Argumentative Relation Classification for Argumentative Dialogue Systems

Bachelor Thesis

by

Carolin Schindler

Reviewer:	Prof. Dr. Dr.-Ing. Wolfgang Minker
Co-Reviewer:	Prof. Dr.-Ing. Dr. h.c. Stefan Wesner
Supervisor:	M. Sc. Niklas Rach

Institute of Communications Engineering
University of Ulm
20 November 2020

© 2020

This work is licensed under the CC BY-NC-ND 4.0 (Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International) License.
To view a copy of this license, visit https://creativecommons.org/licenses/by-nc-nd/4.0

Abstract

Yet, many argumentative dialogue systems are capable of talking about a limited set of specific topics. In the herein presented work, we retrieve arguments from an argument search engine regarding any topic and organize them automatically in an argumentation structure that can serve as a database for such systems. In order to structure the arguments, we utilize methods from the field of argument mining and propose different approaches for each step. Our main contribution is the implementation of an argumentative relation classification pipeline that queries the argument search engine for arguments regarding a specific topic, performs argumentative relation classification and outputs the resulting structure. It is evaluated in an annotation study with a new combination of annotation guidelines. The carried out study yields promising results for the quality of the generated argumentation structure.

Declaration of Authenticity

I certify that I have prepared this Bachelor Thesis by my own without any inadmissible outside help.

Ulm, 20 November 2020

(Carolin Schindler)

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

Why should we establish flexible argumentative dialogue systems? Argumentation is an essential element of human communication and reasoning [20]. Whenever information is incomplete or inconsistent, we start to argue [3]. This can take place internally or externally with other people. Whether we try to make a decision, persuade somebody or in a negotiation or deliberation, for example. Currently, most dialogue systems interacting with a human in an argumentation are only aware of a few predefined topics [29, 30, 32, 44]. However, in order to handle the variety of topics the user could be interested in, an argumentative dialogue system is in general required to be flexible in that manner.
Recently, two broad approaches to realize this flexibility have been introduced. On the one hand, the system can use a trained model to generate its own arguments on demand [18]. On the other hand, already existing arguments can be collected from texts, e. g. in the Web, using techniques from the field of argument mining [17]. For this purpose, the output of argument search engines [1], which emerged in the recent years, can be utilized [28]. Instead of representing the structure of a single argumentative text, these search engines identify arguments from heterogeneous texts in the Web concerning the queried topic and present them to the user in a ranked order. Additionally, they may also provide a classification of the arguments according to their stance towards the topic. The arguments retrieved by the search engine can be exploited by an argumentative dialogue system to make points. The system is still left with the decision which arguments to use at which state of the discussion. Hence, many systems require a structuring of the arguments [29, 32, 44] that goes beyond the ranked list provided by the argument search engine.
In a study, Chalaguine et al. [7] showed that arguments addressing the user's concerns are more effective. Therefore, retrieved arguments should be selected according to the user's input using an appropriate agent strategy that fits the task of the system, for example persuasion [32], discussing controversial topics [18, 29, 30] or improving people's argumentation skills [44]. The interaction between the user and the system can be modelled as a dialogue game [3] where moves correspond to utterances and rules specify which moves are allowed. The underlying representation of the arguments has to match the game and in order to conform to the flexibility requirement, it has to be generated automatically. Argumentation structures [20] are well suited for the representation as they capture argument components and their relations. An argument component models a single argument which can be defined on sentence-level or as a union of claim and evidence, for example. The relations can also be of various types including supportive and

attacking. Moreover, the structure may define types for the single argument components. The arguments available to the system are represented by the argument components and the relations are necessary to apply the game formalism [26] which is based on logics for defeasible argumentation.
When the generation of an argumentation structure has to be automated, argumentative relation classification comes into play [38]. This includes classifying whether there holds a relation for all possible pairwise combinations of argument components. In case of a relation, its type has to be identified. When the structure captures the types of the components themselves, they have to be determined automatically, as well.

In this thesis, we contribute an argumentative relation classification pipeline implemented in Python that

- queries the argument search engine ArgumenText [35] for arguments concerning a topic selected by the user.
- organizes the retrieved arguments in a predefined argumentation structure that encodes the relations support and attack and also assigns a type to each argument component. For this task, we exploit methods from the field of argument mining [20] where the automated structuring of arguments has already been investigated. The pipeline implements different approaches which are evaluated in an annotation study that introduces a new composition of guidelines.
- outputs the structure in a machine-readable file that can be utilized by an argumentative dialogue system.

With this, we extend the argumentative dialogue system EVA designed by Rach et al. [29] with topic flexibility.

1.2. Overview

The remainder of this thesis is structured as follows: An overview of existing research on argumentative dialogue systems and relation classification in the context of argument mining is given in Chapter 2. In order to have a common understanding, Chapter 3 explains background information which is relevant to this thesis. Chapter 4 reviews the required steps and within these different approaches to gain an argumentation structure from the output of an argument search engine. The results of this chapter are used in Chapter 5 for the implementation of the pipeline that covers the whole process from querying the argument search engine over argumentative relation classification to an argumentation structure that is dumped for further utilization by the argumentative dialogue system. In Chapter 6, the different approaches are evaluated and compared by carrying out an annotation study. We conclude this thesis in Chapter 7 with a summary and an outlook on future work.

2. Related Work

In this chapter, we outline different argumentative dialogue systems and especially focus on the type of argumentative representation used. Subsequently, we consider the role of relation classification in the field of argument mining and discuss results that were achieved when investigating methods to automate this task.

2.1. Argumentative Dialogue Systems

A prominent example for a debating system is the IBM Project Debater[1] that engages in a debate with a human with the goal to improve their skills in argumentation and decision-making. Thereby, each side states their position to the discussed topic in an opening, reacts to their counterpart in a rebuttal and closes with a final statement. In every part of the debate a speech of several minutes is held. The Project Debater creates these automatically taking the topic and the opponent's speeches into account. In contrast, the systems discussed in the following interact with the user in a continual alternation by interchanging a single argument in each turn.
Debbie [30] is a chatbot that can interact with a human in a discussion on three different topics. The arguments in favour and against the respective topic were collected and stored in separate databases. The chatbot adopts the user's opposing position and selects the statement with the highest similarity to their input as an answer. To speed up the procedure, the databases are internally split into groups of similar arguments with each having a representative. With this hierarchical agglomerative clustering, firstly, a cluster is selected by calculating the similarity scores between the input and the head of each cluster. Afterwards, the input is compared with all arguments contained in the selected cluster. As no relations between the arguments are represented by the databases, there is no relation classification needed. The output of an argument search engine with stance information would be enough to enable Debbie to argue about a wide range of topics.
Le et al. [18] investigated how to generate arguments during a discussion. Therefore, they built the chatbot Dave with a generative model on the one hand and a retrieval-based model which is analogues to Debbie's one on the other hand. In the former model, the utterances are generated by an in advance trained hierarchical recurrent encoder-decoder network (for architectural details the interested reader is referred to Sordoni et al. [34]) that takes the context, the user's current input and the history of the discussion into account. With this approach, Dave is able to talk about any topic but no predefined arguments and facts are used.

[1] `https://www.research.ibm.com/artificial-intelligence/project-debater`

Another system aimed at educational discussions was implemented by Yuan et al. [44] where the user has an interface to select a move type and the content to be said. The underlying representation is a knowledge base containing all statements and encoding a consequence relationship between them. In order to automatically create the knowledge base representation from previously retrieved arguments, an approach analogous to the herein presented one for argumentative relation classification would be required.
The strategical partially observable Markov decision process agent [32], for short SPA, is able to persuade humans to change their attitude and behaviour. The arguments were modelled by hand according to the weighted bipolar argumentation framework. In this framework, every argument is assigned a justification level depending on the user's believes and can support or attack other arguments to a certain extend. The root of the structure is built by the argument of interest which is evaluated in the argumentation. To automate the task of creating the framework, relations between the arguments and their degree would have to be determined.
Finally, we discuss the argumentative dialogue system EVA [29]. It is multimodal as it uses natural language to output speech and text and an avatar to display mimics and gestures. During the interaction, the user and EVA try to convince the respective other side of their stance towards the discussed topic. Both can select their arguments according to a manually created argumentation tree that encodes the natural language representation of the arguments and their relations towards each other. The structure of the tree [27] is adopted from the task of essay analysis in the field of argument mining and is explained in more detail in Section 3.1.
As we exploit existing argument mining techniques for the relation classification task, EVA was selected as a reference system among the possible options.

2.2. Relation Classification in the Field of Argument Mining

Argument mining generally aims at detecting argument components from a given text and predicting the structure of these arguments [20]. The detection is often split into an argumentative sentence detection followed by a more precise argument component boundary detection. We are particularly interested in the structuring as this part contains relation classification.
Stab and Gurevych [37] identified argumentative relations between arguments extracted from essays distinguishing the types supportive and non-supportive. They obtained features for each pair of arguments and used those as an input for a traditional binary classifier. This basic strategy is also considered in Section 4.1.3 of this thesis.
Another approach is to classify the relations jointly with other data that contains additional information about the relation, e. g. stance information [15] or the argument component types [38].
Hou and Jochim's [15] work is based on arguments from different texts and defines the relations agree and disagree. They built a binary classifier for stance and relation prediction each. In one approach, they directly concluded from the stance classification to the type of the relation by assuming that within a topic, arguments with opposing stance

disagree with each other and arguments with the same stance agree with each other. In another approach, they used Markov logic networks to combine the output of the two local classifiers. In their experiment, they found the latter method to perform best, followed by drawing direct conclusions from the stance which outperformed the extra classifier for the relation prediction. Since argument search engines giving stance information exist [28], we use the stance directly to classify the type of the relation. Thus, we do not have to implement a classifier for the relation type and assume that the stance classification gained from the search engine is correct.
In the domain of essay analysis, Stab and Gurevych [38] used the method of Integer Linear Programming to combine the output of a ternary argument type classifier and a binary relation classifier that only predicts whether there is a link between the presented pair of arguments. This was done to establish a consistent assignment between the component type classification and the existence of a relation according to the predefined argumentation structure. Our approach to gain consistency between the identification of relations and component types is the following: We predict the relational links taking the rules of the argumentation structure into account; then we determine the argument types from the resulting structure. Moreover, the arguments retrieved by argument search engines are out of context and originate from various documents. We consider that classifying argument types without information about the content of the original source is not effective due to the large number of contextual features that were selected for the component type classification by Stab and Gurevych [38].

The argument mining community also started to utilize argumentative relations between sentences to decide whether they are argumentative or not. The basic idea by Carstens and Toni [6] is that a sentence is argumentative when it stands in a supporting or attacking relation to any of the other collected sentences. Rocha and Cardoso [31] showed that classifying pairs of sentences as argumentative and then drawing conclusions for each sentence yields better results than a classification in a single-sentence fashion.
In our case, the argument search engine already succeeds in the task of detecting arguments. We use these retrieved arguments and relate them towards each other in an argumentation structure that also captures the type of the argument component.

3. Background

At first, we provide a definition of the argumentation structure that we are going to achieve. Afterwards, we investigate existing argument search engines in order to select the one that fits our scenario best. Their output serves as a starting point for this work. Furthermore, we discuss candidate features for the relation classification problem and give a brief introduction into the neural model BERT, which is fine-tuned as a classifier and used for feature extraction later in this work. In the following parts of this thesis, an argument is an argumentative sentence, unless stated otherwise.

