Using the web to explain claims

Master Thesis

Nick van Bremen (5953790)

A thesis presented for the degree of
Master of Information Science

First supervisor: Dr. I.R. Karnstedt-Hulpus
Second supervisor: Dr. A. Gatt

Department of Information and Computing Sciences
Utrecht University
The Netherlands
February 2023

Using the web to explain claims

Master Thesis

Nick van Bremen

Abstract

An important part of human interaction is argumentation. Arguments can be found anywhere and have been studied in various disciplines, going all the way back to ancient times. With the rise of the world wide web, and more specifically the social web, researchers have gained access to a seemingly endless resource of arguments. The study of argumentation has also gained interest in the field of computer science, where researchers have made it their objective to automatically detect arguments in natural language and store them in a structured way. This area of research is known as argument mining. In order to understand an argument, one often needs reasoning, commonsense, and contextual knowledge. Sometimes an argument is based on believes and assumptions that are not known to someone. For a system that performs an argument mining task, it may also be beneficial to receive more information than just the text that makes up the argument. To help people as well as computers to better understand an argument, multiple systems have been proposed that try to capture the implicit parts of an argument and make them explicit. This has for instance been done by training machine learning models with annotated datasets and by the use of structured knowledge graphs. We expect that many of the underlying believes and assumptions of an argument can be found in unstructured natural language on the internet. Therefore, in this thesis, a pipeline is proposed that may help in explaining an argument by retrieving text fragments from the web using different techniques from the field of Natural Language Processing (NLP). The performance of the pipeline is analysed through a user study, in which people had to choose between the explanation of the pipeline and a baseline. Results show that the task of choosing the best explanation is somewhat ambiguous, making it unclear whether the proposed system outperforms a baseline. The process of building the pipeline has also made it clear that certain NLP tasks still need some progress to be used in a downstream task like this.

Contents

1 Introduction **5**
- 1.1 Problem statement . . . 6
- 1.2 Contributions . . . 6
- 1.3 The pipeline . . . 7
- 1.4 Layout . . . 8

2 Related Literature **9**
- 2.1 The structure of arguments . . . 9
  - 2.1.1 The Toulmin Model . . . 10
  - 2.1.2 Argumentation schemes . . . 10
  - 2.1.3 Structures in Argument Mining . . . 11
- 2.2 Argument Mining . . . 12
- 2.3 Using background knowledge . . . 13
  - 2.3.1 Argument identification . . . 13
  - 2.3.2 Relation identification . . . 14
- 2.4 Making implicit knowledge explicit . . . 14
  - 2.4.1 Annotation of implicit knowledge . . . 14
  - 2.4.2 Enthymeme reconstruction . . . 15
  - 2.4.3 Claim matching . . . 16
  - 2.4.4 Explicitation . . . 16

3 Background **18**
- 3.1 Phrase extraction . . . 18
  - 3.1.1 Part-of-Speech tagging . . . 18
  - 3.1.2 Dependency parsing . . . 19
- 3.2 Web querying . . . 19
  - 3.2.1 Crawled datasets . . . 20
  - 3.2.2 Search engine APIs . . . 20
  - 3.2.3 Web scraping . . . 21
- 3.3 Snippet extraction . . . 21
  - 3.3.1 Rhetorical Structure Theory . . . 21
  - 3.3.2 Discourse markers . . . 22
- 3.4 Text summarization . . . 22
- 3.5 Stance detection . . . 23
- 3.6 Clustering . . . 23

4 Implementation **25**
4.1 Phrase extraction . . . 25
4.2 Web querying . . . 28
4.3 Snippet extraction . . . 29
4.3.1 Summarization . . . 31
4.4 Stance detection . . . 31
4.5 Snippet ranking . . . 32

5 Evaluation **33**
5.1 Set up . . . 33
5.1.1 The baseline . . . 33
5.1.2 Data gathering . . . 33
5.1.3 The annotation site . . . 35
5.2 Results . . . 36
5.2.1 Descriptive statistics . . . 38
5.2.2 Inter-annotator agreement . . . 38
5.2.3 Performance . . . 39
5.3 Discussion . . . 46

6 Conclusion **49**

A Argument structure **56**

B Task description **57**
B.1 Introduction . . . 57
B.2 The task . . . 57
B.3 Important terms . . . 58
B.4 How the data is stored . . . 59

C Planning **60**

Chapter 1

Introduction

For probably as long as mankind has existed, people have been expressing their views and opinions, have tried to persuade each other, and have sought to sort conflicts. This has been done with the use of arguments [44]. Expressing these believes may be done through gestures or facial expressions, but argumentation is mostly considered to be written or spoken [71]. Scholars from all sorts of disciplines have been interested in argumentation for millennia [45, 51, 30, 77], with the most famous traditional work coming from Aristotle [71, 44]. After him, many others have tried to capture a definition and model of arguments. Most models have in common that an argument is made up of one or multiple *premises* that relate to a certain *claim* [73, 28, 77, 30].

Besides classical fields of research like law, logic, and philosophy, argumentation also became a subject of interest in the field of computer science. The automatic retrieval of arguments from unstructured natural language became a new area of research known as *argumentation mining* [45]. The structure of arguments has become an important aspect in this area, as computational implementation requires some sort of formalization [77]. Early advances in argumentation mining were made in the legal field, as the retrieval of arguments from precedent cases is an important aspect in judging a current case [51]. This, in combination with the relatively structured nature of legal cases, made the legal field a good starting point for argumentation mining and the use of *argumentation schemes* [77, 20].

Apart from court decisions, argumentation can be found in plenty of places, like scientific papers, medical texts, patent requests, and so on [51]. The rise of the World Wide Web in the past decades, and specifically the coming of social media, has given researchers access to a seemingly endless resource of arguments. A difference between arguments found online and arguments found in the aforementioned texts, is that online arguments often do not have a clear structure and may lack all the information necessary to completely understand the argument.

With the coming of large language models like BERT, tasks within argument mining and *Natural Language Processing* (NLP) have made impressive performance gains. But it is argued that these language models do not have a real understanding of the meaning of what they produce and merely work with superficial features derived from the text [17, 55]. Furthermore, it was found that most tasks in argument mining will benefit from the use of background knowledge [44].

1.1 Problem statement

Arguments that lack one or multiple premises, or even the conclusion, are called *enthymemes* [77]. Enthymemes pose a difficult problem in argumentation mining, as it requires a system to reason and understand the context to make sense of an argument. Besides the argument missing certain parts, there may be other information that is assumed to be known by the one asserting the argument. This may be commonsense knowledge, believes, assumptions, or just simple facts.

All this implicit information may not be known by a human, and certainly not by a computer. Therefore, it might be interesting to find a way to automatically make explicit the implicit parts of an argument. Gathering the information underlying an argument has been tried in multiple ways. Most research focuses on finding this information in structured knowledge bases or unstructured factual sources like Wikipedia. In this study, we assume that important implicit information does not have to be factual or stored in knowledge graphs, but may very well be believes and assumptions that are not necessarily true. Therefore we want to use unstructured natural language from sources found on the internet that may help in explaining the underlying assumptions in an argument.

In this research, the main aim will be to explore whether our approach produces good results and can be used as an alternative way to make implicit parts of an argument explicit. This leads to the main research question:

> **Research Question:** *How can arguments be explained using text sources from the web?*

This research question can be divided into two sub questions. First, a means is needed to automatically retrieve fragments of text from the internet. To this end, the following question is posed:

> **Sub Question 1:** *How can potential explanations for an argument be retrieved from the web?*

In order to answer this question, a pipeline will be proposed using multiple existing methods and techniques from the fields of argument mining and NLP. Different approaches will be tried and a qualitative assessment will be done on the different components, modifying the component where needed. As a final step, the system as a whole should be evaluated. As the text snippets that will be retrieved may be anything, it is not possible to use a dataset to analyse the performance of the pipeline. The pieces of text are unstructured and content may be retrieved one would not have thought of in advance. It would be insufficient to judge the output of the pipeline on one or a few references [14]. Therefore, a method is needed to thoroughly analyse the pipeline, leading to the second sub question:

> **Sub Question 2:** *What methods can be used for analysis of the proposed solution?*

1.2 Contributions

The main contributions of this research are the following: 1) an alternative approach to finding explanations for arguments that is not restricted to a specific kind

of explanation and may result in information on believes and assumptions that is not modelled in knowledge bases, 2) a pipeline for finding these explanations, that takes as input a claim, topic, and stance, and outputs multiple text fragments that explain the connection between the topic and claim, and 3) a means of analyzing the unstructured output of the pipeline in a quantitative way.

1.3 The pipeline

In this section, the different components of the proposed pipeline will briefly be introduced. This is done to provide an overview of the rest of the document. Pointers will be given to where information can be found on the background of the task (Chapter 3) and on the implementation of the component (Chapter 4).

Figure 1.1: High-level overview of the proposed pipeline. It takes as input a claim, topic, and stance, and outputs text snippets.

Figure 1.1 shows the main components of the pipeline. The pipeline takes as input a claim, a corresponding topic, and the stance of the claim towards the topic. In the first step, important words and phrases are extracted from the claim and the topic which will be used in subsequent steps. This will be done by means of Part-of-Speech (POS) tagging and dependency parsing for which background is given in Section 3.1. The implementation is explained in Section 4.1. Next, the web will be searched for relevant articles. Some background on the multiple considered approaches is given in Section 3.2, while the final implementation is described in Section 4.2. After the articles have been collected, an algorithm is needed to extract relevant pieces of text from the full document. The created algorithm makes use of simple word matching, using the terms extracted in the first step, while taking into account paragraph boundaries. The full implementation is detailed in Section 4.3, while some background on used and considered techniques is given in Section 3.3. As the algorithm sometimes extracts lengthy snippets, a sub task of this component is to summarize snippets that are deemed too long. Some background on summarization is given in Section 3.4. Now that all snippets have been gathered, the stance towards the topic will be determined for every snippet. Stance detection will further be explained in Section 3.5, while the used model is presented in Section 4.4. As a last step, a selection of relevant snippets has to be made. This is done by clustering all the snippets that have the same stance towards the topic as the claim and then extracting the snippet closest to the centroid of the cluster, which will be explained in Section 4.5 with some background in Section 3.6. This leaves us with a number of snippets which is equal to the number of clusters. As a last step, the snippets are ranked based on the size of the cluster they are found in.

