Classification of schemes

Philipp Heinisch
Bahnhofstr. 80
33102 Paderborn
phhei@mail.uni-paderborn.de

Abstract

The identification, verification and indexing arguments in natural language is a very complex task. Solving it would lead to easier, more varied and trustworthy argumentation. One step to complete this task is to classify the arguments to argumentation schemes that are common patterns. This paper will present two machine learning approaches to solving this classification task automatically. In addition, this paper presents an approach in the semantic web, with which you can infer in an argument network argumentation schemes based on description logic. We will compare the results of the machine learning classification with the human classification. The automatic classification on different text corpus databases for the presented approaches has an average accuracy of $\approx 75\%$.

1 Introduction

There is a continuing growth in the volume of data (natural language) that we can find on the internet. In (product) reviews, forums, discussions, papers and so on we can find lots of arguments for different topics. For example, a key feature of scientific writing is the use of argumentation (Green, 2015). However, because of the fact that most of the arguments are written in natural language, we must think about which way a computer can understand that – for higher goals, too. A good step to understand automatically and compute on arguments in natural language is to apply argumentation schemes on it. Argumentation schemes are common and abstract patterns for arguments of humans (Lawrence and Reed, 2016). In this paper we will define argumentation schemes (chapter 2.1) and scheme sets (chapter 2.2). Then we will face the question, how we can automatically classify arguments to such argumentation schemes. Therefore, we will present two machine learning approaches (3) and then a semantic web approach in chapter 4. This approach based on description logic can be used to complete networks of argument components by logical inference.

Classifying arguments results in a bunch of new possibilities. We can automatically identify complex argumentative structures (Lawrence and Reed, 2016). Individual agents can reason about and develop arguments that employ schemes (Reed and Walton, 2005). And for example for reviews we can improve our fact-checking: ”It has been shown that argumentation schemes are useful in evaluating common arguments as fallacious or not (van Eemeren and Grootendorst, 1992). In order to judge the weakness of an argument, a set of critical questions are asked according to the particular scheme that the argument is using, and the argument is regarded as valid if it matches all the requirements imposed by the scheme.” (Feng and Hirst, 2011).

A further research goal with the help of argumentation schemes is to automatically detect and complete enthymemes (Feng and Hirst, 2011). An enthymeme is a missing unstated premise that is paraphrased by presenting generally accepted background knowledge of the human reader (Green, 2015)

2 Foundations

In this chapter we want to have a look at the foundations. Especially we will define the concept of argument schemes (2.1) including an example in chapter 2.1.

2.1 Definition and formalization of argumentation schemes

Different definitions exist, two of them are:

> "Argumentation schemes are *patterns* of *non-deductive reasoning* that have been the focus of extended study in argumentation theory [...] a handy guide to the ways and means of persuading an audience" (Reed and Walton, 2005)

> "Argumentation schemes are abstract descriptions of acceptable, but not necessarily deductively valid, forms of argument used in everyday conversation" (Green, 2015)

To sum up, argumentation schemes are frameworks (or furthermore guidelines) for the natural language. Typically, an argumentation scheme consists of one conclusion and at least one (often two) premises. Each part – so each premise and conclusion – is called *proposition* or *scheme component* (Lawrence and Reed, 2016).

An example

Many of argumentation schemes were already defined. The two main papers (Feng and Hirst, 2011) and (Lawrence and Reed, 2016) present four common argument schemes together: Argument from Example, Argument from Cause to Effect, Argument from Verbal Classification, Practical reasoning[1].

We want to have a detailed look at one of the schemes: *Argument from Cause to Effect*. The abstract description of that scheme contains three components, which we can see in Figure 1.

Let us a assume we read in a forum about environmental pollution and a user argues against pollution by the following argument: *You know, generally, if all glaciers melt, then the sea level will increase by 66m. Actually, the glaciers are melting, faster and faster! So, the sea level will increase significantly and e.g. the Netherlands will have a big problem!*

If we analyze this argument, we can see that the major premise is: "If *glacier melt* occurs, then *the sea level increase* will occur". We can find the minor premise, too: "The *glacier melt* occurs". So,

[1]You can find an overview about a few schemes at https://www.reasoninglab.com/patterns-of-argument/argumentation-schemes/waltons-argumentation-schemes/