3.1. Argumentation Structure

EVA's argumentation structure [27] was adopted from Stab and Gurevych [36] who designed it to analyse written essays. They defined three different argument types, namely major claim, claim and premise, and a directed so-called target relation that can be supportive or attacking. The arguments build the nodes and the relations the arcs of the graph.

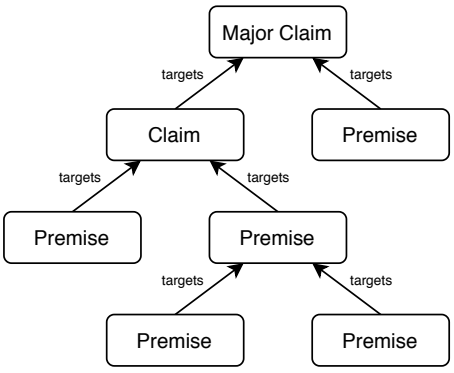

Figure 3.1.: Exemplary argumentation structure

In such a structure, exactly one major claim exists which is the utterance that started the argumentation and hence is always the root of the graph. Claims are controversial statements that need to be proven right or wrong in the further course of the discussion and express an attitude towards the major claim. Consequently, claims can only target the major claim. A premise approves or refutes another argument and can target the major claim, claim or another premise. Every argument, except the major claim, targets exactly one other argument. Therefore, the resulting structure is a tree as depicted in Figure 3.1. An argument cannot target itself but can be targeted by any number of arguments.

3.2. Argument Search Engines

Recently, Rach et al. [28] evaluated the usage of argument search engines in the context of argumentative dialogue systems. The engine has to provide an API (application programming interface) in order to be accessible in a convenient way and stance information which is required by the herein considered approach to argumentative relation classification. They discovered that out of IBM Debater [19], TARGER [9], PerspectroScope [8], args [42] and ArgumenText [11] only the last two fulfil both requirements. In their study, they asked the participants to assign the arguments, which were presented within a topic and engine in a random order, to the following categories: interesting, convincing, comprehensible, related. The first two criteria are task-related and aim at cooperative and non-cooperative setups. With these, they want to find out how suitable the retrieved arguments are for the respective task. The other two criteria are related to the quality of the arguments. The last one also takes the relation and the stance of the argument component to the queried topic into account.
We are now examining the argument search engines args and ArgumenText and outline the result of the discussed work for each of them. Afterwards, we reason our decision to use ArgumenText as the basis of our work.

3.2.1. args

The args search engine [42] depends on nearly 300k arguments crawled from five different online debate portals. Thereby, the structure of the discussions in the different portals was exploited to obtain more reliable results.
The extensible argument model contains information about the argument component and its context. The basic output of the args API is shown in Table 3.1. As can be seen, an argument consists of several premises that are related to a conclusion.

Information	**Description**
Conclusion	text span
Stance	stance of the conclusion towards the query
Premises	text span for each premise
Stances	stance of each premise towards the conclusion
Context	surrounding arguments position of the conclusion and premises

Table 3.1.: Basic output of the args retrieve arguments API[1]

The advantage over sentential arguments is that there is more context given when combining the text of the conclusion with its premises. On the other side the utterances of the dialogue system become indeterminably long and therefore might be hard to follow and understand by only listening to them. To overcome this disadvantage, Rach et al. [28]

[1] `https://www.args.me/api-en.html`

limited the overall number of words contained in the text span of an argument component to 60. With this setup, args outperformed ArgumenText in the category related.

3.2.2. ArgumenText

The search engine ArgumenText [35] uses the English part of CommonCrawl[2] to retrieve relevant documents. Consequently, the structure of the collected documents is unknown and can therefore not be utilized. Arguments are defined on sentence-level and have a stance towards the queried topic. The Search API additionally provides scores for the confidence of the stance classification and the certainty of the sentence to be argumentative. This argument model is shown in Table 3.2.

Information	Description
Argument	sentence
Stance	stance of the argument towards the query
Confidence	argumentative and stance

Table 3.2.: Basic output of the ArgumenText Search API[3]

ArgumenText was evaluated by Stab et al. [35] on three randomly selected topics. Across these, it covered almost 90% of expert-generated arguments. However, the authors reported that precision is an issue as on average 47% of the detected arguments are not argumentative, nonsensical or have a wrong stance prediction. In the user study by Rach et al. [28], ArgumenText outperformed args in the convincing criterion and yielded better results in the category interesting.

3.2.3. Selection of an Argument Search Engine

We choose the search engine ArgumenText for the following reasons:

- Arguments are defined as sentences.
  EVA's argumentation structure contains argumentative sentences as nodes and not a sum of sentences leading to a conclusion. Therefore, the argument model of ArgumenText fits our task better.

- Better performance in the task-related criteria.
  Since EVA [29] is a non-cooperative argumentative dialogue system, it is important that the arguments are convincing.

In order to compensate the precision issue, we use the confidence scores to create an individual ranking. An option is to multiply them [28]. Then it holds the following: The higher the score, the more certain it is that the sentence is argumentative and has a correct stance classification.

[2] `http://commoncrawl.org`
[3] `https://api.argumentsearch.com`

3.3. Classification of Relations between Sentences

We use machine learning approaches to find a solution to the relation classification problem which includes identifying the direction and the type of the relation in case one exists. As neural networks are mathematical models, we have to find a way to represent argumentative sentences as numerical vectors.

3.3.1. Feature Extraction for a Sentence Representation

There was existing research on which features to use in order to classify discourse relations before this task got relevant in the field of argument mining. The features extracted in the following papers are summarized and briefly explained in Table 3.3.

Group	**Feature**	**Details**
"basics"	bag-of-words	compare terms in sentence with predefined list of terms
	embeddings	vector representation of words, n-grams or sentences
"tools"	part-of-speech	identification of nouns, verbs, ...
	parse trees	dependency parsing, constituency parsing
	background knowledge	extract information from knowledge graphs
	aspects	extracted from sentence
preprocessing	stemming	reduction to word stem
	lemmatisation	derive dictionary form of word
	lower casing	-
	removing stop words	remove the most common words of the language
entity features	...	no benefit over or with lexical features
structural features	token statistics	count of tokens, ...
		argumentative domain: premise in average longer than claim
	punctuation statistics	count of punctuations, ...
	position within sentence	of tokens, ...
contextual features	position in context	distance between sentences, surrounding sentences, ...
	indicators	discourse markers, ...
lexical features	word pairs	combinations of word embeddings, e. g. cross-product
	first-last-first3	first, last and first three words of sentence
		often contain connectives
	modality	count of modal adverbs, ...
	verbs	main verb, verb classes, tense, ...
		argumentative domain: premise past, claim presence
	numbers	count of numbers, ...
syntactic features	production rules	extract rules from parse tree, e. g. depth
sentiment features	polarity	count of words with positive, negative or neutral sentiment
	inquirer categories	count of words for each category (more fine-grained)
similarity features	cosine similarity	similarity between vector representations
	common terms	shared nouns, lemma overlapping, ...
	distance measures	edit distance, ...
	entailment	one sentence can be inferred from the other
other features	type of sentence	argumentative domain: major claim, claim, premise

Table 3.3.: Summary of features for relation classification

To recognize implicit discourse relations, Park and Cardie [23] utilized results of other works and used word pairs, first-last-first3, polarity, inquirer tags, verbs, modality, production rules and contextual features. In their experiments, word pairs did only provide little to no benefit when combined with the other task-specific features. They also showed that preprocessing with stemming and hand-crafted rules improves the classification task immensely, whereas binning decreases the performance.
Louis et al. [21] tried to classify implicit discourse relations by exploiting entity features with little success. The selected entity features were outperformed by lexical features and neither could complement them.
According to Lippi and Torroni [20], one of the most used features in the argument mining domain is the bag-of-words representation. In the simplest implementation, a 1 is denoted when the sentence contains the respective n-gram (uni-gram in case of a word) and a 0 otherwise. Thereby, the position of n-grams in the sentence is ignored. Another drawback is that semantic similarity is not considered, but this can be mitigated by incorporating knowledge, e. g. from lexical databases. In general, background knowledge [16] from knowledge graphs can be used to compensate for missing contextual features and improve the results when distinguishing between supportive and attacking relations. The bag-of-words feature [20] can also be applied to grammatical information collected by constituency or dependency parsers or part-of-speech taggers.
In order to classify whether there is a supportive relation between two argument components extracted from an essay, Stab and Gurevych [37] proposed token statistics, punctuation marks and positioning as structural features, word pairs, first word, modal adverbs and common terms as lexical features, production rules and tense of verb as syntactic features and discourse markers and predicted type as further features. Counting tasks were realized for each individual argument component and the absolute difference between the respective counters was included, as well. They concluded that lexical features perform best followed by syntactic features and indicators. Structural features were only effective when combined with other features and the impact of the predicted type was negligibly low. Similar features were used in their later work [38] to identify whether there holds a relation between two argument components.
Menini and Tonelli [22] classified utterances in the political domain pairwise into the categories agreement and disagreement. For this task, they used sentiment scores, word embeddings for keywords, cosine similarity, entailment, lemma overlapping and negation features. In their experiments, they achieved the best results when using all features together for an election dataset as well as for a Debatepedia[4] dataset.
A ternary classification into supportive, attacking and no-relation was investigated by Carstens and Toni [6]. They applied similarity scores, the edit distance measure, textual entailment, discourse markers and sentiment scores among other features.
Another possible feature for relation classification are aspects extracted from the sentences. Schiller et al. [33] recently trained a model for argument aspect detection in order to generate arguments according to a query consisting of topic, stance and the aspect of interest.

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

3.3.2. BERT Model

The abbreviation BERT [12] stands for Bidirectional Encoder Representation from Transformers and is a pre-trained language representation model. It supplies embeddings that represent the given natural language input as a vector. Furthermore, it can be fine-tuned to be used in a specific task which is the classification of the relation between two arguments in our case. The BERT model achieved state-of-the-art results in eleven natural language processing tasks and is thus used for feature extraction and considered as a model for the argumentative relation classification in this thesis.
Its architecture corresponds to the encoder part of the Transformer proposed by Vaswani et al. [40] which is depicted in Figure 3.2. The stacked layers consist of two sublayers with the function

$$out = \text{Norm}(in + \text{sublayerFunction}(in))$$

each. The multi-head self-attention mechanism interconnects all positions in the sequence and thus enables to learn long-range dependencies more easily. The position-wise fully connected feed forward network implements the function

$$out = \text{linearTransform2}(\text{ReLU}(\text{linearTransform1}(in))) = \max(0, in \cdot a_1 + b_1) \cdot a_2 + b_2$$

where a_1, a_2, b_1 and b_2 denote the parameters of the linear transformations differing from layer to layer. $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ alter in the number of stacked layers and in the number of layers and heads within the self-attention mechanism [12].

