Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Medieninformatik

Semi-supervised Cleansing of Web-based Argument Corpora

Bachelor's Thesis

Jonas Dorsch

1. Referee: Prof. Dr. Benno Stein
2. Referee: Jun.-Prof. Dr. Henning Wachsmuth

Submission date: August 31, 2020

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, August 31, 2020

.....
Jonas Dorsch

Acknowledgements

Foremost, I would like to express my special thanks to my supervisor Jun.-Prof. Dr. Henning Wachsmuth for his ongoing support during the creation of this thesis.
I also wish to thank Prof. Dr. Benno Stein and Jun.-Prof. Dr. Henning Wachsmuth for examining my work.

Special thanks also to Jana Puschmann, Milad Alshomary and Wei-Fan Chen for supporting my work by using their expertised knowledge in argumentation to manually annotate a sample of the generated results.

Many thanks also go to Jana Puschmann, Lucas Hübner and my father for proofreading this work and for making suggestions for improvement. Last but not least, I would like to express my endless gratitude to my family, who supported me throughout my studies and the creation of this thesis.

Abstract

Current research has yet to find a way to automatically detect and extract arguments from the world wide web in a reliable way. In the meantime, applications working on argumentative data, such as the argument search engine args.me developed by the Webis Group[1] and researchers analysing argumentative structures have to rely on pre-structured arguments extracted from so called debate portals. Given the varying quality of these arguments a cleansing of these corpora is necessary. Due to the high amounts of data collected from these platforms, manually processing the datasets becomes too time consuming. This thesis therefore introduces an approach to automatically filter unwanted segments of the argument texts. Based on a small predefined set of ground truth patterns, the proposed approach automatically searches and classifies sentences according to their importance towards an argument. Making use of this classification new patterns will be generated. These new patterns will then function as new ground truth for an iterative process. Until no new patterns are found, the process will extend the classified sentences. The sentences that are clearly classified as irrelevant are finally removed from the corpus. This way the argument corpora quality is increased highly precise while only requiring a fraction of the manual labor necessary in comparison to manually evaluating these corpora. A manual evaluation of a sample of the sentences classified as irrelevant resulted in a majority agreement of 97% on average between annotators and our approach. Based on the classified sentences we managed to improve the quality of 48 089 out of roughly 380 000 arguments in our chosen corpus.

[1] `https://webis.de/`

Contents

Chapter 1

Introduction

Argumentation is part of our every day lives and has been for the whole human existence. Studies on different aspects of argumentation reach more than 2000 years back to Aristotle[1]. Analysing the science behind argumentation, ascertaining what impact different factors are having on argument quality and exploring what argument quality even means promises great benefits. Convincing the population of different ideas and approaches plays a huge role in politics and negotiation. In modern democratic societies politicians are mostly elected based on their opinion and how well they can give reasoning. Argumentation does however not only play an important role in political scenarios. In fact, most areas of human life are influenced by argumentations. Decisions about the outcome of a legal proceeding, competing company contracts or trivial things like deciding whether tonight's dinner should be vegetarian will all be made based on the underlying reasoning. Not only are we arguing with other people, but also with ourselves on a daily basis. Taking the previous example, the decision whether we should or should not eat vegetarian will be influenced by different factors in our mind. Aspects such as the price, calories and more will play a significant role in the decision making process. In the end, the factors that hold the most important aspects in our point of view will define the outcome a decision.

Considering the impact of argumentation on our lives, it is hardly astonishing that researchers are attracted to this area for millennia now. Especially in the last decade, this interest transferred from linguistic research to the research area of computer science. As a subsection of the rapidly growing field of Natural Language Processing (NLP) different aspects of argumentation are being analysed.

[1] `https://owl.purdue.edu/owl/general_writing/academic_writing/historical_perspectives_on_argumentation/classical_argument.html` (accessed 08.08.20)

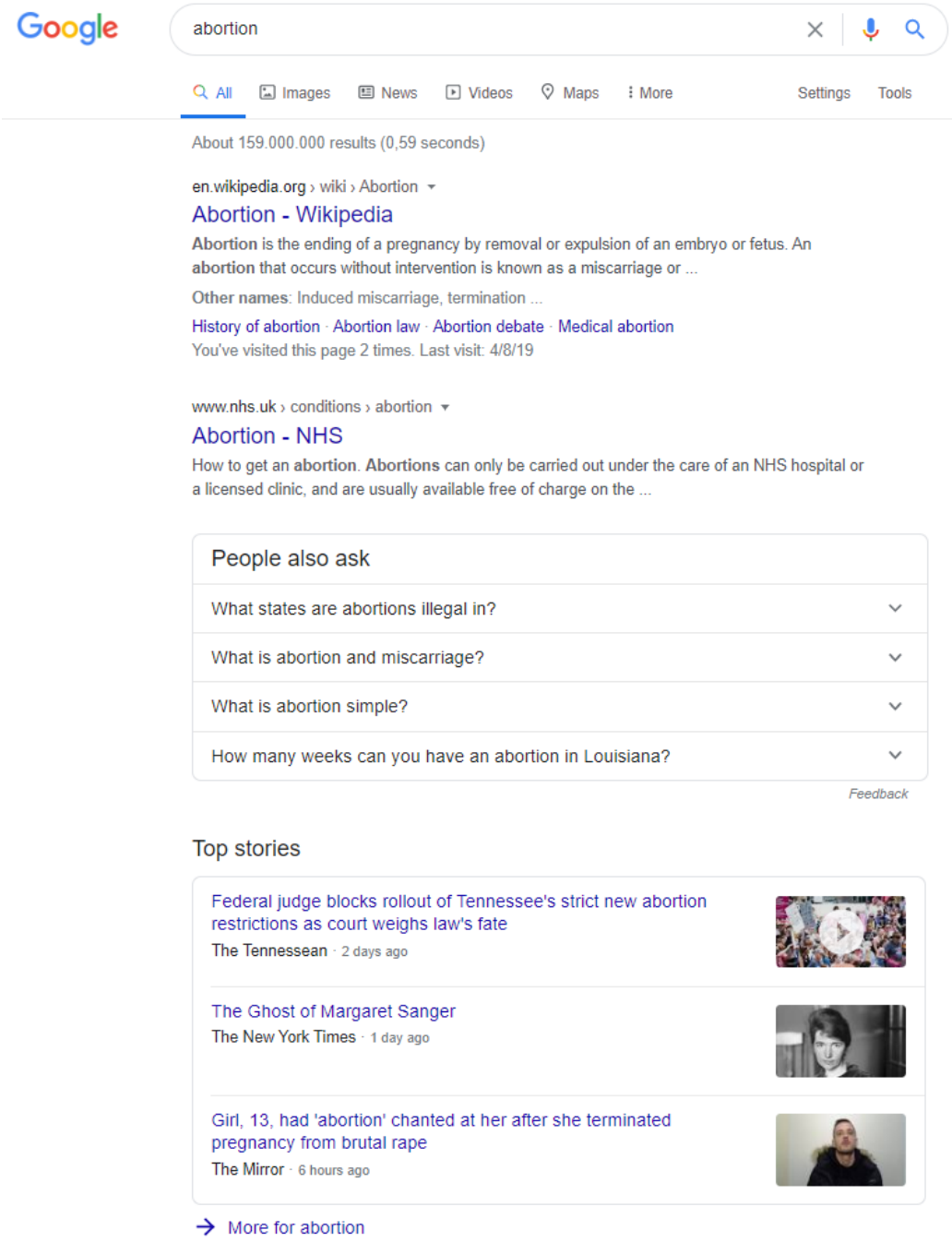

Figure 1.1: Google result page for query *abortion* (accessed 27.07.2020)

This development is also fueled by the growing impact of the World Wide Web on our every day lives. Information retrieval and knowledge sharing becomes

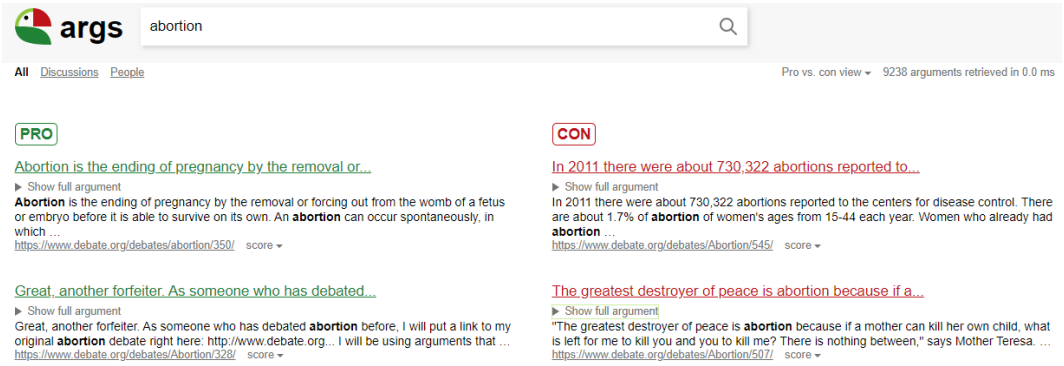

Figure 1.2: Args.me result page for query *abortion* (accessed 27.07.20)

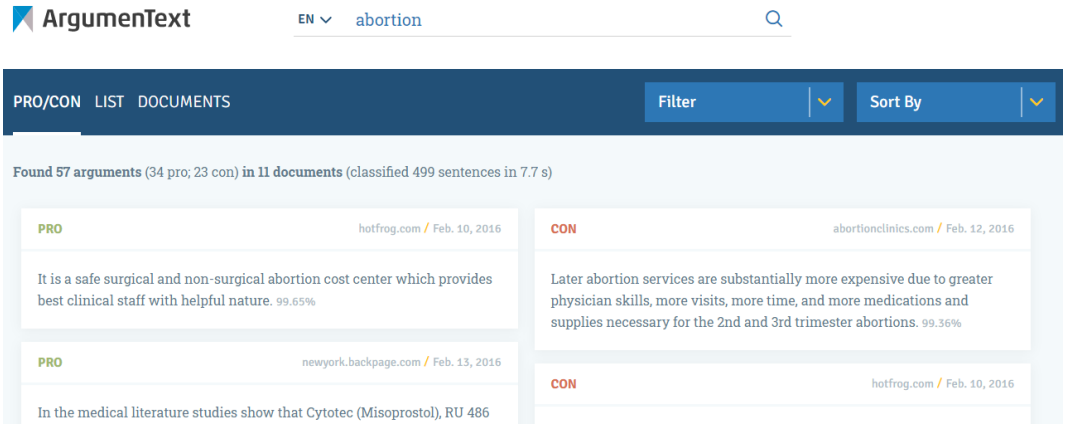

Figure 1.3: Argumentsearch result page for query *abortion* (accessed 27.07.20)

more and more accessible. Anyone can publish his or her opinion online and make their knowledge accessible to the whole web. Search engines such as Google[2] make use of this data to provide the user with resources to answer basically any question. However, while providing factual data in a well designed and easily accessible way, these factual search engines have their limitations when it comes to argumentation.
When looking for different points of view on a controversial topic, such as *abortion*, one will find the use of Google fairly limited. As shown in the Figure 1.1, Google will provide definitions, current news and info pages. While all these links might contain argumentative information, the user is required to analyse each individual entry and extract the arguments him- or herself.

Current approaches such as args.me[3] and argumentsearch[4] try to fill this gap

[2] `https://www.google.com/`
[3] `https://www.args.me/index.html`
[4] `https://www.argumentsearch.com/`

by explicitly offering the user arguments regarding the controversial topic. Figure 1.2 and Figure 1.3 show the top results returned for the query *abortion*. In contrast to Google, these search engines rely on different strategies on how to retrieve the results shown. Each search engine makes use of a different argument mining approach to collect and process data. We will take a quick look at these approaches in chapter 2.
While these search engines introduce novel concepts on how the internet can be used in an argumentative context, the giant amounts of data provided by web sources give plenty opportunity to further research as well. Never before could we analyse a large base of arguments, or data in general, in such an straightforward way. However, in order to achieve the most benefit from computational analysis well defined rules and datasets are required.
While humans can understand and detect argumentative structures and rules quiet easily, this task has proven to be non-trivial to any computer application. In order to automatically generate large argument corpora one first has to analyse different aspects of discussions and arguments. The gained insights from these datasets can then be used to teach the computer important rules to automate the creation of these datasets in return.
However, most of current automatic approaches for argument detection and extraction currently do not reach a satisfying level of performance, therefore, one has to look for alternative ways to gain qualitative argument corpora. A different approach of extracting arguments is by relying on pre-defined structures such as debate portals. These debate portals give the user the opportunity to discuss a controversial topic with a counter-party. Exploiting the structures of these portals can help extracting arguments in a straightforward way.
Relying on these structures implies new challenges. Sentences or text passages irrelevant to the topic or discussion are part of most, if not all, debate portals. These irrelevant by-products are common for online conversations in general and decrease the argument quality. While some of these text passages are related to the originating portal, such as structural or logical explanations, others are simply based on human behaviour in conversations (greetings, insults, spam, and similar). Figure 1.4 gives one example of such text passages in an argument. Inspecting the two highlighted sentences one can easily see they hold no relevant information towards the important information provided in between.
In order to improve the overall quality of these debate portal-based corpora one has to filter out these irrelevant sentences. This will also help to improve subsequent research and presentation quality based on these datasets.

Tackling the challenge of cleansing these impure argument datasets, we developed a semi-supervised approach to remove irrelevant data from such corpora.