1.4 Layout

In the following chapter, related literature will be presented. The chapter starts with a broad overview of the structure of arguments (Section 2.1) after which the field of argumentation mining is introduced (Section 2.2). Then, the use of background knowledge in different argumentation mining tasks is discussed (Section 2.3) and the chapter ends with the specific task of making implicit knowledge explicit (Section 2.4). Background on the different methods and techniques used in the pipeline is given in Chapter 3, after which the implementation is specified in Chapter 4. Chapter 5 first presents the approach for evaluation of the pipeline, after which the results of the evaluation are given, and ends with a discussion of the results. Lastly, in Chapter 6, the thesis is concluded.

Chapter 2

Related Literature

In this chapter, an overview of the related literature is given. First, the structure of an argument will be discussed and the different terms used in the field of argument mining, followed by an overview of this field. Then, studies that use background knowledge will be discussed followed by studies on the task of making implicit knowledge explicit.

2.1 The structure of arguments

There are two main levels of detail at which one can look at the structure of arguments: the *micro-structure* and *macro-structure*. At the micro-structure level, an argument is split into different *argumentative units* (also referred to as *argumentative discourse units* or *argument components*) that have a relation to one another. At the macro-structure level, arguments as a whole are considered, and the relations between different arguments is examined. Within these two granularity levels, Bentahar, Moulin, and Bélanger [18] define three perspectives: *monological*, *dialogical*, and *rhetorical*. The monological perspective is concerned with the internal structure of an argument and looks at the different components that make up a single argument and the relations between these components, and is thus at the micro-structure level of argumentation. The dialogical and rethorical perspective both consider the macro-structure of arguments and therefore how complete arguments relate to each other. The difference is, that the dialogical perspective is more concerned with formal relations between arguments, whereas the rethorical perspective is concerned with convincing an audience and discursive techniques to do so. In this study, we are concerned with single arguments and are thus interested in the monological perspective of arguments. Therefore, the rest of this section will deal with models on the micro-structure of arguments.

There are different ways to explain the micro-structure of an argument. A classical way to explain an argument which has been used since the time of Aristotle is by separating the argument into *premise* and *conclusion* [73, 71]. The premises can than be divided into *minor premise* and *major premise*, where the minor premise refers to a specific case, while the major premise asserts a generalization. A famous example of such an argument is the following:

> Socrates is a man;
> Every man is mortal;
> Therefore, Socrates is mortal.

An important aspect of the classical logic, is that a conclusion is accepted when both premises are assumed to be true, like in the case of the example. This kind of argument is called a *syllogism* [73, 71]. By means of deductive reasoning, the argument can be accepted or disproved. This view has been adopted by logicians and was seen as a good way to test whether an argument is sound. A problem with this view is that an argument may only be true for certain cases, or may be made tentatively and be refuted with the coming of new information. An argument that does not hold indefinitely, is called *defeasible* [71, 77].

2.1.1 The Toulmin Model

Using traditional inference to either accept or reject an argument has its problems. Toulmin argues that pure syllogisms are not common in everyday argumentation and believes that the representation of arguments using two premises and a conclusion does not suffice [73]. At the core of his criticism lies the form of the major premise in what he calls an *analytic* argument, which is either "All A's are B's" or "No A's are B's". He asserts that in most cases this does not hold and that this premise generally has the form of "Most A's are B's" or "Barely any A's are B's". Using traditional inference, no certain conclusion can be drawn from the last two forms of premises. An argument that does not implicitly or explicitly hold its conclusion in its premises, Toulmin calls *substantial*.

He then goes on to describe a more fine-grained structure that may be used to model substantial arguments. In this model, he also uses some sort of premise and conclusion, which he calls *data* and *claim*. To make the inference from some piece of data to a claim, one or multiple *warrants* are needed. To support a warrant, a *backing* may be added. Where data can be compared to the minor premise, a warrant together with backing can be considered the major premise. Two more elements are now added to the model: the *qualifier* and *rebuttal*. The qualifier is used to indicate the certainty of the inference from data to claim, whereas one or multiple rebuttals may be added to present exceptions on the inference. A much cited example of the complete model from Toulmin [73] can be seen in Figure 2.1.

When writing his book, Toulmin set out to challenge the view that any argument can be formally described in some sort of model, specifically the syllogism. It was never his intention to produce a new model for argumentation. However, the terms proposed in his book to describe an argument have come to be known as the Toulmin model which has been widely used in various disciplines, among which computer science. Especially in the field of argument mining, numerous variations of the Toulmin model have been applied, as will become clear in Section 2.1.3.

2.1.2 Argumentation schemes

The classical model and Toulmin model mentioned in the previous sections are relevant to model arguments in general. But there are multiple types of arguments. A comprehensive overview of the structure of different types of arguments has been given in Walton, Reed, and Macagno [77]. They describe sixty-five schemes in terms of premise and conclusion. Depending on the scheme, different types of premises are used. They may use a structure in the classical sense, with a major and minor premise, or add more premises and just number them. In some cases, premises

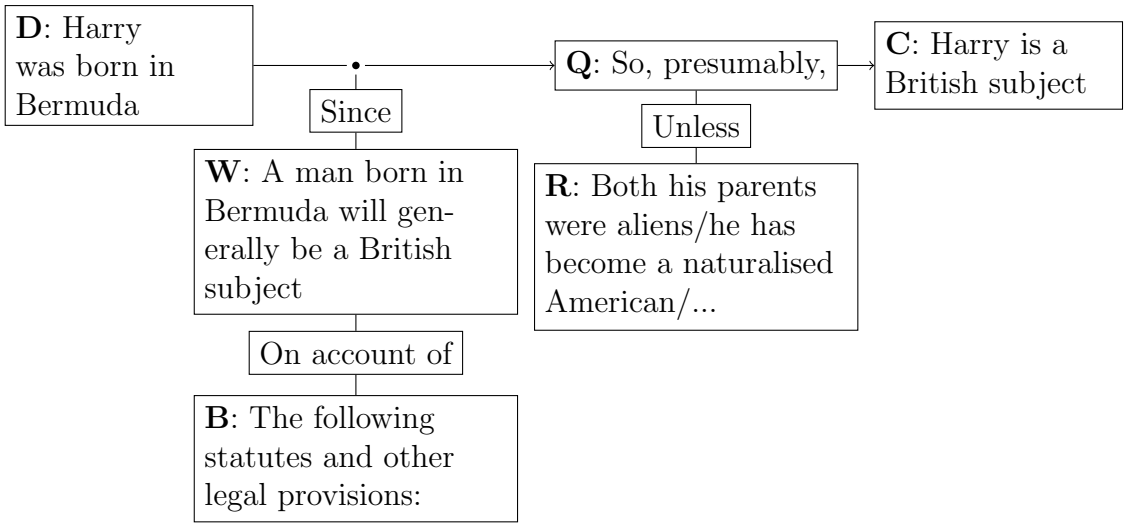


Figure 2.1: Example of the complete model taken from Toulmin [73]. Where D stands for data, W for warrant, B for backing, Q for qualifier, R for rebuttal, and C for conclusion.

may have specific names. For instance, for the argument from analogy, besides the major and minor premise, they define a *relevant similarity premise* to complete the scheme.

Besides the premise and conclusion structure, every scheme has corresponding *critical questions*. The questions can be used to determine whether an argument that fits in a certain scheme is sound or whether it is fallacious. Argumentation schemes have been widely used in computer science and may help in the automatic analysis of arguments.

2.1.3 Structures in Argument Mining

In the field of argument mining (which will be further discussed in Section 2.2), the model by Toulmin [73] and the more specif schemes by Walton, Reed, and Macagno [77] have often been used. Apart from these models, other ways of describing and structuring argumentation have also been used. These structures are often inspired by the Toulmin model or the classical model of argumentation, where an argument always at least consists of a premise and a conclusion. But, since most authors choose to use different terms, it can become quite confusing what is actually being referred to. For this reason, a table was created that can be found in Appendix A. In this table, we took the classical and Toulmin model as the basis, and tried to connect terms used in other papers to these models. This table is based on our interpretation of how the authors described the terms they were using, and does not try to suggest all these terms have a one-on-one relation or are entirely interchangeable. It is merely for clarification.

This table only takes into account the micro-structure of an argument. Certain papers further use the term *topic* [39, 67, 63, 20, 38, 68, 29, 7] to which an argument has a certain *stance*. Furthermore, not all papers are concerned with the full micro-structure of an argument and only refer to a specific part of an argument (e.g. claim [38, 7]).

Looking at Table A.1, it seems most structures are mainly concerned with some

sort of premise and a claim (or conclusion). The qualifier and rebuttal present in the model by Toulmin [73] do not seem to be of importance in other structures. In general we can say that an argument consists of premises and a conclusion. The premise overlaps with the minor and major premise in the classical model, and with the data, warrant, and (sometimes) backing in the Toulmin [73] model. The conclusion is also referred to as the claim.

For the purpose of this study, we will mainly be concerned with the claim (or conclusion) of an argument and how it relates to a certain topic. Now that it has been clarified how arguments are structured, which terms are used, and how they relate, we can go on to discuss argumentation within computer science.