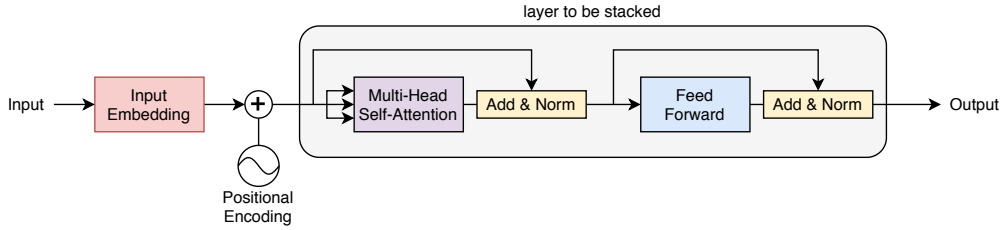

Figure 3.2.: Architecture of the Transformer encoder
Source: based on [40]

During pre-training, BERT was trained in an unsupervised manner in two steps. Firstly, a masked language model, that randomly masks some tokens of the input, was used with the goal to predict the original tokens from the masked input. With this, a contextual representation is learned. Secondly, the model was trained on a next sentence prediction task where it has to classify whether Sentence A could directly be followed by Sentence B in a text. This was done in order to learn the special classification token ([CLS]) which is required by BERT for classification tasks in the domain of natural language processing.
The fine-tuning is performed by adding an additional layer to the model that uses the token representation of the input-sequence or in case of a classification task the [CLS] token as an input. The parameters of the extended model are fined-tuned by supervised training in an end-to-end manner. The pre-training and the fine-tuning for our task is sketched in Figure 3.3.

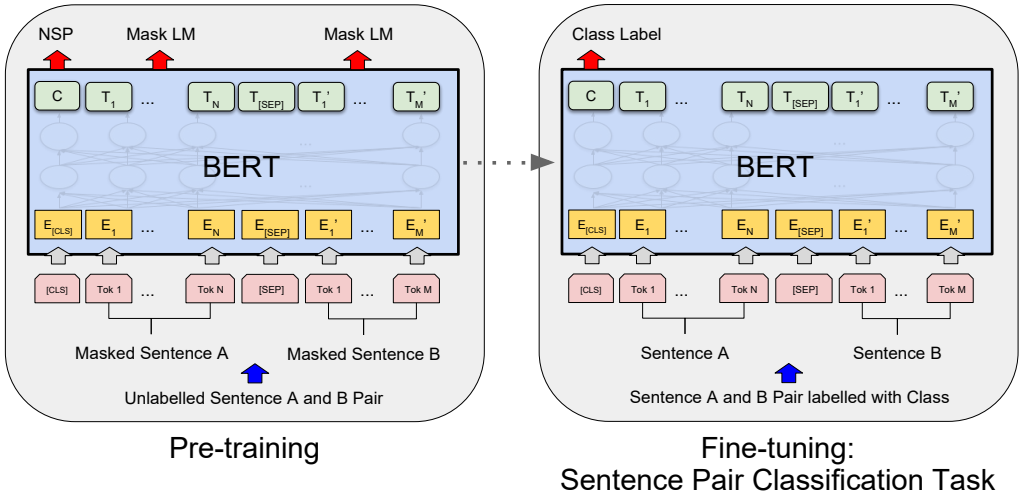

Figure 3.3.: Pre-training and fine-tuning process for the BERT model
Source: adapted from [12]
CC BY 4.0, https://creativecommons.org/licenses/by/4.0

Devlin et al. [12] suggested setting the dropout probability to 0.1 and trying out all combinations of the subsequent parameters for the fine-tuning task and then picking the model with the best performance:

- batch size: 16, 32
- learning rate (Adam): 5e-5, 3e-5, 2e-5
- number of training epochs: 2, 3, 4

This exhaustive search is proposed since fine-tuning is commonly fast and therefore computationally inexpensive. Especially small datasets are sensitive to the choice of parameters.
Another way to exploit the pre-trained BERT model is to use the resulting representation as an embedding feature. The best performance with this feature-based approach was reached for the CoNLL-2003 Named Entity Recognition task [39] by concatenating the token representations from the top four hidden layers.

4. Steps to Present Argumentative Relations in a Single Tree

The objective of this chapter is to decide whether an argumentative sentence targets another one in compliance with the constraints given by the argumentation structure introduced above. In the first step, we train a neural network to classify if there is a directed relation from one sentence to another. Afterwards, we build a single argumentation tree using the output of the classifier.
As mentioned earlier, the type of the relation and the argument type are subsequently derived from the stance information and the position of the argument in the tree and are therefore considered in Section 5.3.

4.1. Prediction of a Relation between Sentences

In order to train a neural network in a supervised manner on this task, a corpus with labelled sentence pairs for the training is required. We investigate two different approaches: The first is to fine-tune a BERT model with an additional classification layer; the other is to extract features and input these to a traditional classifier. The final selection of a model for each approach follows in Section 5.2.1, where we use them in our pipeline for argumentative relation classification.

4.1.1. Selection of a Corpus

The corpus has to fulfil the following requirements for our task: It has to distinguish between no relation and argumentative relation, ideally supportive and attacking ones, while taking the direction into account. Relations have to be defined on sentence-level as the selected argument search engine provides argumentative sentences.
These demands are met by the corpus created by Carstens and Toni [6] for relation-based argument mining. Other corpora, for example the Debatepedia corpus prepared by Kobbe et al. [16], only differ between support and attack and hence are not suitable to detect if there is a relation or not. The corpus created by Stab and Gurevych [36] fulfils the first requirement but is not defined on sentence-level.
Consequently, we select Carstens and Toni's [6] corpus[1]. We transform the ternary classification between support, attack and no-relation to a binary classification between no-relation and relation with the latter including support and attack. Even though the corpus enables us to directly predict the type of the relation, we decided to stay with our

[1] `https://www.doc.ic.ac.uk/~ft/softwareArg.html`

approach in order to be consistent with the stance information provided by the argument search engine. Putting all 4,606 annotated sentence pairs together, we count 2,632 pairs for relation and 1,974 for no-relation. Randomly, every fourth pair within the attacking and supporting category is discarded to achieve a perfectly balanced dataset. This also leads to a perfect balance between supportive and attacking relations in the related category, which is important in order to classify both relation types as a relation later on. For the following considerations, we have to keep in mind that the corpus does not contain sentences from ArgumenText which are used as an input later on.

4.1.2. Fine-tuning a BERT Model

We use the simpletransformers[2] library, which is based on the HuggingFace's Transformers [43] library, to fine-tune a BERT model for our task. The additional layer added to the pre-trained BERT model for classification is a linear layer.

Selection of a Pre-trained BERT Model

There exist several types of pre-trained BERT models. We already mentioned that a base and a large variant are available. Furthermore, we have to decide on the language and whether the model is case-sensitive. The language of the input sentences is always English and as all words usually start with a lower-case letter in this language, we choose a BERT uncased model. Another parameter to be set is the length of the input sequence. The longest combination of two sentences contains 247 words in our dataset. The single argumentative sentences retrieved by ArgumenText are not longer than 30 words by default. Rach et al. [28] limited the number of words in an argument to 60 in their evaluation. Therefore, we use an input sequence length of 256. Moreover, a spot-check with the parameters batch size 16 and learning rate 4e-5 on $\text{BERT}_{\text{base}}$ showed that the lengths 128 and 512 do not perform better than the same model with length 256. Now the question is still open whether to use a base or a large BERT model. To answer this, we again performed a spot-check with the same parameters but this time differing the model size. Unexpectedly, we found $\text{BERT}_{\text{large}}$ not outperforming $\text{BERT}_{\text{base}}$ and as the large model requires more memory and training time, we decide to finally select the $\text{BERT}_{\text{base}}$-256 uncased model for further fine-tuning.

Results

In the exhaustive search, we vary the batch size and the Adam learning rate as proposed by Devlin et al. [12] to find the parameters that fit our dataset best. Additionally, we use the learning rate 4e-5 as this is the default value of the selected library. We train the model with each parameter combination for five epochs and evaluate and save the fine-tuned model in each of them. The performed 5-fold cross validation uses 80% of our dataset for training and the remaining 20% for evaluation of the model. The accuracy results achieved in the evaluation are shown in Table 4.1. The higher the mean-value

[2] `https://github.com/ThilinaRajapakse/simpletransformers`

and the lower the variance, the better the result. All mean-values greater or equal to 0.8 and all variances smaller or equal to 10e-5 are highlighted in bold. We can see that models with batch size 16 outperform models with batch size 32 for our dataset and the parameters batch size 16 and learning rate 4e-5 yield the best results.

Batch Size	**Learning Rate**	**Epoch**									
		1		2		3		4		5	
		mean-value	variance	mean-value	variance	mean-value	variance	mean-value	variance	mean-value	variance
16	2e-5	0.794	13e-5	**0.800**	35e-5	0.792	12e-5	0.792	12e-5	0.794	18e-5
16	3e-5	0.790	60e-5	0.788	27e-5	0.792	12e-5	0.794	**8e-5**	0.792	**7e-5**
16	4e-5	0.778	17e-5	0.796	43e-5	**0.800**	**10e-5**	**0.800**	**10e-5**	**0.802**	17e-5
16	5e-5	0.778	87e-5	**0.800**	25e-5	0.790	25e-5	0.794	23e-5	0.792	17e-5
32	2e-5	0.792	57e-5	0.798	17e-5	0.792	12e-5	0.794	**8e-5**	0.792	**7e-5**
32	3e-5	0.794	43e-5	0.792	27e-5	0.790	20e-5	0.796	13e-5	0.794	13e-5
32	4e-5	0.794	28e-5	0.792	17e-5	0.790	25e-5	0.792	17e-5	0.794	23e-5
32	5e-5	0.792	57e-5	0.794	43e-5	0.796	33e-5	0.796	43e-5	0.794	43e-5

Table 4.1.: Accuracy results for the exhaustive search on BERT parameters

4.1.3. Features and Classifiers

In this section, we use feature vectors as an input for traditional classifiers. The numerical feature vector is created out of the natural language representation of the respective sentence pair. Thereby, different characteristics of the sentences are exploited.

Features

First of all, we have to extract features from the sentence pair which is done after lower casing. Out of the features summarized in Table 3.3 we

- do not examine entity features as they did not improve the results in the originating work [21].
- consider structural features.
  - The token and punctuation statistics count the number of tokens or punctuations respectively for each sentence and afterwards the difference is calculated as a feature. These might be helpful to determine the direction of the relation.
- cannot use contextual features as we do not know the context of the sentences.
- investigate lexical features.
  - Instead of word pairs, we gain sentence embeddings from the $\text{BERT}_{\text{base}}$-256 uncased model using the simplerepresentations[3] library, which is built upon the HuggingFace's Transformers [43] library. The last four hidden layers are concatenated to achieve optimum results [12]. Besides a sentence pair representation, we also obtain a concatenation of the representation of each sentence in the pair. We did not sum the single sentence representations as concatenating them yielded better results for Cocarascu and Toni [10] in their setup.

[3] `https://github.com/AliOsm/simplerepresentations`

  - First-last-first3 is not considered as this feature is used to capture connectives, which are not meaningful without a context.
  - For each sentence, the number of modal adverbs and numbers is counted and the respective difference is calculated as a feature. These features may provide information about the direction of the relation.

- do not use sentiment features as we are not interested in the type of the relation.
- examine similarity features.
  - We calculate the Jaro similarity and Jaro-Winkler similarity between the two sentences to be classified.
  - After excluding stop words and stemming, the words that both sentences have in common are counted as the common words feature.
  - The edit distance between the sentences in the pair is calculated.
- cannot use the type of the argument as it is unknown at this stage. Moreover, this feature was shown not to be helpful for the task [37].