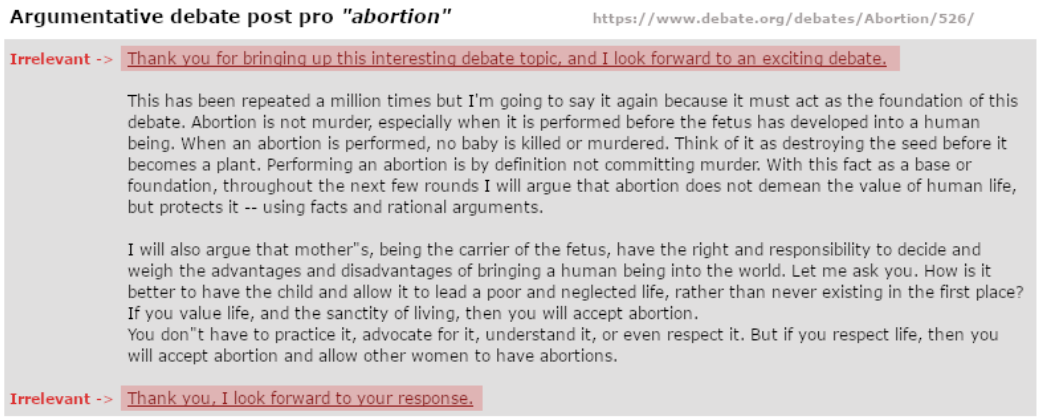

Figure 1.4: Example text taken from its original source on a debate portal. Text is also returned as a result for querying "abortion" to args.me. Highlighted text passages are considered to be irrelevant towards the authors argument.

Choosing the args.me corpus as the biggest argument dataset known to us, we first evaluated different approaches to select fitting patterns for classifying sentences correctly. Based on our investigations we selected n-grams from length two to five as suitable candidate to generate patterns from. To further improve the pattern generation process, we removed all stopwords from our potential patterns. Based on a sample of the corpus, we then generated and manually selected a set of ground truth patterns. This ground truth is then used to generate new patterns from in an automatic process. By extracting sentences containing patterns from our ground truth, we are able to search for new common patterns in these sentences. These possible new pattern candidates then have to be evaluated according to their suitability in our classification process. When a pattern is classified as a suitable candidate it is added to the ground truth patterns. After evaluating all newly found patterns, the process of extracting new patterns from the sentences matching the ground truth is then repeated based on the new ground truth defined in this round. This iterative process will be repeated until no new patterns are found. As a last step, the process of actually cleansing the arguments from all sentences classified as irrelevant has to be performed. Since all sentences matching our patterns are already extracted, a simple sentence matching step is sufficient to find and remove all sentences not relevant to the arguments.
In a concluding study we asked three argumentation experts to annotate a sample of 600 sentences classified as irrelevant by our approach. The results show that our approach identifies irrelevant sentences in 97% of the cases correctly. The classified sentences are then used to remove irrelevant sentences from around $48k$ of the total $380k$ arguments.

In chapter 2, we will discuss some current research related to the challenge first. Additionally, we will give a short introduction to the two state of the art argument search engines. Afterwards, we will introduce the corpus we used to develop and evaluate our approach. In the third chapter (chapter 3) we will explain each step of our tripartite algorithm individually. We will give a general overview of our concept and explain the separate steps of the proposed ground truth definition (seed pattern selection), the automated pattern detection process and the final corpus cleansing step. Later, we will discuss specific aspects regarding the implementation (chapter 4) of these individual steps. We will present some optimization steps and possible ways to increase performance further. Thereupon, we will describe the experimental setup of our evaluation (chapter 5) and discuss the first results generated by our pipeline. Finally, we will summarize the process and take a look into possible future improvements and optimizations (chapter 6).

Chapter 2

Background and related work

In the following we will give a short introduction into the research areas directly related to this thesis. First, we will briefly discuss argument mining in general and then move on to a possible use-case of making use of said argument mining process. By introducing two state of the art argument search engines we give a short introduction in different techniques and approaches for argument mining, processing and presentation. Based on the necessity of well defined argument corpora, we present a number of approaches aiming at generating such high quality argument corpora. Afterwards, this chapter will be concluded with some details about different machine learning approaches an how we developed a semi-supervised approach to perform the cleansing of web-based argument corpora.

2.1 Argument mining

Natural language processing represents a big part of the currently rapidly growing field of artificial intelligence (AI). Learning the rules of the human language enables the computer to fulfil tasks that could otherwise only be done by humans. Applications like grammar checker, chat bots or personal assistants like Siri[1] or Alexa[2] all are build on NLP approaches. While one could claim that all these tasks could easily be handled by human workers, no one can argue the increased efficiency and cost reduction achieved by outsourcing such processes to computer applications.
On the other hand, there exist tasks which simply could not be performed by human beings in a reasonable amount of time. As the rapid growth of the big

[1] `https://www.apple.com/siri/`
[2] `https://www.alexa.com/`

data market shows, the amount of available data increases with every moment passing by. It is of great interest for companies and scientists all over the world to analyse the respective parts of this information. While in some cases it is mandatory to use human workers to perform every single step of the analysis, this becomes more and more impractical the bigger the dataset gets. Teaching the computer to perform processing and analysis steps and validating the results by human workers greatly reduces the time and work required.
One of the uprising fields of NLP is called *argument mining (AM)*. It aims at the automatic detection and extraction of argumentative structures from text. In their paper Cabrio and Villata [2018] analyse five years of research regarding this topic. According to the authors, the field of AM gained increased interest around 2010 when approaches where made to extract argumentative structures from legal texts and scientific articles. This increased interest is supported by the authors by mentioning the *workshop on argument mining (ArgMining)*[3] taking place annually. The first workshop took place in 2014 and since then was re-hosted every year. It is known as one of the most significant events in the argument mining community ever since. Additionally, the increasing importance of argument mining as part of NLP is supported by its appearance on major AI and NLP conferences.
In the following Cabrio and Villata [2018] highlight the two main steps of every argument mining framework namely *argument extraction*, identifying arguments within a given text sample, and *relations prediction*, describing the relationship between two arguments. Furthermore, they emphasize the importance of high quality annotated corpora in order to evaluate the resulting data. The diversity of fitting corpora and data valuable to argument mining is shown afterwards. Approaches from education, such as persuasive essays and scientific articles, as well as web-based content, such as Wikipedia articles, debate portals, online product reviews, legal documents and political debates and speeches are utilized for argument mining.
Finally, the authors emphasize the great strengths of argument mining and its benefits, such as fact checking. On the other hand, some disadvantages are presented. To give an example, the authors mention that only very limited research related to argument mining is done in other languages besides English. Due to the diversity of datasets there are also slightly different definitions and proceedings in each approach making it hard to unify the datasets but also emphazising the broad variety of possibilities to apply AM.

[3] `https://argmining2020.i3s.unice.fr/`

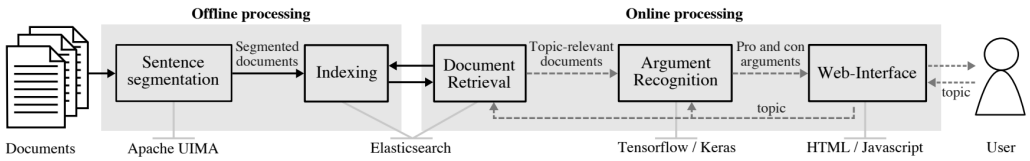

Figure 2.1: System architecture introduced by Stab et al. [2018a]

2.2 Argument search engines

Extracting arguments can be useful in many different scenarios. While companies could aim for improved products or service quality by analysing the argumentative structure in reviews, the results can also be presented for different use cases as well.
Search engines such as Google or Bing[4] do an excellent job in providing the user with factual data. When looking for reasoning, these search engines might provide the web sites matching the request, but they will not provide the reason itself. In order to close this gap a new type of search engine was developed: the so called argument search engines. An argument search engine is supposed to provide the user with arguments supporting or attacking any given controversial topic.
Rach et al. [2020] compared two state of the art search engines in their paper *Evaluation of Argument Search Approaches in the Context of Argumentative Dialogue Systems*.
First, we will take a brief look at the search engines compared by Rach et al. and later we will discuss the insights found in their publication.

ArgumenText The first search engine mentioned was introduced in *ArgumenText: Searching for arguments in heterogeneous sources* in Stab et al. [2018a] and is called ArgumenText. As the name already suggests, the main focus lies on extracting argumentative structures from heterogeneous documents. While other approaches regarding argument mining are focusing on specific types of documents or pre-annotated data, the authors are proposing a way to extract relevant sentences providing reasoning and evidence from different sources at the same time. Following this approach, the underlying corpus is based on the English documents of CommonCrawl[5], an initiative to provide everyone with large scale web crawls.

[4] `https://www.bing.com/`
[5] `http://commoncrawl.org/`

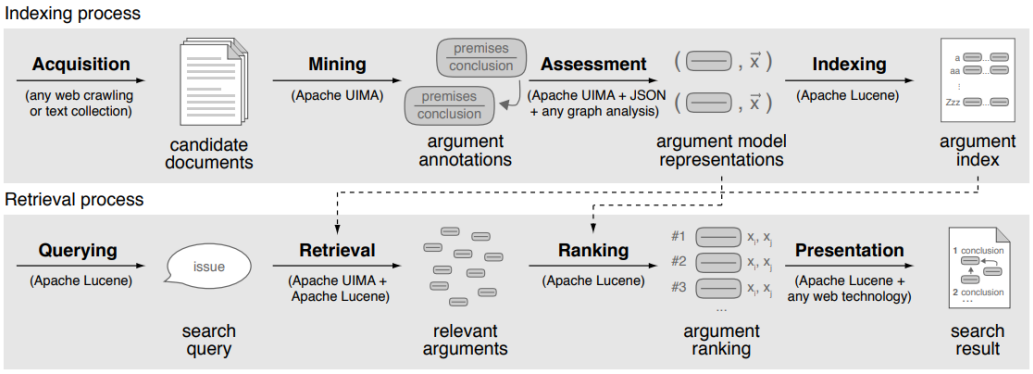

Figure 2.2: System architecture introduced by Wachsmuth et al. [2017]

They propose a concept separating the processing into two steps: First, the time consuming process of tokenization, segmentation and indexing of the roughly 683 GiB large corpus is done offline with the help of the Apache UIMA Framework[6]. And second the steps of retrieving and processing topic related documents which can be completed online afterwards. A graphical overview of the approach is provided by Stab et al. in Figure 2.1.
The argument recognition process is based on an approach introduced in a former paper by the author [Stab et al., 2018b].
Stab et al. introduce a way to classify sentences as *argument* or *no argument* in respect of a given topic. The approach makes use of an *attention-based neural network* to classify each sentence. Additionally, a BiLSTM[7] model is applied to determine the stance of given sentence to the related topic. Sorted by an average between argument extraction and stance recognition model confidence, the arguments are then presented to the user through a web interface[8].

args.me The second search engine compared by Rach et al. [2020] was introduced in 2017 by Wachsmuth et al.. While Wachsmuth et al. [2017] introduces a prototype for a search engine, the main goal of the publication is to introduce an argument search engine framework. The authors emphasizes the focus on creating an adaptable framework defining the important steps required for each argument search engine. The introduced framework is visually represented by Figure 2.2.
Due to the diversity in argument corpus annotations at that time, a common

[6] `https://uima.apache.org/`
[7] C.Olah:`http://colah.github.io/posts/2015-08-Understanding-LSTMs/` (accessed 09.08.20)
[8] `https://www.argumentsearch.com/`

argument model is introduced as well. Based on the introduced model, the argument processing flow is designed. Every step of the framework is briefly described and current challenges and research goals are highlighted. While Wachsmuth et al. [2017] introduce the model and framework, they invite the community to participate in the development of each step by providing an adaptive scaffolding which can be applied by everyone.
The lack of reliable approaches to mine arguments from web sources led the authors to create a corpus from pre-structured data. In order to generate a significant corpus five big debate portals were crawled. The resulting data was then fitted onto the argument model introduced previously, leading to the creation of the largest argument index known to that time. As a proof of concept the argument search engine prototype args.me[9] is introduced. First insights in the areas of argument mining, retrieval and presentation are provided and the importance of further research is highlighted.

The beforehand mentioned publication from Rach et al. aims for a general evaluation of argument retrieval systems regarding their suitability for dialogue system applications. In order to achieve this goal, the previously described search engines are compared to a baseline in a user study. Based on this ground truth both search engines are compared to each other.
Later in the publication the authors also mention some advantages and disadvantages of both search engines. While args.me provides some contextual information due to the nature of its retrieval process, its structure makes it more difficult to use the data directly in a dialogue system context. ArgumenText, on the other hand challenges the retrieval process from another angle, giving it a high coverage on controversial topics compared to expert-curated collections. While this approach allows the search engine to retrieve data from a broad variety of sources, it lowers the precision.
Based on the comparison on the ground truth, the importance of argument search engines is enforced. The individual strengths and weaknesses of the search engines can be traced back to their respective retrieval process.

As highlighted in multiple of the previously presented publications, the argument mining process obviously plays a critical role in the argument search process. Supplying argumentative search engines with suitable resources to extract arguments from and approaches on how to extract these resources is a critical step of research around argument mining and argument search. As discussed in previously mentioned publications, multiple ways exist on how to tackle the task of argument mining. While there exist plenty of different

[9] `https://www.args.me/index.html`

approaches and solutions, there is still a general need for research in this area as current solutions most times either provide a high level of precision and lack recall or the other way around.

2.3 Argument search engine corpus generation

The research field of argument search extends the one of argument mining and includes other areas as well. Analysing the best ways of argument ranking, evaluation and presentation as well as cross argument relations, stance related research and multiple other areas that require data to research on. While the coverage of the widest possible range of topics seems to be imperative in a practical application context, such as argument search engines, highly precise datasets are very desirable in a research aspect.
The importance and the need for well annotated large scale corpora is omnipresent in the area of argument processing. Multiple publications focus on introducing or making use of such corpora for different applications. Walker et al. [2012] introduce such a corpus created from the former debate portal *4forums*[10]. In their publication about *Argumentation Mining in User-Generated Web Discourse* Habernal and Gurevych [2017] mention debate portals as possible valuable source for argumentation mining research. Durmus and Cardie [2019] introduce a dataset soley based on *debate.org*[11] focusing on the user traits and their connection to the argumentation instead of technical aspects such as detecting and classifying arguments. Recognizing the far reach and frequent use of debate portals as basis for argument corpora and argument research highlights the importance of improving these datasets.