2.2 Argument Mining

The research area of *argument* (or *argumentation*) *mining*, is mainly concerned with deriving arguments from unstructured natural language and storing them in a structured way [45]. But this is not just one unified task. To go from natural language to structured data, multiple tasks have to be conducted that all have their separate sub- and supporting tasks. Now, on the basis of an overview paper by Lippi and Torroni [45] and the a book by Stede and Schneider [71], an overview will be given of the different tasks that encompass argument mining. A typical argument mining system would roughly consist of three main tasks: finding argumentative text, finding argument components, and relating the argument components and complete arguments.

Given multiple natural language texts, the first task would be to determine for each text, whether it includes arguments at all. This may be done by classifying texts as argumentative or not, but could also be done by detecting whether a text is opinionated, or whether it is subjective or objective. After this first classification of full texts, the same may be done for individual sentences or for larger fragments of text.

Now that it is established which sentences contain (parts of) arguments, the next task would be to determine what argumentative components are present in the sentences, and where each component begins and ends. Depending on the chosen argument model, different components may be identified, but generally speaking a system will look for a claim and premises. Although finding claims and premises is often done in parallel, Stede and Schneider [71] also describe the tasks separately (using the term *statements* for all argument components that are not the claim). A statement belonging to a claim is usually thought of as supporting the claim, but it may as well be an opposing statement. Looking at the model by Toulmin, this may for instance be a rebuttal.

Finally, the relation between all argumentative units and arguments themselves has to be predicted. There are different schemes to annotate arguments. Simpler schemes only use support and attack relations to describe how arguments or argumentative units are linked. Others are more fine-grained and divide the attack relation in rebuttal and undercut. Some will take into account only full arguments, while others will relate arguments via their different components. Lippi and Torroni [45] say that this is the most challenging task that may require knowledge of the context that is not present in the arguments.

The previous paragraphs have given a high level overview of a hypothetical argument mining system, but these are not the only tasks within the field of argument mining. Although the complete task of argument mining has now been split up into more manageable parts, some of these separate tasks are still hard to accomplish and may require a complete system on their own. There are also multiple supporting tasks that may help one or more of the main tasks. The task of *stance detection* may for instance be helpful to determine whether a premise is against or in favor of a certain claim. *Enthymeme reconstruction* may help in completing an argument so that a downstream task can better relate the different argumentative units. And all these tasks may benefit from the use of background knowledge to improve their results [44].

In this study, we are interested in explaining arguments by finding what connects a claim to a topic using unstructured sources from the internet. This extra information may help a human better understand an argument and may improve downstream tasks. The remainder of this chapter is divided into two main parts: first, studies that use background knowledge to improve an argument mining task will be discussed, after that, research will be discussed that specifically focuses on finding this implicit knowledge and making it explicit.

2.3 Using background knowledge

In this section, two main tasks of the argument mining pipeline will be discussed that may benefit from the use of contextual knowledge.

2.3.1 Argument identification

The task of *argument identification* is one of the first tasks in a typical argument mining pipeline and is described as the task of finding premises and conclusion of an argument in a text [45]. This task may benefit from the use of background knowledge.

In their paper, Fromm, Faerman, and Seidl [29] show that taking into account the topic when detecting arguments may improve the task of argument identification. They claim that most argument search engines only rely on the sentence itself to determine whether it is an argument to a certain topic. In these search engines, the assumption is made that everything is relevant to the topic, as the search is based on a certain query that holds the topic. However, Fromm, Faerman, and Seidl [29] prove that including the topic in the identification task improves the results, and argue that adding extra context with the use of knowledge graphs may further boost the task.

Using knowledge graphs in combination with a topic and a sentence was done by Abels et al. [1]. Their task was similar to that of the previous paper: given a topic and a sentence, determine whether the sentence is an argument towards the topic. They believed the use of contextual knowledge could enhance the system. To find this knowledge, entities were extracted from the topic and the sentence, and the structured knowledge graph Wikidata was used to find what connects these entities. This would give a starting point for building a topic related knowledge graph, that would further be augmented by extracting triples from a web search. Adding this

knowledge graph as input to a classifier, together with the topic and the sentence, improved the classification.

For our research, especially the last study is relevant, as it not only demonstrates the usefulness of contextual knowledge for an argument mining task, but it also shows a method for retrieving this knowledge from structured as well as unstructured data. Specifically the extraction of entities from the topic and sentence with which the web is queried, is similar to our research. A difference is that we are not concerned with a specific task (in this case argument identification) and we take into account the stance of the sentence towards the topic.

2.3.2 Relation identification

Finding the relations between argumentative units or complete arguments is known as *relation identification* (also referred to as *relation prediction* and *relation classification*) [45, 44, 71]. The aim is to determine for two arguments, whether they are in a support or attack relation, but more fine-grained classifications may be used as well. In a typical argument mining pipeline, this is seen as one of the last, and more complex tasks [45, 71].

To improve this task, contextual or commonsense knowledge may be needed. To this end, Kobbe et al. [39] made use of knowledge graphs, such as ConceptNet and DBpedia, as extra input to a Siamese Neural Network that classifies the relation of pairs of argumentative units. The knowledge graphs are used by linking all entities in a pair of arguments to the knowledge graph, and then extracting paths (up to a length of three) between every entity pair from both argumentative units. The relations themselves, together with some features derived from the paths, are then added to the input of the classifier. The experiments conducted in the study prove that augmenting a classifier with this background knowledge improves the performance compared to a non-augmented baseline.

2.4 Making implicit knowledge explicit

Different tasks have been proposed to uncover the underlying premises that help explain an argument. These tasks have in common that the assumption is made that not all information needed to understand an argument is present within the argument [13]. Various terms are used to describe what is missing in an argument and the search for these implicit parts has been conducted in multiple ways. Whereas in the previous sections tasks have been discussed that use some sort of external knowledge to improve the task, we will now discuss research that is specifically set out to make implicit information explicit.

2.4.1 Annotation of implicit knowledge

It is often mentioned in literature that there are few large datasets for argumentation. This is considered a problem, as a lot of data is needed to train a machine learning model [58]. To counter this, multiple studies propose methods for annotating implicit knowledge and various annotated datasets have been published.

Becker, Korfhage, and Frank [13] propose a structured way of annotating implicit knowledge to pairs of argumentative units (from the Microtexts Corpus [58]). They

claim that most annotation studies are too unstructured, making the differences in annotations added by different annotators too big. Therefore a multi-step annotation process is proposed in which annotators review each others annotations [16]. The argumentative units in the dataset as well as the added implicit knowledge are then further annotated using semantic clause types and commonsense relations.

Another annotation study that uses a structured representation to add the implicit knowledge that connects a premise to a claim, was proposed by Singh et al. [66]. They derive what they call *action* and *outcome entities* from claims and premises, and then add *implicit causal knowledge* that explains the step from premise to claim. To structure this, they add the relation between the income entity, the implicit causal knowledge, and the outcome entity. This is either a cause or suppress relation. The relations are added because the authors believe that only adding the implicit knowledge does not tell enough about how the premise and claim are related.

A task that is closely linked to uncovering knowledge that is left implicit in an argument, is the *reasoning comprehension task* as introduced by Habernal et al. [31]. Given a reason and a claim, the task is to find the warrant connecting them, choosing from two opposing warrants. To this end, a dataset is constructed consisting of a claim, reason, and two warrants. One warrant is correct, while the other is the opposite.

The problem with implicit knowledge is that it may come in any form. It may be warrants or causal relations, but it can also be facts, believes or other arguments that are assumed to be known. Therefore, uncovering implicit knowledge using datasets can be limiting, although it does facilitate a good evaluation.

2.4.2 Enthymeme reconstruction

Most arguments do not explicitly state everything that is needed to understand the argument. Often, certain argument components are missing. Arguments from which a premise, multiples premises, or even the conclusion is missing, are called *enthymemes* [77, 76, 71]. For the understanding of arguments by a system or a person, it is useful to have all the argumentative components explicit, as well as any extra information that is relevant. This task is called *enthymeme reconstruction*.

A first step in reconstructing an enthymeme was taken by Rajendran, Bollegala, and Parsons [60]. They created a binary classifier that could distinguish between implicit and explicit opinions. They argue that an implicit opinion is closely related to an argument with a missing premise. A dataset with opinions about hotel rooms is augmented with either the conclusion that the reviewer is in favour of (an aspect of) the hotel, or against. Furthermore, for every statement it is annotated whether it is implicit or explicit.

But finding an enthymeme is only the first part. Chakrabarty, Trivedi, and Muresan [24] try to generate an unstated premise by using a commonsense model. They use a dataset that contains two observations and a hypothesis connecting the observations. They use one observation as the explicit premise, the other observation as the claim, and the hypothesis as the implicit premise. Their goal is to automatically generate the implicit premise using a pre-trained language model, fine-tuned on the aforementioned dataset and a commonsense knowledge model. The commonsense model returns a commonsense inference for a given sentence. Adding this

commonsense knowledge for the explicit premise as input for the model gave better results compared to only inputting the explicit premise and the claim.

This study is especially interesting, as it not only makes explicit an implicit part of the argument, but also demonstrates the benefits of using commonsense in this task. The difference between the task of enthymeme reconstruction and our research, is that we are not concerned with specifically finding premises to complete an argument. We want to find anything, specifically believes and assumptions, that may help in explaining an argument. Of course, this may include missing premises.

2.4.3 Claim matching

Boltužić and Šnajder [20] argue that user-generated claims found online are often noisy and do not have a clear structure. As lots of claims are made online, there is a need to group claims together. This may be done using superficial information like text similarity, but it may be that two claims arguing for the same point use totally different wordings. To match a claim to a *main claim*, the authors propose to create a dataset consisting of claims, main claims, and premises that connect the claim to the main claim. It was found that the premises help in correctly matching user-generated claims to overarching main claims. The problem is that there will not always be a dataset with premises for all the user-generated claims. Therefore, there should be a way of automatically finding these premises, which makes the approach to claim matching somewhat similar to that of enthymeme reconstruction. The authors also provide a preliminary attempt to do this, but do not outperform the baseline model.