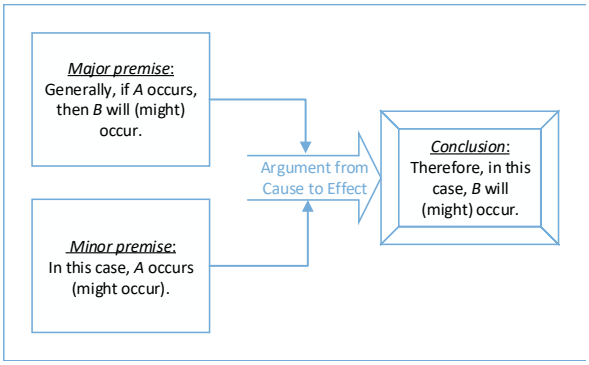

Figure 1: The abstract description of the argumentation scheme *argument from cause to effect*

we have a reasonable conclusion: "*Sea level increase* will occur".

Formalization of arguments and argumentation schemes with AML

We sum up the approach of the paper (Reed and Walton, 2005) in this chapter.

The first step of the formalization of argumentation schemes is the use of the propositional logic (PL). PL is a kind of logic with all logical operators and propositions. A proposition is a premise or conclusion. In the equation 1 we can find an example, each line is a proposition – so first we have two premises and then the conclusion.

$$\frac{A \rightarrow B \qquad A}{B} \quad (1)$$

For our example (2.1), attribute A is the glacier melt and B the sea level increase.

Formalization of arguments with AML The ***A**rgument **M**arkup **L**anguage* is an annotation language based on XML. With AML we can annotate arguments, for example we can specify the used scheme, enthymemes, premises and the conclusion.

The Araucaria software, an argument software, stores its content in AML. The Araucaria corpus contains a lot of annotated arguments from parliamentary records, online discussion boards, newspapers and magazines (Green, 2015).

Formalization of argumentation schemes with AML DTD Next step unto that is to write the

schemes down in *AML **D**ocument **T**ype **D**efinition*. The goal is to comprise each scheme in the AML DTD into a tuple of the form $< SName, SConclusion, SPremises >$. $SName$ is a unique identifier for the scheme. In $SConclusion$ we define the type of the conclusion and in $SPremises$ a set of types of the premises.

2.2 Scheme sets

A scheme set is a catalogue of argumentation schemes (Green, 2015). A good scheme set covers (nearly) all existing arguments with exactly one fitting argumentation scheme of the set.
Everyone can define his own scheme set by selecting a bunch of schemes. However, we can find scheme sets in the research area, which are often referred to.
One of the best-developed scheme sets in argumentation theory is the scheme set by Douglas Walton, known as Walton's scheme set (Feng and Hirst, 2011). It's focused on presumptive arguments (Walton, 2013) and contains more than 60 schemes, increasing year by year.
Other known scheme sets are:

- by Joel Katzav and Chris Reed (Katzav and Reeds scheme set) with over 100 schemes (Katzav and Reed, 2004)
- by John Pollock (Pollocks scheme set) (Feng and Hirst, 2011)
- by Grennan (Reed and Walton, 2005) and more

3 Machine learning classification approaches

There are several machine learning approaches to classify argumentation schemes. The basic principle is to train a set of classifiers with already annotated training data. Notice that not all approaches can handle raw natural language, but requires preprocessing. For example, some approaches require that another algorithm already detects arguments in natural language including the classification of premises and conclusion (Feng and Hirst, 2011). But this is another well-researched topic.
In this chapter we will have a look at two approaches: the first one was published by Feng& Hirst in 2011 (3.1) and the second one from Lawrence& Reed in 2016 (3.2).

3.1 Approach from Feng& Hirst

In this sub chapter, we summarize the machine learning approach of (Lawrence and Reed, 2016).

The expected input for this approach is an argument in natural language, where the propositions are already classified. The order and the content of the premises and conclusion are already known.

The output will be a guess of the name of argumentation scheme that was used in that argument. The approach took only the five most frequently occurring schemes of Walton's scheme set into account (Green, 2015): *argument from example*, *argument from cause to effect*, *practical reasoning*, *argument from consequences* and *argument from verbal classification*. If the user inputs an argument, which uses a different scheme, the approach will return one of the five names incorrect.