Unless otherwise stated, the Natural Language Toolkit [5] is used to extract the features. We wanted to exploit aspects either as word embeddings or to count common aspects between the sentences, but to our knowledge there exists no accessible tool or trained model for this.

Classifiers

As we do not know which classifier is best for our task, we implement a bench of classifiers using classifier models from the scikit-learn library [24]. Among these are a

- Random Forest classifier, which was used by Carstens and Toni [6] on the original corpus achieving an accuracy of 77.5%.
- Support Vector Machine (SVM) applied by Stab and Gurevych [38] to a task equal to ours.
- Decision Tree, which was part of Stab and Gurevych's [37] benchmark for classifying supportive and non-supportive relations but did not perform better than an SVM.

Cocarascu and Toni [10] obtained an accuracy of 89.53% on the corpus by training embeddings for it. We did not implement their architecture using Long Short-Term Memories for the subsequent two reasons: The quality of the embeddings depends on the training dataset. As the sentences in our dataset are not comparable with the sentences retrieved by ArgumenText, learning the embeddings most likely will not improve our results in practical application. Without trained embeddings, an accuracy of 68.25% was reached which is worse than the results achieved on the same data by Carstens and Toni [6] with a Random Forest classifier.

Results

To investigate the possible combinations of features and classifiers, we apply a 5-fold cross validation with the same split as used above for the BERT models. The compositions of features, which are relevant in the following, are encoded by a number that can be looked up in Table 4.2.

Feature Set	Features contained in the Set
0	sentence pair
1	sentence pair, token and punctuation statistics, common words
2	sentence pair, edit distance, Jaro similarity, Jaro-Winkler similarity
3	sentence pair, token and punctuation statistics, common words, edit distance, Jaro similarity, Jaro-Winkler similarity
9	sentence pair, single sentences, token and punctuation statistics, common words

Table 4.2.: Encoding of feature combinations as feature sets

Classifier	mean-value	Feature Set
NuSVC with radial kernel	0.776	9
NuSVC with linear kernel	0.776	9
SVC with linear kernel	0.772	1, 2, 3
Random Forest	0.766	3
SVC with radial kernel	0.748	9
Decision Tree	0.696	2, 3

Table 4.3.: Best mean-value accuracy for classifiers with the feature set that obtains the result
NuSVC: SVM with nu-parameter
SVC: SVM with C-parameter

Table 4.3 shows the best mean-value accuracy result for each classifier and the feature set by which the result is obtained. The Decision Tree classifier performs worst and Random Forest is slightly outperformed by Support Vector Machines. We conclude sentence embeddings to be an essential feature, with the sentence pair representation leading to better results than the concatenated embeddings for every single sentence in the pair. The extracted features are not useful without any sentence embeddings, which is probably due to the small size of the resulting feature vector without any embeddings.
We continue with tuning the respective parameter of the SVC (SVM with C-parameter) and NuSVC (SVM with nu-parameter) on the kernels and feature sets they yield best results. Additionally, the feature set 0 is considered as this provides the smallest useful feature vector and enables the comparison of a fine-tuned BERT model with its feature-based approach. The best accuracy results obtained during the experiment are shown in Table 4.4 for each classifier and parameter setting. The highest achieved

mean-value accuracy with feature set 0 is listed in Table 4.5 for each classifier. We could not obtain a higher mean-value than in our earlier setup.

Classifier	C- or nu-value	Feature Set	mean-value	variance
NuSVC with radial kernel	0.5	9	0.776	38e-5
	0.05	9	0.728	37e-5
	0.005	9	0.618	112e-5
NuSVC with linear kernel	0.5	9	0.776	43e-5
	0.05	9	0.722	22e-5
	0.005	9	0.714	3e-5
SVC with linear kernel	1.0	3	0.772	47e-5
	1.0	1, 2	0.772	52e-5
	10.0	3	0.768	32e-5
	100.0	1	0.748	37e-5
	100.0	3	0.748	17e-5

Table 4.4.: Best mean-value accuracy for SVC and NuSVC with C- or nu-parameter and the feature set that obtains the result
NuSVC: SVM with nu-parameter
SVC: SVM with C-parameter

Classifier	C- or nu-value	mean-value	variance
NuSVC with radial kernel	0.5	0.774	53e-5
NuSVC with linear kernel	0.5	0.770	35e-5
SVC with linear kernel	1.0	0.770	50e-5

Table 4.5.: Best mean-value accuracy for SVC and NuSVC with C- or nu-parameter using feature set 0
NuSVC: SVM with nu-parameter
SVC: SVM with C-parameter

4.1.4. Discussion

We did not extract further features for the latter approach as the improvements of appending additional features to the sentence pair representation were in most cases negligibly low, if any occurred. Furthermore, the classifier can be improved and easily interchanged in future work.
As our dataset is small, we did not implement an own deep neural network. The BERT model was considered as it is pre-trained and therefore only requires data to be fine-tuned. The fine-tuned BERT models yield better accuracy values on the dataset, but this does not imply that they will perform better on our task later on since the corpus does not contain sentences from the argument search engine.

4.2. Generation of a Tree

After training and investigating models that predict if there is a directed relation between two sentences, we still have to create an argumentation tree that conforms to the constraints given by the argumentation structure. The relational requirements [27] are the following to recap:

1. No argument can target itself.
2. Every argument targets exactly one other argument except the major claim, which is the root of the tree.
3. A single tree has to be constructed and hence there are no circular relations allowed.

The major claim and therefore the root of the tree is always known for certain: It is the topic of the discussion and used to query the argument search engine. Instead of using the class labels, we retrieve the probability for a relation from the classifier. The higher the probability, the likelier the relation.

4.2.1. Traversing and Modifying Graphs

This self-provided approach firstly determines and uses the most probable relation for every argument, except the major claim, to all other arguments. The possible resulting graphs are represented in Figure 4.1. With this, Requirement 2 is fulfilled and thus every graph that does not include the major claim as its root contains exactly one circle with any number of branches.

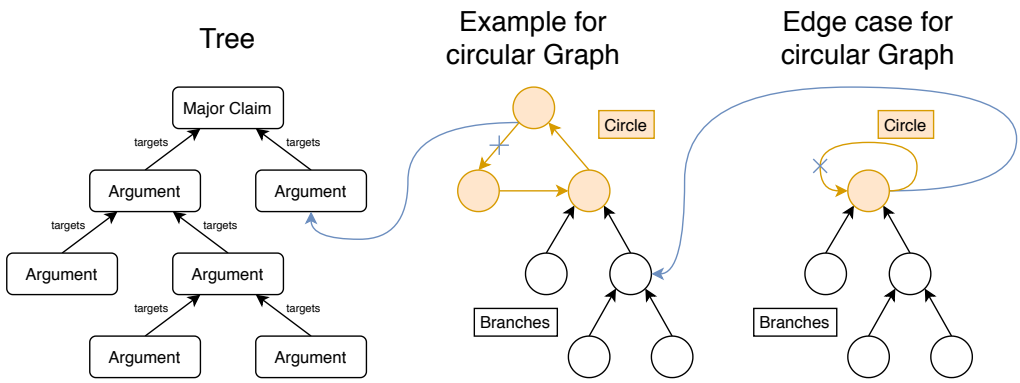

Figure 4.1.: Concept of Traversing and Modifying Graphs

Our aim is to link all graphs to the existing tree without any circles resulting in a satisfaction of Requirements 1 and 3. In order to do so, we loosen one arc in the circle at its parent and link it to an argument in one of the other graphs or directly to a node in the tree. This is shown exemplary by the blue arrows in the figure. For this procedure, the node in the circle with the most probable relation to any node outside the graph is selected and its target relation is changed to this node. If we would allow the arc to link

to a node within the same graph, we could construct another circle and the algorithm therefore might not terminate. After iterating over all circles, a single argumentation tree fulfilling all requirements is constructed.

4.2.2. Binary Integer Linear Programming

This approach is adapted from Stab and Gurevych [38] where Integer Linear Programming was used to join and globally optimize the results of a ternary argument type classifier and a binary relation classifier like ours. With this, they additionally ensured that the resulting structures are trees. The major claim was not part of their problem formulation as they connected all arguments without an outgoing relation directly to it.
We use this idea to fulfil the relational constraints of the argumentation structure including the major claim. The objective function σ to be maximized over all $r_{t,f} \in \{0, 1\}$ is

$$\sigma = \underset{r}{\operatorname{argmax}} \sum_{t=1}^{n} \sum_{f=1}^{n} p_{t,f} \cdot r_{t,f} \tag{0}$$

with $r_{t,f}$ denoting a boolean value for the relation from argument f to argument t and $p_{t,f}$ being the probability for this relation. n is the total number of arguments. With this function, we maximize the sum of the probabilities of the arcs in the generated tree under the following constraints:

$$\forall i : r_{i,i} = 0 \tag{1}$$

$$\forall f \neq \arg\{\text{MajorClaim}\} : \sum_{t=1}^{n} r_{t,f} = 1 \tag{2a}$$

$$\forall t : r_{t,\arg\{\text{MajorClaim}\}} = 0 \tag{2b}$$

Constraint (1) guarantees that Requirement 1 and the Constraints (2a) and (2b) that Requirement 2 is satisfied. We still have to fulfil Requirement 3. Therefore, the subsequent formulas are devised with $h_{t,f} \in \{0, 1\}$ representing indirect relations between the arguments analogous to $r_{t,f}$:

$$\forall t \forall f : r_{t,f} \leq h_{t,f} \tag{3.0 a}$$

$$\forall i \forall j \forall k : h_{k,i} - h_{j,i} - h_{k,j} \geq -1 \tag{3.0 b}$$

$$\forall i : h_{i,i} = 0 \tag{3.1}$$

$$\forall f \neq \arg\{\text{MajorClaim}\} : h_{\arg\{\text{MajorClaim}\},f} = 1 \tag{3.2}$$

Equation (3.0 a) ensures that at least all direct relations are captured in the $h_{t,f}$ values. Formula (3.0 b) propagates the indirect relations. To understand this better, we assume the scenario depicted in Figure 4.2. $h_{k,i}$ has to be 1 as there holds an indirect relation

from argument i to argument k over argument j. Hence, the formula is true for $h_{k,i} = 1$ and false for $h_{k,i} = 0$. Constraint (3.1) specifies that there exists no indirect relation to an argument itself. This means that the resulting structures do not contain any circles and are therefore trees. In addition, we need Equation (3.2) to ensure that a single tree is constructed with all arguments at least indirectly linked to the major claim.

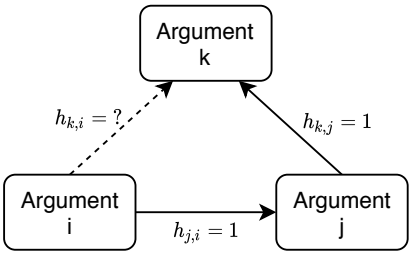

Figure 4.2.: Scenario to understand Equation (3.0 b)

After solving the Binary Integer Linear Programming problem defined above, the $r_{t,f}$ values state which target relations hold. Due to the problem formulation, the resulting argumentation tree fulfils all relational constraints straight away.