2.4.4 Explicitation

The task of argument explicitation was proposed by Hulpus et al. [35] and defined as the task of providing information that explains the implicit parts of an argument to help either a person or system better understand and interpret a given argument. These implicit parts may be anything, it could be missing premises, relevant facts, or information that connects entities within or between arguments. The authors divide the task into two sub-tasks: argument analysis and argument reconstruction. The first sub-task is mainly about understanding the structure of the argument and identifying argument components (also referred to as argumentative units). The second sub-task is about adding any information in an argument that may be missing.

In the study, three types of knowledge are mentioned that may be used to analyse and reconstruct arguments: 1) knowledge that is captured within the text of an argument itself, this is quite shallow knowledge that uses e.g. word embeddings or discourse markers, 2) knowledge about argumentation, for instance the structure of arguments or the relation between them, and lastly 3) background knowledge, which may be anything that explains the argument.

Some other sub-tasks of argument mining are discussed in the paper, of which *knowledge enhancement-based explicitation* is the most relevant task for our study. This task is similar to enthymeme reconstruction, but it differs in that all kinds of knowledge and facts may be added to the argument, as opposed to enthymeme

reconstruction, where specifically missing parts of an argument (premises or conclusion) are to be found and added.

Implicit parts of an argument may also be commonsense knowledge. In order to make this knowledge explicit, Becker et al. [15] propose a framework that creates paths of commonsense knowledge between concepts from two sentences. Instead of using static knowledge bases like ConceptNet to find relations, the authors use *relation classification* and *target prediction* to dynamically create knowledge paths between concepts. This results in the addition of highly structured information that helps in understanding what connects two sentences. The aim of our study, finding information that further explains a claim by for instance connecting it to the topic, is somewhat similar, but it is done with unstructured data.

Chapter 3

Background

In this chapter, some background information will be given on the different methods and techniques used in the pipeline. The description of the implementation of the various components can be found in Chapter 4. It is not the aim of this study to improve the techniques mentioned here, but to combine them and use them to try and solve a downstream problem.

3.1 Phrase extraction

In the first step of the pipeline, relevant words and phrases are extracted from the topic and claim which are to be used in other steps. To extract these words and phrases, two techniques from the field of NLP are used: Part-of-Speech tagging and dependency parsing. These will now briefly be explained.

3.1.1 Part-of-Speech tagging

Understanding the function of a word in a sentence is an important task that forms the basis of many other NLP tasks [50, 25]. Assigning labels (or tags) to words to describe their grammatical function in a sentence is called *Part-of-Speech (POS) tagging*. The idea behind POS tagging is that all parts of a sentence can be categorized in a certain class. Examples of these classes are "noun", "verb", and "adjective" [74, 25].

Although every natural language makes use of word classes to distinguish words in a sentence, there are differences between the used classes and the number of classes [65]. The Universal Dependencies[1] project brings together the annotation for many different languages and is based on existing schemes. It provides an overview of 17 POS tags which they name the universal part-of-speech tags [49].

The automated tagging of words in a sentence is not entirely straightforward, as natural language is ambiguous [50]. Multiple approaches have been tried for POS tagging. Early work focused on rule-based approaches, but creating the rules is time-consuming, requires linguistic knowledge, and is prone to error [25, 12].

With the help of these early methods started the annotation of large corpora, like the Brown corpus [74]. Annotated corpora could then be used in stochastic methods for POS tagging. These methods make use of statistical information and

[1] https://universaldependencies.org/

probability to assign a tag to a word and are implemented based on models like the Hidden Markov Model and the Maximum Entropy [50, 12, 25]. Furthermore classic Machine Learning and Deep Learning approaches are used for POS tagging [25]. State-of-the-art performance in many NLP tasks, and thus also in POS tagging, is achieved with Transformer models like BERT [47].

3.1.2 Dependency parsing

Related to POS tagging is the task of *dependency parsing*. A sentence is not a random collection of words, but a coherent structure of connected parts [41]. With dependency parsing, the aim is to make this structure explicit by finding the relations between words. The output of a dependency parser is a set of directed edges going from the parent word (or head) to the child word (or dependent). These edges have labels describing the grammatical relation between the two words [49, 32].

To train a model for dependency parsing, large annotated datasets, called *treebanks*, are often needed. Manually annotating this data is time consuming. Famous large datasets are the Penn Treebank and the Brown Corpus, which contain annotations on dependencies as well as POS-tags [78]. The earlier mentioned Universal Dependencies project was set out to create a standard for linguistic treebank annotation. Based on efforts by Stanford and Google, a universal standard for annotating dependencies has been created [56].

Generally, models for dependency parsing can be divided into two broad classes: graph-based and transition-based, which Zhang [78] further divides into two broad approaches: the statistical and neural approach. The models have a comparable performance, but require large annotated treebanks. Other approaches have therefore focused on unsupervised dependency parsing, although these do often rely on the availability of POS-tags [32].

3.2 Web querying

In order to find relevant pieces of text, the internet has to be searched for relevant articles. To do this, a web search engine, like Google [22], can be used. These search engines generally consist of three main components [10]. The first component is a *crawler* that starts at a certain web page and searches for links to other web pages where it will continue the process. This way, all the different web pages on the internet can be mapped [37]. The second component is the *indexer*. After having found all the links to web pages, it is important to understand what a web page is about. This is done by processing all information on a web page and storing it in a database that is easy to be searched [69]. The final component, which may be divided into two components, is the *query engine*. Here, web pages are retrieved based on some query. As a high number of pages may fit the query, part of this component is to rank the pages [10].

Services like Google are built for human users that will search for something and open one or a few links from the results page to read, and not for computer systems that try to obtain as many URLs from the results page as possible. As existing search engines do not make it easy to programmatically use them, and it is not feasible to implement a complete web search engine in this project, other ways have to be explored. Three approaches have been considered which all have their pros

and cons. Firstly, available crawled datasets can be used to index and build a search engine on top of to retrieve relevant articles. Another option would be to use and API from and existing search engine. Lastly, a web scraper can be built to scrape an existing search engine. All three approaches will now shortly be discussed.

3.2.1 Crawled datasets

Big commercial search engines do not publicize the crawled, and certainly not the indexed, data, as it is the core of their business. Open source initiatives have been set up that do make this data available which can be used to build your own search engine on top of. One such project is Common Crawl[2], a non-profit that makes vast amounts of crawled data accessible to everyone. This provides a way to bypass the commercially available search engines, although the data does still require a custom search engine to extract any relevant information.

The advantage of using this approach is that you have a lot of control over what information is retrieved and there are no limitations on the amount of results that can be retrieved or number of queries that can be made. But there are major downsides. First of all, you are dependent on the amount of the crawled data, which is often lower than that of the big commercial search engines. Secondly, computational power may be lacking to index the enormous amounts of data that is available. Lastly, the results of a search will most likely not be as good as the results of a big commercial search engine, as they constantly work on bettering their algorithms.

ChatNoir[3] is a freely accessible search engine that has indexed publicly available crawled datasets [19]. This search engine is especially interesting, as it offers unrestricted API access. This means no custom search engines has to be made. Although the performance of ChatNoir seems to be sufficient, a disadvantage is that the latest indexed data is from 2015.

3.2.2 Search engine APIs

An *Application Programming Interface* (API) provides an easy access point for one program to communicate with a another one [61]. Most commercially available search engines provide an API which offers a way to programmatically obtain search results in a structured way, bypassing the user interface.

Search engines like Google[4] and Bing[5] provide such APIs. This would be the ideal solution, as it gives the quality of the results of a commercial search engine, while offering it in a structured way. The problem is that most of the API's are either payed or very limited. For example, the Google API only retrieves the first 100 results of a query and allows a maximum of 100 API calls per day. One API call will always at maximum provide 10 results, which means 10 calls are needed to get the first 100 results of one query, effectively only allowing 100 results for 10 queries per day. After this, it will cost $5 per 1,000 API calls.

[2]https://commoncrawl.org/
[3]https://www.chatnoir.eu/
[4]https://developers.google.com/custom-search/v1/overview
[5]https://www.microsoft.com/en-us/bing/apis/bing-web-search-api

3.2.3 Web scraping

To programmatically get data from the internet, web pages can be scraped. This is the process of loading a web page and going through the HTML code to extract the needed data based on some search command [79]. Scraping a search engine can be done by performing a query, loading the results page, and extracting all URLs from the page.

Although this is a bit of a workaround compared to using the API, as a web scraper has to be built, the advantage is that there are no limitations in the number of results or requests that can be done. The downside is that a scraper may break when the search results page is altered. Furthermore, search engines, especially Google, do not want bots to scrape their search results. Therefore, if a scraper is detected, chances are that the IP address will be blocked. To counter this, proxy servers can be used, although the available free proxies have often already been blocked or are highly unreliable in general.

3.3 Snippet extraction

To extract relevant pieces of text from full articles simple word matching can be used. The problem is that it is now unclear what the boundaries of the piece of text should be. The size of a piece of text may be anything from a few words, to multiple paragraphs. It may help to figure out the structure of an article to find coherent pieces of text. Furthermore, apart from using word matching to find pieces of text, there may also be certain signal words that help in finding more relevant parts. Therefore, in the following sections a brief overview will be given on Rhetorical Structure Theory and Discourse markers.

3.3.1 Rhetorical Structure Theory

Just like sentences are not an arbitrary collection of words (see Section 3.1.2), neither are complete texts. *Rhetorical Structure Theory (RST)* is based on the idea that a text can be divided into non-overlapping parts (referred to as *text spans*) through a tree like structure starting at the full text and ending with parts of sentences (referred to as *Elementary Discourse Units (EDU)* [23]). Text spans have relations to each other which can be used to uncover the structure of a text. Examples of relations that can hold between two text spans, are for instance “evidence”, “summary”, or “contrast” [48].