Description of the approach/ algorithm

The algorithm measures a set of features. The features that are independent of the applied argumentation scheme (3.1), are called *general features*. The values of each feature in combination (general trained classifier) return an argumentation scheme guess. In addition, there is a set of scheme-specific features. Each scheme has its own set of features that give a hint for or against a particular scheme (3.1).

General features are the location of the conclusion in the argument, the location of the first premise, the gap between the conclusion and the first argument and the number of explicit premises. In addition, the ratio between the character length of all premises and all conclusions is calculated. Furthermore, the question, whether the conclusion is for the first premise or not is answered. A last general feature is the *type*. If there are two or more inter-dependent premises, but only those who are necessary for the validation of the conclusion, we call the argument a *linked argument*. One example of a linked argument is the one in chapter 2.1. If there is only exactly one premise which is sufficient for the argument, it is called a *convergent argument*. It is a further research topic to determine the value of this feature, but it has a huge impact. Experiments show that the accuracy is increased by -0.5% to 20.1% by using that feature.

Scheme-specific features are in general relation patterns, grammar, count of positive and negative

propositions, punctuation cues, keywords, similarity and central words (Stanford parser is used). For the example of the scheme *argument from cause to effect* we have 22 keywords/ cue phrases like *result, related to, lead to* etc. Furthermore, we take ten relation patterns into account for this scheme.
Let us say that two schemes share the same keyword or punctuation key. To avoid rash guesses for particular scheme-specific features, we compute the degree of belief (confidence value c) that a particular argument A belongs to a particular scheme S. Therefore we use the distribution characteristics of the cue phrase or pattern in our training data, like we can see in Formula 2, where cp means cue phrase and m_i the number of scheme-specific cue phrases. In *boolean mode*, the number indicating how often the cue phrase is found in the argument, is limited to 1. Later we normalize the confidence values.

$$c_S = \sum_{k=1}^{m_S} P(A \in S | cp_k \in A) \cdot \#cp_k \in A \quad (2)$$

Experiments and results

The experiments used the AraucariaDB corpus.
The paper presented two setups to train and measure the approach: the first one is the *one-against-others classification*. In this setup, the data contains 50% arguments of one particular scheme (target scheme) and 50% arguments of other schemes. The task was to determine whether a specific argument belongs to that target scheme or not. The approach is better then a random guess (baseline 50%): the best average accuracy for the scheme *practical reasoning* is 90.8%, for *argument from cause to effect* 70.4% and the worst one is *argument from consequences* (62.9%). By using the general feature *type* and the count mode for the confidence of scheme-specific features we reach in overall average an accuracy of 75.5%.
The second measurement approach is *pairwise classification*, where we have only arguments of two schemes (50% of each). The task is to distinguish the arguments regarding their schemes. Here, the best average accuracy that we reach is 98.3% by distinguishing *practical reasoning* and *argument from verbal classification* because the schemes are very different from each other. The worst result of this setup is 64.2%, in overall average 85.6%.

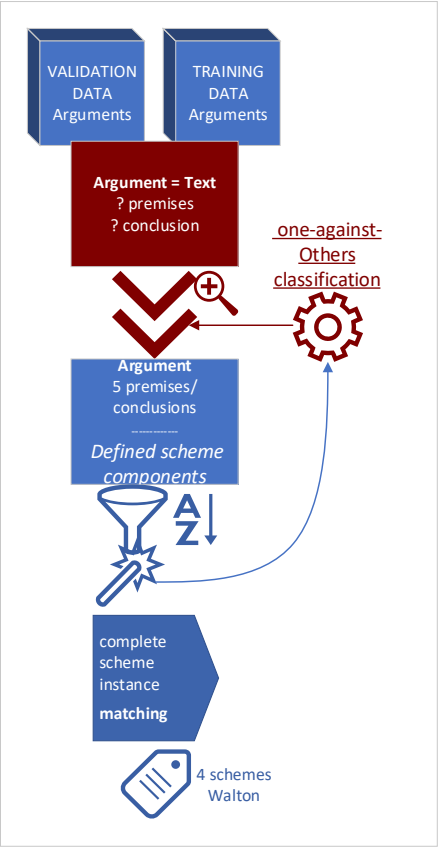

Figure 2: Abstract control flow of the approach of (Lawrence and Reed, 2016)

3.2 Approach from Lawrence& Reed

In this sub chapter, we summarize the machine learning approach of (Lawrence and Reed, 2016).

The expected input for this approach is a set of arguments – the raw natural language, segmented into propositions.

