,IN
IN,OUT,IN,IN,OUT,UNDEC,UNDEC

Tabelle 7: Diese Tabelle zeigt den Aufbau eines ARFF-Files für das exemplarische Argumentationsframeworf **AF** aus Beispiel 2. Die beiden zulässigen Labelings l_1 sowie l_2 (siehe auch Tabelle 1) werden im **@DATA**-Abschnitt genannt.

4.2.4 Klassifikationsalgorithmen anhand der Labelings trainieren und anwenden

Jeder in Abschnitt 2.2 genannte Klassifikationsalgorithmus ist in Weka zuerst mit dem Trainingsdatensatz zu trainieren, bevor anschließend der trainierte Algorithmus auf den Testdatensatz angewendet wird. Die verwendeten Parameter der Klassifikationsalgorithmen werden im nächsten Unterabschnitt erörtert.

4.3 Parameter der Klassifikationsalgorithmen für die Testreihen

Es erfolgt nun ein Gesamtüberblick über die selbst getroffene Auswahl der Parameter für die Klassifikationsalgorithmen. Auf diese Auswahl können die Klassifikationsalgorithmen in den einzelnen Testreihen zugreifen:

- Naive Bayes: Hier sind keine Anpassungen an Parametern möglich, da die höchste a-posteriori-Wahrscheinlichkeit selbstbestimmt berechnet wird.
- K-Nächste Nachbarn:
  - K (Anzahl Nachbarn) kann folgende Werte annehmen: $k \in \{1, 3, 5, 7, 9, 12\}$.
  - Beide Distanzmaße aus Abschnitt 2.2.3:
    - Euklidische Distanz.
    - Manhattan-Distanz.

- Entscheidungsbaum: Hier sind keine Anpassungen an Parametern möglich, da der verwendete ID3-Algorithmus selbstbestimmt anhand der Entropie den höchsten Informationsgewinn berechnet.
- Neuronales Netz:
  * Bezüglich Anzahl der Zwischenschichten sowie der Fehlerwert F (Abschnitt 2.2.5) kamen folgende Werte zum Einsatz:
    · Zwischenschichten = 3 Stück a 20 Knoten, Anzahl Epochen = 50, F = 0,6.
    · Zwischenschichten = 3 Stück a 20 Knoten, Anzahl Epochen = 50, F = 0,3.
    · Zwischenschichten = 2 Stück - 10 und 5 Knoten, Anzahl Epochen = 50, F = 0,3.
- Supportvektormaschinen - alle 3 Kernelfunktionen aus Abschnitt 2.2.6:
  * Radiale Basisfunktion.
  * Polynomieller Kern.
  * Sigmoidfunktion.

4.4 Abnehmende Vollständigkeit der Testdaten

In dieser Masterarbeit wird unter anderem untersucht, wie es sich mit der Vorhersagbarkeit bei abnehmender Vollständigkeit der Argumente verhält. Die abnehmende Vollständigkeit bezieht sich auf die Testdaten (Anwendungsphase) der ungesehenen Teil-Labelings. Hier wird es mehrere Labels mit fehlenden Werten geben.

Definition 17 *Für die ungesehenen Teil-Labelings der Anwendungsphase werden bei den Labels der Argumente B_1 bis B_n fehlende Werte simuliert.*

Mit der Weka-Funktion **ReplaceWithMissingValue** können per Zufall Labels in den Testdaten, für eine vorher definierte Anzahl an Argumenten, als fehlende Werte simuliert werden (in Weka mit einem Fragezeichen gekennzeichnet, siehe Tabelle 8). Somit soll evaluiert werden, ab welcher Prozentzahl an vorhandenen Argumenten in den Testdaten ein trainierter Klassifikator k valide Ergebnisse berechnen kann. Die abnehmende Vollständigkeit der Testdaten wird in 3 Prozentstufen evaluiert:

- 70 % Vollständigkeit Testdaten.
- 50 % Vollständigkeit Testdaten.
- 30 % Vollständigkeit Testdaten.

@RELATION Beispiel

@ATTRIBUTE attribute_A {IN,OUT,UNDEC}
@ATTRIBUTE attribute_B {IN,OUT,UNDEC}
@ATTRIBUTE attribute_C {IN,OUT,UNDEC}
@ATTRIBUTE attribute_D {IN,UNDEC}
@ATTRIBUTE attribute_E {OUT,UNDEC}
@ATTRIBUTE attribute_F {IN,OUT,UNDEC}
@ATTRIBUTE attribute_G {IN,UNDEC}

@DATA

?,?,IN,IN,OUT,IN,?
?,?,IN,IN,OUT,UNDEC,?

Tabelle 8: Der bekannte Aufbau des ARFF-Files aus Tabelle 7 zeigt nach der Anwendung der Funktion **ReplaceWithMissingValue** die zulässigen Labelings l_1 sowie l_2 mit 3 fehlenden Werten. Die beiden Labelings sind nur noch zu 57,14 % vollständig.

4.5 Evaluation

Nachdem die Klassifikationsalgorithmen anhand der Labelings trainiert und angewendet wurden, sind nun Kennzahlen für die Evaluation zu erstellen. Für die Evaluierung der Ergebnisse sind alle in Abschnitt 2.3 vorgestellten Maße zu verwenden. Weka stellt alle diese Maße zur Verfügung, somit ist nach dem Anwenden der trainierten Algorithmen die Klassifikationsgüte direkt erstellbar. Final sind nun alle Ergebnisse (der jeweiligen Algorithmen) miteinander zu vergleichen und zu evaluieren, wie gut das Zielargument vorhergesagt werden kann (und somit den Ausgang einer Diskussion).

5 Datengrundlage ICCMA'19 Datensatz

Hier erfolgt nun eine Gesamtübersicht über den ICCMA'19 Datensatz [4]. Aus diesem werden anschließend 50 Argumentationsframeworks ausgewählt, welche die Datengrundlage für die Testreihen dieser Masterarbeit bilden.

5.1 Metadaten

Der ICCMA'19 Datensatz besteht aus insgesamt 326 Argumentationsframeworks, welche hinsichtlich der Vorauswahl auf folgende Kennzahlen untersucht werden: Anzahl Argumente, Anzahl Attacken sowie der Knotengrad (Verhältnis Attacken zu Argumente).

Die Abbildung 9 zeigt die höchsten bzw. kleinsten Werte dieser Kennzahlen.

	Anzahl Argumente	**Anzahl Attacken**	**Knotengrad**
Maximum	10.000	1.039.471	1.011,16
Minimum	5	8	1,04

Tabelle 9: Hier erfolgt eine Auflistung der maximalen bzw. minimalen Werte (der zuvor genannten Kennzahlen) aus den Argumentationsframeworks des ICCMA'19 Datensatzes.

Zu einem Argumentationsframework mit über 10.000 Argumenten die zulässigen Labelings zu erstellen ist hinsichtlich der folgenden zwei Punkte problematisch:

- Großer Speicherplatzbedarf für die Labelings (z.T. mehrere Terrabytes pro Argumentationsframework).
- Weka kann nur eine gewisse Größe an Megabytes verarbeiten.

Aus diesen Gründen ist es sinnvoll, die Anzahl der Argumente auf 1000 Stück zu begrenzen. Dadurch bleiben von den 326 Argumentationsframeworks noch 276 Stück übrig. Um sich die zulässigen Extensionen für alle 276 verbliebenen Argumentationsframeworks berechnen zu lassen, wird auf ein speziell dafür ausgelegtes Software-Tool zurückgegriffen, und zwar auf die ASPARTIX System Suite [10]. Via Tweety können nun im Anschluss die zulässigen Extensionen eingelesen und anschließend die zulässigen sowie vollständigen Labelings ausgegeben werden.

Da das Data-Mining-Tool Weka (Abschnitt 4.2.3) die zulässigen Labelings auf einem lokalen Rechner zu verarbeiten hat, ist hier ferner noch eine Größenbegrenzung (in Megabyte) unumgänglich. Zusammengefasst ist bei der Betrachtung der restlichen 276 Argumentationsframeworks folgendes zu berücksichtigen:

- Mit ASPARTIX werden die zulässigen Extensionen erstellt. Auf Grund des Speicherplatzes gilt hier eine Grenze von 30.000 Extensionen.
- Die via Tweety ausgegebenen zulässigen und vollständigen Labelings dürfen 40 MB nicht übersteigen, falls doch ist hier ein Teilauszug der Labelings zu verwenden.

Bei 45 der zu bestimmenden 50 Argumentationframeworks wurde sich an die drei Kennzahlen aus Tabelle 9 orientiert und zur besseren Übersichtlichkeit in 3 Test-Sets eingeteilt. Pro Kennzahl (= Test-Set) sind 15 Argumentationsframeworks auszuwählen, jeweils 5 aus dem höchsten, kleinsten sowie mittlerem Wertebereich der jeweiligen Kennzahl:

- Anzahl Argumente (Set 1).

- Anzahl Attacken (Set 2).
- Knotengrad (Set 3).

Anschließend werden nach dem Zufallsprinzip noch 5 weitere Argumentationsframeworks ausgewählt (Set 4), um in Summe auf 50 Stück zu kommen.