To aid the development of automated RST parsing systems, Carlson, Marcu, and Okurowski [23] created the *RST Discourse Treebank (RST-DT)*. This is a dataset consisting of 385 articles from the Wall Street Journal that were taken from the Penn Treebank (see Section 3.1) and annotated with discourse structure as described in RST. In order to do so, the text needs to be split up in EDUs, which is the starting point of any RST parser [72].

Hou, Zhang, and Fei [34] divide the implementations of RST parsing systems into four approaches: cue phrase-based, rule-based, shallow machine learning-based, and deep learning-based. Some of the machine learning and deep learning-based approaches have been implemented into one system by Neumann [54], who has made

them publicly available via a website[6] and through the use of Docker containers.

3.3.2 Discourse markers

Cue phrases, from the cue phrase-based approaches (see Section 3.3.1), are defined as words that signal a relation between two text spans [34]. They are also referred to as *Discourse Markers* (DM) and multiple indexes of these kind of words have been created. Examples of DMs are "although", "because", and "nevertheless".

Although DMs explicitly express the relation between two text spans, the function of a single word may be ambiguous or vague [70]. This makes it hard to to simply apply DMs in tasks like RST parsing. In an effort to increase the accessibility of available lexicons from different languages, Connective-Lex[7] has been created. It provides a user interface that makes it easy to search for specific types of words in different languages and gives pointers to where the structured lexicons are stored.

3.4 Text summarization

The output of the snippet extraction component should be concise, as the goal of a single snippet is to give one possible explanation or piece of background. It might happen that long pieces of text are extracted anyway. Therefore, a sub-task of the snippet extraction is to summarize snippets deemed too long. This section gives a brief overview of the different methods of automatic text summarization.

There exist two main approaches to automatic text summarization: *extractive* and *abstractive*. Extractive summarization is based on extracting relevant sentences from a text in order to form a summary, while with abstractive summarization, new text is generated to create a summary of a larger text [6]. As may be apparent, extractive summarization methods are more straightforward to implement, as it does not require the complex task of text generation. For that reason, early summarization systems were mainly extractive. Abstractive summarization has especially risen to prominence with the coming of large language models [36].

A typical extractive summarization system consists of three main components. First, the text to be summarized needs some sort of abstract representation, which can for instance be done by looking at word frequency to determine important words. Next, using this representation, the sentences of a text are scored based on the occurrence of words in the sentence. Lastly, based on some threshold, a number of sentences are selected to create the summary [6]. An obvious problem with this type of summarization is that it does not necessarily result in a coherent story, as sentences from all over a document may be put together to form the summary [36].

Abstractive summarization consists of two main tasks. Like with extractive summarization, a structured representation of a text is needed. The second step is to use this representation to generate a summary using a natural language generation technique. Abstractive summarization may be done through graph-, rule- or template-based systems, but most recent work focuses on the use of deep-learning [36], as is the case with many tasks within NLP [46].

[6] https://rst-workbench.arne.cl/
[7] http://connective-lex.info/

3.5 Stance detection

The goal of the proposed pipeline is to extract text that not only provides some background to a claim, but moreover, explains why a claim is made. This could be factual background information, or assumptions and believes that underlie a claim. Most importantly, the goal of this study is not to retrieve counter arguments to get a broad overview of a discussion, but rather supporting evidence to gain a deeper understanding of a claim. Therefore, *stance detection* may be an interesting technique to add to the pipeline.

The goal of stance detection is to find out whether a piece of text is for or against a certain target, or neither [42]. Sometimes, apart from the for, against, and neither class, a neutral class is added [53]. In other systems, only the for and against class are used.

Earlier work on stance detection often used supervised machine learning for the classification task. For these implementations, large annotated datasets are needed. A widely used dataset is that of the SemEval-2016 task [52]. A problem with these systems is unseen topics. Generally, for every target, a classifier would be trained to predict the stance of a piece of text towards that target [43]. Unsupervised approaches have also been attempted, for which no annotated datasets are needed. Still, these implementations would often also be topic specific [3].

Ideally, a system would be topic-independent, but creating a topic-independent stance detection system has challenges, as the used vocabulary between topics can greatly differ [3]. Interesting results have been achieved using pre-trained transformer models like BERT, although there exist differences in performances on different topics [62].

3.6 Clustering

Chances are that multiple hundreds snippets are found in the first steps of the proposed pipeline. As the goal is to provide a few pieces of information on the given claim and topic, a means is needed to reduce the number of snippets. The assumption is made that there will be groups of snippets that contain roughly the same information, but it is not clear up front what this information will be, how many different subjects the set of snippets cover, and how much snippets will cover one subject. Therefore, *clustering* will be used.

The goal of clustering is to find groups of data in a multidimensional dataset based on similarity [57]. This similarity is calculated using some similarity measure like Euclidean distance or cosine similarity. Clustering algorithms can be grouped into two categories: hierarchical and partitional techniques [64].

Hierarchical clustering methods start out with all items of a dataset in the same cluster and split up the clusters until every item has their own separate cluster, or the other way around. This means there is no need to determine the number of clusters beforehand, but these algorithms are also computationally expensive. Partitional clustering techniques on the other hand divide the data into a given number of clusters using some criterion. Typically, a partitional algorithm randomly creates a given amount of *centroids* and then computes for every item in the dataset to which centroid it belongs. It then recalculates the cluster centroids until a stopping criterion is met. Partitional clustering techniques are generally more popular, from

which the k-means algorithm is most commonly used [57].

Determining the number of clusters beforehand may not be a problem, for instance if a dataset needs to be split up in a known number of categories. However, it may not always be clear beforehand how many clusters should be found in the data, for instance when exploring unseen data. Therefore, measures are needed to determine the number of clusters. Multiple variations of existing algorithms have been proposed that determine the optimal amount of clusters. However, instead of having to set a fixed number of clusters, these algorithms often require other parameters to be set [57].

Other methods to determine the optimal number of clusters rely on the algorithm being run multiple times with different cluster sizes and calculating some score for each run. A well known method is the *elbow method*, which calculates the distortion for a fixed number of runs and relies on there being a turning point in the fall of this distortion score. Plotting this in a graph, the turning point looks somewhat like an elbow. The problem is that this point is not always present [40].

In order to cluster text, an important step it to create a numeric representation of the text to be clustered [5]. This can for instance be done by using Bag-of-Words or Term-Frequency Inversed-Document-Frequency. Although these simpler methods preform well with larger texts, short texts may require a more sophisticated method that captures the semantics of the words as well, because similar short texts may use different words, which would be overlooked using the more shallow methods. Pre-trained language models may help in capturing these semantics [2].

Chapter 4

Implementation

In this chapter, the implementation of the different components of the pipeline is explained. Where applicable, the initial approach of the component is discussed including the reasons for changing it. The high-level overview of the pipeline from the introduction is provided again in Figure 4.1. In Chapter 3 some background on the different tasks that make up the components is given.

Figure 4.1: High-level overview of the proposed pipeline. It takes as input a claim, topic, and stance, and outputs text snippets.

The input of the pipeline is a claim, the topic of the claim, and the stance of the claim towards the topic. To illustrate every step of the pipeline, a running example is created using the following input:

Topic: Bombing Hiroshima and Nagasaki
Claim: Soviet victories in Manchuria were core cause of Japanese surrender.
Stance: CON

4.1 Phrase extraction

The first task of the pipeline is to extract relevant terms and phrases from the topic and claim which will be used in subsequent components. The input of this component is the topic and claim, and the output is two lists: one with words from the topic, and one with words and phrases from the claim. The extracted words will be used to search articles for relevant parts. Extracting these terms will be done with the help of POS tagging and dependency parsing. More specifically, spaCy will be used, which is an open source Python library that has pre-trained transformer models for the mentioned tasks[1]. Claims are often well-formed, complete sentences. A topic on the other hand may not be a full sentence and could in some cases be just one word. Therefore, two different methods are used to extract the terms.

[1]https://spacy.io/usage/v3

To decide which words of the topic are relevant, the function of the word in the sentence is determined by POS tagging all the words. By examining the explanations on POS tags provided by the Universal Dependencies project[2], a list was created with tags that a word must have to be extracted from the topic. Most importantly, words like "the", "a", and "on" should not be included. The list contains the following POS tags: adjective (ADJ), adverb (ADV), interjection (INTJ), noun (NOUN), numeral (NUM), pronoun (PRON), proper noun (PROPN), symbol (SYM), verb (VERB), or other (X). Table 4.1 shows the extraction and selection for the example topic.

Table 4.1: All words from the topic together with the POS tag and whether the word is selected based on this POS tag. POS tagging was done using spaCy.

Word	POS-tag	Selected
Bombing	PROPN	✓
Hiroshima	PROPN	✓
and	CCONJ	✗
Nagasaki	PROPN	✓

To get relevant phrases from the claim, *noun chunks* are extracted. Noun chunks can be described as a noun together with the words describing a noun. A noun chunk can consist of one or multiple words. For the example claim, the noun chunks are "Societ victories", "Manchuria", "core cause", and "Japanese surrender".

But when using only the noun chunks, not all useful parts of the claim may be extracted. For instance, the *root* of a sentence may convey an action that is important to the context of the claim. Furthermore, a noun chunk may consist of three or more words which hold multiple other phrases that make perfect sense to look for in an article. Therefore, phrases are added based on two more criteria. First, if the root of the sentence has a POS tag that appears in the POS tag list as described above, it is added to the list of extracted noun chunks. For the second check, a list was created consisting of dependencies by reading the descriptions as provided by the Universal Dependencies project[3] as well as by going through different claims and examining the output. If a word in the claim has a dependency that is present in the list, it is added to the list of extracted noun chunks, and if applicable, the root. The selected dependencies are adverbial modifier (advmod), adjectival modifier (amod), compound (compound), and prhasal verb particle (prt). The main criterion for choosing a dependency was that the two words combined should make a coherent phrase that may help in searching an article for relevant parts.