The output will be a guess of the used argumentation scheme for each argument. In addition, the algorithm determines the scheme components, for example, we will know, which premise belongs to which part of the abstract description of the scheme and where the conclusion is. This approach took only four most frequently occurring schemes of Walton's scheme set into account: *argument from example*, *argument from cause to effect*, *practical reasoning* and *argument from verbal classification*.

Description of the approach/ algorithm

You can find an abstract control flow of the algorithm in Figure 2. In this chapter we will look at the specific steps in detail.

First step of the algorithm: Identifying scheme components The approach provides a range of classifiers for each proposition type. Each classifier uses a subset of the following features:

- Unigrams: take a look at every single word in the proposition, e.g. keywords
- Bigrams: always takes two successive words in the proposition, e.g. for searching cue phrases
- length (number of words) of the proposition
- the average length of each word in the proposition
- parts of speech by Python NLTK POS-tagger
- punctuation (cues)
- similarity: the maximum similarity[2] of a word in the proposition to pre-defined words corresponding to each proposition type. E.g., the scheme component *major premise* of our example scheme (2.1) should contain similar words to generally and occurs.

The paper presented three different types of classifiers: Multinomial Naive Bayes classifier, Support Vector Machines (SVMs), Decision Trees. Since the Multinomial Naive Bayes classifier type achieves the overall best results, we choose this type.
Besides this *one-against-other classification* approach for training the classifiers and annotating the arguments, the paper presented the *pairwise classification*, too. Here we assume that we already know the argumentation scheme and the task is to match the premise scheme components (like minor and major premise in the *argument from cause to effect* scheme) in a right way to the set of premises of the argument.

Second step of the algorithm: Match scheme instance For this step, we assume that we already annotated the scheme components (done by the first step[3]). The idea is to have a look at each five successive scheme components. If we find a set of scheme components equal to a set of scheme components of one particular scheme in that frame, we bundle these scheme components under that scheme. If only one component is missing, we search for this missing component again in this frame by lowering the threshold for the classifier for this scheme component. If the missing component is still not found, we assume that there is an enthymeme in the argument.

Experiments and results

First, the paper investigated the fitness of the single classifiers for the scheme components. They used an extended AraucariaDB corpus. The results for finding out a particular scheme component from a random set of scheme components (baseline: 50%) are between 44% and 88% for Naive Bayes, in average 71%. If we take all three classifier types into account, we can observe 57% - 91%. In addition, the paper used the proposition corpus from the Digging by Debating project[4] for the experiences. It contains many arguments from a psychology textbook. Based on this experiment the paper sums up that they came close to identify at least where a scheme is occurring with a general high accuracy.

3.3 Evaluation of classification approaches

The second approach of Lawrence& Reed claims to have similar accuracy values to determine argumentation schemes like the first approach of Feng& Hirst (Lawrence and Reed, 2016). To prove that, they used the same dataset, but nevertheless they only distinguished between four and not five argumentation schemes. The big advantage of the second approach is that it does not only compute the used scheme names but the location of components in the text, too.
Overall the average accuracy of $\approx 75\%$ (the one of four guesses is wrong) does not seem very reliable. However, it is a good first step in this research topic. In addition, we should take into account that it is even for humans it is not easy to classify arguments. There was for example an experiment, where students and expert should classify arguments in a scientific text about genetics. The task was to match every argument to the one right out of 10 schemes. The schemes were not common because of this special genre of the text. There were a bunch of different experiment setups. In average, only 49% of the students completed the task successfully and 95% of the experts (Green,

[2] the similarity score is calculated using WordNet: https://wordnet.princeton.edu/

[3] used one-against-other classification with type Multinomial Naive Bayes classifier

[4] http://diggingbydebating.org/

2015).
To understand that it is sometimes hard to distinguish, one can have a look at *argument from example*[5] and the *argument from verbal classification*[6]

4 Graph analysis: Argumentation schemes in Semantic Web

In this chapter, we summarize the approach of (Rahwan et al., 2011).
For understanding this approach, we first need to understand, how we can model argumentation schemes in a graph, or more specific: in the semantic web.