Neben den bereits erwähnten Kennzahlen ist vor allem die zusätzlich berechnete durchschnittliche Anzahl der Labels eine weitere wichtige Information und kann Werte W im Bereich $W = [1; 3]$ annehmen. Ein Wert $W = 1$ bedeutet hierbei, dass jedes Argument im Argumentationsframework genau ein Label über alle Labelings besitzt, die Vorhersage ist hier trivial. Eine mögliche Beobachtung für den Wert $W = 3$ wäre, dass es einen kleinen Prozentsatz an Argumenten gibt, welcher alle anderen Argumente angreift und teilweise im Gegenzug von diesen auch zurück angegriffen wird. Dieses Angriffsverhalten verursacht einen sternförmigen Aufbau der Argumentationsframeworks (siehe Abbildung 10).

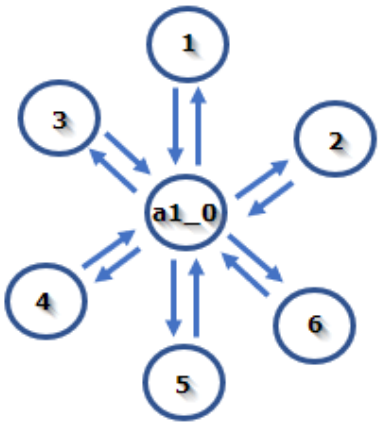

Abbildung 10: Hier schematisch die sternenförmige Anordnung der Argumente eines Argumentationsframeworks bei $W = 3$.

Auf Grund der Anordnung ist es für einen Klassifikator sehr schwierig für ein zufällig ausgewähltes Argument das korrekte Label zu erlernen, außer es ist das zentrale Argument in der Mitte.

Tabelle 10 zeigt exemplarisch die 5 Argumentationframeworks mit der höchsten Anzahl an Argumenten. In Abhängigkeit der Testreihen ist es sinnvoll, die Tabellen mit den Metadaten (siehe Tabelle 10) mit den Ergebnissen der jeweiligen Testreihe zu erweitern, um so interessante Zusammenhänge zu beobachten.

AF	Anzahl Argumente	Anzahl Attacken	Knoten-grad	Anzahl zulässige Labelings	Durchs. Anzahl Labels
A-1-admbuster_1000	1000	1498	1,50	750	2,23
A-4-afinput_exp_acyclic_depvary_step8_batch_yyy09	971	893586	920,27	6000	1,03
T-2-ferry2.pfile-L3-C3-03.pddl.2.cnf	933	1838	1,97	6000	1,21
B-2-afinput_exp_cycles_indvary2_step1_batch_yyy01	834	672442	806,29	7500	1,03
T-3-afinput_exp_acyclic_indvary1_step2_batch_yyy09	812	644253	793,42	4033	2,01

Tabelle 10: Hier die 5 Argumentationsframeworks mit der höchsten Anzahl von Argumenten (Test-Set 1). Es wurde auch zusätzlich die durchschnittliche Anzahl von Labels pro Argument mitbestimmt.

6 Testreihe: Zulässige Labelings

In der ersten Testreihe werden die zulässigen Labelings der in Abschnitt 5 gewählten 50 Argumentationsframeworks näher analysiert. Ziel ist es, erste Aussagen über die Vorhersagbarkeit des Labels eines zufällig gewählten Arguments treffen zu können. Basierend auf den ersten Erkenntnissen werden anschließend Rückschlüsse für die weiteren Testreihen gezogen.

6.1 Aufbau und Durchführung

Vorab eine Übersicht der wichtigsten Parameter zur Durchführung dieser Testreihe:

- Semantik: Zulässige Semantik
- Anzahl Argumentationsframeworks: 50 (ICCMA'19 Datensatz)
- Vollständigkeit Testdaten: 100 %

Für alle Argumentationsframeworks wurde die Precision, der Recall sowie der F-Wert (siehe Abschnitt 2.3) der jeweiligen Klassifikatoren (siehe Abschnitt 2.2) ermittelt. Pro Argumentationsframework sind 70 % der zulässigen Labelings für das Trainieren und 30 % für das Testen zu verwenden. Die 50 Argumentationsframeworks sind in 4 Test-Sets (siehe Abschnitt 5.1) eingeteilt. Für die Durchführung dieser Testreihe wurde auf die in Abschnitt 4 vorgestellten Wertebereiche der einzelnen Klassifikationsalgorithmen zurückgegriffen:

- Naive Bayes
- K-Nächste Nachbarn (KNN)
- Entscheidungsbaum
- Neuronale Netze

- Supportvektormaschinen

Bei jedem der 50 ausgewählten Argumentationsframework sind die zulässigen Labelings komplett in Weka einzulesen. Dort wird eine Randomisierung der Labelings vorgenommen, bevor Sie in einen Trainings- sowie Testdatensatz aufgeteilt werden. Per Zufall ist ein Argument auszuwählen, das im weiteren Verlauf die vorherzusehende Klasse bildet. Bei diesem Argument sollte in den Labelings des jeweiligen Argumentationsframeworks mindestens einmal das Label In, Out sowie Undec vorkommen. Falls dies nicht möglich ist, müssen mindestens zwei der genannten Labels vorkommen. In Weka ist immer zuerst der Trainingsdatensatz mit dem jeweiligen Zielargument als vorherzusehende Klasse zu öffnen, bevor anschließend der Klassifikator ausgewählt und der Trainingsvorgang gestartet werden kann. Anschließend ist der Testdatensatz zu öffnen, wo ebenfalls das Zielargument als vorherzusehende Klasse auszuwählen ist. Nun kann die Klassifikationsgüte des Klassifikators berechnet werden. Die Ergebnisse der jeweiligen Klassifikatoren sind pro Argumentationsframework zu dokumentieren.

6.2 Ergebnis

Im Folgenden werden die Ergebnisse der erste Testreihe analysiert, bevor anschließend auf die beobachtete Unteranpassung eingegangen wird.

6.2.1 Ergebnisse der Testreihe

Tabelle 11 zeigt den durchschnittlichen F-Wert, Recall sowie Precision der 50 Argumentationsframeworks. Die Precision (und somit folglich der F-Wert) konnte in ein paar Fällen nicht berechnet werden, da auf Grund der Unteranpassung nicht alle Labels der vorherzusehenden Klasse berechenbar waren (siehe Abschnitt 6.2.2). In solchen Fällen wurde die Precision (und der F-Wert) gleich null gesetzt (Nullwerte). Da der Recall nur tatsächlich gefundene Instanzen berücksichtigt und eine mögliche Unteranpassung der Daten keinen Einfluss auf den Berechnungsvorgang hat, wird dieser ebenfalls in Tabelle 11 ausgewiesen. Da nur ein einziges Argument bestimmt wird und die Ausgangslage hinsichtlich Vollständigkeit der Labelings sehr gut ist, ist ein hoher F-Wert (ein F-Wert von 1,0 ist maximal erreichbar) zu erwarten.

50 % der Argumentationsframeworks haben über alle Klassifikatoren hinweg einen maximalen F-Wert von 1,0 erreicht. 20 % der Argumentationsframeworks haben einen durchschnittlichen F-Wert von unter 0,5. Dies wird in Abschnitt 6.2.3 näher analysiert. Die in Abschnitt 4 genannten Auswahlmöglichkeiten an den Parametern haben überwiegend zu keinen signifikanten Verbesserungen oder Verschlechterungen geführt (F-Wert lag oft sehr hoch). Folgende Beobachtung wurden punktuell festgestellt:

- K-Nächste Nachbarn: Für $k \in \{7, 9\}$ gab es die besten Ergebnisse.

- Supportvektormaschinen: Die Kernelfunktion Polynomieller Kern konnte oft durch die Unteranpassung nicht bestimmt werden.
- Bei den neuronale Netzen wurden die besten Werte oft mit folgenden Parametern erzielt: Zwischenschichten = 3 Stück a 20 Knoten, Anzahl Epochen = 50, Fehlerwert F = 0,6.

	F-Wert - ohne Nullwerte	**F-Wert - mit Nullwerten**	**Recall**	**Precision - ohne Nullwerte**	**Precision - mit Nullwerten**
Naive Bayes	0,835	0,819	0,838	0,843	0,826
KNN	0,807	0,790	0,806	0,808	0,792
Entscheidungsbaum	0,772	0,772	0,775	0,770	0,770
NN	0,928	0,631	0,841	0,963	0,655
SVM	0,865	0,675	0,833	0,887	0,692

Tabelle 11: Die Tabelle zeigt die verschiedenen Maße der Klassifikationsgüte (Durchschnitt der 50 Argumentationsframeworks) bezogen auf die einzelnen Klassifikatoren.

Tabelle 11 zeigt, dass sich bei einer Berücksichtigung von Nullwerten der F-Wert verschlechtert. Vor allem bei den neuronalen Netzen sowie den Supportvektormaschinen ist es offensichtlich, dass die Precision (und somit der F-Wert) oft nicht berechnet werden kann und der Durchschnitt sich somit verschlechtert. Insgesamt ist festzuhalten, dass die Klassifikationen trotz 100 % Vollständigkeit der Labelings nicht trivial waren und der maximale F-Wert von 1,0 nicht erreicht wurde. Vor allem bei den zuvor erwähnten 20 %, welche näher in Abschnitt 6.2.3 analysiert werden, war es schwierig korrekte Klassifikationen vorzunehmen.

Der Klassifikator Naive Bayes konnte permanent gute Ergebnisse erzielen, da das Problem der Unteranpassung (siehe F-Wert mit Nullwerten Tabelle 11) einen geringen negativen Einfluss hatte. Die K-nächsten Nachbarn waren ebenfalls selten von der Unteranpassung betroffen und haben stets gute Ergebnisse geliefert. Der Entscheidungsbaum ist unanfällig gegenüber Unteranpassung, jedoch ist hier im Durchschnitt der geringste F-Wert berechnet worden. Der Klassifikator Neuronale Netze konnte die besten Ergebnisse erzielen, aber nur dann wenn es keine Nullwerte gab. Dieser Klassifikator ist genauso wie die Supportvektormaschine sehr anfällig für Unteranpassung.