4.2.3. Discussion

A direct evaluation for each of the approaches is not possible as we do not have data that provides us with a control input.
Additionally, an argument could be discarded if the resulting class labels are no-relation for all other arguments or the probabilities for the relations are all below a certain threshold. This auxiliary can be added in a pre-processing step for both approaches if required. In our case, no arguments are discarded as we assume that every argument retrieved by the search engine makes a relevant contribution to the discussed topic.

5. Implementation of the Argumentative Relation Classification Pipeline

In this chapter, we present our pipeline for argumentative relation classification, which includes the complete process from querying the argument search engine over relation classification to the output of the defined argumentation structure. This resulting structure can be utilized by an argumentative dialogue system like EVA.
The pipeline includes different selection options: One can decide which APIs from ArgumenText are queried, which model predicts the probabilities for the relations and which approach generates the argumentation tree. Additionally, this setup enables us to evaluate the different combinations in an annotation study, which is discussed in Chapter 6.

5.1. Interface to the ArgumenText APIs

Daxenberger et al. [11] created a service infrastructure for ArgumenText that allows accessing single services via REST APIs[1].
In the first step, we either use the Search API to collect arguments from a static web crawl or the Classify API in order to retrieve arguments from our own input source which can be a text, a list of sentences or an explicit URL of a web page. When the Search API is used, we additionally provide the option to select how many arguments are further processed out of the retrieved ones. The arguments are split according to their stance towards the search query, which is the major claim, and are ranked according to their multiplied confidence scores. If possible, we extract the same amount of arguments with stance pro and con in order to create a balanced discussion.
Afterwards, the arguments can optionally be grouped using the Cluster API. This has the following effect in our pipeline: Generally, all collected arguments have to be paired with each other leading to an effort of $\mathcal{O}(n^2)$ if we process n arguments. When clustering is applied, we only combine all arguments within a cluster reducing the effort to $\mathcal{O}(n \cdot m)$ where m denotes the average number of arguments in a cluster. Hence, clustering can speed up the computation but also effects the results.

5.2. Presentation of Argumentative Relations in a Single Tree

This part is the centrepiece of the argumentative relation classification pipeline. For every step presented in the previous Chapter 4, we implement the different approaches using the theory and the results obtained there.

[1] `https://api.argumentsearch.com`

5.2.1. Prediction of the Probability for a Relation between Arguments

We reason our selection of one BERT model and one feature-based Support Vector Machine and briefly explain how a measure for the probability instead of the class label is retrieved for each of them. The models employed in the pipeline are trained on the complete dataset.

Fine-tuning a BERT Model

The $\text{BERT}_{\text{base}}$-256 uncased model fine-tuned with the parameters batch size 16 and learning rate 4e-5 is selected as it yields the best results in the evaluation on the dataset. We train the model for three epochs in order to avoid overfitting. As shown in Table 4.1, there is no improvement in epoch four and only a tiny improvement in the mean-value of the accuracy in epoch five.
The probability for the class labels is derived by applying the softmax function to the output of the model. We only use the result for the label relation.

Features and Classifiers

We decide to use the NuSVC with radial kernel, nu-value 0.5 and feature set 0 even though the NuSVCs with nu-value 0.5 and feature set 9 yield an accuracy that is 0.2% better as can be seen in Tables 4.4 and 4.5. Our choice is due to the computational effort to calculate feature set 9: It takes at least three times longer than calculating feature set 0. Table 5.1 shows the computation time needed to calculate the respective features averaged over nearly 1,000 argument pair combinations created by querying ArgumenText for arguments concerning the topic "nuclear energy is good".

Feature	**Average Computation Time in ms**
sentence pair	586.5
single sentences	1174.0
token and punctuation statistics	2.8
common words	1.1
edit distance	8.3
Jaro similarity	0.6
Jaro-Winkler similarity	0.6

Table 5.1.: Computation time for features averaged over nearly 1,000 argument pair combinations

The measure we gain for the probability is not the actual probability for the label relation. The user guide[2] points out that the Platt scaling [25], which is used to estimate the probability, not only has theoretical issues but also is an expensive operation and may

[2] https://scikit-learn.org/stable/modules/svm.html#scores-and-probabilities

yield results that are inconsistent with the classified label. As suggested by the user guide, we use the `decision_function` which returns a score for the confidence of the classification and therefore an ordered measure for the likelihood of a relation. Unlike probabilities, the scores are not in range from 0 to 1. Instead, the following holds in our case: Positive values imply the class label relation and negative values the class label no-relation. The higher the absolute value of the score, the more confident is the estimation for this label. Consequently, a relation is likelier, the higher the signed value of the score is.

5.2.2. Generation of a Tree

Both approaches are implemented as described in Section 4.2 with technical improvements explained in detail below. The probability measures are represented in a probability matrix $P = (p_{t,f})_{t=0...n,f=0...n}$ where $p_{t,f}$ is the likelihood for a relation from argument f to argument t, n the number of arguments without the major claim and 0 denoting the major claim. Our aim is to gain the relation matrix $R = (r_{t,f})_{t=0...n,f=0...n}$ with $r_{t,f} \in \{0, 1\}$ interpretable as a boolean value representing the existence of a target relation in the argumentation structure. We additionally define a relation row r with $r[f] = t$. In the following, we ignore any $r_{t,0}$ or $r[0]$ values as we know the major claim, which has no outgoing relation, in advance. This leads to less special cases that are only existent due to the major claim being the root of the tree.

Traversing and Modifying Graphs

The complete implementation is depicted as a flowchart in Figure 5.1. After creating a relation row containing the position of the first occurring maximum for each column in the probability matrix, we have to implement algorithms to find and process circles.

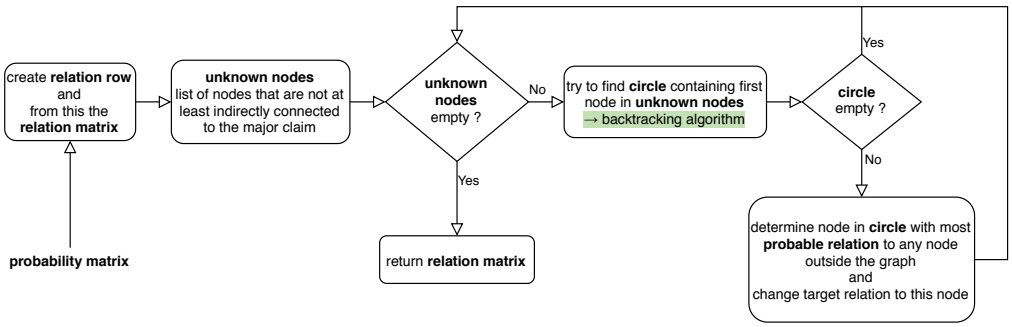

Figure 5.1.: Implementation of Traversing and Modifying Graphs as a flowchart

In order to find all nodes that are part of a circle, we use the method of backtracking. Figure 5.2 shows the corresponding flowchart. Whenever `true` is returned, the backtracking stops. This is the case when a circle is found and therefore the start node is part of the circle. It is also possible that the algorithm cannot find a circle in case the start

node is not contained in a circle. This occurs when the start node is part of a branch or when not all branches of a previously processed circle were visited and thus are not deleted from the unknown nodes list even though they are already indirectly connected to the major claim. `False` is returned when nodes belonging to a branch are detected. These nodes are removed from the circle list and are already not part of the unknown nodes list anymore.

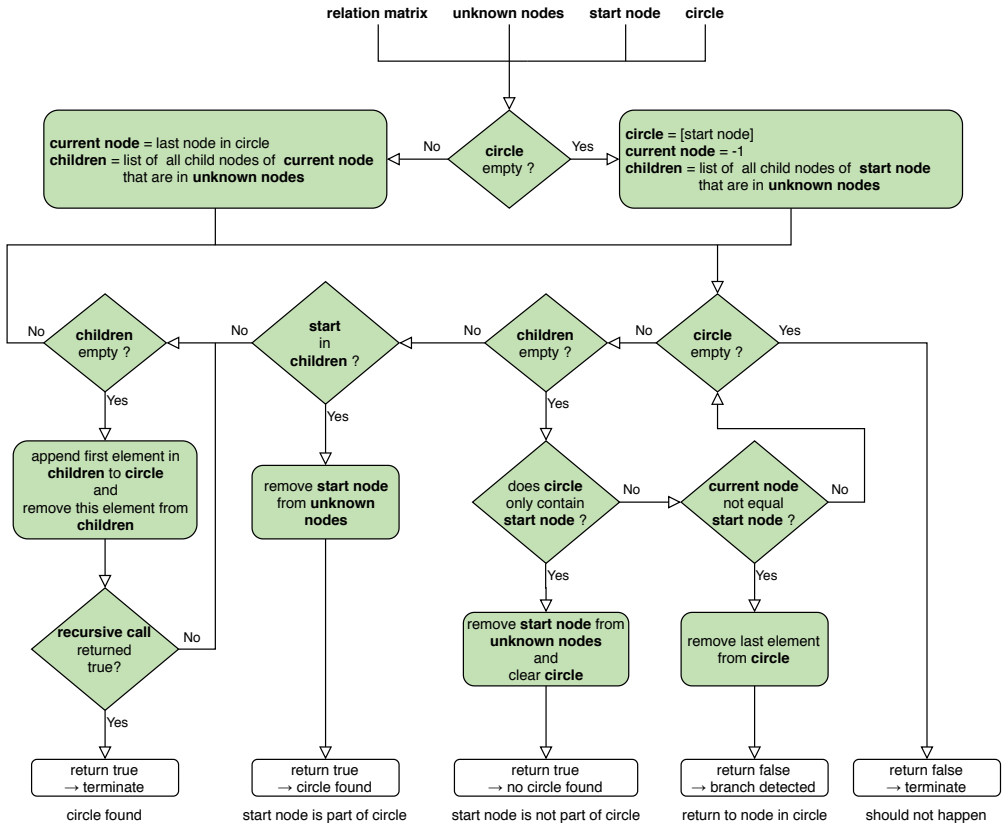

Figure 5.2.: Flowchart of backtracking algorithm to find circles

Initially, the backtracking is called with the created relation matrix, the current unknown nodes list, the start node which is the first node in the unknown nodes list and an empty circle list. All lists are handed over via call-by-reference and consequently the manipulations in the backtracking algorithm have a global effect.
When a circle is found, which is equivalent to the circle list not being empty, the node in the circle with the most probable relation to a node outside the graph is determined and the target relation is changed to this node.
The detection and processing of the circles are implemented modularly, so they can be interchanged independently.