4.1 Foundations of argumentation in Semantic Web

Semantic Web is a word for linked data. One basic concept of this is the Resource Description Framework (RDF) that arranges all data into triples, consisting of a subject, a predicate (the link) and an object. The formalization of argumentation schemes is based on the Argument Interchange Format (AIF), a core ontology of argument-related concepts. It leads to an argument network (directed graph) with argument entities as nodes. Each node can contain a bunch of attributes, additional information to the argument. Firstly there are the information nodes filled with one proposition. Two information nodes must not be connected directly to each other. In addition, scheme nodes (S-Nodes – patterns of reasoning) are defined, e.g the *inference application nodes* (RA-node).
Until now, we can only model specific examples, because of the fact that in AIF nodes represents instances, not classes. For that, the ARGDF ontology (depends to RDF) was founded. The ontology extends the AIF with two types of classes: the Form node (F-node) for representing a general form of a proposition and the scheme class node with the name of the argumentation scheme. That gives us the opportunity to model argumentation schemes. An example of the argumentation scheme of *Argument from Cause to Effect* (related to the example in chapter 2.1) is visualized in Figure 3.

[5] https://www.rationaleonline.com/map/s5tfy6/argument-from-example
[6] https://www.rationaleonline.com/map/tsh4n9/argument-from-verbal-classification

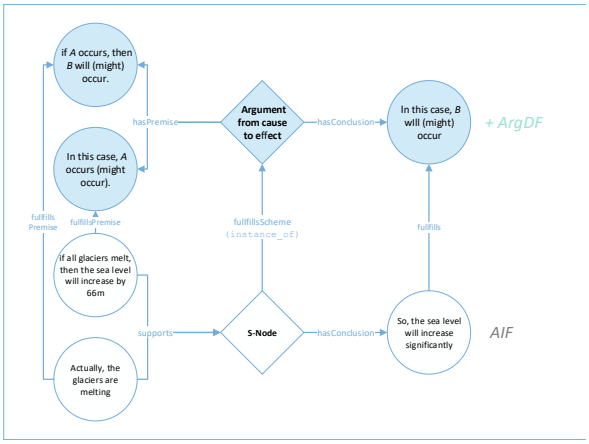

Figure 3: Formalization of the argumentation scheme *Argument from Cause to Effect* with ARGDF. The upper shaded part represents the argumentation scheme in general, the lower unshaded part represents a specific instance of the scheme – like we can find in a AIF argument network. The instance can be further connected with other arguments.

The paper (Rahwan et al., 2011) does a further step and presents a way to describe argumentation schemes in description logic. DL is a family of logics for reasoning and knowledge representation. It decides between terminology and concrete facts (for the world description like the specific arguments of a discussion). We will use the DL of the Web Ontology Language (OWL). OWL is a semantic web language that extends RDF.
The interesting part in OWL-DL is the idea of classes including inheritance (terminology) and instances of these classes (argumentation schemes or propositions). First, in OWL-DL, all schemes (corresponds to AIF S-Nodes) and their classes, propositions (corresponds to AIF I-Nodes) and their classes (called statement) or meta information are things. In each subclass layer (inherited from a more general class), we describe more specifically, what we want to describe.
In addition, OWL-DL offers a bunch of properties (so-called roles). With those, we can add meta information to our classes and instances like the creation date, the title, the author, the claim text for instances (propositions) and more. Each scheme class must be related to a creation date and a title. Furthermore, a scheme class must have a relation (modelled with a certain property) to one statement as the conclusion and at least one relation to a statement as premises. With

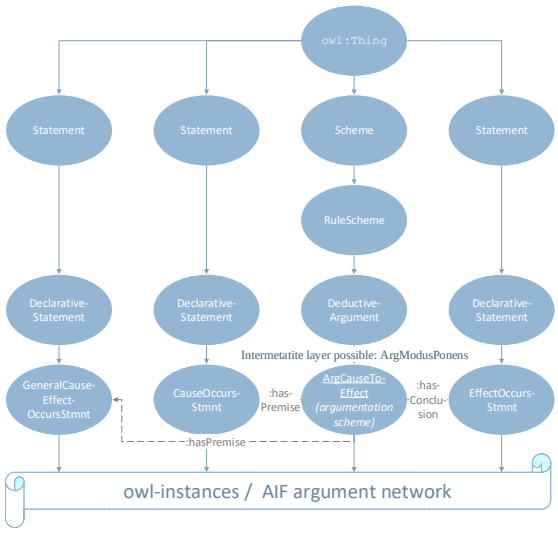

Figure 4: A snapshot of a tree in OWL-DL regarding the example of 2.1. Notice that the tree is not complete, but contains only all necessary paths and nodes for describing the argumentation scheme *Argument from Cause to Effect*. Of course, we can and should reuse any statement that was used for this particular scheme.