6.2.2 Unteranpassung

Bei den Ergebnissen der Lernvorgänge fällt auf, dass bei manchen Argumentationsframeworks kein F-Wert bestimmt werden konnte, was auf den fehlenden Precision-Wert zurückzuführen ist. In solchen Fällen lagen dem Klassifikator oft unausgegli-

chene Daten zum Trainieren vor. Hierbei war entweder eine Klasse überproportional oft vertreten oder es kam eine Klasse innerhalb der Trainingsdaten nie vor (siehe Abbildung 12). In solchen Fällen wurde des weiteren festgestellt, dass die Anzahl Labelings im Verhältnis zur Gesamtmenge der Argumente gering war. Es kommt somit zur Unteranpassung [13] (engl. Underfitting). In so einem Fall wurde die Precision und somit der F-Wert gleich null gesetzt (Nullwerte).

	Trainingsdaten	**Testdaten**	**Gesamt**
In	21	16	37
Out	549	230	779
Undec	20	7	27

Tabelle 12: Verteilung der zulässigen Labels des Argumentationsframeworks **n160p3q24e**, Argument **a74**. Das Label Out kommt überproportional oft vor.

Tabelle 12 zeigt exemplarisch die Labelverteilung des Arguments **a74** des Argumentationsframeworks **n160p3q24e**. Die Gesamtmenge der zulässigen Labelings beläuft sich auf 843, was bei 160 Argumenten gering ist. Es ist zu sehen, dass das Label Out in Summe 779-mal vorkommt und damit überproportional vertreten ist. Bei der Anwendung des Klassifikators Supportvektormaschine (Kernelfunktion: Radiale Basisfunktion) werden die 16 In-Label sowie die 7 Undec-Label des Testdatensatzes fälschlicherweise als Out klassifiziert, siehe Konfusionsmatrix in Abbildung 13.

		Aktuelle Klasse		
		IN	OUT	UNDEC
Vorhergesagte Klasse	IN			
	OUT	16	230	7
	UNDEC			

Tabelle 13: In der Konfusionsmatrix ist zu sehen, dass innerhalb der Testdaten die 16 In-Label sowie die 7 Undec-Label fälschlicherweise als Out-Label klassifiziert wurden. Es wurde der Klassifikator Supportvektormaschine (Kernelfunktion: Radiale Basisfunktion) verwendet.

6.2.3 Analyse der Argumentationframeworks mit der schwächsten Performance

Es ist auffällig, dass die niedrigsten F-Werte (unteren 20 % der Argumentationsframeworks) aus der Small-Result- sowie aus der BA-Reihe stammen. Für diese Argumentationsframeworks erfolgt hier (siehe Tabelle 16) eine nähere Betrachtung. Dort ist ersichtlich, dass es bei allen 10 Argumentationsframeworks trotz der hohen Anzahl an Labelings sehr wenig Argumente und Attacken gibt. Ferner fällt bei 5 der 10 Argumentationsframeworks auf, dass die durchschnittliche Anzahl der Labels

W nah am Wert $W = 3$ liegt. Dies lässt vermuten, dass es einen kleinen Prozentsatz an Argumenten gibt, welcher alle anderen Argumente angreift. Dies soll am Beispiel des Argumentationsframeworks Small-result-b9 genauer analysiert werden, da hier die Anzahl Attacken gering ist und somit eine nähere Betrachtung des Aufbaus, insbesondere der Angriffsrelationen, stattfinden kann. Es ist zu beobachten, dass das im Argumentationsframework existierende Argument **a1_0** alle anderen Argumente attackiert und im Gegenzug von diesen auch zurück angegriffen wird. Dieses Argument bildet somit den Kern des Argumentationsframeworks (siehe Abbildung 11).

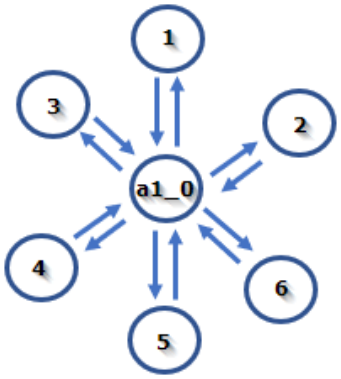

Abbildung 11: Schematische Abbildung des Argumentationsframeworks Small-result-b9. Das Argument **a1_0** bildet den Kern des Argumentationsframeworks.

Tabelle 14 zeigt für das Argumentationsframework Small-result-b9 einen Auszug der zulässigen Labelings, der die Behauptung des sternenförmigen Aufbaus des Argumentationsframeworks stützt. Wird das zentrale Argument **a1_0** bei den Klassifi-

	1	**2**	**3**	**a1_0**	**4**	**5**	**6**	**In(l)**	**Out(l)**	**Undec(l)**
l_1	Out	Out	Out	In	Out	Out	Out	{$a1\_0$}	{1,2,3,4,5,6}	$\emptyset$
l_2	In	In	In	Out	In	In	In	{1,2,3,4,5,6}	{$a1\_0$}	$\emptyset$
l_3	In	Undec	In	Out	Undec	In	In	{1,3,5,6}	{$a1\_0$}	{2,4}

Tabelle 14: Teilauszug der zulässigen Labeling l_1 - l_3 aus dem Argumentationsframework Small-result-b9 (Auszug beschränkt auf 7 Argumente). Es ist zu sehen, dass das Argument **a1_0** das zentrale Argument ist.

zierungsvorgängen als das zu bestimmende Argument ausgewählt, verbessert sich der F-Wert bei allen Klassifikatoren auf den Maximalwert von 1,0. Dies ist nachvollziehbar, da das Argumentationsframework alleinig auf das Angriffs- bzw. Abwehrverhalten dieses Arguments ausgelegt ist. Dies bestätigt auch Tabelle 15, welche auszugsweise die Anzahl eingehender sowie ausgehender Attacken der Argumente **a1_0**, **1** sowie **2** anzeigt. Wenn ein einzelnes Argument im Verhältnis zu den an-

	a1_0_E	a1_0_A	1_E	1_A	2_E	2_A
Small-result-b9	8	15	1	1	1	1

Tabelle 15: Die Tabelle zeigt sinngemäß für die 3 Argumente (**a1_0**, **1** sowie **2**) die unproportionale Verteilung der Anzahl an ausgehenden Attacken (_A) sowie eingehenden Attacken (_E) des Argumentationsframeworks Small-result-b9.

deren Argumenten überproportional oft bei den ausgehenden sowie eingehenden Attacken vorkommt, ist nur für dieses eine Argument eine gute Klassifikationsgüte bestimmbar. Bei den restlichen 9 Argumentationsframeworks aus Tabelle 16 wurden bei einer ersten Analyse ebenfalls sternenförmige Anordnungen von mehreren Argumenten beobachtet. Es sind jedoch weitere Faktoren nicht auszuschließen.

AF	Anzahl Argumente	Anzahl Attacken	Anzahl Admissible Labelings	Durchs. Anzahl Labels	KNN
Small-result-b29	16	27	30000	3,00	**0,435**
T-1-BA_100_10_3	101	111	30000	1,19	**0,417**
Small-result-b36	19	96	30000	2,89	**0,413**
Small-result-b39	17	56	30000	3,00	**0,388**
C-1-BA_60_60_2	61	97	30000	1,31	**0,387**
B-1-BA_40_30_3	41	53	30000	1,46	**0,381**
Small-result-b28	383	32768	30000	1,04	**0,379**
Small-result-b21	23	128	30000	1,87	**0,322**
Small-result-b9	16	23	30000	2,94	**0,189**
Small-result-b85	24	129	30000	2,29	**0,000**

Tabelle 16: Hier die schlechtesten 20 % der Argumentationsframeworks gemessen anhand des F-Werts des Klassifikators K-Nächste Nachbarn.

6.2.4 Schlussfolgerung

Wird auf die in Tabelle 16 genannten 10 Argumentationsframeworks verzichtet, verbessern sich die F-Werte (mit Nullwerte - siehe Tabelle 11) der jeweiligen Klassifikatoren um mindesten 0,1. Interessant ist hierbei, dass die beiden Klassifkatoren Neuronale Netze sowie Supportvektormaschinen weiterhin sehr anfällig für Unteranpassung sind. Zusammengefasst sind die wichtigsten Erkenntnisse dieser Testreihe, dass wenn auf unausgeglichene Labelings zurückgegriffen wird oder die Anzahl an ausgehenden sowie eingehenden Attacken innerhalb eines Argumentationsframeworks unausgeglichen ist, sich dies negativ auf die Klassifikationsgüte auswirkt.

7 Testreihe: Zulässige Labelings mit abnehmender Vollständigkeit

In dieser Testreihe wird untersucht, wie sich eine abnehmende Vollständigkeit der Testdaten bei der Klassifizierung anhand der zulässigen Labelings auf die Klassifikationsgüte auswirkt. Grundlage bilden hierbei die 50 Argumentationsframeworks aus Abschnitt 6.2.1. Die Klassifikatoren werden nochmals mit vollständigen Trainingsdaten trainiert, die Testdaten hingegen sind dieses Mal unvollständig (70 %, 50 % und 30 % Vollständigkeit, siehe Abschnitt 4). Anschließend wird die Klassifikationsgüte, die mit den vollständigen Testdaten (100 %) berechnet wurde, mit dem Ergebnissen aus den unvollständigen Testdaten (70 %, 50 % und 30 %) verglichen.