Binary Integer Linear Programming

The implementation is realized using the cvxpy [13] library. We define the relation matrix as a `cvxpy.Variable` that only contains boolean values. The helper matrix $H = (h_{t,f})_{t=0...n, f=0...n}$ to capture the indirect relations is defined equally. In order to minimize the number of single constraints, we transform them from the presented summation form into matrix form whenever possible.
The objective function is to maximize the sum over all elements after multiplying the probability matrix element-wise with the relation matrix. The Constraints (1) and (3.1) of the form $\forall i : x_{i,i} = 0$ specify that the diagonal of the matrix X has to be a vector of zeros. Constraints (2a) and (2b) transform into the following: The column-wise sum of the relation matrix is an all-ones vector with only the first entry being a zero. In order to fulfil Equation (3.2), the same vector has to result for the first row of the helper matrix. $R \leq H$ represents the transformation of Formula (3.0 a). For Constraint (3.0 b), we could not find an equivalent matrix notation.
This results in the implementation of the problem formulation captured in Figure 5.3.

```
matrix_shape = probability_matrix.shape
matrix_nm = matrix_shape[0]
vector_nm_zeros = numpy.zeros(matrix_nm, dtype=int)
vector_nm_zero_ones = numpy.ones(matrix_nm, dtype=int)
vector_nm_zero_ones[0] = 0

relation_matrix = cvxpy.Variable(matrix_shape, boolean=True)
helper_matrix = cvxpy.Variable(matrix_shape, boolean=True)

# (0)
objective_function
  = cvxpy.Maximize(cvxpy.sum(cvxpy.multiply(probability_matrix, relation_matrix)))

constraints = [
  cvxpy.diag(relation_matrix) == vector_nm_zeros,              # (1)
  cvxpy.sum(relation_matrix, axis=0) == vector_nm_zero_ones,   # (2a) and (2b)
  relation_matrix <= helper_matrix,                            # (3.0a)
  cvxpy.diag(helper_matrix) == vector_nm_zeros,                # (3.1)
  helper_matrix[0] == vector_nm_zero_ones                      # (3.2)
]

# (3.0b)
for i in range(matrix_nm):
  for j in range(matrix_nm):
    for k in range(matrix_nm):
       constraints += [
         helper_matrix[k][i] - helper_matrix[j][i] - helper_matrix[k][j] >= -1
       ]
```

Figure 5.3.: Implementation of Binary Integer Linear Programming using the cvxpy library

5.3. Generation of the Debating Ontology

Out of the relation matrix created in the previous step, we generate the debating ontology. Since the argumentation structure is encoded as an OWL [4] ontology in EVA's case [29], we proceed the same way. The file with all component and relation definitions as used in Rach et al.'s [27] work was provided by the author.
Firstly, we transform the relation matrix into a relation row to extract the respective target relations. Secondly, we have to determine the type of each relation. When the related arguments have the same stance towards the major claim, the relation is supportive, otherwise it is attacking. For arguments directly related to the major claim, the relation is supportive when their stance is pro and attacking in the other case. Finally, the type of each argument has to be determined. The major claim is the topic of the discussion. All arguments targeting the major claim are claims when they are targeted by at least one argument. The remaining arguments are premises.
The created OWL file can be utilized by EVA as an underlying argumentation structure. A sample discussion between two agents concerning the topic "nuclear energy is good" is given in Appendix A.

5.4. Discussion

The argumentative relation classification pipeline provides the user with a command line interface which requires the topic of the discussion and enables to select

- between the Search and Classify API to retrieve arguments.
- whether clustering is applied.
- which model is used for the relation prediction: the Support Vector Machine with the sentence pair feature or the fine-tuned BERT model.
- which approach generates the argumentation tree: our self-created Traversing and Modifying Graphs approach or the Binary Integer Linear Programming algorithm.

Due to the modular implementation, the pipeline can be easily extended with further models for relation classification and approaches to generate an argumentation structure out of the probability matrix. The APIs to the search engine and the generation of the output can likewise be interchanged or extended. This quality of the pipeline implementation allows for easy use in future research and adoption to differently defined argumentation structures and output formats.
We investigated the computation time for the different approaches. In Table 5.2 the times for predicting the probability of a relation from one argument to another with the classification models are noted. At the first glance it seems as if the SVM is significantly faster than the BERT model but the SVM does require feature extraction in advance which the BERT model does not. Therefore, there are no noteworthy differences in computation time between the models.

Model	Average Computation Time in ms
BERT	561
NuSVC with radial kernel	2
NuSVC with radial kernel + sentence pair representation	589

Table 5.2.: Average computation time for BERT model and NuSVC with radial kernel and feature set 0 for one relation

Matrix	Average Computation Time
no circle	26 s
maximum number of circles	30 s
single circle	4 min 27 s

Table 5.3.: Average computation time for Binary Integer Linear Programming with 20 retrieved arguments

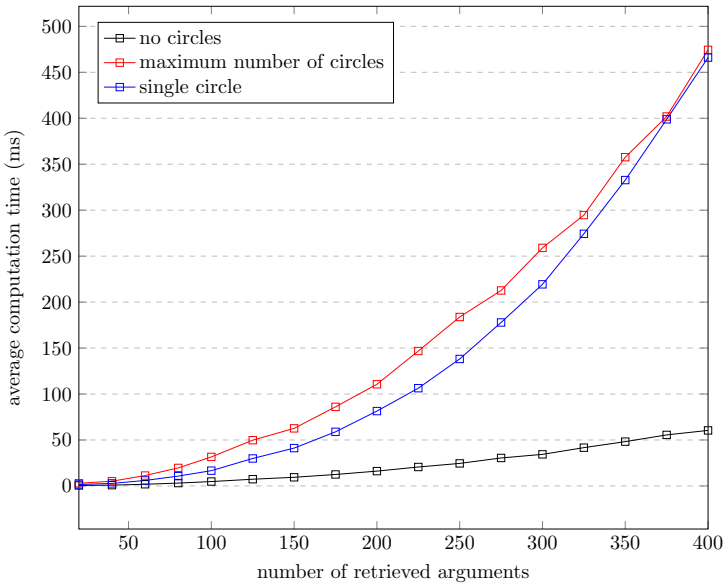

Figure 5.4.: Average computation time for Traversing and Modifying Graphs

For comparing the computation time between the approaches to generate the tree, we construct three different types of probability matrices. The values that are important for the structure of the respective matrix are set to 1, the other elements are assigned a random value in range $[0, 1)$. One matrix contains the maximum number of circles,

meaning that every argument initially targets itself. Another has no circles and is hence already an argumentation tree after determining the most likely outgoing relation for each argument. In the last matrix, all arguments, except the major claim, build a single circle. We stopped early for Binary Integer Linear Programming as it is significantly outperformed by Traversing and Modifying Graphs. Table 5.3 shows the result for Binary Integer Linear Programming with 20 retrieved arguments, whereas Figure 5.4 plots the computation time for Traversing and Modifying Graphs from 20 up to 400 retrieved arguments without crossing the second mark.

6. Evaluation

In order to evaluate the outcome of the previously presented argumentative relation classification pipeline, we carried out an annotation study. We take different combinations of models and tree generation approaches and also clustering into account. The study is necessary for the evaluation as there exists no ground truth that we can compare our results to.
Since we focus on investigating the contributions of our own work, we evaluate the created pairs of related sentences individually and not in a sample dialogue. An annotation concerning a sample discussion, like the one shown in Appendix A, would take the agent strategy and the quality of the arguments themselves into account [27], which we both cannot influence.
After discussing our set of annotation guidelines, we reason our decision for multi-κ as an inter-annotator agreement measure. Finally, we present the results of the executed annotation study.

6.1. Annotation Guidelines

Annotation guidelines have to be formulated as clearly as possible in order to retrieve a result that is as objective as possible. Hence, we search for existing guidelines that allow to conclude whether the predicted target relation holds.
The guidelines provided by Stab and Gurevych [36], that were used to create the argumentation structure for EVA by hand, are not suitable for our study. The original version[1] and the updated version[2] for their later work [38] both require the type of the argument component as a prerequisite and guide the user to identify which component is supported or attacked by the currently considered one. In contrast, we present the annotators with argument pairs in order to determine if the predicted relation holds, regardless of whether the relation is supportive or attacking.
Instead, we utilize the Local Relevance criterion proposed by Wachsmuth et al. [41] which questions whether an argument component leads to the acceptance or rejection of its conclusion. From this we can conclude if the relation is meaningful in an argumentation. The other guidelines for argumentative quality assessment defined in that paper are not usable in our study as they either take the quality of the argument itself or the quality of the resulting argumentation into account.

[1] https://www.informatik.tu-darmstadt.de/ukp/research_6/data/argumentation_mining_1/argument_annotated_essays/index.en.jsp
[2] https://www.informatik.tu-darmstadt.de/ukp/research_6/data/argumentation_mining_1/argument_annotated_essays_version_2/index.en.jsp

Gold et al. [14] investigated the interactions between the semantic relations Contradiction, Textual Entailment, Specificity, Paraphrase and Semantic Similarity on sentence-level. We exploit Contradiction to capture attacking relations, Entailment for supportive relations and the Specificity guideline is used to model the general relation between a parent and a child node as the targeting argument can give further detail on the subject. Arguments should not just be related due to their similarity. Therefore, we do not include Similarity in our annotation. Instead, Paraphrase is considered for the following reason: The arguments retrieved by ArgumenText should be free of duplicates [35]. In case two sentences with approximately the same meaning are retrieved, it might be likely that a relation is predicted between these and this should not distort our results.
The set of annotation guidelines containing the annotation categories Local Relevance, Contradiction, Entailment, Specificity and Paraphrase is presented in Appendix B. The general instructions additionally refer to our spreadsheet implementation that is used to guide through the annotation process where previously annotated categories are not accessible. This is done to prevent the annotators from looking up their answers and thereby creating an artificial correlation between the categories. Moreover, it ensures that the given answers comply to the labels yes and no. In the annotation study, Sentence 1 is always the parent node and Sentence 2 the child node of the relation, which means that we evaluate whether Sentence 2 targets Sentence 1.
An odd number of annotators is required to be able to build a conclusive majority voting for each argument pair in each category. When out of the majority at least one category is labelled with yes, we conclude that the predicted relation for the respective argument pair is correct as each category yields a sufficient criterion for a target relation by itself.

6.2. Inter-Annotator Agreement

To ensure the validity of our study, we select a measure to calculate the inter-annotator agreement for more than two annotators. In Gold et al.'s [14] crowd-sourcing annotation, the average percentage agreement and an averaged kappa-score measure were calculated. The percentage agreement measure [2] does not consider that agreement can occur by chance. Consequently, annotation tasks with fewer answer possibilities gain a higher agreement only due to chance and seemingly high values may actually be worse than the agreement value expected by chance. Therefore, chance-corrected measures are employed [2]. Wachsmuth et al. [41] applied Krippendorff's α [2] to measure the inter-annotator agreement. In their study, a scale was used and α captures the following in a distance function: A rating of 1 and 3 yields more disagreement than a rating of 1 and 2, for example. As we are not using a scale but the binary answers yes and no, we are not in need of this feature.
Two measures treating all agreements and disagreements the same way and allowing for multiple annotators are multi-π [2] and multi-κ [2]. The former uses a chance distribution differing for each answer possibility but being the same for each annotator. With this, it is more concerned about the reproducibility of an annotation. Instead, multi-κ assumes that each annotator has an individual chance distribution. Due to this, it is a better measure

for trustworthiness and thus for the validity of our evaluation-related annotation. Hence, we select multi-κ as an inter-annotator agreement measure for our study.
Since the annotation study is executed with the help of a spreadsheet program, we implement the multi-κ measure in this program, as well. A κ value above 0.4 yields moderate agreement, above 0.6 it is substantial and above 0.8 perfect agreement is reached [2]. Additionally, the average percentage-agreement and Krippendorff's α, with a distance function that treats all equal and unequal annotations the same, are implemented in order to be able to compare the agreement values to the original works.