The code of the scheme would be $ArgCauseToEffect \equiv (DeductiveArgument \sqcap \exists hasConclusion.EffectOccursStmnt \sqcap \exists hasPremise.GeneralCauseEffectOccurs-Stmnt \sqcap \exists hasPremise.CauseOccursStmnt)$

the `formDescription`, an annotation property in OWL-DL, we can describe the statements (the components of an argumentation scheme) in an abstract way.

An inheritance tree based on OWL-DL is visualised in Figure 4, again with the example from 2.1. This inheritance including the relation will establish many reasoning methods.

4.2 Approach and ideas for scheme classification

For the approach of (Rahwan et al., 2011), we assume an argumentation network/ graph, based on OWL-DL. So we assume that each node in the network is already specified (related to a defined class), at least to the most general class `owl:Thing`. For retrieving more specific information, so more specific class relations, we assume furthermore that at least the nodes of one argumentation scheme instance have more specific class annotations. The approach is useful if, for example, a user or machine (like the machine learning approaches in chapter 3) already specifies more specific classes regarding a subset of nodes. The approach will help us to fill the gaps of types in the graph or at least to determine a specific set of class suggestions for an instance node. In addition to each scheme class in DL, we can refer a class of assumption statements and exception statements that can be used for verifying and attacking a particular argument of that scheme with another argument or an argument component (Rahwan et al., 2011).

So, how can we match an argument (in an AIF argument network) to his particular argumentation scheme? The paper (Rahwan et al., 2011) presented a bunch of approaches, which based on the inference techniques of the OWL-DL. For example, the OWL-DL is able to detect transitivity. For example, if the conclusion of an argument supports a premise of a second argument and the conclusion of the second argument supports a premise of a third argument, OWL-DL is able to determine the indirect support from the first argument to the third[7]. Same with the subclasses: if A is a subclass of B and B a subclass of C, OWL-DL detects that A is a subclass of C, too. For example, if the user queries for all statements of `DeductiveArgument`, he will get the statements of `ArgumentFromCauseToEffect`, too.

This technique in the combination of the conditions defined on each scheme will help us to infer the matching statement classes for each particular proposition and so in the end to infer the argumentation scheme.

The first technique is to find scheme subclasses. We assume that we already know the classes of the instances of the scheme components (statements) of 2 schemes: S_1 and S_2. Without loss of generality we assume that the number of premises of S_2 is equal to or higher than the number of premises of S_1. If for the conclusion C is hold that $(C_{S_1} \equiv C_{S_2}) \vee C_{S_2} \sqsubseteq C_{S_1}$ (conclusion of S_2 is of the same class or a subclass of the conclusion of S_1) and for each premise P_{S_1} we can find a corresponding premise P_{S_2} with $(P_{S_1} \equiv P_{S_2}) \vee P_{S_2} \sqsubseteq P_{S_1}$, then we infer that $S_2 \sqsubseteq S_1$ (the scheme class of S_2 must be the same or a subclass of S_1[8]). With

[7] such arguments sequences are called *chained arguments*

[8] If the number of premises are equal and if there is no subclass relationship between the propositions, we have the special case of $S_1 \equiv S_2$

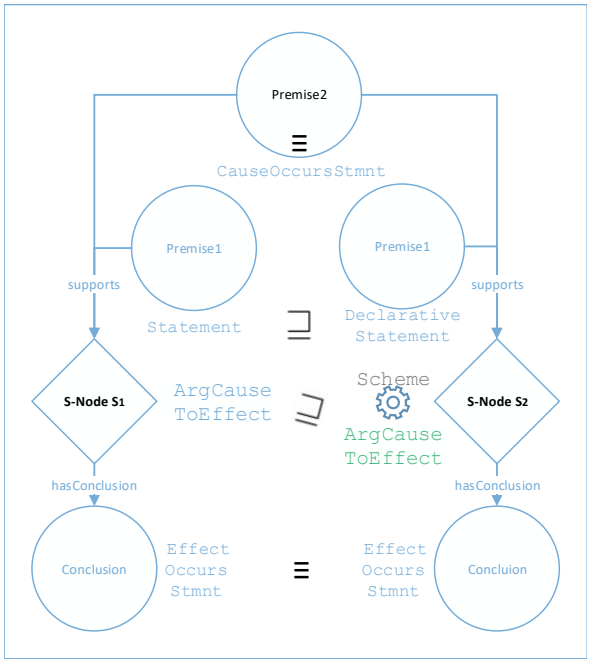

Figure 5: Example (in an AIF argument network) of the approach of (Rahwan et al., 2011) that infers the argumentation scheme of the left argument. Notice that this approach doesn't require a node that both arguments use.