7.1 Aufbau und Durchführung

Vorab eine Übersicht der wichtigsten Parameter zur Durchführung dieser Testreihe:

- Semantik: Zulässige Semantik
- Anzahl Argumentationsframeworks: 20 (ICCMA'19 Datensatz)
- Vollständigkeit Testdaten: 100 %, 70 %, 50 %, 30 %

Aus den 50 Argumentationsframeworks aus Abschnitt 6.2.1 werden für diese Testreihe 20 Stück ausgewählt, die wiederum ursprünglich aus dem Set 1 und 2 stammen (Wertebereiche mit Fokus Anzahl Argumente bzw. Anzahl Attacken - Abschnitt 5.1). Tabelle 17 zeigt die kummulierten Recall-Werte. Der Recall ist als sehr gut zu interpretieren (da knapp bei 1,0), die Testsdaten lagen hier zu 100 % vollständig vor. Die Recall-Werte (100 % Vollständigkeit Testdaten) sind für den angestrebten Vergleich auch in Abbildung 12 dargestellt. In der zweiten Testreihe soll die Frage näher analysiert werden, wie gut ein trainierter Klassifikator trotz abnehmender Vollständigkeit der Testdaten die zulässigen Labels eines Zielarguments vorhersagen kann. Bei

Klassifikator	**Naive Bayes**	**KNN**	**Entscheidungs-baum**	**NN**	**SVM**
Recall 100 % Testdaten:	0,941	0,944	0,952	0,939	0,924

Tabelle 17: Hier die kummulierten Recall-Werte aus den 20 ausgewählten Argumentationsframeworks. Die Testdaten liegen zu 100 % vollständig vor.

jedem Argumentationsframework werden die Klassifikatoren analog wie in der ersten Versuchsreihe mit vollständigen Trainingsdaten (70 % der zulässigen Labelings) trainiert. Anschließend ist jedoch mit den folgenden prozentualen Vollständigkeiten der Testdaten (30 % der zulässigen Labelings) zu evaluieren, wie stark sich der Recall aus Tabelle 17 verschlechtert:

- 70 % Vollständigkeit Testdaten.
- 50 % Vollständigkeit Testdaten.
- 30 % Vollständigkeit Testdaten.

7.2 Ergebnisse

Bei der Durchführung dieser Testreihe ist als erstes aufgefallen, dass für den Klassifikator Entscheidungsbäume (mit dem ID3-Algorithmus) sich bei abnehmender Vollständigkeit der Testdaten keine Ergebnisse berechnen ließen. Bei genauerer Analyse des ID3-Algorithmus ist dieses Verhalten erklärbar. In der Trainingsphase werden bei diesem hierarchischen Ansatz die jeweiligen Attribute (der Algorithmus interpretiert die Argumente als Attribute) mit dem höchsten Informationsgewinn (siehe Abschnitt 2.2.4) bestimmt. Wenn diese Attribute bei den Testdaten fehlen, kann der Algorithmus in Weka keine Ausgabe erzeugen. Dieses Verhalten ist beim maschinellen Lernen durchaus bekannt [16]. Für diese Testreihe ist der Entscheidungsbaum somit kein geeigneter Klassifikator.

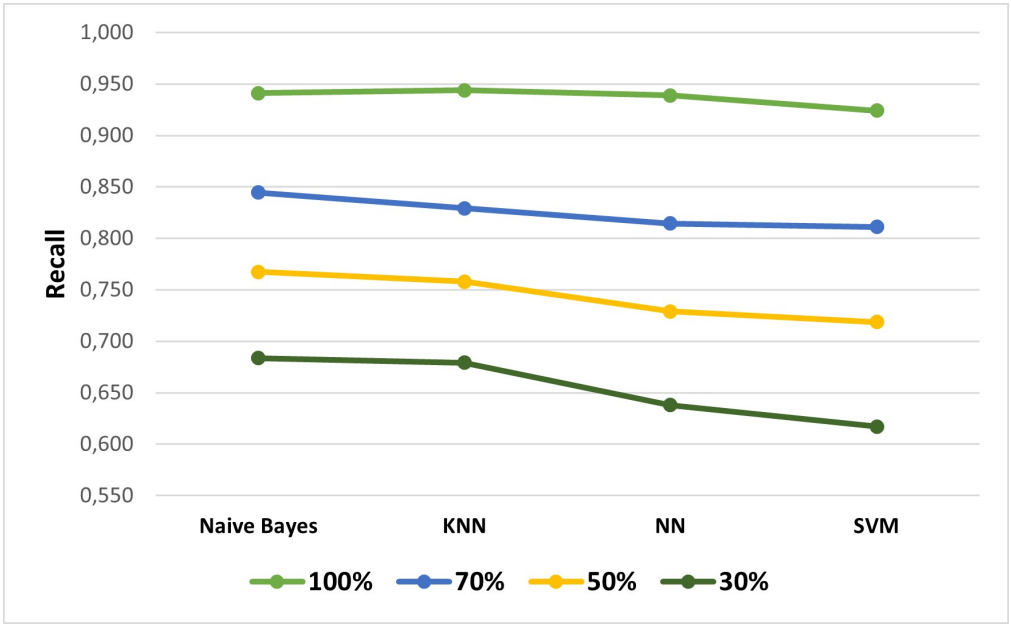

Abbildung 12: Die Abbildung zeigt grafisch den Recall bei abnehmender Vollständigkeit der Testdaten der verbliebenen 4 Klassifikatoren.

Abbildung 12 zeigt den Recall der 4 Klassifikatoren (basierend auf den Klassifikationen anhand der zulässigen Labelings der 20 Argumentationsframeworks) bei abnehmender Vollständigkeit der Testdaten. Wie aus dem grafischen Verlauf aus Ab-

bildung 12 ersichtlich ist, sind bei 70 % Vollständigkeit der Testdaten gute Recall-Werte ermittelbar (Recall > 0,8). Ein rein hypothetischer F-Wert würde hier ebenfalls sehr gute Ergebnisse (F-Wert > 0,8) liefern, da die Werte der Precision bei den 20 Argumentationsframeworks (bis auf 3 Stück mit Nullwerten - siehe Unteranpassung) sehr nah am Recall lagen - siehe Abbildung 13.

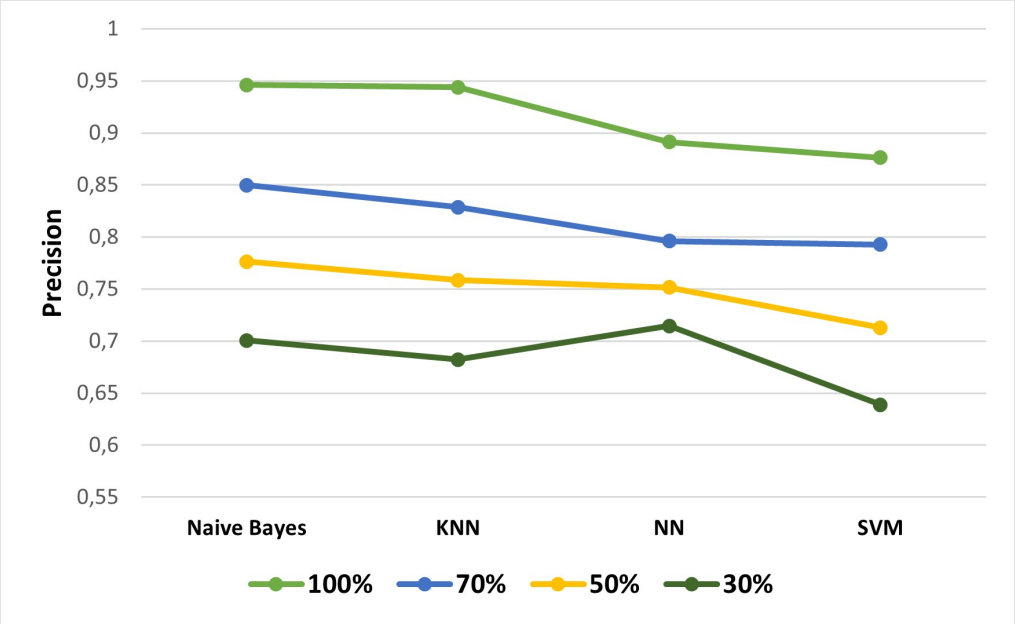

Abbildung 13: Hier grafisch die Precision (mit Nullwerten) bei abnehmender Vollständigkeit der Testdaten der verbliebenen 4 Klassifikatoren.

Bei 50 % Vollständigkeit der Testdaten können die beiden Klassifikatoren Naive Bayes sowie K-Nächste Nachbarn ebenfalls gute Werte abbilden, da der Recall knapp bei 0,8 liegt. Die beiden Klassifikatoren Neuronale Netze sowie Supportvektormaschinen schneiden hier etwas schlechter ab. Bei 30 % Vollständigkeit sind über alle Klassifikatoren hinweg keine guten Ergebnisse mehr ermittelbar. Es ist auffällig, dass die beiden Klassifikatoren Neuronale Netze sowie Supportvektormaschinen bei allen prozentual abnehmenden Vollständigkeiten etwas schlechter abschnitten. Es ist anzunehmen, dass sich mit einer Optimierung der Parametereinstellungen der beiden Klassifikatoren Neuronale Netze sowie Supportvektormaschinen noch bessere Ergebnisse erzielen lassen.