6.3. Results for the Argumentative Relation Classification Pipeline

We retrieved 20 arguments for each of the topics "nuclear energy is good" and "animal testing is good" and used different combinations of the approaches in the argumentative relation classification pipeline to create the argumentation structure.
Five annotators without task-related background were asked to label the resulting argument pairs, which were predicted to hold a relation, in each annotation category. In order to eliminate outliers and retrieve a result that is as objective as possible, we consider the labels of the three most agreeing annotators for each category. These are annotators 2, 4 and 5 for Contradiction and Local Relevance, annotators 2, 3 and 4 for Specificity and Paraphrase and annotators 3, 4 and 5 for Entailment. The achieved inter-annotator agreement is provided in Table 6.1 with the mulit-κ values yielding substantial up to perfect agreement. The average percentage and Kirppendorff's α are included in the table for a comparison to the original works in the following Section 6.4.

	Contradiction	**Entailment**	**Specificity**	**Paraphrase**	**Local Relevance**
multi-κ	72.4%	69.3%	76.1%	81.9%	66.0%
percentage	81.9%	79.9%	83.3%	88.9%	74.3%
Krippendorff's α	41.8%	35.9%	57.9%	50.6%	48.8%

Table 6.1.: Inter-annotator agreement for most agreeing annotator triplets in the annotation study

We split the annotation study into two parts. Firstly, all resulting argument pairs for each possible combination without clustering were annotated. Table 6.3 shows the correlation of the related arguments between the combinations of the approaches for both topics. From this we can conclude the degree of difference between the approaches. The selected model for relation classification has a major impact since approximately only 12.5% of the predicted relations were the same between the BERT model and the SVM. The algorithms to generate the tree just differed in three relations for the same model. In the second part of the study, we investigate the impact of clustering, which was only done for the better performing model due to limited resources.

The results of the complete study are given in Table 6.2 with the percentage values stating the following: For a certain percentage, the arguments in the argumentation structure are indeed related as declared and therefore the automatically created structure enables the argumentative dialogue system to respond with a related argument to the previous one in those cases. The BERT model outperforms the SVM and thus clustering was investigated with the former one. In Table 6.3, we see that nearly 50% of the relations were differing only due to clustering. The BERT model with clustering performs worse than without clustering but its results are still up to 10% better than the ones of the SVM. The different tree generation approaches are quite close in the results, which is expectable as the argument pairs were nearly the same for them. For the SVM, they achieve the same result and for the BERT model, the difference is negligibly low under the consideration that 2.5% are corresponding to one argument pair that was related correctly by the respective approach.

SVM		**BERT**		**BERT Cluster**	
TMG	**BILP**	**TMG**	**BILP**	**TMG**	**BILP**
62.5%	62.5%	75.0%	77.5%	72.5%	70.0%

Table 6.2.: Correctly classified relations within a combination of approaches
TMG: Traversing and Modifying Graphs
BILP: Binary Integer Linear Programming

		SVM		**BERT**		**BERT Cluster**	
		TMG	**BILP**	**TMG**	**BILP**	**TMG**	**BILP**
SVM	**TMG**	40	37	5	6	7	7
	BILP	37	40	4	5	6	6
BERT	**TMG**	5	4	40	37	22	22
	BILP	6	5	37	40	22	22
BERT Cluster	**TMG**	7	6	22	22	40	38
	BILP	7	6	22	22	38	40

Table 6.3.: Correlation of the related arguments between combinations of approaches
TMG: Traversing and Modifying Graphs
BILP: Binary Integer Linear Programming

6.4. Discussion

Taking the quality of the output and the computation time into account, we recommend the use of the following approaches: The BERT model should be used as it yields better results in the annotation study and has no disadvantages over the SVM in computation time. The Traversing and Modifying Graphs algorithm should be used since there

are no significant quality differences in the study and this approach highly outperforms Binary Integer Linear Programming in computation time as shown in Section 5.4. When a large number of arguments has to be structured, clustering should be considered as it minimizes the number of possible argument combinations in our pipeline. The BERT model with clustering yields worse results in our study but when other parameters or another clustering algorithm is used, results will differ and hence may improve.
The annotators stated that the given examples in the guidelines are much easier to annotate than the actual argument pairs in the study. This is probably due to the following reasons: The examples have to be unmistakable. Therefore, they have to be formulated as clearly as possible in order to prevent errors due to misconception. Furthermore, the retrieved arguments originate from various texts with different authors and are thus out of context and have a heterogeneous structure.
Nevertheless, the agreement between the triplets of annotators is high and comparable to the agreement achieved in the original works. For the category Local Relevance, Wachsmuth et al. [41] reported a Krippendorff's α of 47% for the most agreeing annotator pair which is close to our value of 48.8%. Thereby, we have to keep in mind that the study setups differ: Wachsmuth et al. [41] let the annotators vote on a scale instead of offering the answers yes and no. The average percentage agreement in Gold et al.'s [14] work is directly comparable to our percentage values which are shown above in Table 6.1. All results are plus/minus 3% in comparison, only Contradiction yields an agreement of 81.9% in our study compared to 94% in the original one. As this measure lacks the consideration of chance, no clear conclusions can be drawn. Additionally, they reported an averaged kappa-score in adoption to their crowd-sourcing annotation with agreements from 56% up to 71%.
In any case, our multi-κ values yield a high enough agreement to qualify the executed annotation study as valid and we therefore conclude that the results are reliable.

7. Conclusion

7.1. Summary

In this work, we presented our argumentative relation classification pipeline that exploits techniques from the field of argument mining to automatically create an argumentation structure regarding any topic.
It queries the argument search engine ArgumenText either through the Search API or the Classify API for arguments concerning the topic of the discussion. Optionally, the Cluster API can be used in order to decrease the number of possible argument combinations and thus reduce the computational complexity. The relation classification comprises the following steps. Firstly, we predict the probability for a relation between all possible pairwise combinations of arguments. For this step, the pipeline provides a fine-tuned $\text{BERT}_{\text{base}}$-256 uncased model and a NuSVC with radial kernel using an embedding representing the sentence pair as a feature vector. Secondly, the relational constraints of the argumentation structure have to be satisfied. The pipeline again implements two different approaches to this. Traversing and Modifying Graphs determines the most probable outgoing relation for each argument and then detects and processes circles. Binary Integer Linear Programming maximizes the sum of the probabilities of the relations that hold under the relational constraints given as formulas. Both approaches output an argumentation tree that fulfils all constraints given by the argumentation structure. Finally, the argumentative relations attack and support are derived from the stance information provided by the argument search engine and the type of the argument components is determined from the position of the component in the tree. The automatically generated argumentation structure is output as an OWL file which can be utilized as a database by an argumentative dialogue system like EVA.
The pipeline offers a command line interface to the user to select between the APIs and approaches. Due to the modular implementation, the pipeline can be easily extended and facilitates further research in that area.
In order to evaluate the output of our pipeline, we carried out an annotation study containing the new composition of annotation categories Contradiction, Entailment, Specificity, Paraphrase and Local Relevance. A pair of arguments is concluded to actually hold a relation when it is labelled with yes in at least one of the categories.
Considering the results of the annotation study and the investigated computation times, we conclude the BERT model in combination with the Traversing and Modifying Graphs algorithm to perform best. Clustering only slightly decreases the results in our study, while at the same time, offering a computationally efficient alternative to a comparison of all sentence pair combinations in the case of a large argument pool. In the annotation

study, the above-named setup without clustering identifies 75% of the relations correctly which yields promising results.
Consequently, our work enables argumentative dialogue systems to automatically collect and structure arguments regarding any topic. The computer-generated argumentation structures can be utilized to train the agent's strategy on various topics and serve as a database for a discussion with a human or another agent. Hence, we substantially contribute to the flexibility of such systems.

7.2. Future Work

Based on this thesis, the following opportunities for future work arise:
The existing pipeline could be improved especially by advancing the relation classification model that predicts the probability for a relation between two arguments. Further research is possible in the area of feature extraction. There are more characteristics of sentences that can be included in the extraction than investigated in this work. Table 3.3 gives an overview over possible features and not yet considered features, e. g. aspects or background knowledge from knowledge graphs, may improve the quality of the classifiers. Collecting a corpus of sentential arguments retrieved by ArgumenText that are arranged as directed pairs and labelled with relation or no-relation, gives the opportunity to train the relation classification models directly on the aimed data. Furthermore, deep neural networks with an architecture designed for the explicit task of relation prediction between arguments can be created and trained from scratch when the corpus is large enough. Additionally, task-specific clustering approaches can be examined. The ArgumenText Cluster API uses the similarity of the sentences as a criterion. Clustering according to the extracted aspects of the arguments is a possible approach.
When we look at the bigger picture, the herein presented work is also a starting point for including user's utterances that cannot be mapped to an existing argument in the generated structure. Via the Classify API provided by ArgumenText we are able to validate whether the sentence is argumentative and in case it is, we also retrieve its stance towards the topic of the discussion. When we not only keep the obtained structure but also the predicted probabilities in memory, we can include the new argument with $\mathcal{O}(2n)$ comparisons into the argumentation structure. Thus, the argumentative dialogue system is able to select a suitable answer to the so far unknown utterance and to exploit received arguments in later discussions.

Bibliography

[1] Yamen Ajjour, Henning Wachsmuth, Johannes Kiesel, Martin Potthast, Matthias Hagen, and Benno Stein. 2019. Data Acquisition for Argument Search: The args.me corpus. In *42nd German Conference on Artificial Intelligence (KI 2019)*, pages 48–59. Springer.

[2] Ron Artstein and Massimo Poesio. 2008. Survey article: Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

[3] Katie Atkinson, Pietro Baroni, Massimiliano Giacomin, Anthony Hunter, Henry Prakken, Chris Reed, Guillermo Simari, Matthias Thimm, and Serena Villata. 2017. Towards artificial argumentation. *AI Magazine*, 38(3):25–36.

[4] Sean Bechhofer. 2009. *OWL: Web Ontology Language*, pages 2008–2009. Springer US, Boston, MA.

[5] Steven Bird, Ewan Klein, and Edward Loper. Natural language processing with python [online]. 2019. updated for Python 3 and NLTK 3.

[6] Lucas Carstens and Francesca Toni. 2015. Towards relation based argumentation mining. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 29–34, Denver, CO. Association for Computational Linguistics.

[7] Lisa Andreevna Chalaguine, Anthony Hunter, Henry Potts, and Fiona Hamilton. 2019. Impact of argument type and concerns in argumentation with a chatbot. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1557–1562. IEEE.

[8] Sihao Chen, Daniel Khashabi, Chris Callison-Burch, and Dan Roth. 2019. Perspectroscope: A window to the world of diverse perspectives. *Computing Research Repository*, arXiv:1906.04761.

[9] Artem Chernodub, Oleksiy Oliynyk, Philipp Heidenreich, Alexander Bondarenko, Matthias Hagen, Chris Biemann, and Alexander Panchenko. 2019. TARGER: Neural argument mining at your fingertips. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 195–200, Florence, Italy. Association for Computational Linguistics.

[10] Oana Cocarascu and Francesca Toni. 2017. Identifying attack and support argumentative relations using deep learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1374–1379, Copenhagen, Denmark. Association for Computational Linguistics.

[11] Johannes Daxenberger, Benjamin Schiller, Chris Stahlhut, Erik Kaiser, and Iryna Gurevych. 2020. ArgumenText: Argument classification and clustering in a generalized search scenario. *Datenbank-Spektrum*, 20(2):115–121.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[13] Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5.