In *Data Acquisition for Argument Search: The args.me Corpus* Ajjour et al. [2019] present a new version of the args.me corpus introduced previously by Wachsmuth et al. [2017]. Due to its retrieval process from dedicated online debate portals, the corpus consists of a large amount of high-quality arguments. It covers a total of 387 606 arguments from 59 637 different debates. To support the initial approach of creating a collaborating environment made by Wachsmuth et al. [2017], the data is freely provided to the community. In order to evaluate the coverage of the dataset, one can inspect the inquired queries to the search engine. As one can see in Figure 2.3 (a), the top queries in the period of time from September 2017 to May 2019 are mainly focused around current controversial political, economical and ecological discussions.

[10] `http://www.4forums.com/`
[11] `https://www.debate.org/`

(a) Query	Absolute	Relative
climate change	251	3.5%
feminism	193	2.7%
abortion	158	2.2%
trump	146	2.0%
brexit	128	1.8%
death penalty	73	1.0%
google	58	0.8%
vegan	57	0.8%
nuclear energy	56	0.8%
donald trump	47	0.7%

(b) Conclusion	Absolute	Relative
Abortion	2 838	0.7%
Gay Marriage	1 558	0.4%
Rap Battle	1 396	0.4%
Death Penalty	972	0.3%
God exists	790	0.2%
Gun control	719	0.1%
Ivf debate	432	0.1%
Animal testing	372	0.1%
I will not contradict myself	357	0.1%
Euthanasia	331	0.1%

Figure 2.3: (a) Top 10 queries from args.me query log. (b) Top 10 conclusions of arguments in the args.me corpus. Each presented with absolute and relative frequencies by Ajjour et al. [2019]

As one can see in Figure 2.3 (b), many of these topics are also frequently discussed on the debate portals crawled for the args.me corpus, making it a suitable candidate for current research. Ajjour et al. present the corpus as a basis for future research in the field. Additionally, they mention multiple options for further research projects to increase user experience and improve the dataset further. As demonstrated in Figure 2.3 the coverage and size of the corpus gives it great potential for further usage. For this reason we decided to build our approach based on this corpus.

2.4 Machine learning

At the beginning of this chapter we emphasized the connection between the areas of NLP and AI and therefore machine learning (ML) as a sub field of AI. In most of the cited publications ML is mentioned in one way or another, supporting the importance of it in this research area. Since ML plays a big role in argument mining and argument search, it will be briefly introduced in the following.
We want to focus onto the four main approaches made by machine learning in general. We differ between *supervised*, *unsupervised*, *semi-supervised learning* and *reinforcement learning* models.[12]

[12]David Fumo: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861 (accessed 09.08.20)

In *supervised learning*, a classifier is fed with input data as well as the expected output data. This data is called the *training set*. The classifier learns to recognize structures in the training set leading to the expected results. Later, the learned rules will be evaluated on a *test set*. While the training set provides the algorithm with input and expected output, the test set lacks the respective output. The classifier has to predict the output based on what it learned while being trained. The resulting predictions can then be used to evaluate the classifier and later predict results in the context of the application.

In contrast in *unsupervised learning* the algorithm is only provided with input data. The classifier is then expected to find patterns and structures in the data on its own. Afterwards these insights are applied in applications such as cluster analysis.

The third area, called *semi-supervised learning*, combines the two previously mentioned fields. A classifier has to learn from a combination of annotated and not annotated data. In most cases the not annotated data is considerably bigger in quantity then the annotated section. The small sample of annotated data will be used to define new rules and self-improve the approach.

Finally, in *reinforcement learning* a software agent is meant to learn from its environment and evaluate the possible interactions with the environment in order to find the individual best behaviour in this specific environment, or at least minimize the risks taken.

Annotating and evaluating corpora often times require expertised annotators capable of performing specific tasks at a highly precise level. In order to assure quality of the annotations, multiple individuals have to evaluate the same data several times. When scaling up corpus size, this procedure becomes impractical quickly. To face the challenge of big amounts of data containing only little annotation researchers often times rely on semi-supervised approaches. This way, way more information can be made use of in contrast to relying on fully annotated data. Multiple current research projects in NLP and NLU (Natural language understanding) therefore build semi-supervised approaches. Qiu et al. [2019] for example introduce a graph based semi-supervised approach providing NLU applications with additional training data by including unlabeled datasets. In *Cross Language Text Classification by Model Translation and Semi-Supervised Learning* the authors Shi et al. [2010] are making use of a semi-supervised approach to solve the problem of missing labeled data in a given target language. These two examples only give a small impression of the

great benefits one can obtain from making use of semi-supervised approaches. Similar to previously mentioned applications our given scenario simply does not allow for a manual annotation of the whole 380k arguments of the args.me corpus and therefore we wanted to find a automatic solution capable of performing this task. In order to automate the process of corpus cleansing, we need to define some rules the algorithm can follow to automatically perform the task given. Based on the concept of semi-supervised learning to provide the classifier with partially annotated data. We want to provide our approach with a well defined ground truth of such annotated data. This ground truth can then be used to analyse the whole corpus in an iterative process. As our main goal is to remove irrelevant sentences from our corpus, the classifier needs to learn how to differentiate between irrelevant and relevant sentences. The initial task of our approach therefore lies in defining a set of annotated sentences classified as either irrelevant or relevant. By manually selecting specific patterns which represent one of the sentence types, we provide the classifier with a sample of such annotated data. These patterns and the sentences they occur in can then be used in an automated approach to find new patterns and sentences matching either side. Under perfect conditions, this way each sentence of the corpus can be classified as irrelevant or relevant based on only a small amount of manually selected data. In the following we will present the exact approach we have taken in detail.

Chapter 3

Approach

Different approaches introduced in chapter 2 present how versatile argument corpora can be. Diverse concepts result in more or less precise corpora. Automatically retrieving high quality arguments from heterogeneous sources is still subject to research. Until further breakthroughs happen, we have to rely on partially pre-annotated sources. As presented by Ajjour et al. [2019], high quality corpora suitable for further research can be created by crawling debate portals. While this approach greatly reduces the complexity of the argument retrieval process, it leads to different kinds of new challenges. One of the resulting problems lays in the nature of these debate portals. On *debate.org* for example, debates are structured in rounds. In each round, both sides have the opportunity to present arguments on the previously defined topic. This can lead to "platform specific phrases" included in the text, as shown in Figure 3.1. The first argument on *con* side does not include any argumentative information at all but defines the basic rules for the debate. In return, the discussion partner starts his argument with a topic unrelated sentence himself. While the argument presented afterwards might be of high quality and reasoning, these sentences are neither interesting nor relevant in the context of an argument search engine or other applications investigating the argument.

Inspired by this problem, we wanted to develop an algorithm capable of cleansing specific sentences from a given text passage. Applied to this scenario it should remove the sentences that do not exhibit relevant data to the topic. While our approach is based on the issue presented, our goal was to come up with a more general solution applicable in different scenarios as well.
In order to do so, we had to split the process into three steps challenging different parts of the problem. At first, we have to find a way to differentiate between *relevant* and *irrelevant* sentences in our corpus. We were searching for the most straight forward and intuitive way to classify sentences by their

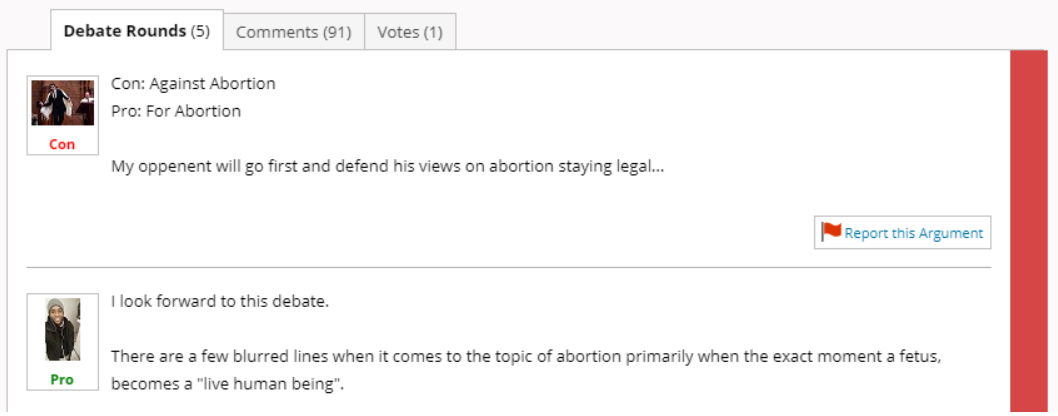

Figure 3.1: Snippet of a discussion about abortion on debate.org[1]

importance and followed the basic assumption that there might be specific keywords or phrases appearing frequently in respective sentence types. Therefore, our first goal was to recognize and extract these specific *patterns* which were classifying these sentences. We decided to name the patterns related to irrelevant sentences *negative patterns* while the opposed patterns are called *positive patterns*. In section 3.1 we present the initial step of extracting and selecting these significant patterns based on a corpus sample.
For obvious reason we were not interested in searching for all relevant patterns manually. We wanted to define a ground truth, a set of seed patterns, representing both relevant and irrelevant sentences. This ground truth then has to be utilised to initialize an automated candidate pattern extraction process. In contrast to the seed pattern selection, we did not want to evaluate the potential new patterns manually but instead use an automatic system to evaluate the newly generated pattern candidates. This evaluation process is based on validating whether sentences matching a given negative or positive pattern also contain patterns of the opposite type. The resulting patterns will then recursively function as new set of ground truth patterns and the process of searching and evaluating new patterns will be repeated until no new patterns are found. We will explain the exact procedure in section 3.2.
The cleansing of the initial corpus then follows as the last step. Since we already got all relevant information, a simple pattern matching algorithm to find the irrelevant sentences suffices to decide on which sentences can be removed. While there are some options for additional interaction here, we decided to keep it as simple and intuitive as possible. We focus on the details in section 3.3.

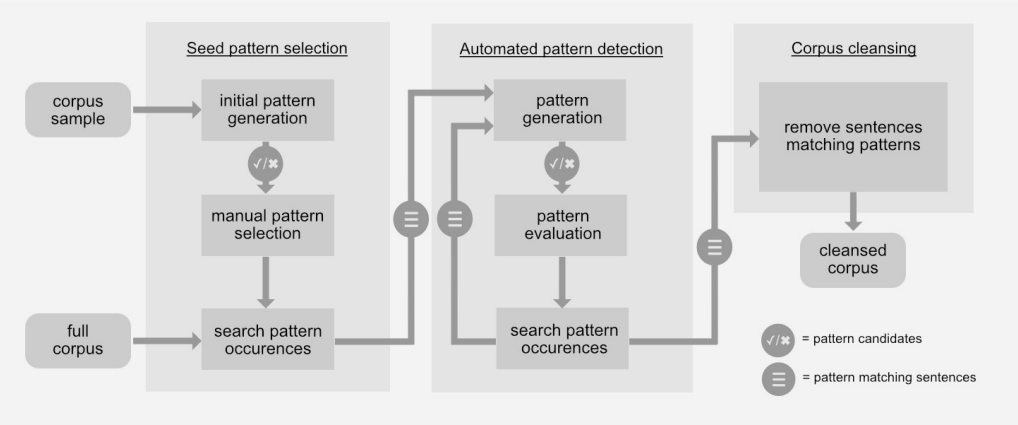

Figure 3.2: The system architecture of the introduced automated corpus cleansing approach. Separated into the three main steps: *Seed pattern selection*, where a corpus sample is used to generate a small list of patterns which will then be evaluated manually. Following these patterns will be used in the *automated pattern detection* step, where the algorithm autonomously searches and evaluates new patterns. Lastly, all sentences matching the negative patterns solely will be removed in the *corpus cleansing*.

Giving a short verbal and visual (Figure 3.2) summary of all three steps:

- **Seed pattern selection**: Initial process of manually selecting a set of patterns defined as ground truth for the automated approach to follow. Selecting a reasonable amount of patterns well suited to classify the sentences is essential to guarantee a good performance of our approach.
- **Automated pattern detection**: Automated generation and evaluation of new patterns based on the previously selected ground truth. Candidates classified as fitting patterns are then added automatically to the previous ground truth and form the basis for the next iteration. This process will be repeated until no new patterns are found.
- **Corpus cleansing**: Final step of removing all sentences classified as irrelevant in the previous steps. Additional metrics might be applied to fine tune the removal process.

3.1 Seed pattern selection

Since the basic idea is to find specific elements highlighting the relevance or irrelevance of a sentence, we had to come up with a way to detect and extract

these elements. While there might be complex approaches making use of advanced algorithms we were searching for simple and efficient solutions suitable for solving the task.
Let us assume that we are looking for a group of sentences, which contains some kind of similarity. Some possible similarities could be same sentence structures, sentence length, same words or word groups. We would expect these structures to be an indication for either relevant or irrelevant sentences. Meaning we are looking for structures mostly appearing in only one of those two sentence types. We want to define these sentence structures as our patterns.
Applied to the *args.me* corpus problem this would mean we are looking for similarities in sentences which are not relevant to our argument. Finding patterns that are contained in a large portion of these potentially irrelevant sentences would provide us with the option to automatically filter all these sentences from our corpus. On the other hand, looking for sentences which are an integral part of an argument would also provide us with valuable information. Assuming we could classify all sentences but one as relevant to the argument, this would conclude the one sentence might in return not be relevant to the argument and therefore might be removed. Taking both positive and negative candidates into account will prove useful in subsequent proceedings as well.
Following the general approach of simplicity, we chose to search for frequent words or consecutive word groups in order to find these patterns. This sequence of items, in our case words, is called *n-gram*[2] in computational linguistics. While we expected plain n-grams to be a fitting metric to generate patterns from, we wanted to compare its performance to some alternative. One of the basic concepts in *information retrieval (IR)* and text mining to determine importance of a given term in a corpus is called *term frequency-inverse document frequency (tf-idf)*[3]. Due to its characteristic to find important terms we considered it as a fitting candidate to find good patterns for our application. As tf-idf is an established algorithm in IR it also seemed as a fitting candidate to roughly measure the performance of our approach that uses only n-grams for the pattern generation.