8 Testreihe: Vollständige Labelings

In dieser Testreihe wird untersucht, wie gut sich die Klassifikatoren mit vollständigen Labelings der 50 Argumentationsframeworks aus Abschnitt 6.2.1 trainieren lassen. Es wird ähnlich wie in den vorangegangen Testreihen die Klassifikationsgüte bestimmt sowie weitere Auffälligkeiten erörtert. Die Ergebnisse dieser Testreihe bilden die Grundlage für die nächste Testreihe, wo das Verhalten mit unvollständigen Testdaten (bei vollständigen Labelings) analog wie in der zweiten Testreihe näher analysiert wird. Es ist zu erwarten, dass der Recall, die Precision sowie daraus folgend der F-Wert bei den vollständigen Labelings mindestens genauso gut ist wie bei den zulässigen Labelings. Er müsste sogar leicht höher liegen, da die vollständigen Labelings höhere Anforderungen an die Argumente selbst stellen und somit selektiver sind (siehe Abschnitt 2). Dadurch, dass die vollständige Semantik restriktiver ist, wird vermutlich die Anzahl der vollständigen Labelings stark sinken.

8.1 Aufbau und Durchführung

Vorab eine Übersicht der wichtigsten Parameter zur Durchführung dieser Testreihe:

- Semantik: Vollständige Semantik
- Anzahl Argumentationsframeworks: 6 (ICCMA'19 Datensatz)
- Vollständigkeit Testdaten: 100 %

Für die Durchführung dieser Testreihe wird wieder auf die in Abschnitt 4 vorgestellten Wertebereiche der einzelnen Klassifikationsalgorithmen zurückgegriffen:

- Naive Bayes
- K-Nächste Nachbarn (KNN)
- Entscheidungsbaum
- Neuronale Netze
- Supportvektormaschinen

Zu den 50 Argumentationsframeworks der ICCMA [4] sind die vollständigen Extensionen ebenfalls verfügbar, weshalb diese direkt mit Tweety eingelesen und als vollständige Labelings augegeben werden können. Für die Duchführung der Testreihe ist eine Datengrundlage von mindestens 1.000 vollständigen Labelings sinnvoll, was leider auf Grund der höheren Anforderungen an die Argumente nur bei 6 Argumentationsframeworks zutrifft. Tabelle 18 zeigt die 6 Argumentationsframeworks. Bei 2 Argumentationsframeworks sind mehr als 30.000 vollständige Labelings vorhanden, beide liegen mit ihrer Größe jedoch unter 40 Megabyte und werden somit vollständig für die Klassifizierung verwendet. Analog der vorherigen Testreihen, wird pro Argumentationsframework 70 % der vollständigen Labelings für das Trainieren und 30 % für das Testen verwendet.

AF	Anzahl Argumente	Anzahl Attacken	Knotengrad	Anzahl Vollständige Labelings
A-1-BA_120_30_3	121	157	1,30	19683
A-1-caravan-or-us.gml.80	19	37	1,95	1624
B-1-massachusetts-archiver_ 20090912_0208.gml.50	34	58	1,71	33534
C-1-BA_60_60_2	61	97	1,59	6561
T-1-BA_80_30_3	81	105	1,30	43740
T-2-ferry2.pfile-L3-C3-03.pddl.2.cnf	933	1838	1,97	4627

Tabelle 18: Hier die 6 Argumentationsframeworks, die jeweils mindestens 1.000 vollständige Labelings besitzen und in dieser Testreihe näher analysiert werden.

8.2 Ergebnisse

Tabelle 19 zeigt den durchschnittlichen F-Wert, die Precision sowie den Recall der 6 Argumentationsframeworks aus Tabelle 18. Es kamen bei keinem einzigen Nullwerte vor. Wie erwartet sind die Ergebnisse als sehr gut zu bewerten, da der F-Wert sehr nah bei 1,0 liegt. Für die genannten Argumentationsframeworks aus Tabelle

	Naive Bayes	KNN	Entscheidungsbaum	NN	SVM
Recall:	0,940	0,984	1,000	1,000	1,000
Precision:	0,958	0,985	1,000	1,000	1,000
F-Wert:	0,931	0,982	1,000	1,000	1,000

Tabelle 19: Hier die kummulierten Ergebnisse der 6 Argumentationsframeworks aus Tabelle 18. Die Ergebnisse sind als sehr gut zu interpretieren.

18 wurde ebenfalls der durchschnittliche Recall der zulässigen Labelings berechnet, um diesen Wert dann mit dem Recall aus Tabelle 19 zu vergleichen. Dieser Vergleich ist grafisch in Abbildung 14 zu sehen. Wichtig ist es hierbei zu erwähnen, dass sich die zu klassifizierenden Zielargumente bei den vollständigen Labelings gegenüber den zulässigen Labelings unterscheiden. Ein Zielargument, das gemäß Anforderung aus Abschnitt 6.1 mindestens einmal das Label Out, In sowie Undec bei den zulässigen Labelings besitzt, kann für die Vorhersage bei den vollständigen Labelings ungeeignet sein. Aus diesem Grund wurden zum Teil bei den vollständigen Labelings andere Zielargumente bestimmt (welche die Kriterien aus Abschnitt 6.1 erfüllen). Über alle Klassifikatoren hinweg wurden bei den vollständigen Labelings bessere Ergebnisse erzielt als bei den zulässigen Labelings.

Die durchschnittliche Recall-Abweichung betrug hierbei 0,129. Dieser Vergleich wurde analog mit dem F-Wert durchgeführt, siehe Abbildung 15. Hierbei ist jedoch anzumerken, dass bei den zulässigen Labelings der 6 Argumentationsframeworks die

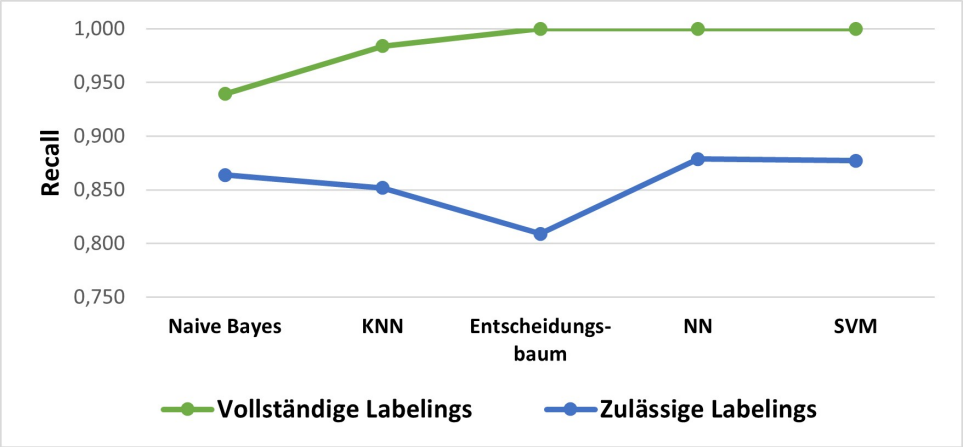

Abbildung 14: Der Recall schneidet mit vollständigen Labelings als Datengrundlage besser ab als mit zulässigen Labelings.

Precision auf Grund von häufigen Nullwerten sehr niedrig war. Somit ist der F-Wert bei den zulässigen Labelings um einiges niedriger als bei den vollständigen Labelings. Bezüglich der Parameter für die jeweiligen Klassifikatoren haben sich einige Unterschiede, verglichen zu den zulässigen Labelings, hinsichtlich der Klassifikationsgüte ergeben.

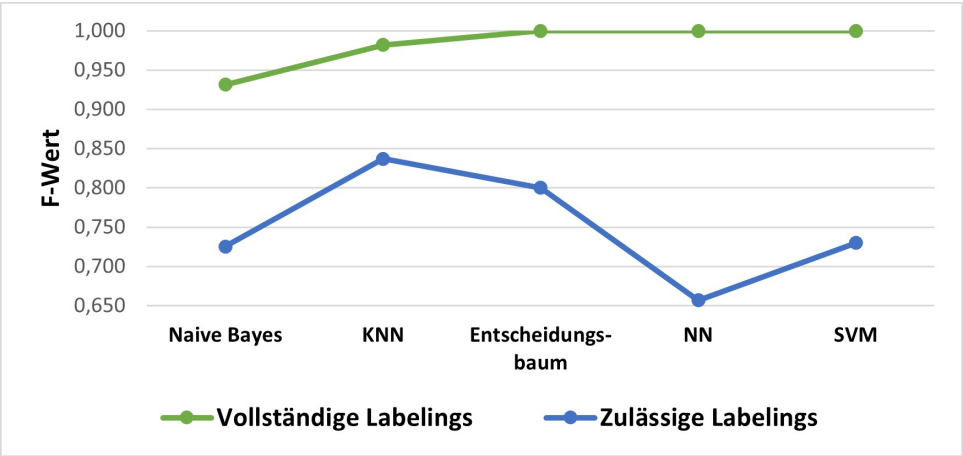

Abbildung 15: Der F-Wert schneidet mit vollständigen Labelings als Datengrundlage erheblich besser ab (zulässige Labelings hatten oft Nullwerte).