To illustrate the added steps, take the following claim as an example:

Universal health care systems incentivize improving patient health.

When only considering the noun chunks, the following phrases are extracted: "Universal health care systems" and "patient health". The first noun chunk consists of four words, which may contain other valuable phrases that help in extracting

[2]https://universaldependencies.org/u/pos/
[3]https://universaldependencies.org/u/dep/

snippets. In this case, the extra step extracts the phrases "universal systems", "health care", and "care systems", which are all relevant in the context of this claim. Table 4.2 shows for every word in the claim its dependency relation, the head of the relation, and whether the phrase is selected based on the criteria. The phrase "patient health" is also selected, but this one was already extracted as a noun chunk. Furthermore, the word "incentivize" is the root of the sentence, which signals an action that is at the core of the claim, it would make sense to also use this word for extracting text from articles. As the POS tag of the root is "VERB", which is present in the POS tag list, it can be added to the extracted noun chunks.

Table 4.2: The dependency relation of all words from the claim together with the head of the dependency and whether it is chosen based on the criteria

Word	Dependency	Head	Selected
Universal	amod	systems	✓
health	compound	car	✓
care	compound	systems	✓
systems	nsubj	incentivize	✗
incentivize	ROOT	incentivize	✗
improving	xcomp	incentivize	✗
patient	amod	health	✓
health	dobj	improving	✗
.	punct	incentivize	✗

In the process of extracting words and phrases from the topic and claim, steps are taken to prevent that any words that have nothing to do with the actual content of the topic and claim are extracted. This mostly concerns so called *stop words*, words that have no significance to the meaning of a sentence [9]. As a final step in the component of entity extraction, any stop words that have somehow slipped in anyway, will be removed. This is done by using the stop words list from the NLTK[4] python library. Although it was found that stop words are rarely extracted, this step is still crucial, as using stop words in the snippet extraction phase greatly changes the type and number of extracted snippets.

Initial approach For the claim, only specific noun chunks were extracted. A list with relevant dependencies was created and only noun chunks that had one of these dependencies were used. The dependency relations were chosen based on their description on the Universal Dependencies website and a qualitative assessment of the extracted phrases. After a more elaborate analysis of the extracted noun chunks, it was found that not enough important phrases were extracted. Therefore, all noun chunks were extracted and a new list of dependencies was created to extract even more relevant phrases. This also resulted in the addition of the stop word removal step, as in some cases stop words were extracted as well.

[4]https://www.nltk.org/

4.2 Web querying

In this component, a web search engine is used to scrape articles from the internet. The input of this component is the topic and the claim, and the output is a list of web links. The links will be used to extract snippets from in the snippet extraction component. The used search engine is Bing[5].

In order to scrape URLs, a search query has to be created. This is done by combining the topic and the claim and appending "language:en" at the end of the query, which should ensure that all retrieved articles are in English[6]. Using the example topic and claim, the query looks like this:

> bombing hiroshima and nagasaki soviet victories in manchuria were core cause of japanese surrender language:en

It is possible that the claim contains words that are also found in the topic. In these cases, the double words are deleted from the topic while the claim stays the same. It was chosen to do this as the claim often forms a complete sentence, while the topic is just one or a few words. Therefore, it seemed better to keep the claim intact.

Using the created query, the results page of Bing is scraped. Bing retrieves ten results per page, this means that for every ten results, a request has to be made. Ideally, all results retrieved by Bing for a given query should be extracted. But not only will this take a lot of time in this step, every following step will also take considerably more time. Although we are not concerned with the efficiency of the pipeline, there are certain boundaries. In the end it was decided to scrape the results of 25 pages, which should lead to the extraction of 250 links. The number of 25 was chosen rather arbitrarily, and could have also been 20 or 30. The results page is then scanned for every element with the class called "b_algo", which in Bing holds one result[7]. From this result, the URL is extracted and added to a list with every unique URL found in any of the results pages.

Even though 250 URLs are expected to be found, the scraper never actually retrieves all of them. This may have to do with the fact that there are duplicate links, which are not stored, but more likely it has to do with the fact that scraping a search engine is highly unreliable. If the URL list did contain 250 results, these would all be "Bing URLs", which are URLs that lead to a Bing page before redirecting to the actual result. Although all of these links would be unique, they often all lead to the same few results.

It was found that running the scraper again and again with the same query, would eventually lead to getting URLs, even if no results were found the first few times. Therefore, if less than 25 or exactly 250 articles are found, the scraper runs again until it retrieves anything between 25 and 250 results. This will ensure that the next steps can be executed.

[5]https://www.bing.com/

[6]https://support.microsoft.com/en-us/topic/advanced-search-keywords-ea595928-5d63-4a0b-9c6b-0b769865e78a

[7]This is currently (January 2023) the case, but may change in the future, if Bing decides to update their results page and change class names.

Initial approach Besides scraping, other approaches have been considered. These are explained in Section 3.2 together with their advantages and disadvantages. But also within the scraping approach some changes were made. At first, the goal was to scrape Google. But is was soon found that Google actively monitors for scraping bots, and blocks IP addresses. To bypass this, free proxy servers were tried. A list of free proxy servers would be scraped from a website and the scraper would rotate through these proxies so that every request would come from a different IP address. Unfortunately, Google had already blocked most of these proxies, which made them useless. Therefore, the final implementation was built using Bing.

The creation of the query has also changed. In first instance, it was decided to use advanced search operators, with which one has more control over the retrieved web pages. The idea was to use the words and phrases extracted in the entity extraction component to retrieve articles that contained exactly these words. The example topic and claim would produce the following query:

> "bombing" AND "hiroshima" AND "nagasaki" AND ("soviet victories" OR "manchuria" OR "japanese surrender")

Having the terms between quotation marks and using search operators like "AND" and "OR" would make sure that every retrieved article contains exactly all words from the topic, plus at least one phrase from the claim. This idea was later dropped, as Bing does not support these operators. Furthermore, search engines already do a lot of processing of the query, so it is unnecessary to do the processing yourself. Only the language constraint has been added to the query, as only English text should be retrieved, although it is unclear to which extent Bing supports this[8].

4.3 Snippet extraction

In this component, snippets are extracted from the web pages found in the previous component. The component takes as input a list of URLs (from the web querying component), a list of words from the topic, and a list of phrases from the claim (both from the phrase extraction component). First, a basic web scraper is used that extracts all the text from a web page, after which an algorithm goes over all the sentences of an article to find relevant pieces of text.

The scraper takes a URL of the list and retrieves the HTML of the web page. It then extracts all the text inside the HTML p element. Every block is considered to be a paragraph. This is important, as the algorithm created for snippet extraction uses paragraph boundaries to determine the length of a snippet. An article is stored as a list of paragraphs with each paragraph containing a list of sentences.

As said, only the text between p elements will be extracted. A problem with this approach is that there may be text on the web page that is not extracted, since it is not between a p element. However, it was found most text in HTML is written within a p block. Furthermore, it also means no headings are extracted, as these are in heading blocks (h1 to h6). Since the aim is to find coherent pieces of text, headings are not relevant, therefore this is a good thing.

[8]Bing claims to support advanced search queries, but the results tell otherwise.

But apart from missing certain text on a page, another problem is that too much may be extracted. Since all text within p elements is extracted, this means text from for instance the footer of a web page may also be extracted, which mostly holds general information about a website and is often irrelevant to the topic and claim. However, during the extraction of the snippets, the algorithm should ensure no irrelevant text is extracted from the articles, so this should not have to be a problem.

Using the retrieved content from a web page, the algorithm will go over the text and extract pieces of text based on a few rules. The most important factor in the extraction of a snippet is a phrase from the list of claim phrases (from here on referred to as *claim phrase*). If a sentence contains a claim phrase, the sentence including any surrounding sentences will no matter what be extracted as a snippet. The size of the snippet depends on the length of the paragraph. If the paragraph consists of just one sentence, the last sentence from the previous and the first sentence from the next paragraph will be combined to form a snippet. If the paragraph has a maximum of four sentences, the complete paragraph will be extracted as a snippet. In case the paragraph has more than four sentences, the sentence containing the claim phrase plus two surrounding sentences in the same paragraph will be extracted. If a word from the topic list (from here on refered to as *topic word*) is in the proximity of the sentence with a claim phrase, a snippet will be formed from the sentence with the topic word to the sentence containing the claim phrase. Otherwise, the previous and next sentence are chosen.

In order to extract more snippets that may be relevant to the topic and claim, without them containing any words from either, discourse markers are used. If the beginning of the opening sentence of a paragraph that comes directly after a snippet contains a discourse marker, a snippet will be created with this sentence. The idea is that this discourse marker signals that something is being written that somehow relates to the previous part of the text. The used discourse markers were taken from the Connective-Lex website[9]. Here, a collection of discourse markers is given for different languages [70]. The used lexicon was presented by Das et al. [26] who made a structured version publicly available[10]. This file makes it possible to effortlessly traverse the different types of discourse markers, although in this study no distinction is made between the different types.

Initial approach In the first implementation, the retrieved articles would be stored as a list of sentences, with disregard of paragraphs. Snippets would be extracted based on the proximity of a sentence with a claim phrase to a sentence with a topic word. Only if these sentences were near each other, a snippet would be extracted. The idea was that this would ensure that the snippet was relevant to the claim in relation to the topic. A dataset was created for testing using the example claim and topic. After examining the output of this first implementation, a few interesting observations were made.