that technique, we can determine a more and more concretely set of matching argumentation schemes – from `owl:Thing` to the most specific matching scheme class. There is an example in Figure 4.2

You can think about more techniques. For example, let us assume, we know the class of $n-1$ propositions of an argument with n propositions in a good detailed degree (maybe the most specific subclass), but not that for one proposition. By iterating about all known sensible argumentation-schemes, we can maybe reveal or guess a more specific class for that one proposition. And since we are working in an AIF argument network and scheme components are sometimes chained or reused, we can infer more and more specific information.

However, there was no fully automatic algorithm presented in (Rahwan et al., 2011) that has as input only the pure AIF argument network and as output a graph, completely labelled with specific argumentation-schemes.

Instead, the paper presented the Web-based system *Avicenna*. The system allows to explore available arguments, listed by their titles. Furthermore, we can create new arguments. The user can refer the new argument to an argumentation scheme and can reuse already existing propositions (called claims).

5 Conclusion

We looked on the (automatic) classification of argumentation schemes. Further research focused mainly on fundamental text mining tasks, but there is the need to extract complete arguments to automatically identify complex argumentative structures (Green, 2015). We presented a milestone in this task with the argumentation schemes.

The research of scheme classification with machine learning is advanced – so we presented two approaches in that area: the approach of Feng& Hirst (chapter 3.1) and of Lawrence& Reed (chapter 3.2). Both used a feature set for their classifiers. Feng& Hirst distinguished between general features and scheme-specific features. Lawrence& Reed used classifiers to determine scheme components, so that they can conclude schemes out of the scheme components. Unfortunately, this two-step approach didn't lead to significant improvement of the accuracy. Approximately one of four guesses for the scheme is wrong. Even for humans, it is hard to classify sometimes – we can observe that by looking to the long discussions of the annotators in the Araucaria database (Feng and Hirst, 2011). However, if one doesn't have a critical application based on scheme classification, we think that it is sufficient. For example, a tool that helps to verify arguments by asking critical questions or a tool that makes suggestions for enthymemes, can work with that accuracy without critical considerations.

In addition, the success of the computer computation approaches depends on the genre and topic of the text, due of various argumentation styles and domain-specific language rules in different fields. For example, the computation in scientific papers about genetics needs further research, even though there is already a BioNLP existing, where we can crawl articles about genetics (Green, 2015).

Finally, we can conclude that it is useful and in some areas required to have further research in argument scheme classification – for example in the Semantic Web (4) – but we can already use the existing approaches to fulfil tasks that require classified arguments.

References

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying Arguments by Scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 987–996, Portland, Oregon. Association for Computational Linguistics.

Nancy Green. 2015. Identifying Argumentation Schemes in Genetics Research Articles. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 12–21, Denver, Colorado. Association for Computational Linguistics.

J. Katzav and C. A. Reed. 2004. On Argumentation Schemes and the Natural Classification of Arguments. *Argumentation*, 18(2):239–259.

John Lawrence and Chris Reed. 2016. Argument Mining Using Argumentation Scheme Structures. *Frontiers in Artificial Intelligence and Applications*, 287(Computational Models of Argument):379–390.

Iyad Rahwan, Bita Banihashemi, Chris Reed, Douglas Walton, and Sherief Abdallah. 2011. Representing and classifying arguments on the Semantic Web. *The Knowledge Engineering Review*, 26(04):487–511.

Chris Reed and Doug Walton. 2005. Towards a Formal and Implemented Model of Argumentation Schemes in Agent Communication. *Autonomous Agents and Multi-Agent Systems*, 11(2):173–188.

Douglas Walton. 2013. Applying Argumentation Schemes. In *Methods of Argumentation*, chapter 4, pages 93–121. Cambridge University Press.