In the following, we will explain the individual aspects of the initial pattern detection process and introduce some parameters. The exact parameters named and used in the process were fitting best in our usecase. Different values might be more fitting for different scenarios, we will however discuss these when introducing the parameters themselves.

[2] `https://www.lexico.com/definition/n-gram` (accessed 09.08.20)
[3] `http://www.tfidf.com/`

3.1.1 n-grams

While n-grams can by definition consist of subsequent characters or syllables, it is most common to define a *1-gram (unigram)* as a single word. Consequently a *2-gram (bigram)* describes two successive words and so on. According to this definition we will assume that a *n-gram* is a pattern of n subsequent words.
Length n=1 as the smallest possible pattern length results in a diverse quantity of pattern candidates. Naturally this leads to the largest amount of patterns one can generate from a text following this approach. Additionally, the pattern frequency on 1-gram long patterns in a corpus can in general be expected to be the highest. Since we are trying to find frequent patterns in irrelevant sentences, high frequency is exactly what we are looking for. On the other hand, patterns consisting of just one word do not reflect any sentence specific formulations or structures but only single words. Extracting 1-grams therefore can be used to search for frequent words that might appear more often in irrelevant sentences. Given the structure of some debate portals, keywords like "round" could be frequent in irrelevant sentences mainly.
In contrast, increasing the pattern length by 1 or more words will give more insight about context. While the 1-gram pattern frequency will segment into multiple smaller n-gram constellations and therefore each individual patterns appearance will be reduced, these patterns will hold more contextual information and classifying the pattern becomes more and more easy, even without knowing the sentence the pattern was extracted from.
Assuming we selected "round" as a possible pattern candidate, longer patterns could be "first round", "round two", "round table". The first two examples could be related to the previously mentioned debate portal specific structure, the latter on the other hand does not seem to be related to this structure. While this does not necessarily make the "round table" pattern a better or worse candidate, we can clearly see the advantage a 2-gram pattern has over a 1-gram pattern.
However, increasing the length of a pattern too much can result in another problem. While 1-grams might be too general to be used as pattern candidates, long patterns will at some point get too specific. Choosing patterns of increased length will result in less re-appearances in our corpus. Picture defining the pattern length equal to the sentence length. We would be able to find all sentences identical to the one we chose as a pattern, but in order to find all irrelevant sentences, we had to look for each individual sentence, define it as a pattern, potentially find all its appearances and restart the process for the next sentence. This process would be similar to a manual annotation of the whole corpus.

Choosing the right pattern length is an important step to prevent the results from being to general on the one side, or *overfitting* on the other. In our approach, we chose to generate n-grams of length two to five in order to filter out too general words and prevent overfitting at the same time. In order to generate our initial pattern set, we took a sample of our corpus and extracted the most frequent patterns from it for each different pattern length. To keep the sample statistically relevant, we chose a sample size of 10% of the original corpus size.

3.1.2 tf-idf

Term frequency-inverse document frequency is a common metric used in NLP. It is used to calculate how well a specific term represents a document in a corpus. Assuming one argument in our corpus to be one document, tf-idf will return the specific term best presenting this argument (document). We believed most, if not all, of these high ranked sentences represent important information in our argument. This approach would therefore provide us with great help in finding the positive pattern candidates and in reverse possibly negative candidates as well.
Following this idea, we had to specify the terms we wanted the algorithm to rank. Due to our insights regarding the n-gram pattern generation, we decided to define terms identical to how we defined the n-grams. We took the same sample generated for the previous n-gram extraction step and again extracted n-grams in length two to five. Afterwards we ranked these n-grams using the tf-idf algorithm.

3.1.3 Stopword filtering

Stopwords play different roles in NLP scenarios. The basic idea of stopword removal is to take out the most common words used in the language the filter is applied to. In english, common examples are "the", "of", "to", etc. These words are expected to contain little to no information. Removing them should in most cases not alter the context and logical structure of the information contained. Given the following sentence:

- Steve Jobs, Steve Wozniak and Ron Wayne are the founders of Apple Inc.

All the important information in the above sentences is contained in the words *Steve Jobs, Steve Wozniak, Ron Wayne, founders* and *Apple Inc.* If we now remove all the stopwords from it, the sentence will go as follows:

- Steve Jobs, Steve Wozniak Ron Wayne founders Apple Inc.

As we can see, the relevant information provided in the first sentence is not lost after removing the stopwords it contains. Since we are trying to find structures presenting specific types of sentences, we want to focus on the most significant parts of a sentence. This leads to the question whether stopwords are important to our scenario at all.
On the other hand, there are different cases where stopword removal could alter the sentence meaning. Taking the above example and changing it slightly, we can see the contextual information provided by stopwords in some cases:

- Steve Jobs is a founder of Apple Inc.
- Steve Jobs founder Apple Inc.

The second sentence leads to the conclusion that Steave Jobs is the sole founder of Apple Inc. The first sentence however states that he is *a* founder, suggesting there are several.
Additionally, negations are often times removed by stopword filtering as well. Similar to the previous example, this has significant impact on the sentence meaning.

- Elon Musk is not a founder of Apple Inc.
- Elon Musk founder Apple Inc.

Given the previous examples one has to analyze the importance of stopwords in the context of the application. In our specific scenario we decided to compare both the results of the approach with and without filtering out all stopwords in the pattern generation process and evaluate the results.

3.1.4 Final pattern selection

```
# 1. SEED PATTERN SELECTION
# 1.1. INITIAL PATTERN GENERATION
# Parameters:   sample = (corpus_size / 10),
#               n_gram_length = 1,...,5,
#               top_100 = 100
get sample from Corpus
for each of the four n-gram extraction approaches:
    for each n_gram_length:
        get top_100 most frequent n-grams in sample

# 1.2. MANUAL PATTERN SELECTION (GROUND TRUTH DEFINITION)
for each of the four approaches:
    get all n-grams that expert A considers clearly NEGATIVE
    keep only those n-grams that expert B also considers
   clearly NEGATIVE
```

```
15
16 determine most promising approach:
17     create one pattern from each negative n-gram of best-judged
       approach
18
19 for each of the four approaches:
20     get all n-grams that expert A considers clearly POSITIVE
21     keep only those n-grams that expert B also considers
      clearly POSITIVE
22
23 determine most promising approach:
24     create one pattern from each positive n-gram of best-judged
       approach
```

Listing 3.1: Simplified pseudocode representation of the initial pattern generation

	***n*-gram w/ Stopwords**		***n*-gram w/o Stopwords**	
	raw	tf-idf	raw	tf-idf
1-gram				
2-gram				
3-gram		m most frequent patterns		
4-gram				
5-gram				

Table 3.1: Strongly simplified visualization of all initially generated pattern lists. For each combination of n-gram and approach type a list of the most frequent occurring patterns in the 10% sample was generated. In our specific use-case we chose the top 100 most frequent patterns to be analysed ($m = 100$).

According to the previously described processes, we selected a sample of our corpus. In order to produce significant results, we chose the sample to be exactly 10% of the corpus size. Based on this sample, we generated a list of the 100 most frequent patterns for the n-gram length one to five. We repeated the process applying the tf-idf algorithm. Both steps were completed once with and once without filtering the stopwords beforehand. This generated a total of 2000 patterns spread over the different approaches. A simplified visual overview is provided in Table 3.1.

As described previously, we expected the 1-grams to not contain enough contextual information to be used as pattern candidates. During our manual analysis we found this theory to be correct, especially in regard to the negative pattern candidates. Since most 1-gram candidates were included in the longer patterns anyways, we decided to drop the 1-gram lists for all negative

		Counts w/ Stopwords		Counts w/o Stopwords	
Type	**n-gram**	**Pattern**	**Count**	**Pattern**	**Count**
Positive	1-gram	people	36286	people	36286
		god	30367	god	30367
	2-gram	the world	6210	united states	3906
		the bible	5918	death penalty	1868
	3-gram	the fact that	4206	big bang theory	251
		the united states	3659	united states america	218
	4-gram	in the united states	977	life liberty pursuit happiness	102
		of the united states	698	shall surely put death	82
	5-gram	has nothing to do with	494	make law respecting establishment religion	69
		like to point out that	384	shall make law respecting establishment	63
Negative	1-gram	opponent	29088	opponent	29088
		—	—	—	—
	2-gram	my opponent	27149	thank opponent	1494
		this debate	9567	first round	1491
	3-gram	my opponent s	3983	first round acceptance	617
		my opponent has	3454	like thank opponent	509
	4-gram	thank my opponent for	1251	would like thank opponent	359
		i thank my opponent	746	thank opponent accepting debate	128
	5-gram	i thank my opponent for	682	round 1 acceptance round 2	82
		i would like to thank	665	would like thank opponent accepting	59

		TF-IDF w/ Stopwords		TF-IDF w/o Stopwords	
Type	**n-gram**	**Pattern**	**TF-IDF**	**Pattern**	**TF-IDF**
Positive	1-gram	heterosexual	1.0	abortions	1.0
		poetry	1.0	designates	1.0
	2-gram	and weaponry	0.555	americans like	1.0
		is religion	0.5	god make	1.0
	3-gram	an infinite regression	1.0	able kill others	1.0
		in recent years	1.0	abortion adopted kids	1.0
	4-gram	abusive education and domestic	1.0	accidentally killing equivalent purposely	1.0
		across the globe power	1.0	according british home office	1.0
	5-gram	acceptance of metapyhsical space that	1.0	able disprove evolution instead creationists	1.0
		acttions lean more to pain	1.0	abolish sentences thing life sentences	1.0
Negative	1-gram	—	—	—	—
		—	—	—	—
	2-gram	actually forfeiting	1.0	—	—
		arguement right	1.0	—	—
	3-gram	—	—	—	—
	4-gram	—	—	—	—
	5-gram	—	—	—	—

Table 3.2: Sample of the two best fitting positive and negative n-gram seed patterns for each of the four initially considered pattern types, with their *count* or *TF-IDF* extracted from the mentioned 10% sample of the corpus.

pattern candidates. A small set of positive 1-gram candidates seemed to be independently relevant however, therefore we chose to keep this set. The automatic processing step introduced in the next section (section 3.2) will however not make use of any 1-grams due to the fact that these 1-grams can mostly not be clearly interpreted as positive or negative.

Type	n-grams	Seed Patterns
Positive	1-grams	(government, 94198), (states, 85388), (state, 68123), (law, 59609), (society, 58695), (money, 54314), (death, 52327), (universe, 49412)
	2-grams	(big bang, 9370),(minimum wage, 8650), (human rights, 8592), (god exists, 7959), (health care, 7537), (years ago, 7165), (global warming, 6399), (high school, 6235), (opponent claims, 6160), (believe god, 6151), (human beings, 6136), (video games, 6051), (god exist, 5810), (existence god, 5636), (jesus christ, 5592), (supreme court, 5513), (new york, 4890), (human life, 4838), (old testament, 4640), (years old, 4632), (god created, 4621), (god would, 4495), (self defense, 4089), (merriam webster, 2015)
	3-grams	(new york times, 1071), (world war ii, 1062)
	4-grams	(life liberty pursuit happiness, 791), (shall surely put death, 192), (law respecting establishments religion, 370), (make law respecting establishment, 352)
	5-grams	—
Negative	2-grams	(first round, 10113), (thank opponent, 10018), (vote con, 6048), (round acceptance, 4698), (vote pro, 4585), (new arguments, 4056), (accepting debate, 4040), (accept debate, 3432), (kfc kfc, 15), (thinking bee, 3), (wonyou wonyou, 1), (ham ham, 1)
	3-grams	(debate good luck, 335), (debate look forward, 863), (hi hi hi, 1), (dan small penis, 1)
	4-grams	—
	5-grams	(every one wrong every one, 2)

Table 3.3: The full list of positive and negative seed patterns used for each n-gram type, along with the number of sentences they match in the whole corpus, ordered by number of matches.

The results generated by applying the tf-idf algorithm seemed to return great candidates for the positive list especially. Given the concept of tf-idf however, the manual analysis showed the results mostly did not include candidates for the negative pattern list (as shown in Table 3.2). Since we are mainly looking for negative patterns, these results were not ideal. Additionally, the algorithms approach of finding the most significant term in a single argument in relation to the other arguments inevitably results in a pattern badly suitable for finding common patterns especially in irrelevant sentences. Following this conclusion we decided to drop the results provided by tf-idf entirely as well.

This left us with patterns of n-gram length two to five once with and once without stopwords. We analysed these lists manually and selected possible candidates for positive and negative seed patterns based on our individual judgement. Afterwards, we compared the list of possible candidates containing stopwords to the list containing no stopwords and selected the one giving

us the bigger ground truth to work with. While both approaches returned nearly the same amount of pattern candidates, the ones generated with stopwords removed returned the most fitting candidates and was therefore chosen. Since we wanted to apply the most promising approach to our automated pattern extraction process as well, we also chose the stopword removed approach based on the over all results generated. While both approaches gave us a similar amount of seed pattern candidates, the one containing stopwords returned many candidates occurring at a high frequency but also holding close to no information regarding the sentence stance. Patterns such as *(there is no, 5400)*, *(the fact that, 4206)* and *(would like to, 3315)* do not indicate their stance clearly. Taking the latter example, the fragment could originate from following sentences:

- I would like to highlight the importance of ...
- I would like to thank my opponent for ...

As indicated by the under-linings, equivalent 3- or 4-grams from the stopword filtered approach would contain much more information regarding the sentence type. The final candidates chosen as our ground truth patterns can be inspected in Table 3.3.

The whole process is visualized in a simplified form in Figure 3.2 and the individual steps can also be traced in the *pseudocode (Listing 3.1)* at the beginning of the section.