First of all, it was found that topic words occurred considerably more often in the articles than claim phrases. In the used dataset, an article contained on average 4 sentences with a claim phrase, while having 47 sentences with topic words. This makes sense, as words from the topic are often more general while phrase from the

[9] http://connective-lex.info/

[10] https://github.com/discourse-lab/en_dimlex/blob/master/en_dimlex.xml

claim are more specific. Furthermore it was found that sentences with claim phrases often held more important information regarding the claim and topic than sentences with topic words. Because of these observations, it was decided that sentences with claim words should always be extracted while putting less emphasis on sentences with topic words.

Secondly, the decision on snippet boundaries was rather arbitrary. This could be seen in the extracted snippets, that sometimes seemed to miss sentences to make a point, or had extra sentences that did not add to the main point of the snippet. Therefore it was decided to take into account paragraph boundaries in the extraction of snippets. This may help in extracting more coherent snippets.

Lastly, through examination of the text surrounding an extracted snippet, it was determined that sentences and paragraphs following a snippet often held relevant information to the claim. To better understand how the different parts of a text relate, it was attempted to use RST parsing (see Section 3.3.1). Unfortunately, existing systems for RST parsing were not as readily available as hoped. Still this theory contributed to a change in the algorithm, as discourse markers (see Section 3.3.2) were now added to extract extra snippets.

4.3.1 Summarization

This is a sub task of the snippet extraction component. Because of the used methods to extract snippets, it may happen that a snippet becomes quite long. This is unwanted, as the goal is to have a short statement as explanation as opposed a long paragraph. Therefore, as part of the extraction, snippets that are deemed too long will be summarized. A brief background on summarization can be found in Section 3.4.

The length of a snippet is measured in sentences and a threshold was set by manually going through the retrieved snippets. A snippet is deemed too long if it has more than seven sentences. A main problem with this measure is the variability in sentence length. A snippet may consist of ten short sentences or five lengthy ones. Furthermore, the seven sentence mark is chosen rather arbitrarily, and could just as well be six or eight. In future implementations, a different measure could be used.

If a snippet has more than seven sentences, it will be summarized. To do this, an implementation from the Hugging Face[11] library is used. Hugging Face has brought together all kinds of different models and datasets and made them easily accessible. Pre-trained models can be loaded for different tasks. For summarization, the T5 model has been used as proposed by Raffel et al. [59]. This model has been trained to perform a number of NLP tasks, among which summarization. The input snippet is tokenized using the same model.

4.4 Stance detection

In this component, the stance of a snippet towards the give topic is determined. Only snippets that hold the same stance towards the topic as the claim will be selected and used in subsequent steps.

[11] https://huggingface.co/

The DeBERTa large model was used for stance detection. DeBERTa is based on BERT [33]. The model was trained on a dataset scraped from Debatepedia, as described in Kobbe, Hulpuș, and Stuckenschmidt [38]. The dataset consists of claims with a corresponding topic and the stance of the claim towards the topic. The system predicts whether a given input text is in favor of (PRO) or against (CON) a given topic.

4.5 Snippet ranking

In this component, the snippets are clustered and ranked, as the execution of the previous components may result in a list of multiple hundreds of snippets. In the end, the goal is to have a few pieces of text that provide an explanation. Therefore, the snippets will be grouped using a clustering technique and ranked based on the size of the cluster. In order to apply a clustering technique , a numeric representation of the snippets is needed.

The snippets are cleaned by making the words lowercase, removing punctuation and digits, stemming the words, and removing stop words. As the snippets are extracted based on certain words and phrases, these words and phrases will be present in most of the snippets. This will probably not help with the clustering, and therefore, these words and phrases are also removed. The words that are left are tokenized using the tokenizer from the NLTK library and subsequently vectorized. The vectorization is done with a pre-trained Word2Vec model from the Gensim library[12].

The vectorized snippets are provided as input to a k-means clustering algorithm from the scikit learn library[13]. It is not clear in advance how many clusters of snippets will be found. Therefore, the optimal number of clusters has to be established first. This is done by running the algorithm multiple times, starting with two clusters, and ending with twenty. Of every run, the distortion is stored which is used to perform an automated version of the elbow method. Once the optimal number of cluster (the optimal K) is found, clustering is performed one more time using this K.

The snippet closest to the centroid of a cluster is chosen to represent that cluster. This leaves us with a number of snippets equal to the number of clusters. These are subsequently ranked based on the size of the cluster. Two assumptions are made in this step: snippets from the same cluster cover roughly the same subject, and a subject of which more snippets are found is more important than a subject of which less snippets are found.

A problem that was found when selecting the snippet closest to every centroid, is that this snippet does not necessarily belong to the cluster to which the centroid belongs. This sometimes results in the same snippet being selected for two clusters.

Initial approach To create a numeric representation of the snippet, *Term Frequency-Inverse Document Frequency (TF-IDF)* has been tried. This method seemed to shallow to really capture the semantics of the snippets, therefore in the final implementation, a pre-trained Word2Vec model was used.

[12] https://radimrehurek.com/gensim/models/word2vec.html
[13] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Chapter 5

Evaluation

In this chapter, the evaluation of the pipeline will be discussed. To evaluate the performance of the pipeline, the output will be compared to a baseline. The pipeline will be used to gather snippets for a number of claim - topic pairs. For every snippet, a baseline will be created for comparison. The task of the annotators will be to, given a claim - topic pair, choose whether the pipeline snippet or the baseline explains the claim and topic best. First, the setup of the evaluation will be explained, after which the results are analysed. The chapter concludes with a discussion of the results.

5.1 Set up

In this section, the set up of the user study is described. As the output of the pipeline will be compared to a baseline, it is first explained how the baseline is created. Then, the process of gathering all necessary data is outlined, and the section ends with a description of the website used in the study.

5.1.1 The baseline

To gather snippets, the pipeline uses a web search engine (in this case Bing) to find articles, which is queried using the topic and the claim. To create the baseline snippets with which the pipeline snippets will be compared, the same query is used on Bing. But, instead of extracting text from as many articles as possible and going through the text using specific terms and signal words to find relevant pieces, only the first search page is queried and the preview text that is visible on the overview page is used to create snippets. An example of which texts will be extracted can be seen in Figure 5.1. As these preview texts are often not complete sentences, the full text from the article is used to find the paragraph from which the preview text is taken. The surrounding sentences are then added to create a baseline snippet.

5.1.2 Data gathering

To gather the snippets and baseline for the annotation study, claim - topic pairs are needed together with the stance of the claim towards the topic, as this is the input for the pipeline. To this end, the dataset[1] created by Kobbe, Hulpuș, and Stuckenschmidt [38] was used. This dataset consists of arguments scraped from the

[1] https://github.com/dwslab/StArCon/blob/master/data/datasets/debate.txt

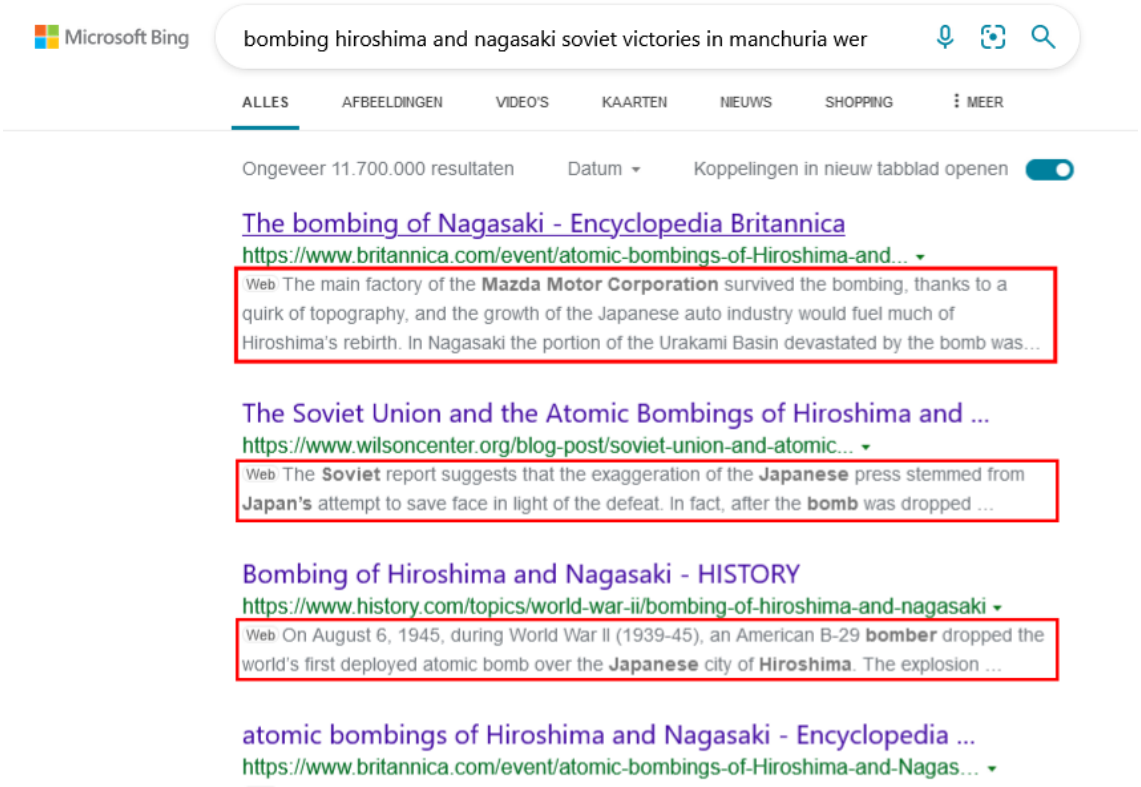


Figure 5.1: Screen capture of the Bing search page using the query based on the example topic and claim. The text extracted for the baseline is outlined in red.

debating platform Debatepedia, which has since gone offline. A selection of the complete dataset was made by manually filtering the claim - topic pairs based on a few loosely defined criteria.