Beim Klassifikator K-Nächste Nachbarn gab es bei allen Argumentationsframeworks für $k \in \{3, 5\}$ die besten Ergebnisse. Für Werte $k > 7$ hat sich der Recall stark verschlechtert. Die neuronalen Netze haben bei allen Argumentationssystemen den

maximalen Recall von 1,0 erreicht. Hierbei ist aufgefallen, dass die Parameter mit der geringeren Anforderung an die Zwischenschichten (Zwischenschichten = 2 Stück - einmal 10 sowie 5 Knoten, Anzahl Epochen = 50, F = 0,6) bereits ausgereicht hat, um den höchstmöglichen Recall zu bestimmen. Bei den zulässigen Labelings hatten neuronale Netze die längste Berechnungszeit bei den Trainingsvorgängen (bis zu 10 Minuten), wohingegen bei den vollständigen Labelings die Laufzeit wesentlich schneller war. Bei den Supportvektormaschinen mit der Kernelfunktion Polynomieller Kern ist aufgefallen, dass das in Abschnitt 6.2.2 genannte Problem der Unteranpassung mit einer hohen Zahl an Labelings (mindestens 30.000 Labelings) nicht mehr vorkommt. Leider hat sich dafür die Berechnungszeit mit bis zu 20 Minuten auf die bisher längste gemessene Berechnungszeit pro Trainingsvorgang erhöht.

9 Testreihe: Vollständige Labelings mit abnehmender Vollständigkeit

In dieser Testreihe wird untersucht, wie sich eine abnehmende Vollständigkeit der Testdaten bei der Klassifizierung anhand der vollständigen Labelings auf die Klassifikationsgüte auswirkt (ähnlich zweiten Testreihe, jedoch auf vollständige Labelings bezogen). Grundlage bilden hierbei die bekannten 6 Argumentationsframeworks aus Tabelle 18.

9.1 Aufbau und Durchführung

Vorab eine Übersicht der wichtigsten Parameter zur Durchführung dieser Testreihe:

- Semantik: Vollständige Semantik
- Anzahl Argumentationsframeworks: 6 (ICCMA'19 Datensatz)
- Vollständigkeit Testdaten: 100 %, 70 %, 50 %

Bei jedem der 6 Argumentationsframework aus Tabelle 18 werden die Klassifikatoren mit den jeweils vollständigen Trainingsdaten (70 % der vollständigen Labelings) trainiert. Anschließend ist jedoch mit den folgenden prozentualen Vollständigkeiten der Testdaten (30 % der vollständigen Labelings) zu evaluieren, wie stark sich der Recall sowie die Precision verschlechtert:

- 70 % Vollständigkeit Testdaten.
- 50 % Vollständigkeit Testdaten.

In der zweiten Testreihe wurde festgestellt, dass es bei einer 30-prozentigen Vollständigkeit der Testdaten zu sehr schlechten Ergebnissen kommt. Erste Tests mit den 6 Argumentationsframeworks aus Tabelle 18 haben dies bestätigt, weshalb dieser Prozentsatz in dieser Testreihe nicht weiter verfolgt wird. Wie bereits ebenfalls

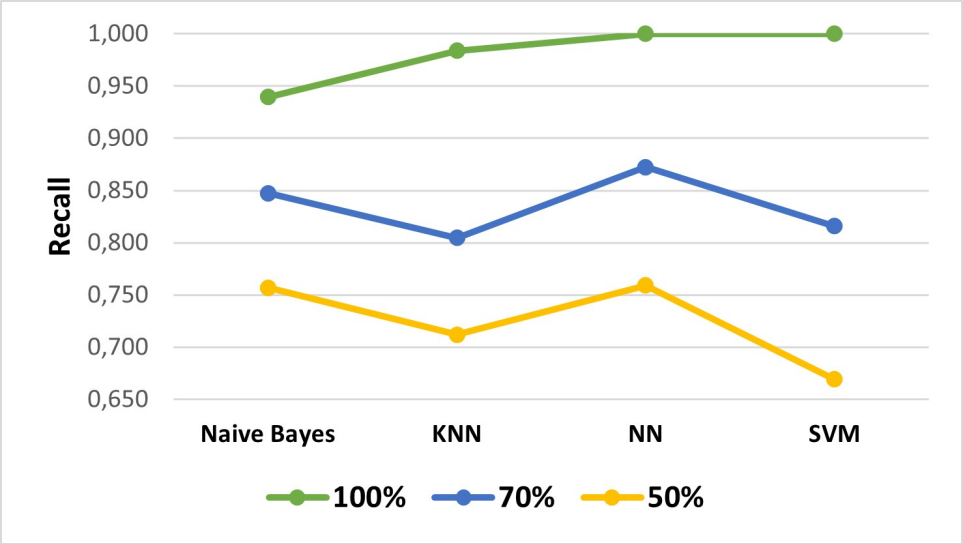

Abbildung 16: Die Abbildung zeigt grafisch den Recall bei abnehmender Vollständigkeit der Testdaten der verbliebenen 4 Klassifikatoren basierend auf den vollständigen Labelings aus Tabelle 18.

in der zweiten Testreihe erörtert, ist der Klassifikator Entscheidungsbäume bei abnehmenden Vollständigkeiten der Testdaten nicht anwendbar und wird somit auch in dieser Testreihe nicht weiter berücksichtigt.

9.2 Ergebnisse

Abbildung 16 zeigt den Recall der 4 Klassifikatoren basierend auf den Klassifikationen anhand der vollständigen Labelings der 6 Argumentationsframeworks. Wie aus dem grafischen Verlauf ersichtlich ist, sind bei 70 % Vollständigkeit der Testdaten ebenfalls gute Recall-Werte ermittelbar (Recall > 0,8). Der F-Wert würde hier ebenfalls gute Ergebnisse (F-Wert > 0,8) liefern, da die Werte der Precision bei den 6 Argumentationsframeworks sehr nah am Recall liegen (siehe Abbildung 17). Bei 50 % Vollständigkeit der Testdaten konnte beim Recall kein Klassifikator den Wert 0,8 erreichen und ist somit nicht mehr als gut zu interpretieren. Bei der Precision kommen nur die beiden Klassifikatoren Neuronale Netze sowie Supportvektormaschinen auf einen Wert größer 0,8. Es lässt sich somit schlussfolgern, dass sich das Verhalten der Klassifikatoren bei abnehmender Vollständigkeit bei den vollständigen Labelings ähnlich verhält wie bei den zulässigen Labelings.

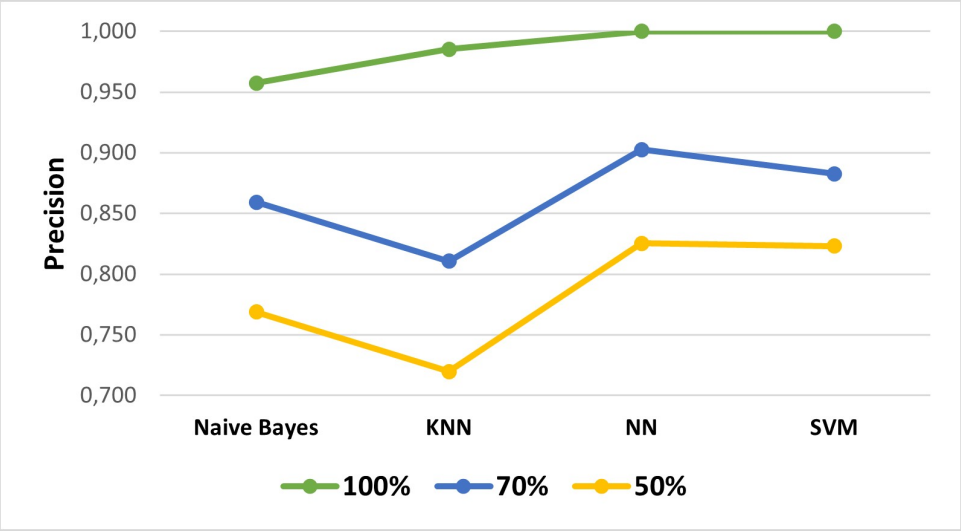

Abbildung 17: Die Abbildung zeigt grafisch die Precision bei abnehmender Vollständigkeit der Testdaten der verbliebenen 4 Klassifikatoren basierend auf den vollständigen Labelings aus Tabelle 18.

10 Testreihe: Zulässige Labelings mit Zusatzinformationen

Abschließend soll in dieser Testreihe untersucht werden, ob sich die Klassifikationsgüte anhand der zulässigen Labelings bei der Hinzunahme von Zusatzinformationen verbessert. Die zulässigen Labelings werden hierbei um Informationen hinsichtlich des Knotengrads erweitert. Es ist anzunehmen, dass sich die Klassifikationsgüte durch die Zusatzinformationen verändert, im besten Fall sogar verbessert.