3.2 Automated pattern detection

```
# 2. AUTOMATED PATTERN GENERATION
# Parameters:   k_negative = 20
#               k_positive = 20
# 2.1. BALANCE PARAMETERS (UNIQUE SUBSTEP)
determine sentence frequency ratio between negative and positive sentences
    adjust k_negative and k_positive accordingly
determine sentence length ratio between negative and positive sentences
    adjust k_negative and k_positive accordingly

# 2.2. PATTERN GENERATION
for each matched-sentence containing only negative pattern matches:
    generate n-grams
```

```
return all n-grams >= k_negative as new negative pattern candidates

for each matched-sentence containing only positive pattern matches:
    generate n-grams
return all n-grams >= k_positive as new positive pattern candidates

# 2.3. PATTERN EVALUATION
# Parameters:   p = 0.95 / 0.99
# Precision calculation:
# Sentences-ONLY-negative/Sentences_pattern_appears_total

for each negative pattern candidate:
    for each sentence matching the pattern:
        count appearance of positive pattern in negative matching sentence
    calculate precision for given pattern

for each positive pattern candidate:
    for each sentence matching the pattern:
        count appearance of negative pattern in positive matching sentence
    calculate precision for given pattern

for all patterns:
    if pattern precision <= p and not pattern is in ground truth:
        remove pattern
    else:
        add pattern candidate to regarding pattern list

# 2.4. LOOP STEP
# Parameters:   r = X
for r rounds repeat:
    if positive and negative list equals r-1 positive and negative pattern list:
        end loop
    else if positive and negative pattern list enter "loop state" repeating itself after Y rounds:
        end loop
    else
        generate new pattern by repeating Step 2.2., 2.3., 2.4.
```

Listing 3.2: Simplified pseudocode representation of the automated pattern generation

While we had to manually select fitting candidates in our initial step, we wanted to find a way to automatically extract new patterns based on the ones

we defined as ground truth. Therefore, we designed the algorithm to repeat a cycle of pattern generation and evaluation steps. One cycle of generating new pattern candidates and evaluating them will be called *round*. Each round, the previously found patterns are used as ground truth for the following round. Based on this ground truth new patterns will be extracted and evaluated. This self repeating process will be continued until no new patterns are found in one round. At this point all resulting patterns will be used to detect and remove all sentences matching these patterns. We call this process *corpus cleansing*. In the following we will explain the two steps of each round.

3.2.1 Pattern generation

First, we have to generate new patterns based on the ones that were selected as ground truth. In order to do so, all sentences containing at least one of our patterns are searched and extracted. In this step we search for matching sentences for both negative and positive patterns. The resulting list of sentences is then separated into sentences containing only negative patterns, and sentences containing only positive patterns. Sentences that include patterns of both sides are not included in the process of generating new patterns but will be used to calculate the pattern precision later. Since these lists only contain sentences solely related to their seed patterns orientation, as negative or positive, they provide the optimal ground to search for similarities regarding the specific stance.
Based on the insights won in the initial pattern generation step, we repeat the approach introduced in the previous section. The seed sentences first have to be stopword filtered and afterwards patterns of length two to five have to be generated from each seed pattern sentence. The resulting patterns are then counted and each pattern that occurs over a pre-defined frequency is selected as new pattern candidate.
In the initial pattern extraction step, we searched the top 100 most frequent patterns of our sample. Since this sample was one tenth of the size of our original corpus and the lowest frequency a pattern of our ground truth occurred was just above 20 appearances, we defined the lower border frequency to be $20 * 10$ fitting to our initial pattern extraction step. Even though we could have simply chosen the lower border of 20 matching the ground truth creation process, we decided to increase the minimum required frequency in benefit of performance and run time by a suitable factor.
Since the whole process has to be concluded for the negative as well as for the positive side independently, it was easy to detect a certain imbalance between the newly generated patterns quickly. Defining the border for both sides to be set at 200 led to extreme discrepancy between those. In initial tests the

	Total Sentences	Irrelevant Sentences (IS)	IS A1 ∧ IS A2	IS A1 ∨ IS A2	Cohen Kappa Similarity
Annotator 1 (A1)	1294	147 (11,3%)	111	175	0.748
Annotator 2 (A2)		139 (10,7%)			

Table 3.4: Manual sentence classification of the top 10 arguments derived for the top 10 queries issued to args.me by Annotator 1 (Jun.-Prof. Dr. Henning Wachsmuth) and Annotator 2 (myself).

amount of newly found positive patterns would be 10-50 times as high as the ones found for the negative side. This imbalance comes from the very definition of the task itself. While we are analysing some text, we only want to find very specific irrelevant sentences. We assume that most part of the text is important however. We therefore have to expect a much higher frequency in positive patterns than in negative ones. This sentence frequency relation has to be considered in the new pattern generation.
In order to analyse the frequency differences Professor Wachsmuth and I analysed the top 10 arguments derived for the top 10 queries issued to args.me (see Figure 2.3[4]). We both individually marked each sentence of each argument as either relevant or irrelevant in order to calculate the ratio between both. In total we annotated 1294 sentences each. Professor Wachsmuth marked 11.3% of the sentences as irrelevant, while I myself marked 10.7% (Figure 3.4). Following, we calculated the Cohen's Kappa coefficient[5] and achieved substantial agreement with a value of 0.748. This calculation showed us that relevant sentences are roughly 10 times more frequent then irrelevant ones. Applying this ratio to our frequency border, we redefined the minimum frequency for positive patterns to 10 times the initial value, resulting in a minimum of 2000 appearances and kept the original value of 200 for the negative patterns. We therefore had restored the balance between positive and negative sentences. Testing the adjusted values showed great improvement of the initial results.

3.2.2 Pattern evaluation

After searching and selecting new possible pattern candidates, we had to evaluate them somehow. At this point, the importance of extracting both side patterns comes into play. While we do not want to manually evaluate our patterns each round, we have to guarantee the algorithm is doing what it is supposed to do. In order to do so we decided to evaluate the performance of the patterns against the opposite side of patterns. For this reason we perform a precision analysis. As mentioned previously, one sentence can contain nega-

[4][Ajjour et al., 2019]
[5]K. Pykes: `https://towardsdatascience.com/cohens-kappa-9786ceceab58` (accessed 09.08.20)

tive patterns as well as positive patterns. Therefore, each individual sentence containing a negative/positive pattern occurrence is counted and compared to the total sentences including the pattern as well as opposite patterns. Putting these values into the standard precision formula, we come up with the following:

$$\text{Pattern Precision} = \frac{\text{Sentences containing only same-side pattern matches}}{\text{Total sentences containing pattern}}$$

The resulting pattern precision is expected to be at least 95% in order to be a relevant positive or negative pattern candidate. With other words, out of 100 sentences including a negative/positive pattern, only five of these sentences can include patterns of the opposite side at the same time.

We perform this evaluation twice per round. At the beginning of each round we generate new patterns based on the sentences we already extracted. These patterns are then evaluated the first time. For this evaluation step, we compare the newly found patterns of each side against this rounds ground truth. The ground truth consists of the initial patterns defined as ground truth and all patterns approved from previous rounds. All occurrences of our pattern candidates are searched in the sentences matching previous patterns. We do not search the whole corpus for every pattern occurrence in this step yet. As described above, we then calculate the precision of each individual newly found pattern based on the occurrence of opposite side ground truth pattern appearances in the already classified sentences. If the pattern achieves a precision of at least 95%, it will be kept. Otherwise it will be removed. Additionally, if one pattern is found as positive and negative candidate at the same time, it is removed from both lists. Since it is flagged as possible candidate for both sides, we consider it to be in fact not suitable due to its double flag.
In the second step, for each newly found pattern, all sentence appearances in the whole corpus are searched and added. After completing the search process for each new pattern, a second round of evaluation has to take place. While we previously compared the new patterns against the opposite ground truth patterns, we now compare them against the newly added patterns of the opposing side as well. Similar to the first evaluation we calculate the precision and filter out each candidate not reaching the threshold.
The resulting patterns will then be declared as ground truth for the next round. Before restarting the process in the next round, the sentence list on which the new patterns are based on is cleaned from all entries not connected to any of the remaining patterns in at least one of the two sides.

3.3 Corpus cleansing

```
# 3. CLEANSE CORPUS
for pattern in negative pattern list:
    for sentence matching pattern:
        if sentence does not contain positive pattern:
            store sentence

for argument in corpus:
#   3.1.
    if argument starts/ends with stored sentence:
        remove stored sentence from argument
    repeat 3.1. with shortened version of argument until non
   removed
```

Listing 3.3: Simplified pseudocode representation of the corpus cleansing

The last step of our approach comes down to a simple corpus cleansing process. Based on the previously defined pattern extraction process, we already extracted all sentences including our patterns. To guarantee maximum precision in our cleansing process, we selected all of these sentences including **only** negative patterns and no positive patterns.
Additionally, we assume that most of the irrelevant information is at the beginning or end of an argument. Following this assumption, we first evaluate the outer sentences of an argument. Only if an outer sentence is flagged as irrelevant, it will be removed and the process will be repeated recursively with the next "outer" sentence following. This process will be repeated for each individual sentence as long as sentences are removed. If no more sentences are found, the next argument will be analysed.
However, we did not analyse whether one argument in the annotated corpus corresponds to one actual argument. Assuming the annotated argument contains multiple actual arguments, one could choose different rules on how to remove and keep sentences. While this requires further analysis, the detection and removal of irrelevant sentences in the middle of an argument could even help detecting whether multiple arguments are contained in an annotated text span.

This process is highly focused on the context of argument corpora however. While we believe it to be a solid approach for our specific scenario, we want to highlight the ease on how to adapt this process to different applications. Due to the previously taken steps in the automated process, we already extracted the exact sentences fitting our scenario. Choosing different cleansing approaches comes down to simply removing these sentences from the corpus while applying the additional metrics one prefers. Adding these additional metrics, such

as only removing at the beginning or end of a text passage, might be useful in some cases, hindering in others.

Chapter 4

Implementation

In this chapter we want to discuss some aspects of the practical implementation of our algorithm. We will mention the tools and libraries used for different parts of our approach. While our algorithm seems to be rather simple in theory, different challenges emerged during the process of this thesis. In order to reduce possible error sources and focus on the concept of our algorithm, we make use of well known and established libraries.
As for the technical environment, we developed our approach on a machine running the Linux distribution *Ubuntu 18.04 LTS*, using *16GB RAM* and a *Intel Core i7-7500U GPU @ 2.7GHz*. Besides the fairly limited resources of the hardware, we expect an increasing growth of dataset size in the context of our application. Therefore, the algorithm should make use of minimal resources and be able to perform the task at the highest possible efficiency. The limited hardware resources enforced the optimization process from the beginning. However, scaling up the computational power will reduce the run time and might give alternative options when coming up with different ways to perform the individual steps of our approach.
In the following section (section 4.1) we will first discuss the practical realisation of our approach, we will mention different tools and libraries used to perform individual steps. We will then give some short instructions on how to actually run the algorithm (section 4.2). Afterwards we will describe a potential way to integrate the algorithm into the args.me pipeline to give one short practical example on how to apply the pipeline introduced in our approach (section 4.3).

4.1 Architecture and realisation

First of all, we want to mention that the whole pipeline is implemented in Python Version 3.6.9. We chose the language for different reasons, but mainly

its wide use in the context of text analysis and machine learning, as well as for personal comfort. It should however be possible to translate it into any other programming language to fit the application as long as comparable algorithms can be used or implemented.
Following, we will discuss different challenges we were facing in the development process. We will orient ourselves on the general flow of the approach and focus on relevant decisions and specific implementations.

4.1.1 Preprocessing

When working with text, especially when the text is created by individuals more or less focused on grammar rules, many processing challenges emerge. Processing such texts often times requires multiple pre-processing steps in order to minimize errors in the following steps. Nevertheless, it is much desirable to reduce these errors as much as possible. Regardless the source of the data, the focus of our approach lies on detecting irrelevant sentences in a text passage regarding an argument. Independent from each sentences' quality, the process of correctly detecting sentence boundaries is crucial to our approach. These boundaries can than be used to split longer text passages into smaller segments. This process is called tokenization. Additionally, this does not only apply to sentence splitting, but also splitting text into passages, phrases, words, or syllables and sometimes classifying these segments along the way.

In our specific scenario, tokenization is used for segmenting the arguments into sentences and subsequently creating phrases (*patterns*) from these sentences. Python provides us with a tool capable of exactly this task. Alongside many different applicable areas in the field of NLP, the *Natural Language Toolkit*[1] *(NLTK)* makes these exact functions available to us. In particular our tokenization process is based on the tokenizer modul *(nltk.tokenize)*[2] making use of the *word_tokenize()* and *sent_tokenize()* function. Additionally, we use the *stopwords* defined in *nltk.corpus* in our stopword filtered approach.
In the early testing phase, we compared the NLTK Tokenizer to the Tokenizer introduced by *SpaCy*[3]. In a small manual analysis we decided to continue working with NLTK instead of SpaCy. However, further analysing the individual advantages and disadvantages and using one over the other in different use cases might proof beneficial. While NLTK is performing well in our scenario, one might take a look at alternatives, depending on the approach and

[1] https://www.nltk.org/
[2] https://www.nltk.org/api/nltk.tokenize.html
[3] https://spacy.io/api/tokenizer

used programming languages.
In order to improve the tokenizers performance, we performed some small pre-processing steps. First, the tokenizer has proven to struggle with URLs. Since we assume URLs as being very much relevant to an argument due to their nature of being mainly used as sources, but not relevant towards our automated approach, we decided to replace them with a string easier process-able by the tokenizer. Later they are added back to the text.
Secondly, we made sure that characters marking a sentence end, such as ".", "!", "?" are followed by a space. Otherwise they would not be detected as individual sentences by the tokenizer.
Lastly, we decided to filter out non relevant characters for our pattern generation process. We noticed different reoccurring patterns which mainly consisted of numbers and special characters, such as brackets and "lesser and greater then" ("<", ">"). These patterns mostly represented undesired sentence structures and kept little to no relevance considering the pattern generation process. In order to not falsify our approach, we decided to remove all these characters and patterns and replace them with white spaces. This would allow us, to keep all patterns consisting of alphabetic characters and at the same time we could remove surplus spaces if necessary.
The resulting tokenized and filtered strings are then stored alongside the original text in the argument. This allows us to perform all pattern creation and evaluation processes without regenerating the pre-processed text in every step. Performing these partially expensive optimization steps in the pre-processing in return grants great benefit in the following steps. Due to the work done in this step, we do not have to do any advanced text processing later on and can rely on the work done in the pre-processing, reducing processing time and complexity in later steps.