[14] Darina Gold, Venelin Kovatchev, and Torsten Zesch. 2019. Annotating and analyzing the interactions between meaning relations. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 26–36, Florence, Italy. Association for Computational Linguistics.

[15] Yufang Hou and Charles Jochim. 2017. Argument relation classification using a joint inference model. In *Proceedings of the 4th Workshop on Argument Mining*, pages 60–66, Copenhagen, Denmark. Association for Computational Linguistics.

[16] Jonathan Kobbe, Juri Opitz, Maria Becker, Ioana Hulpus, Heiner Stuckenschmidt, and Anette Frank. 2019. Exploiting Background Knowledge for Argumentative Relation Classification. In *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 8:1–8:14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[17] John Lawrence and Chris Reed. 2019. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818.

[18] Dieu Thu Le, Cam-Tu Nguyen, and Kim Anh Nguyen. 2018. Dave the debater: a retrieval-based and generative argumentative dialogue agent. In *Proceedings of the 5th Workshop on Argument Mining*, pages 121–130, Brussels, Belgium. Association for Computational Linguistics.

[19] Ran Levy, Ben Bogin, Shai Gretz, Ranit Aharonov, and Noam Slonim. 2018. Towards an argumentative content search engine using weak supervision. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2066–2081, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

[20] Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Trans. Internet Technol.*, 16(2).

[21] Annie Louis, Aravind Joshi, Rashmi Prasad, and Ani Nenkova. 2010. Using entity features to classify implicit discourse relations. In *Proceedings of the SIGDIAL 2010 Conference*, pages 59–62, Tokyo, Japan. Association for Computational Linguistics.

[22] Stefano Menini and Sara Tonelli. 2016. Agreement and disagreement: Comparison of points of view in the political domain. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2461–2470, Osaka, Japan. The COLING 2016 Organizing Committee.

[23] Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 108–112, Seoul, South Korea. Association for Computational Linguistics.

[24] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.

[25] John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

[26] Henry Prakken. 2005. Coherence and Flexibility in Dialogue Games for Argumentation. *Journal of Logic and Computation*, 15(6):1009–1040.

[27] Niklas Rach, Saskia Langhammer, Wolfgang Minker, and Stefan Ultes. 2019. Utilizing argument mining techniques for argumentative dialogue systems. In *Lecture Notes in Electrical Engineering*, pages 131–142. Springer Singapore.

[28] Niklas Rach, Yuki Matsuda, Johannes Daxenberger, Stefan Ultes, Keiichi Yasumoto, and Wolfgang Minker. 2020. Evaluation of argument search approaches in the context of argumentative dialogue systems. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 513–522, Marseille, France. European Language Resources Association.

[29] Niklas Rach, Klaus Weber, Louisa Pragst, Elisabeth André, Wolfgang Minker, and Stefan Ultes. 2018. Eva: A multimodal argumentative dialogue system. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, ICMI '18, page 551–552, New York, NY, USA. Association for Computing Machinery.

[30] Geetanjali Rakshit, Kevin K. Bowden, Lena Reed, Amita Misra, and Marilyn A. Walker. 2017. Debbie, the debate bot of the future. *Computing Research Repository*, arXiv:1709.03167.

[31] Gil Rocha and Henrique Lopes Cardoso. 2017. Towards a relation-based argument extraction model for argumentation mining. In *Statistical Language and Speech Processing*, pages 94–105. Springer International Publishing.

[32] Ariel Rosenfeld and Sarit Kraus. 2016. Strategical argumentative agent for human persuasion. *Frontiers in Artificial Intelligence and Applications*, 285(ECAI 2016):320–328.

[33] Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2020. Aspect-controlled neural argument generation. *Computing Research Repository*, arXiv:2005.00084.

[34] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 553–562, New York, NY, USA. Association for Computing Machinery.

[35] Christian Stab, Johannes Daxenberger, Chris Stahlhut, Tristan Miller, Benjamin Schiller, Christopher Tauchmann, Steffen Eger, and Iryna Gurevych. 2018. ArgumenText: Searching for arguments in heterogeneous sources. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 21–25, New Orleans, Louisiana. Association for Computational Linguistics.

[36] Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

[37] Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar. Association for Computational Linguistics.

[38] Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

[39] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Computing Research Repository*, arXiv:1706.03762.

[41] Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. 2017. Computational argumentation quality assessment in natural language. In *Proceedings of*

the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 176–187, Valencia, Spain. Association for Computational Linguistics.

[42] Henning Wachsmuth, Martin Potthast, Khalid Al-Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59, Copenhagen, Denmark. Association for Computational Linguistics.

[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *Computing Research Repository*, arXiv:1910.03771.

[44] Tangming Yuan, David J. Moore, and Alec Grierson. 2008. A human-computer dialogue system for educational debate: A computational dialectics approach. *I. J. Artificial Intelligence in Education*, 18:3–26.

A. Sample Dialogue between two EVA-Agents

The following discussion between Alice and Bob was held by two artificial EVA-agents utilizing an OWL file created by the argumentative relation classification pipeline. We queried ArgumenText for 70 arguments regarding the topic "nuclear energy is good" and used the BERT model in combination with Traversing and Modifying Graphs to achieve the argumentation structure.

Alice: nuclear energy is good

Bob: As with other types of energy, nuclear energy also has a few major drawbacks.

Alice: I can't agree with that. Nuclear energy has a very good safety record and hundreds of nuclear power plants all over the world have operated for more than fifty years without any serious accidents.

Bob: The most recent disaster has convinced the Germans that nuclear power is dangerous and has begun shutting down all of their nuclear power plants (Power ranges, 2012).

Alice: Could you be more specific?

Bob: I would like to go back to the beginning when you claimed: nuclear energy is good I'm still not convinced. The second reason is reactors accidents and the threat of weapons proliferation easily occurs.

...

Bob: I would like to go back to the beginning when you claimed: nuclear energy is good Why do you think that?

Alice: These next-generation reactors will power scores of businesses and homes - 1.6 million in Georgia alone - and it will do so affordably and reliably.

Bob: Could you be more specific?

Alice: Georgia residents whose power comes from the new reactors could realize up to $20 in savings on each utility bill by 2034.

Bob: Could you please elaborate?

...

B. Guidelines for the Annotation

The following sections contain the guidelines that were used in the annotation study. They are based on the guidelines in the original works by Gold et al. [14][1] and Wachsmuth et al. [41][2]. Whenever their formulations were suitable for our task, we adopted them.

B.1. General Instructions

You are always presented with two sentences: Sentence1 and Sentence2. Take care to not mismatch them in the questions!
In the case of pronouns (he, she, it, mine, his, our, ...) being used, you can assume they reference proper names, if your common sense does not suggest otherwise.

In the document, there is a sheet for each question category with several sentence pairs. You will always see only one sheet at a time named after the question category you are in right now (have a look at the bottom left). When you have answered the question for every sentence pair, click the "NEXT" button in order to go to the next sheet / question category. Keep in mind that once you have clicked the button you are not able to go back to that sheet.
This instruction sheet will always be available. To switch between this sheet and the question sheet, click on the respective slot at the bottom.

B.2. Contradiction

Definition: If Sentence1 is true, Sentence2 is false and the other way round.
Does Sentence2 contradict Sentence1?

Examples for "Sentence2 does contradict Sentence1":

- Sentence1: John bought a new house near the beach.
- Sentence2: John didn't buy the house near the beach.

The second sentence directly contradicts the first one – they can't both be true.

- Sentence1: Mary is on a vacation in Florida.
- Sentence2: Mary is at the office, working.

The two sentences can't be true at the same time Mary is either on vacation in Florida, or at the office. She can't be in two places.

[1] `https://github.com/MeDarina/meaning_relations_interaction`
[2] `http://argumentation.bplaced.net/arguana/data` The Dagstuhl-15512 ArgQuality Corpus

Examples for "Sentence2 does not contradict Sentence1":

- Sentence1: Mary is on vacation in Florida.
- Sentence2: John is at the office.

John and Mary are two different persons. There is no contradiction. Both statements can be true.

B.3. Entailment

Definition: If Sentence2 is true, Sentence1 or parts of Sentence1 are true, too.
Does Sentence2 cause Sentence1 to be true or partially true?

Examples for "Sentence2 causes Sentence1 to be true":

- Sentence1: John has a car.
- Sentence2: John bought a car from Mike.

In that case, the second sentence causes the first sentence to be true, as assuming that John bought a car, it means that he has a car now.

- Sentence1: Boys play games.
- Sentence2: Boys and girls play games.

In that case, the second sentence causes the first sentence to be true, as the second sentence says that both boys and girls play games, it also contains the information that boys play games.

Examples for "Sentence2 does not cause Sentence1 to be true":

- Sentence1: John bought a car from Mike.
- Sentence2: John has a car.

If the first sentence makes the second sentence true (but the second doesn't make the first true), answer with "no".

- Sentence1: He is an English teacher.
- Sentence2: He works as a teacher in Peru.

If you cannot tell if the second sentence causes the first sentence to be true, answer with "no".

B.4. Specificity

Is Sentence2 more specific than Sentence1 or partially more specific than Sentence1?

Examples for "Sentence2 is more specific than Sentence1":

- Sentence1: I like animals.
- Sentence2: I like cats.

As the 2nd sentence gives the more specific information on which animal is liked, it is more specific. Hence, you have to answer with "yes".

- Sentence1: The café sells coffee.
- Sentence2: The cute café has great coffee.

As the 2nd sentence gives the more specific information on both the café and the coffee, it is more specific. Hence, you have to answer with "yes".

Examples for "Sentence2 is not more specific than Sentence1":

- Sentence1: I like cats.
- Sentence2: I like animals.

As the 1st sentence gives the more specific information on which animal is liked, it is more specific. Hence, you have to answer with "no".

- Sentence1: I like cats.
- Sentence2: I like dogs.

Now, as in both cases the liked animal is mentioned, they have the same level of specificity. Hence, you have to answer with "no".

- Sentence1: He saw a blind cat.
- Sentence2: I like black dogs.

Now, as the information is very diverse, it is impossible to say which sentence is more specific. Hence, you have to answer with "no".

B.5. Paraphrase

Do Sentence1 and Sentence2 have approximately the same meaning?

Examples for "Sentence1 and Sentence2 have approximately the same meaning":

- Sentence1: John goes to work every day with the metro.
- Sentence2: He takes the metro to work every day.

In the content of the task, we assume that "He" and "John" are the same person.

- Sentence1: Mary sold her Toyota to Jeanne.
- Sentence2: Jeanne bought her Toyota from Mary Smith.

In the content of the task, we assume that "Mary Smith" and "Mary" are the same person.

Examples for "Sentence1 and Sentence2 have not the same meaning":

- Sentence1: Mary sold her Toyota to Jeanne.
- Sentence2: Mary had a blue Toyota.

The two texts are related but are not the same.

- Sentence1: John Smith takes the metro to work every day.
- Sentence2: John works from home every Tuesday.

The two texts are not closely related except for the person (John).

B.6. Local Relevance

Important: You should be open to see Sentence2 as relevant even if it does not match your own stance on the issue.
Does Sentence2 contribute to the acceptance or rejection of Sentence1?

Examples:
If you think Sentence2 is worthy of being considered as a reason or evidence for Sentence1, you have to answer with "yes".
If you think Sentence2 is similar regarding the conclusion of Sentence1, you have to answer with "yes".