10.1 Aufbau und Durchführung

Vorab eine Übersicht der wichtigsten Parameter zur Durchführung dieser Testreihe:

- Semantik: Zulässige Semantik
- Anzahl Argumentationsframeworks: 5 Stück
  - 3 Argumentationsframeworks aus der ersten Testreihe (ICCMA'19 Datensatz)
  - 2 via Tweety selbst erstellte Argumentationsframeworks
- Vollständigkeit Testdaten: 100 %

Für die Durchführung dieser Testreihe wird wieder auf die in Abschnitt 4 vorgestellten Wertebereiche der einzelnen Klassifikationsalgorithmen zurückgegriffen:

- Naive Bayes
- K-Nächste Nachbarn (KNN)
- Entscheidungsbaum
- Neuronale Netze
- Supportvektormaschinen

Analog der vorherigen Testreihen, werden bei allen betrachteten Argumentationsframeworks 70 % der vollständigen Labelings für das Trainieren und 30 % für das Testen verwendet. Tabelle 20 zeigt die 5 für diese Testreihe relevanten Argumentationsframeworks mit ihren Parametern. Die ersten drei Argumentationsframe-

AF	**Anzahl Argumente**	**Anzahl Attacken**	**Anzahl vollständige Labelings**
A-1-caravan-or-us.gml.80	19	37	4708
C-1-bluestar-or-us.gml.20	24	29	2160
Small-result-b9	16	23	30000
AF1	20	20	1920
AF2	18	11	2496

Tabelle 20: Hier die 5 Argumentationsframeworks, deren zulässige Labelings in dieser Testreihe mit Zusatzinformationen erweitert werden. Erstmalig werden auch zwei eigene Argumentationsframeworks erstellt (AF1 und AF2).

works sind Bestandteil der ersten Testreihe, der F-Wert wurde somit schon ermittelt und ist nochmals in Tabelle 21 zu sehen. Die letzten beiden Argumentationsframeworks in Tabelle 20 wurden via Tweety erstellt. Beim Argumentationsframework AF1 wurde neben der zu erstellenden Anzahl an Argumenten ($Anzahl = 20$) noch die Wahrscheinlichkeit mitgegeben, dass eine Attacke zwischen zwei Argumenten **A** und **B** existiert ($Wahrscheinlichkeit = 5\,\%$). Beim Argumentationsframework AF2 wurde bis auf andere Parameter analog vorgegangen ($Anzahl = 18$, $Wahrscheinlichkeit = 4\,\%$). Die F-Werte aus Tabelle 21 bilden somit die Ausgangslage für diese Testreihe, da hier noch keine Zusatzinformationen enthalten sind.

Die zulässigen Labelings der 5 Argumentationsframeworks werden nun um die Zusatzinformationen Anzahl eingehender Attacken sowie Anzahl ausgehender Attacken erweitert, und zwar für alle Argumente des jeweiligen Argumentationsframeworks. Tabelle 22 zeigt auszugsweise anhand der zulässigen Labelings $l1$ - $l4$ für das Argumentationsframework AF1 an, wie bei 3 Argumenten (**a12**, **a14**, **a15**) die Zusatzinformation eingehende sowie ausgehende Attacken mit in die Labelings fließen. Es werden nun alle zulässigen Labelings der 5 Argumentationsframeworks

AF	Naive Bayes	KNN	Entscheidungs-baum	NN	SVM
A-1-caravan-or-us.gml.80	0,852	0,914	0,910	0,941	0,945
C-1-bluestar-or-us.gml.20	0,843	0,954	0,980	0,985	0,969
Small-result-b9	0,502	0,189	0,188	0,400	0,480
AF1	0,833	0,785	0,719	0,827	0,833
AF2	0,689	0,570	0,472	0,624	0,655

Tabelle 21: In dieser Tabelle ist der F-Wert für alle Klassifikatoren zu sehen, welcher sich in dieser Testreihe durch die Hinzunahme von Zusatzinformationen verbessern soll.

	a14	**a14_E**	**a14_A**	**a15**	**a15_E**	**a15_A**	**a12**	**a12_E**	**a12_A**
l1	In	1	1	Out	2	1	In	0	0
l2	Undec	1	1	Out	2	1	In	0	0
l3	Undec	1	1	Out	2	1	Undec	0	0
l4	In	1	1	Out	2	1	Undec	0	0

Tabelle 22: In dieser Tabelle werden für 3 Argumente (**a12**, **a14**, **a15**) die zulässigen Labelings l1 - l4 des Argumentationsframeworks AF1 (siehe Tabelle 20) angezeigt. Jedes Argument wurde mit der Zusatzinformationen Anzahl eingehende Attacken (_E) sowie Anzahl ausgehende Attacken (_A) erweitert.

mit den Zusatzinformationen erweitert, um anschließend nochmals den F-Wert für alle Klassifikatoren zu berechnen. Die Ergebnisse werden dann im Anschluss mit den F-Werten aus Tabelle 21 verglichen.

10.2 Ergebnisse

Es wurde festgestellt, dass sich der F-Wert aus Tabelle 21 für kein einziges Argumentationsframework verändert hat, die Ergebnisse blieben trotz der Zusatzinformationen überall identisch. Auch der Recall sowie die Precision haben sich nicht verändert. Es wurden ferner unterschiedlichste Parameter für die jeweiligen Klassifikatoren getestet, doch auch hier haben die Zusatzinformationen keine Auswirkung auf die Klassifikationsgüte gehabt und führten zu gleichen Ergebnissen. Zwar haben sich die Erwartungen an diese Testreihe nicht erfüllt, jedoch konnte eine weitere wichtige Beobachtung gemacht werden.

Das Argument **a12** in Tabelle 22 wird von keinen Argumenten angegriffen bzw. greift selbst keine anderen Argumente an. Die Anzahl eingehender Attacken $a12\_E$ sowie ausgehender Attacken $a12\_A$ ist 0. Es ist somit ein einzelnes Argument ohne direkten Bezug zum restlichen Argumentationsframework AF1. Die Klassifikations-

güte für solche Argumente (als vorherzusehendes Zielargument) ist sehr schlecht, da sie keinen Bezug zu den anderen Argumenten hat und die Klassifikatoren somit keine Möglichkeit haben, valide Muster zu erkennen. Dies wird bestätigt, wenn man im Argumentationsframework AF1 das Argument **a12** als das zu bestimmende Zielargument auswählt und anschließend die F-Werte für alle Klassifikatoren ermittelt (alle F-Werte liegen unter 0,5). Diese Beobachtung ergänzt die Erkenntnis aus der ersten Testreihe (siehe Abschnitt 6.2.3), dass eine unproportionale Verteilung der Anzahl an ausgehenden sowie eingehenden Attacken eines Argumentationsframeworks sich negativ auf die Klassifikationsgüte eines Zielarguments auswirkt.

11 Zukünftige Arbeit

Fokus in dieser Masterarbeit waren die Labelings der zulässigen und vollständigen Semantik. Speziell die einzelnen Testreihen hatten immer eine der beiden Semantiken thematisiert. In der zukünftigen Forschung sollten weitere Testreihen mit noch restriktiveren Semantiken, wie zum Beispiel der bevorzugten und stabilen Semantik, durchgeführt und mit den Ergebnissen der anderen Semantiken verglichen werden.

Auf Grund des Speicherplatzbedarfs und der Verarbeitungsmöglichkeiten von Weka (siehe Abschnitt 5) musste eine Begrenzung auf maximal 1000 Argumente sowie 30.000 zulässige Extensionen pro Argumentationsframework festgelegt werden. Dies hatte u.a. zur Folge, dass 50 Argumentationsframeworks aus dem ICCMA'19 Datensatz [4] nicht weiter in Betracht kamen. Diese 50 Argumentationsframeworks gilt es in der zukünftigen Forschung zu berücksichtigen, dadurch wird zwangsläufig auch dem Laufzeitverhalten eine wichtigere Rolle zugeschrieben. Auch das Aufheben des Limits von 30.000 zulässigen Extensionen pro Argumentationsframework würde den Klassifikatoren eine viel größere Datengrundlage ermöglichen, womit in zukünftigen Arbeiten eventuell das Problem der Unteranpassung vermieden oder reduziert werden kann. Bei der Aufhebung der Restriktionen sollten die gestiegenen Anforderungen zur Verarbeitungen von mehreren Terrabytes an Extensionen berücksichtigt werden. Dies hätte auch für zukünftige Arbeiten zur Folge, dass statt dem Data-Mining Tool Weka hier auf eine Big-Data-Software ausgewichen werden müsste, mit welcher ebenfalls Klassifikationen vorgenommen werden können.

Wenn durch eine Aufhebung der zuvor genannten Restriktionen eine größere Datenmenge an Labelings zur Verfügung steht, könnte eine zielgerichtete Optimierung der Parameter der 3 Klassifikatoren K-Nächste Nachbarn, Neuronale Netze sowie Supportvektormaschinen einen größeren Einfluss auf die Klassifikationsgüte haben. In zukünftigen Arbeiten könnten zum Beispiel beim Klassifikator K-Nächste Nachbarn noch zahlreich weitere Distanzmaße evaluiert werden. Des Weiteren wäre mit Sicherheit auch der Klassifikator Neuronale Netze mit dem Fokus auf den Graph

Convolutional Networks (ähnlich der Studie von Kuhlmann [14], Abschnitt 3) sehr interessant. Ganz allgemein wäre eine Hinzunahme weiterer Klassifikationsalgorithmen durchaus denkbar. Statt der in dieser Masterarbeit verwendeten Aufteilung in einen Trainings- sowie Testdatensatz, ist auch die in Abschnitt 2 kurz beschriebene Option der Kreuzvalidierung interessant und hätte eventuell zu noch besseren Ergebnissen geführt. Dies würde sich vor allem für Argumentationsframeworks anbieten, die eine geringe Anzahl an Labelings besitzen.

12 Fazit

In dieser Masterarbeit wurde in verschiedenen Testreihen untersucht, ob es möglich ist mit Hilfe von Klassifikationsalgorithmen, die als Datengrundlage die zulässigen oder vollständigen Labelings eines Argumentationsframeworks verwenden, das Label eines Zielarguments vorherzusagen. Hierbei wurde mit Hilfe des Data-Mining-Tools Weka auf die von der ICCMA [4] bereitgestellten Labelings zugegriffen. Für eine gleichbleibende Auswahl von Klassifikationsalgorithmen wurde anschließend in verschiedenen Testreihen die Maße der Klassifikationsgüte für ein vorherzusehendes Zielargument bestimmt. Die Ergebnisse der einzelnen Testreihen wurden hierbei miteinander verglichen. Ziel war es herauszufinden, inwieweit sich unterschiedliche Semantiken oder auch eine gegebene Unvollständigkeit bezüglich der Labelings auf die Klassifikationsgüte auswirken.