4.1.2 Cleansing

While we do rely on the stored sentences from pre-processing in the automated section of the approach, we will not use them directly in the cleansing processes itself. The main focus in the cleansing step lies on removing the sentences detected as irrelevant while altering the argument as little as possible. All text not directly targeted by the sentence removal should stay as close to the original as possible.
Filtering out special characters and stopwords came in handy when performing the automated pattern detection process, however, it alters the original text quiet heavily. Therefore, instead of performing computational complex steps in order to refactor the original argument from the tokenized text, we chose to

perform the cleansing step based on the original data itself by repeating some of the tokenization steps.
In the initial pre-processing step we made sure to maintain the same amount of sentences compared to the original text, despite processing the text intensely in some cases. This allowed us to only apply the previously mentioned improvement step of adding spaces behind regarding punctuation in case of being missing, in order to improve sentence ending detection. After searching pattern matches in the pre-processed data, we could then simply compare sentence indices and remove the sentence from the original data based on the index of the pre-processed match.

4.1.3 Performance Optimization

Working on large corpora can require enormous amounts of resources very quickly. Especially when one is expecting an increasing amount in data sizes, the development of an approach should partially focus on handling data efficiently. The difficulties coming with huge quantities of data brought multiple challenges to our approach. Redundancies for instance often times are part of text processing. As mentioned in the previous section, we basically store a processed version of the original text alongside the latter. In terms of space consumption this is a drawback and should be avoided when possible. However, performing the pre-processing step every time we required the original text in the processed format would be way to time consuming compared to just storing it. If dataset size grows rapidly however, it might be required to sacrifice some time in benefit of storage capacities.
A second challenge we were facing due to our approach was the search complexity. Searching our patterns in the whole corpus takes up a great chunk of time. For each individual pattern we have to search every occurrence in all sentences and store the sentences matching. To reduce overhead, we do not want to store a sentence multiple times when different patterns occur in a single sentence, therefore, we have to search the stored sentences for an existing entry for this exact sentence. Same comes for generating new patterns. Each pattern has to be checked on previous existence first, before adding it to our pattern list. We therefore chose to store as much information as possible in Pythons dictionaries. These are implemented based on a Hash Table in Python and therefore only require an access time of $\mathcal{O}(1)$. This helped us speed up the lookup of existing sentences and patterns.
Additionally, we tried to speed up the search process of pattern occurrences with the help of *Suffix Trees*[4]. Suffix Trees basically allow to search for pat-

[4]https://www.geeksforgeeks.org/pattern-searching-using-suffix-tree/ (accessed 30.07.20)

terns in a text in an extremely fast manner. Explicitly speaking, traditional search time takes around $\mathcal{O}(n)$ time where n is the length of the search text. Suffix Trees speed up the search time to the Pattern length m ($\mathcal{O}(m)$). However, creating Suffix trees is extremely space intensive and it is generally said to take the quadratic space the original string requires. Given this fact, we dropped the idea of optimizing the search with help of a Suffix Tree for now. One could however try to find efficient solutions tackling this challenge in the future.
As a last optimization step we decided to parallelize the most time consuming steps, namely the initial preprocessing and the pattern search process. We made use of the multiprocessing module[5] of Python introduced in version 2.6. While we did not perform explicit tests measuring the performance increase, we achieved an estimated speed-up of approximately 130% - 150% on the testing machine. Obviously this factor is highly dependent on the hardware provided, but the current implementation is already capable of efficiently using hardware resources to speed up the whole process.

4.2 Running the approach

In the current version of the application all important steps are regulated in one controlling *main.py* file. This file can easily be executed by calling it from the terminal. In contrast to the approaches architecture, this file splits into three slightly altered steps. The first step is focussing on the preprocessing of the corpus. The second one handles the initial pattern generation process as well as the automated approach. The third and last step then concludes the process by performing the corpus cleansing. Mainly in the second step, multiple files are generated on different occasions which can later be used to analyse the approaches behaviour and evaluate the input parameters. The whole code should be self-explanatory in combination with the comments made.

4.3 args.me - Integration into the pipeline

As already introduced in section 2.2, the framework presented by Wachsmuth et al. [2017] gives options to implement steps like the one taken by our approach into their pipeline. The main challenge when applying our approach to specific applications lies in finding most fitting similar functions in the respective context. As to the architecture presented in Figure 2.2, the args.me framework already pre-defines the environment for individual pre-processing

[5] https://docs.python.org/3.6/library/multiprocessing.html

steps. Therefore the prerequisites for integrating corpus pre-processing steps such as our approach are given already. The main challenge in translating our approach into the args.me context will therefore not lie on finding the best possible place to include the concept, but mainly on translating used functions into the Java context.
Since most of our approach makes use of easily transferable concepts, the only real challenge will be to find a fitting, maybe even better performing Java tokenizer. Each individual option has to be assessed in terms of performance and suitability to the task. Additionally the steps of our approach requiring manual interaction will also require to be manually performed in the args.me context. Since we developed an independent concept capable of transforming a given input corpus into a cleansed version of this exact corpus without changing the general structure, one has to think about the necessity of translating the approach in contrast to simply calling it inside the pre-processing steps of the args.me pipeline.

Chapter 5

Evaluation

In the following evaluation we want to discuss the performance of our approach by inspecting different aspects. On the one side we want to guarantee a high precise proceeding in order to maintain argument quality. Therefore we asked three argument expert annotators to conduct a manual evaluation of the results generated by the approach. On the other side we want to discuss the importance of a well chosen ground truth and different aspects of irrelevant sentence position in terms of argument cleansing. We will then discuss the main challenges of the current approach.

Increasing corpus quality without removing important information from the arguments is the most essential goal of our approach. Consequentially, the main focus of our approach lies on a highly precise procedure. Removing only one sentence from the corpus which does not include any relevant data would technically increase corpus quality. However, the approach would not bring a lot of benefit because one could easily remove this sentence manually within way less time and effort. While corpus quality benefits from every single removal, we expect the automatic approach developed to perform better. We do however accept a lower recall in favor of a high precision since even a gradual improvement of the corpus grants benefits for further processing.
In the small study conducted by Professor Wachsmuth and myself in subsection 3.2.1, we analysed the frequency of relevant sentences compared to irrelevant ones in order to fine tune the parameters used in the automatic pattern generation process. This analysis can also be used to roughly estimate the number of irrelevant sentences in our corpus. Even though, neither the analysis conducted nor the estimation provides us with totally accurate numbers, we can still use them to assess the first results obtained in means of the recall. Based on our annotation, we assume roughly 10% of the sentences of the corpus to be irrelevant. This estimation can be used to gain some insight

on the recall values we are achieving.
The corpus has a size of $387k$ arguments and contains a total of roughly 7 million sentences according to our tokenization process. Following the previously mentioned study, this means around 700 000 sentences of the corpus could be expected to be irrelevant. However, multiple factors might invalidate this number. First of all, the corpus consists of multiple sources. We did not investigate whether the analysed arguments in the study all originate from the same debate portal or different ones. If all arguments would stem from one specific portal which might have a significantly higher argument length, this would falsify the value further. Secondly, as we noticed during our approach and mentioned earlier, some arguments may mainly consist of spam. For example of re-occurring irrelevant word groups. These word groups could be separated by sentence delimiters and could be detected as small individual sentences by the tokenizer. This could pump up the sentence numbers as well and falsify the approximation from above significantly. As these questions require further analysis, we want to focus on the notable improvements we achieved.
As mentioned before, a high precision value and therefore a clean removal of irrelevant data in terms of no false-positive removal was the main goal of our approach, therefore it was also the focus of the evaluation.

5.1 Experimental setup

In order to analyse and evaluate the automated process we had to select a sample of sentences and let three annotators analyse whether these autonomous selected sentences are actually irrelevant or not. Since the task at hand requires some background knowledge regarding the topic, we chose three annotators expertised in the argument context. Based on their experience in the field, we were convinced they could solve the task to a satisfying degree.

Annotation data selection Since we wanted to measure the performance of our algorithm over time as well as the precision of each individual step, we decided to extract a sample of 100 new *irrelevant* sentences matching a negative pattern for each round of the algorithm as well as for the initial seed patterns matching. It is important to note that we primarily want to evaluate the precision of our *irrelevant sentence removal* approach. The sentences we are selecting each round are therefore only *clean* irrelevant ones. We count a sentence as *clean* if no pattern of the opposite side occurs in the sentence related to a specific pattern. This means, in our case all extracted sentences for

the manual evaluation are selected based on the fact that they do not contain positive patterns.
As one can see in Table 5.1 the approach generated new patterns over the course of a total of five rounds. While the approach is theoretically designed to run an unlimited number of rounds it took only five rounds to find all patterns given our parameters. In order to represent all rounds we then selected and combined entries from each iteration as well as the seed patterns to a list containing 600 entries in total. This list was then presented to the annotators. As we want to evaluate the precision of our irrelevant sentence removal approach, it is sufficient for us to inspect only negative sentences.
Since the approach generates a mostly decreasing number of new patterns each round (as demonstrated in Table 5.1), it becomes very likely that the sentences in later rounds originate from a very small pool of possible patterns. To reduce the likelihood of creating an unwanted bias by presenting a sequence of similar sentences, we randomized the sentence order before presenting them to the annotators. This would reduce the probability that an annotator would detect the underlying patterns and structures and reduce the risk of creating a bias due to potential common structures of the filtered sentences.

Annotator instructions Alongside the resulting sample dataset, we provided the annotators with a short introduction to the task. As we wanted to minimize the possibility of creating an unwanted bias, we decided not to provide any examples or in depth details to the introduction. We did however define what we assumed to be relevant (5.1.1) and irrelevant (5.1.2) sentences.

Definition 5.1.1 *A Sentence is considered as* ***relevant*** *if it is connected to the argument in any way.*

Definition 5.1.2 *A sentence is considered as* ***irrelevant*** *if it does not support/present the (argumentative) opinion or facts offered by the author. Some examples are: Spam, salutations, insults, meta-comments on the debate, expressions of gratitude or similar.*

Since we wanted to make sure the task is not misinterpreted by our annotators, we highly encouraged them to ask any question regarding the data if necessary. The instructions provided can be inspected in detail in the Appendix B. The annotators than had to classify each individual sentence based on their interpretation of the task as either relevant, or irrelevant.

	Positive Patterns			
Iteration	**Patterns**	**Sentences**	**Auto Prec.**	**Manual Prec.**
Seed	38	600469	0.997	n/a
1	10	7602	0.987	n/a
2	0	-57	n/a	n/a
3	...	-15	...	n/a
4	...	-10	...	n/a
5	...	-6	...	n/a
Total	48	607983	0.984	n/a
	Negative Patterns			
Iteration	**Patterns**	**Sentences**	**Auto Prec.**	**Manual Prec.**
Seed	17	41619	0.972	1.0
1	74	5849	0.983	1.0
2	19	3606	0.982	1.0
3	4	956	0.973	0.96
4	3	594	0.980	0.97
5	5	225	0.983	0.88
Total	122	52849	0.980	0.97

Table 5.1: The number of patterns, number of matching clean sentences, automatically estimated mean precision, and manually evaluated mean precision for positive and negative patterns in each iteration of our approach. The bottom row shows the total numbers and values at the end. For lack of relevance, positive patterns have not been manually evaluated.

5.2 Corpus cleansing evaluation

Automated evaluation In Table 5.1 the total amount of new patterns added in each iteration step alongside the clean sentences added in each round are shown. Furthermore, we calculated automated average precision over all patterns added in the respecting round for both negative and positive patterns. Additionally, the manually calculated precision as majority precision is shown, to demonstrate the performance of the respective round. It is important to mention that the automated and manually calculated precisions do not originate from the same set. The automated precision divides the *sentences containing only same-side pattern matches* by the *total sentences containing the pattern* as introduced in subsection 3.2.2. The set used for the manual evaluation only consists of sentences already clearly classified as irrelevant. Following, the perfect result for the manual evaluation would be 100%.