Anhand der ersten beiden Testreihen lässt sich zusammenfassen, dass basierend auf den zulässigen Labelings eines Argumentationsframeworks die Labels mit den verschiedenen Klassifikationsalgorithmen gut vorhersagbar sind. Dies trifft auch dann zu, wenn die Testdaten nur zu 70 % vorhanden sind (siehe Abschnitt 7). Bei weniger als 70 % Vollständigkeit der Testdaten sind keine guten Ergebnisse mehr zu erwarten. Lediglich die beiden Klassifikatoren Naive Bayes sowie K-Nächste Nachbarn konnten bei einer Vollständigkeit von 50 % noch annähernd gute Ergebnisse liefern. Der Klassifikator Entscheidungsbaum scheidet für alle Testreihen mit einer abnehmenden Vollständigkeit komplett aus (siehe Abschnitt 7). Die beiden Klassifikatoren Naive Bayes sowie K-Nächste Nachbarn schnitten im Allgemeinen mit am besten ab, was die F-Werte der ersten beiden Testreihe bestätigen. Das Problem der Unteranpassung (Abschnitt 6) lässt sich eventuell mit einem anderen Ansatz des überwachten Lernens vermeiden, wurde aber in dieser Masterarbeit nicht weiter verfolgt.

In der dritten und vierten Testreihe wurde festgestellt, dass die Klassifikationsgüte auf Grund der vollständigen Semantik steigt (siehe Abschnitt 8). Leider sinkt durch die höheren Anforderungen an die Argumente die Anzahl der vollständigen Labelings stark. Ist die Anzahl der vollständigen Labelings zu stark gesunken, wird von einer Anwendung von Klassifikatonsalgorithmen abgeraten. Die vollständigen Labelings der Argumentationsframeworks aus der dritten und vierten Testreihe waren

nicht von Unteranpassung betroffen. Hier haben die beiden Klassifikatoren Neuronale Netze sowie die Suportvektormaschinen die besten Ergebnisse geliefert. In der vierten Testreihe wurde ebenfalls das Verhalten bei abnehmender Vollständigkeit der Testdaten der vollständigen Labelings untersucht. Auch hier waren bei einer Vollständigkeit von 70 % gute Ergebnisse hinsichtlich der Klassifikationsgüte ermittelbar. Bei 50 % Vollständigkeit konnte lediglich der Klassifikator neuronale Netze annähernd gute Ergebnisse liefern.

In der letzten Testreihe wurde die Auswirkung der Hinzunahme von zusätzlichen Informationen mit in die zulässigen Labelings eines Argumentationsframeworks untersucht. Es wurde für jedes Argument eines Argumemtationsframeworks die Anzahl an ausgehenden sowie eingehenden Attacken als zusätzliche Information mit in die zulässigen Labelings aufgenommen. Leider hatte dies keine Auswirkungen auf die Ergebnisse, die Klassifikationsgüte blieb unverändert.

Als wichtigste Erkenntnis dieser Masterarbeit bleibt festzuhalten, dass die Analyse der Metadaten eines Argumentationsframeworks entscheidend wichtig ist. Von ihr ist es abhängig, ob die Labels eines Arguments durch Klassifikationsalgorithmen vorhersagbar sind oder nicht. Wichtige Kennzahlen der Metadateninformationen wurden in Abschnitt 5 näher analysiert. Neben der durchschnittlichen Anzahl an Labels eines Argumentationsframeworks ist insbesondere der Knotengrad von hoher Relevanz. Neben dem durchschnittlichen Knotengrad für das gesamte Argumentationsframework ist es sehr empfehlenswert, die Anzahl eingehender Attacken sowie ausgehender Attacken des Zielarguments zu analysieren (siehe Abschnitte 6 und 10). Wenn die genannten Punkte bei der Auswertung der Metadaten berücksichtigt werden, lassen sich die Labels für ein Zielargument anhand der zulässigen oder vollständigen Labelings eines Argumentationsframeworks vorhersagen. Hier sind dann gute Werte hinsichtlich der Klassifikationsgüte erwartbar. Dies gilt auch bis zu einem gewissen Grad, wenn die Labelings (vollständige sowie zulässige) unvollständig sind. Abschließend bleibt jedoch festzuhalten, dass die Klassifikationsalgorithmen selbst nicht die Funktionsweise der jeweiligen Semantiken verstehen. Sie erlernen nur die Muster der jeweiligen Labelings.

13 Anhang

Die Ergebnisse der 5 Testreihen werden dieser Masterarbeit auf einem USB-Stick zur Verfügung gestellt. Anbei eine kurze Auflistung mit den wichtigsten Dateien:

- Testreihe 1 (Abschnitt 6)
  - Excel-Datei: Statistik - ICCMA2019 - Testreihe1.xlsx
- Testreihe 2 (Abschnitt 7)
  - Excel-Datei: Statistik - ICCMA2019 - Testreihe2 Precision.xlsx

  - Excel-Datei: Statistik - ICCMA2019 - Testreihe2 Recall.xlsx
- Testreihe 3 (Abschnitt 8)
  - Excel-Datei: Statistik - ICCMA2019 - Testreihe3.xlsx
- Testreihe 4 (Abschnitt 9)
  - Excel-Datei: Statistik - ICCMA2019 - Testreihe4 Precision.xlsx
  - Excel-Datei: Statistik - ICCMA2019 - Testreihe4 Recall.xlsx
- Testreihe 5 (Abschnitt 10)
  - Excel-Datei: Testreihe5 Knotengrad Verteilung.xlsx

Literatur

[1] Kleo G. Achilleos, Stephanos Leandrou, Nicoletta Prentzas, Panayiotis A. Kyriacou, Antonis C. Kakas, and Constantinos S. Pattichis. Extracting explainable assessments of alzheimer's disease via machine learning on brain mri imaging data. *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 1036–1041, 2020.

[2] Sabrina Bechtel. Maschinelles lernen in der medizin. Universität des Saarlandes, Fachbereich Mathematik, November 2008.

[3] Daniel P. Berrar. Bayes' theorem and naive bayes classifier. In *Encyclopedia of Bioinformatics and Computational Biology*, 2019.

[4] S. Bistarelli, Lars Kotthoff, Francesco Santini, and Carlo Taticchi. Summary report for the third international competition on computational models of argumentation. *AI Mag.*, 42:70–73, 2021.

[5] Stephen M Borstelmann and Saurabh Jha. Confusion in the matrix: Going beyond the roc curve. *viXra*, 2019.

[6] Martin Wigbertus Antonius Caminada and Dov M. Gabbay. A logical account of formal argumentation. *Studia Logica*, 93:109–145, 2009.

[7] Alexander Van Craen. Gpu-beschleunigte support-vector machines. Universität Stuttgart, Institut für Parallele und Verteilte Systeme, 2018.

[8] Pádraig Cunningham and Sarah Jane Delany. k-nearest neighbour classifiers: 2nd edition (with python examples). *ArXiv*, abs/2004.04523, 2020.

[9] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77:321–358, 1995.

[10] Wolfgang Dvorák, S. A. Gaggl, Anna Rapberger, Johannes Peter Wallner, and Stefan Woltran. The aspartix system suite. In *COMMA*, 2020.

[11] Chollet François. *Deep Learning mit Python und Keras - Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH Co. KG, Heidelberg, 2018.

[12] Mahmood Hammoodi, Hasanain Ali Al Essa, and Wial Abbas Hanon. The waikato open source frameworks (weka and moa) for machine learning techniques. *Journal of Physics: Conference Series*, 1804, 2021.

[13] Marlis von der Hude. *Predictive Analytics und Data Mining - Eine Einführung mit R*. Springer Fachmedien Wiesbaden, Wiesbaden, 2020.

[14] Isabelle Kuhlmann and Matthias Thimm. Using graph convolutional networks for approximate reasoning with abstract argumentation frameworks: A feasibility study. In *SUM*, 2019.

[15] Luca Longo and Lucy Hederman. Argumentation theory for decision support in health-care: A comparison with machine learning. In *Brain and Health Informatics*, 2013.

[16] Heizel Rosado-Galindo and Saylisse Dávila-Padilla. Tree-based missing value imputation using feature selection. *Journal of Data Science*, 2021.

[17] Thomas A. Runkler. *Data Mining - Modelle und Algorithmen intelligenter Datenanalyse*. Springer-Verlag, Berlin Heidelberg New York, 2015.

[18] Markus Schlüter. Neuronale netze. Universität Koblenz-Landau, 07 2016.

[19] Nikolaos I. Spanoudakis, Elena Constantinou, Adamos Koumi, and Antonis C. Kakas. Modeling data access legislation with gorgias. In *IEA/AIE*, 2017.

[20] Matthias Thimm. *Tweety - A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation*. 07 2014.

[21] C. Beierle und G. Kern-Isberner. *Methoden wissensbasierter Systeme Grundlagen, Algorithmen,Anwendungen*. Springer Vieweg, Wiesbaden, 2019.

[22] Julius Voggesberger. Evaluation von Zwischenergebnissen in Entscheidungsbäumen. Bachelorarbeit: Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Anwendersoftware, Mai 2019.

[23] Anna von Hopffgarten. Wie lernen maschinen? *Spektrum der Wissenschaft*, Feb 2021.

[24] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *AAAI*, 2019.