The sentences classified as clean in this table do however not represent the total number of sentences matched in this respective round, but the number of unique sentences found in this round. Giving an example: The sentence *"Vote pro."* might occur as possible clean negative candidate in one of the rounds. Since this sentence structure is common on debate portals, adding each entry of this specific sentence would increase the volume of found sentences significantly. We do however want to know how many new *different* sentences are found in each round. This way, we can be sure our approach finds more then just one sentence multiple times in each round. This also leads to the fact that the total sentences we will find in the later evaluation will actually be higher than the 52 849 unique sentences we found over the course of our approach.
As shown in the Table 5.1, the amount of clean positive sentences declines slightly from round two to five. Based on the fact that we continuously add new negative patterns while not adding positive patterns, it is possible, these newly added negative patterns previously already existed in sentences marked as clean positive. Due to adding them as possible negative candidates, the number of clean positive sentences is reduced over the rounds. Since precision calculation is done per round and based on newly added patterns, we did not calculate new precision values for these rounds. While we could inspect the average precision of all patterns in each individual round, we wanted to compare the performance of individual rounds in terms of the precision newly added patterns would achieve. Since we defined a hard set lower border of 95% precision in order to maintain quality. We can assume the average precision of each individual round does never drop beneath that border. However, as seen in Table 5.1, the precision of each round fluctuates between 97.2% and 98.3%. This suggests a constant level of quality of our newly found patterns. As the discussion of the manual evaluation later will however show a decline in terms of precision occurs in the iteration steps. As Table 5.1 also shows, the decreasing precision in later rounds does not affect the largest portion of our filtered sentences. All sentences but the 225 found in the last round do achieve an manually evaluated precision of a minimum of 95%. The largest portion however, roughly 51 000 sentences are selected in the first three rounds. All these rounds have been manually annotated to a precision of 100%. While these annotations obviously only represent the whole data based on a small sample, this result could not be any better.
The high share of clean sentences found in the initial step is an indicator for the well selected seed pattern group. Especially on positive side, nearly all relevant sentences detected in our approach are found based on the seed patterns. While it seems to discredit the use of our approach, we believe it strengthens the well chosen ground truth. Additionally, the parameters we used in the automated pattern extraction process could be adjusted to increase these val-

	Annotator Precision					
	Individual			Combined		
Iteration	**1**	**2**	**3**	**Full**	**Majority**	**At least one**
Seed	1.0	0.99	1.0	0.99	1.0	1.0
1	0.98	0.98	1.0	0.96	1.0	1.0
2	1.0	0.95	0.99	0.94	1.0	1.0
3	0.96	0.96	0.95	0.93	0.96	0.98
4	0.95	0.95	0.97	0.93	0.97	0.97
5	0.84	0.85	0.93	0.79	0.88	0.95
Total	0.955	0.946	0.973	0.92	0.97	0.98

Table 5.2: Individual annotator precision as well as *full*, *majority* and *at least one* annotator agreement for each individual round plus the initial seed step. *Full* describes the case of all three annotators classifying the sentence the same. *Majority* agreement represents the decision made by the majority of the annotators. *At least one* only requires one annotator to vote for a sentence to be irrelevant. A complete agreement of 1.0 is achieved if the annotation of each individual sentence is equal to the classification of our approach (all sentences annotated as irrelevant).

ues. In contrast to the generation of the positive patterns, where we did find only 10 new patterns not included in our ground truth, we found 105 new negative patterns and a significant portion of new sentences. While there is an expected decline in newly found patterns and therefore also new clean sentences, especially the first and second round added a large portion of new data automatically. As the following manual evaluation will show, these results can be assumed to have a near 100% precision as well.

Manual evaluation Due to the composition of the annotated dataset, we could analyse the behaviour of our approach over the course of each round. We therefore calculated the precision of each individual annotator for the respecting round, as well as the *full*, *majority* and *at least one* agreement of our annotators. In order to achieve a *full* agreement of *100% (1.0)* for a given round, all annotators have to classify every sentence of the respective round equal to the approach as irrelevant. Respectively, *majority* agreement of *100%* is achieved when at least two out of three annotators agree with our approach. For *at least one* agreement, only one annotator has to classify the sentence correctly. In order to calculate the chance-corrected inter annotator agreement we calculated the Fleiss' κ and achieved moderate agreement of 0.50 between our annotators. Table 5.2 shows the full results.

Since we chose exactly 100 sentences for each individual round, the percentage values can be translated to sentence numbers easily. As mentioned previously, all sentences annotated have been classified as clearly negative by the algorithm. This means we can directly evaluate our approaches performance.
As the Table 5.2 shows, only one of the three annotators found a single sentence not being classified correctly in the seed sample. Inspecting the initial round in terms of majority agreement, the classification based on our seed pattern sample is expected to classify correctly in 100% of the cases. This strongly highlights the high quality of our seed patterns and reinforces the suitability of the chosen pattern extraction approach.
Furthermore, in the first and second round the full agreement of all three annotators dropped only slightly from 99% (seed) over 96% (1-round) down to 94% (2-round). The majority agreement stays at 100% however. While the full agreement already indicates a slight increasing expectable error, at least two of our three annotators classified all sentences from the seed as well as the first two rounds as correctly irrelevant.
While there is a slight decrease in the following two rounds, all annotators agree in 93% of the cases with our automated approach both times. Since the value for *at least one* annotator agreeing with our approach, drops for the first time from 100% in the third round, this marks the first sentence occurrences which all annotators evaluated as wrongly classified irrelevant. However, in the third round, only two sentences (2%) and in the fourth round, only 3 sentences (3%) are evaluated that way by all annotators same.
In the fifth and last round, the decrease increases rather significantly compared to the previous rounds. While the *majority*-agreement drops to 88%, the *full*-agreement drops down to 79%. Since the sentences manually evaluated in every round are directly related to this rounds' patterns, this indicates one or multiple pattern candidates added in this round are not quiet fitting the quality requirements made initially.
Besides the last round, we consider the results delivered by our approach as extremely promising. Excluding the last round the average error according to the majority precision lies close to 99%. As we will show in the discussion later, we expect this error to reduce further by taking additional optimization steps. While the approaches performance drops significantly in the last round, this drop only influences a very small portion of all sentences filtered. We assume this effect might be reduced if pattern selection criteria are changed in future. By adding more positive patterns in the process, the likely-hood of removing relevant sentences can be reduced further. Even though we will introduce some improvements our approach achieves an over all excellent performance in our opinion.

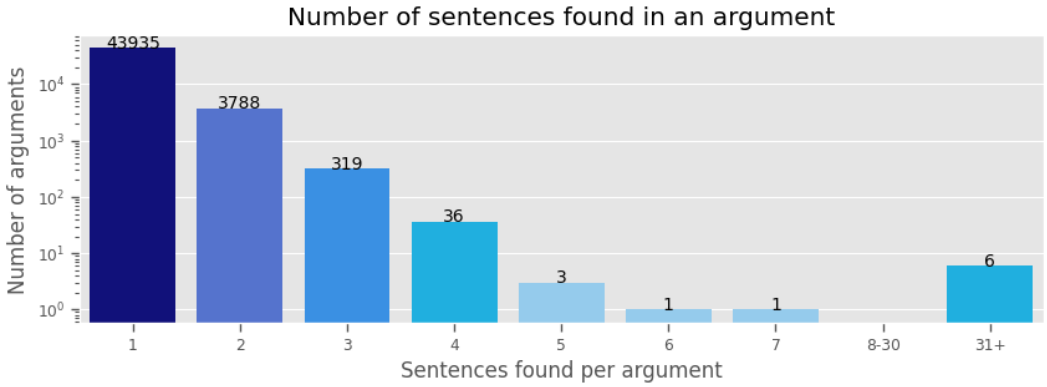

Figure 5.1: The frequency of individual arguments which contain a specific number of matched sentences. Only arguments and sentences we are actually removing are included.

Cleansed sentences per argument Summing up the values from Figure 5.1 we cleansed a total of 48 089 arguments from irrelevant sentences. Considering the Corpus size of roughly 380 000 arguments in total, this means we cleansed more than 12.5% of the arguments contained in the complete corpus from irrelevant sentences. Most arguments cleansed contained between one (43 935 arguments) and three (319 arguments) irrelevant sentences. There are however some cases where the cleansed sentences per argument were much higher. We did not cleanse any argument with more then 7 and less then 30 irrelevant sentences. This strongly suggests the appearance of spam patterns in these extreme cases. The most sentences cleansed in one argument were 642. As a small manual inspection showed, these cleansed arguments containing a high amount of irrelevant sentences often times only consist of spam or end their regular argument with a random amount of copied and pasted reoccurring irrelevant sentences. All extreme cases of more then 30 irrelevant sentences per pattern can be traced back to spam in some form. In total we cleansed 53 502 sentences from these arguments.

Cleansed sentences in relation to sentence position Besides the distribution among arguments, we wanted to inspect the distribution of irrelevant sentences across the sentences of an individual argument. We initially assumed irrelevant sentences to be centered mostly around the beginning and the end of an argument. To support this thesis we calculated the frequency of all pattern matches in the whole argument. Figure 5.2 presents how often a pattern is found in a specific position in an argument. In contrast, Figure 5.3 shows the actual sentences we cleansed following our approach (Subsection 3.3). As

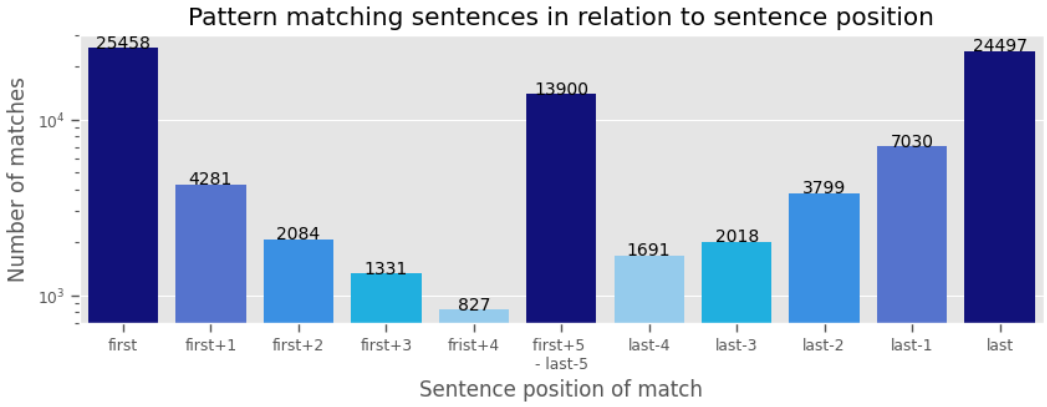

Figure 5.2: Total number of sentences found in the whole corpus in relation to their position in an argument text. *first* to *first+4* describing the first 5 sentences of an argument. *last* to *last-4* describing the last 5 sentences of an argument. The bar in the middle sums up all matches in between the first and last 5 sentences.

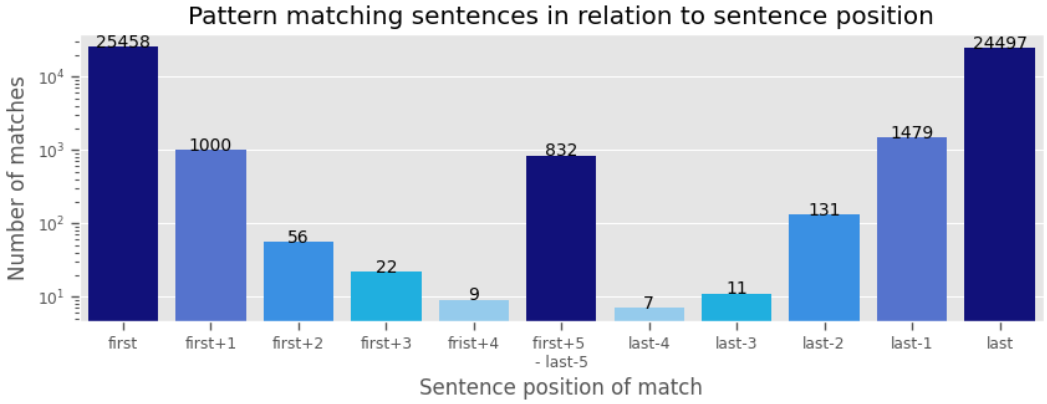

Figure 5.3: Actual number of sentences found in the whole corpus in relation to their position in an argument text. Subsequent sentences have only been inspected if previous sentence was classified as irrelevant. *first-first+4* describing the first 5 sentences of an argument. *last-last-4* describing the last 5 sentences of an argument. The bar in the middle sums up all matches in between the first and last 5 sentences.

indicated by the Figures, in both cases the majority of sentences matched are located at the beginning and end of an argument. While Figure 5.2 shows that 13 900 sentences are found between the sixths and sixths-last sentence, the gradual decrease between the first and fifths as well as the fifth-last and last sentence support our theory of increasing sentence relevance in the center of an argument. Similar to the actually removed sentences, this peak is very likely to originate from unproportionally long arguments which are very likely

to consist of mostly re-occurring spam patterns. From the total of 53 502 sentences cleansed more then 93% where located in the first or last sentence of an argument, supporting the assumption made.
In comparison, we found a total of 86 916 sentences matching our patterns (Figure 5.2) regardless sentence position. These additional 33 414 sentences matched do however slowly decrease from argument beginning and end towards its core similar to the actual removed sentences.

5.3 Discussion

The results shown in the previous section give a detailed insight into the performance of our approach. As we have presented, we are capable to remove a high number of irrelevant sentences from the corpus while maintaining a very high precision. As the manual evaluation confirmed, the process of our automated approach did classify completely correct in half of the process and afterwards only dropped down slightly to around 96%. Only a very small portion of sentences added in the last round of our approach did perform significantly worse with only achieving a manually annotated precision of 88% according to our annotators.

Tokenization In a small manual analysis of the annotator results, we noticed that a portion of the sentences wrongly classified by our algorithm were in fact consisting of multiple sentences. The problem of clean tokenization is an ongoing challenge of NLP and was discussed by us previously. By further improving the tokenization process, we would therefore expect even better results. Additionally, since these combined sentences often times include important information in the other sentences not matching the patterns, an increased sample of positive patterns would at least prevent the algorithm from removing these sentences just now. Since this can only be a temporary solution, the improvement of the tokenization process should be favoured in order to remove the irrelevant sentence segments.

Ground truth selection As also briefly addressed before, the approach is quiet suitable for detecting spam sentences with re-occurring patterns. However, in order to detect new fitting patterns, these patterns have to be included in the sentence set which we create new patterns from, this means these patterns have to be contained in other sentences flagged as irrelevant. This brings

up the currently strongest weakness of our approach. Since our pattern generation process is based on a relation between the patterns previously found and the new ones, we currently have no way to detect sentences not directly connected to any of our previously found patterns. Assuming we have a not yet detected but clearly irrelevant sentence consisting of the exact three same words every time it occurs, we will only be able to filter these sentences out if these three words occur in another sentence already classified as irrelevant. Inspect the following example:

- My opponent is right.
- My opponent is right to say ...

Assuming *opponent right* to be a pattern representing possible irrelevant sentences which we did not find yet. If this constellation should ever only occur in sentences such as the first, we would never be able to filter out these sentences using the current approach since we would never even detect the pattern as a candidate. Assuming this constellation would however exist in a sentence already classified as irrelevant because of matching a different pattern, we would detect all these sentences. While we in general do expect to find these patterns in multiple different sentences, this example highlights the importance of a well-defined ground truth. Only by giving the algorithm enough clean irrelevant sentences it can learn new patterns from will allow him to actually cover a large portion of the text.

Pattern balancing Besides the ground truth, additional factors such as the minimum frequency a pattern is required to appear in order to be considered as a candidate play a significant role in the approaches performance. We consider the ratio of relevant to irrelevant sentences as a relevant factor. We also assume sentence length to be an interesting factor worth a more detailed inspection. In case of detecting specific regularities in terms of sentence length could give additional options in fine tuning the frequency border or introducing new conditions. Also, using as much information as possible to chose reasonable parameters for the frequency border is necessary to keep the balance between negative and positive patterns. Since most of our positive and negative patterns appear in both relevant and irrelevant sentences, one has to make sure no pattern is wrongly classified to the opposite side. When the approach is lacking a sufficient amount of patterns on one side, newly found patterns could be considered for the wrong side since there is a strong imbalance in between its occurrence in already matched sentences. While we challenge this task with

defining a hard set ground truth which will only be enlarged in our approach, this once again enforces the importance of said ground truth.

Cleansing metrics As Figure 5.2 and Figure 5.3 have shown, the current selection of patterns is capable of detecting an even larger portion of sentences which would be classified as irrelevant. The increased frequency of these sentences found in between the first five and last five sentences could have multiple reasons. For one, the calculation sums up all sentences in the center of an argument to one value. As already seen in Figure 5.3 long arguments will lead to such an increase. While spam in long arguments could explain these increased matches in the arguments core other reasons might also be conceivable. Given the origin of the data, it might be very likely that one argument text passage currently does in fact consist of multiple arguments. Our initial assumption bases on the theory that less important text passages are originated at its beginning and end. It would be probable to find irrelevant sentences in between relevant sentences when the text actually consists of multiple arguments. Further analysis of these specific sentences matched in the argument texts core is required to decide whether or not to remove these sentences as well. Investigating this could even help to develop automated approaches to detect argument borders.

Overall improving the quality of large scale corpora often times requires an unreasonable amount of time for manual labourers. Increasing the corpus size will only result in increased processing times. Coming up with an automated approach doing part of this labour does not hold any disadvantages as long one can guarantee a highly precise proceeding. As mentioned previously, all following use-cases and research steps would benefit from small improvements in quality. While we see some challenges and difficulties in our approach, we managed to improve the quality of a large portion of the corpus while minimizing human labour. Inspecting the results from the previous section, making use of our approach in order to improve further research does only hold benefits for everyone.

Chapter 6

Conclusion

6.1 Summary

Current research in the field of argument mining is relying on pre-annotated argument corpora. These corpora are often times based on human created content from so called debate portals. Making use of the given structures of these portals has proven to be very beneficial in terms of simplifying the argument mining process. In some cases however, this leads to impure arguments. Different parts of the argument text can contain irrelevant sentences, such as greetings or personal attacks. These sentences are not related to the argument itself and have no value for the processing of the argument. In order to increase the argument corpus quality, we had to search for a highly precise way to remove these irrelevant sentences from the argument corpus. In terms of scalability, a mostly automated approach is most suitable to challenge this task. Therefore, we developed a semi-supervised learning approach capable of challenging this exact problem. By manually selecting a seed pattern set from a sample of the whole corpus, we established a ground truth on which the automated approach could then detect new patterns matching relevant and irrelevant sentences. In order to ensure a high quality of these selected patterns, we included an automated evaluation process validating the patterns found by our approach. This iterative process of fitting pattern extraction and evaluation then led to a set of patterns precisely classifying sentences and detecting irrelevant text passages. In a manual evaluation conducted by three expert annotators the highly precise performance of our approach is evaluated and confirmed. By cleansing the argument corpus from the detected irrelevant sentences, we generated an improved version of said corpus.

6.2 Contributions

The raw argument data crawled from debate portals is often times used for follow up research regarding argumentation. By introducing a generally applicable approach to cleanse these corpora from irrelevant text passages, we give an opportunity to increase quality of these argument corpora and following research. Due to the creation of a semi-supervised approach, the offered solution is scalable to any corpus size while keeping the required human labour at a necessary minimum. Due to the high precision of said approach, near to no information is lost when applying the approach. By further improving the tokenization process, this loss can be minimized.
Additionally, the introduced approach is meant to define the basic steps of our corpus cleansing process. While we introduce a well performing solution capable of cleansing argument corpora at a very high precision, the individual steps of argument extraction, selection and evaluation can be improved or changed easily. As we focus on extracting irrelevant text based on sentence-specific patterns, improving the extraction and evaluation process directly correlates with researching argument sentence structures. Therefore, both the approach and the research field will prospectively benefit from further research in the respective area.
Besides defining the a general approach of cleansing argument corpora, we applied our algorithm to the args.me corpus and demonstrable increased the corpus quality by a significant degree, as the previous chapter shows (section 5.3). By cleansing this large scale corpus from irrelevant data, we provide a improved corpus version for the args.me search engine as well as all researchers working on the dataset.
Lastly, the patterns generated by our automated approach can be used in different applications as well. While we do not guarantee high performance on other datasets, we assume the patterns to be fitting to detect irrelevant sentences from other sources as well. Individually inspecting the performance in detail might be required in other applications though.

6.3 Outlook

As mentioned in the previous section, we developed a general but highly modifiable approach. The divided structure gives the opportunity to change the initial pattern generation, automated pattern detection and cleansing step independently. In future work, the pattern extraction methods should be reevaluated and different approaches of pattern extraction analysed. By further improving the pattern generation and detection step, the recall of the approach

should increase significantly. A suitable approach that keeps the high precision but increases the recall would be much desirable.
In the current version, both positive and negative pattern generation and evaluation works basically the same way. By analysing different detection and extraction methods for the individual sides (positive or negative patterns) one could come up with an improved approach suitable for classification and pattern generation. Due to the structure of our approach, both processes can be separated and implemented independently without facing bigger challenges.
While the current implementation returns highly precise results on a relatively large scale of arguments, further testing regarding the introduced parameters is required. Adjusting especially the frequency border for newly found pattern candidates could improve the recall even further. Defining different methods on how to select pattern candidates could also prove beneficial.
As one of the biggest challenges of our work was regarding the tokenization process, improving sentence splitting will reduce error and improve sentence matching even further. One has to analyse the fitting metrics for texts originated from any given source (in our case debate portals), in order to find the best rules to tokenize the corpus.
Lastly, we only remove sentences starting or ending a discussion because we expect these sentences to contain the least relevance towards the argument. Since the corpus currently technically could contain multiple arguments in one text passage, one can assume there are irrelevant sentences between those arguments as well. Reevaluating and redefining the argument cleansing process might prove to increase the argument quality even further.

Appendix A

Pseudocode

```
# 1. SEED PATTERN CREATION
# 1.1. INITIAL PATTERN GENERATION
# Parameters:   sample = (corpus_size / 10),
#               n_gram_length = 1,...,5,
#               top_100 = 100
get sample from Corpus
for each of the four n-gram extraction approaches:
    for each n_gram_length:
        get top_100 most frequent n-grams in sample

# 1.2. MANUAL PATTERN SELECTION (GROUND TRUTH DEFINITION)
for each of the four approaches:
    get all n-grams that expert A considers clearly NEGATIVE
    keep only those n-grams that expert B also considers
   clearly NEGATIVE

determine most promising approach:
    create one pattern from each negative n-gram of best-judged
    approach

for each of the four approaches:
    get all n-grams that expert A considers clearly POSITIVE
    keep only those n-grams that expert B also considers
   clearly POSITIVE

determine most promising approach:
    create one pattern from each positive n-gram of best-judged
    approach

# 2. AUTOMATED PATTERN GENERATION
# Parameters:   k_negative = 20
#               k_positive = 20
# 2.1. BALANCE PARAMETERS (UNIQUE SUBSTEP)
```

```
determine sentence frequency ratio between negative and positive sentences
    adjust k_negative and k_positive accordingly
determine sentence length ratio between negative and positive sentences
    adjust k_negative and k_positive accordingly

# 2.2. PATTERN GENERATION
for each matched-sentence containing only negative pattern matches:
    generate n-grams
return all n-grams >= k_negative as new negative pattern candidates

for each matched-sentence containing only positive pattern matches:
    generate n-grams
return all n-grams >= k_positive as new positive pattern candidates

# 2.3. PATTERN EVALUATION
# Parameters:   p = 0.95 / 0.99
# Precision calculation:
# (Sentences-ONLY-negative/Sentences-ALSO-positive)/Sentences_pattern_appears_total

for each negative pattern candidate:
    for each sentence matching the pattern:
        count appearance of positive pattern in negative matching sentence
    calculate precision for given pattern

for each positive pattern candidate:
    for each sentence matching the pattern:
        count appearance of negative pattern in positive matching sentence
    calculate precision for given pattern

for all patterns:
    if pattern precision <= p and not pattern is in ground truth:
        remove pattern
    else:
        add pattern candidate to regarding pattern list

# 2.4. LOOP STEP
# Parameters:   r = X
for r rounds repeat:
```

```
    if positive and negative list equals r-1 positive and negative pattern list:
        end loop
    else if positive and negative pattern list enter "loop state" repeating itself after Y rounds:
        end loop
    else
        generate new pattern by repeating Step 2.2., 2.3., 2.4.

# 3. CLEANSE CORPUS
for pattern in negative pattern list:
    for sentence matching pattern:
        if sentence does not contain positive pattern:
            store sentence

for argument in corpus:
#   3.1.
    if argument starts/ends with stored sentence:
        remove stored sentence from argument
    repeat 3.1. with shortened version of argument until non removed
```

Listing A.1: Simplified pseudocode sample of the Approach

Appendix B

Args.me corpus cleansing evaluation: Annotator guidelines

B.1 Instructions

Argument corpora based on debate portals consist of human created argumentative text spans. Sometimes these text spans contain *information not relevant* to the argument proposed by the author. This irrelevant information can be of different nature. Some possible examples for such **irrelevant** sentences are spam, salutations, insults, meta-comments on the debate, expressions of gratitude or similar.
As part of my bachelor's thesis I am developing a semi supervised approach to automatically detect those sentences in order to remove them from the corpus since they do not hold important information regarding the argument. In order to evaluate the performance of the approach we need to analyse the generated results. By manually annotating the following dataset you will help us assess the approach's efficiency.

You will be presented with a list of sentences from the args.me corpus. Your task is to determine whether each individual sentence contains information related to an argument or has no value to the regarding argument.

Sentences are considered as **relevant** if they are connected to the argument in any way.

We consider a sentence as **irrelevant** if it does not support/present the (argumentative) opinion or facts offered by the author. See above for example kinds of such irrelevant sentences.

Each sentence will be presented individually and the related argument is not shown. The sentence should be classifiable nevertheless. Some sentences may seem to be in relation to each other, but each sentence should be judged individually. It is possible that **all** sentences **or non** are relevant to an argument.

No sentences are included in the purpose of intentionally misleading you.
We will not provide you with examples in order to not create a bias.

If you do have questions regarding the task, do not hesitate to ask!

Bibliography

Yamen Ajjour, Henning Wachsmuth, Johannes Kiesel, Martin Potthast, Matthias Hagen, and Benno Stein. Data acquisition for argument search: The args.me corpus. In *Proceedings of the 42nd Edition of the German Conference on Artificial Intelligence*, page 48â59, 2019. 2.3, 3, 4

Elena Cabrio and Serena Villata. Five years of argument mining: a data-driven analysis. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5427–5433. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/766. URL `https://doi.org/10.24963/ijcai.2018/766`. 2.1

Esin Durmus and Claire Cardie. A corpus for modeling user and language effects in argumentation on online debating. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 602–607, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1057. URL `https://www.aclweb.org/anthology/P19-1057`. 2.3

Ivan Habernal and Iryna Gurevych. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179, April 2017. doi: 10.1162/COLI_a_00276. URL `https://www.aclweb.org/anthology/J17-1004`. 2.3

Zimeng Qiu, Eunah Cho, Xiaochun Ma, and William Campbell. Graph-based semi-supervised learning for natural language understanding. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 151–158, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5318. URL `https://www.aclweb.org/anthology/D19-5318`. 2.4

Niklas Rach, Yuki Matsuda, Johannes Daxenberger, Stefan Ultes, Keiichi Yasumoto, and Wolfgang Minker. Evaluation of argument search approaches

in the context of argumentative dialogue systems. In *Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC)*, 2020. URL https://www.aclweb.org/anthology/2020.lrec-1.65/. 2.2, 2.2, 2.2

Lei Shi, Rada Mihalcea, and Mingjun Tian. Cross language text classification by model translation and semi-supervised learning. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1057–1067, Cambridge, MA, October 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D10-1103. 2.4

Christian Stab, Johannes Daxenberger, Chris Stahlhut, Tristan Miller, Benjamin Schiller, Christopher Tauchmann, Steffen Eger, and Iryna Gurevych. 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, June 2018a. Association for Computational Linguistics. doi: 10.18653/v1/N18-5005. URL https://www.aclweb.org/anthology/N18-5005. 2.1, 2.2

Christian Stab, Tristan Miller, and Iryna Gurevych. Cross-topic argument mining from heterogeneous sources using attention-based neural networks. *CoRR*, abs/1802.05758, 2018b. URL http://arxiv.org/abs/1802.05758. 2.2

Henning Wachsmuth, Martin Potthast, Khalid Al-Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5106. URL https://www.aclweb.org/anthology/W17-5106. 2.2, 2.2, 2.3, 4.3

Marilyn Walker, Jean Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. A corpus for research on deliberation and debate. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 812–817, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/1078_Paper.pdf. 2.3