Criteria for dropping a claim:

- **Claim and topic are only relevant for United States**. Most annotators will not have a lot of experience with it. Although annotators do not need to have any previous knowledge on the claim and topic, it does make the process more interesting if a claim / topic is understood.
- **Claim is too general or obvious**. In this case background information is not needed or cannot be found because of the vagueness of the remark.
- **Claim does not provide any information and is purely emotional**. Although emotional claims may be good candidates for explanations on why someone feels that way, there are claims that solely rely on emotion and provide no extra information.
- **Claim is generally not good**. This is a tricky criterion and should be used with caution so personal preferences do not interfere, but some claims cannot be taken seriously.
- **Claim starts with "General statements in favor of/against"**. This is noise in the dataset because of the means of scraping, as it is not a claim in itself, but a collection of statements/claims.

Example of claim that is both focused on US and is too general:
Topic: Pickens US energy plan
Claim: The Pickens Plan is generally not viable.

Example of a claim that does not provide any information and is purely emotional:
Topic: Assisted suicide
Claim: Most people that believe euthanasia is wrong never sat by a bed side

Example of generally bad claim:
Topic: European Union Expansion
Claim: European Union should include all of Europe; it's in their name.

Furthermore, in this study we are more interested in bigger societal and ideological topics. Therefore all claims of certain topics are dropped, like the topic: "Banning vuvuzela horns at the 2010 World Cup". A random selection from the remaining topic - claim pairs was used for the annotation study.

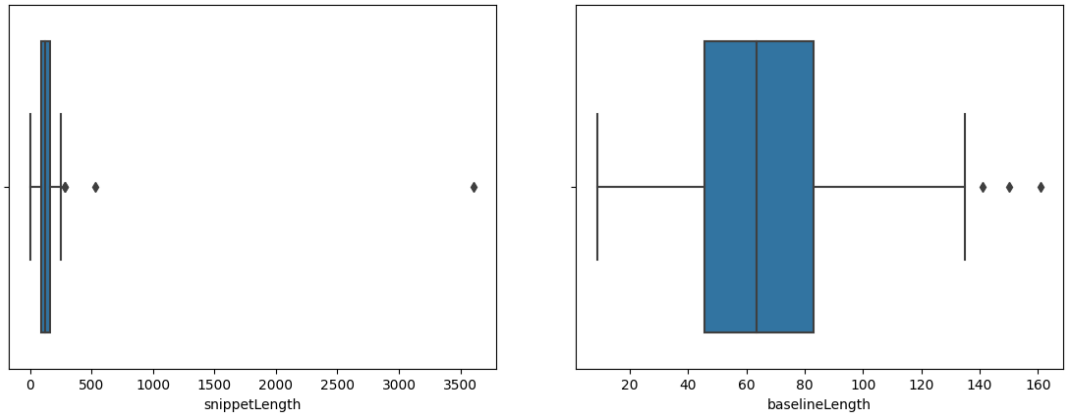
The pipeline and baseline snippets are gathered by randomly selecting claims from the filtered dataset. The system was prepared in such a way that it would extract snippets as described in Chapter 4 and a baseline as described in Section 5.1.1. For every randomly selected claim, multiple snippets are gathered as well as multiple baselines. The top three snippets are paired with the three top results from Bing and stored in a database together with the topic and claim.

In the end, 49 unique claims were selected covering 38 topics, for which a total of 144 snippet - baseline pairs were gathered. A quick examination of the length of the snippets (in words) showed that there was one extreme outlier as can be seen in Figure 5.2a. Upon closer examination it was found that this snippet did not contain any punctuation, therefore it was treated as one sentence. The baseline does not have these extreme outliers as is visible in Figure 5.2b. In the boxplot of the snippets, more outliers can be seen, which are still quite far out, even though the most extreme outlier makes it seem like they are not to far off. Therefore it was decided to drop the most extreme outliers from the snippets by calculating the interquartile range and setting the boundaries at 1.5 times the upper and lower quartile [75]. The pipeline snippets are generally longer than the baseline snippets, with an average of respectively 123 and 69 words.

5.1.3 The annotation site

The snippet - baseline pairs that have been collected as described in the previous section are used for the annotation study. All information is stored in a database consisting of the tables "snippets", "annotators", and "annotations". The snippets table contains for every snippet - baseline pair a unique ID, the text of the snippet, the text of the baseline, the topic, and the claim. The annotator table is made of just one column which is the unique annotator ID. The annotations table consists of a unique annotation ID, an annotator ID which refers to the annotator table, a snippet ID which refers to the snippet table, and the choice that the annotator made for that snippet.

This database is used for the website that has been created for the annotation study. When a user starts annotating, a cookie is set on the device of the user



(a) Snippets (b) Baseline

Figure 5.2: Boxplot figures of the length (in words) of the gathered snippets and baseline for evaluation

containing a unique ID. This unique ID is stored in the annotator table. This way, if someone decides to take a break from annotating snippets and closes the website, this person can come back to the website anytime without problems. The system will still know which snippets have been annotated by this user, which is necessary for the evaluation and ensures that someone does not get the same item twice. It was chosen to use cookies instead of for instance creating an account to make the process as convenient as possible for the user. A potential problem with this approach is that one person may use different devices or that a browser is used that does not store cookies.

Before a cookie is created, the user first arrives at the homepage, which gives a detailed description of the task. The homepage can be seen in Figure 5.3 and the full description can also be read in Appendix B. A short description of the task is given in the next paragraph.

During the annotation, a random snippet is selected from the database and presented to the annotator. The screen displays the topic and claim, followed by two pieces of text that might give extra information on the claim, as can be seen in Figure 5.4. The user has to select which text is deemed a better explanation. It may be that both explanations do not provide any relevant information, in this case the "None" option could be chosen. Another option that is not one of the two explanations is the "Skip" option. This option should be chosen if both are too similar or provide equally good information. The choice of the annotator is thus either snippet, baseline, skip, or none. This is stored in the annotations table.

5.2 Results

In this section, the results from the annotation study will be discussed. The annotation website has been open from Monday the 23rd of January until Sunday the 29th 2023. As no specific knowledge was required to perform the task, anyone with a sufficient knowledge of English could participate. Therefore, the link to the site has been sent out to friends, family, and colleagues. In the following section an overview of the gathered data will be provided, after which the inter annotator agreement

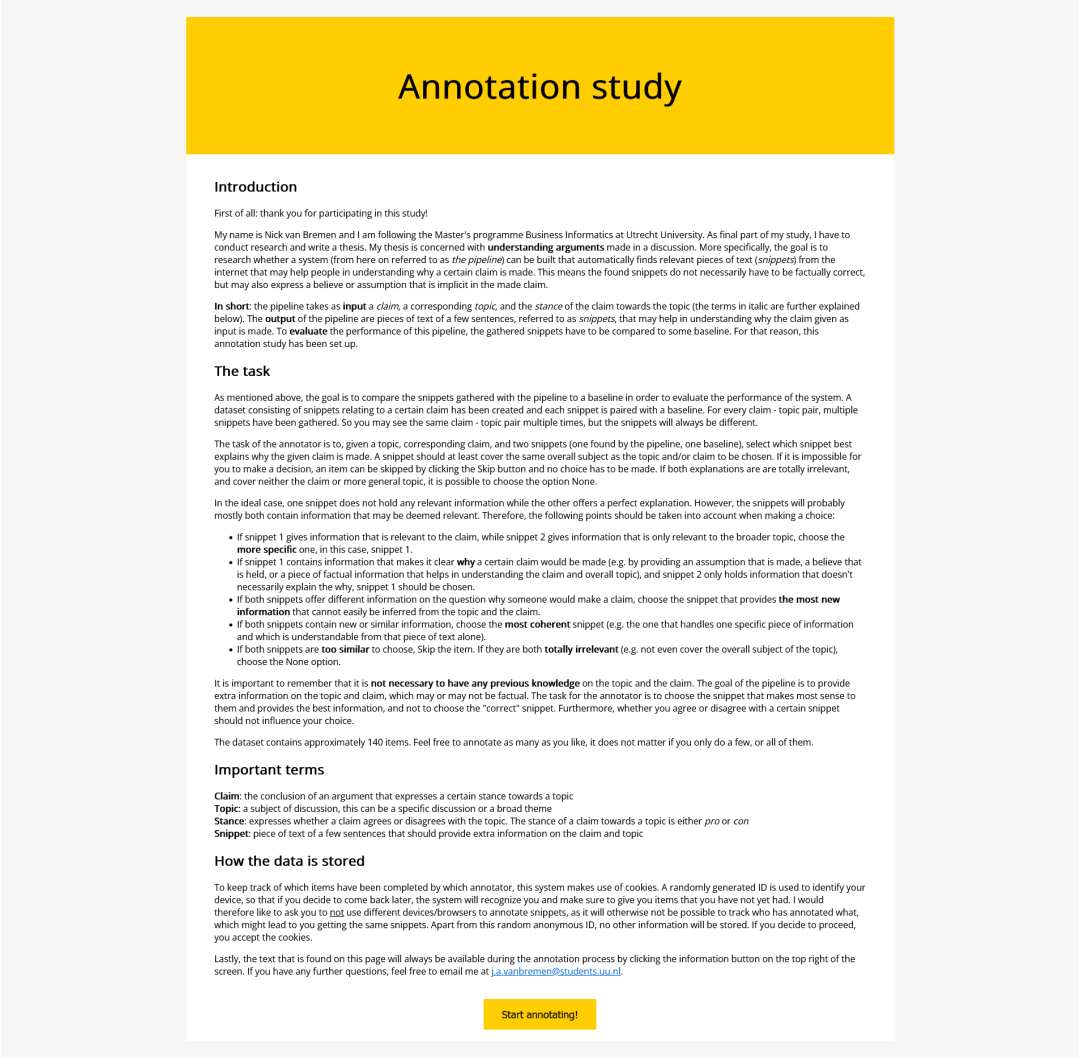


Figure 5.3: Screen capture of the home page of the annotation website with the explanation of the project and task.

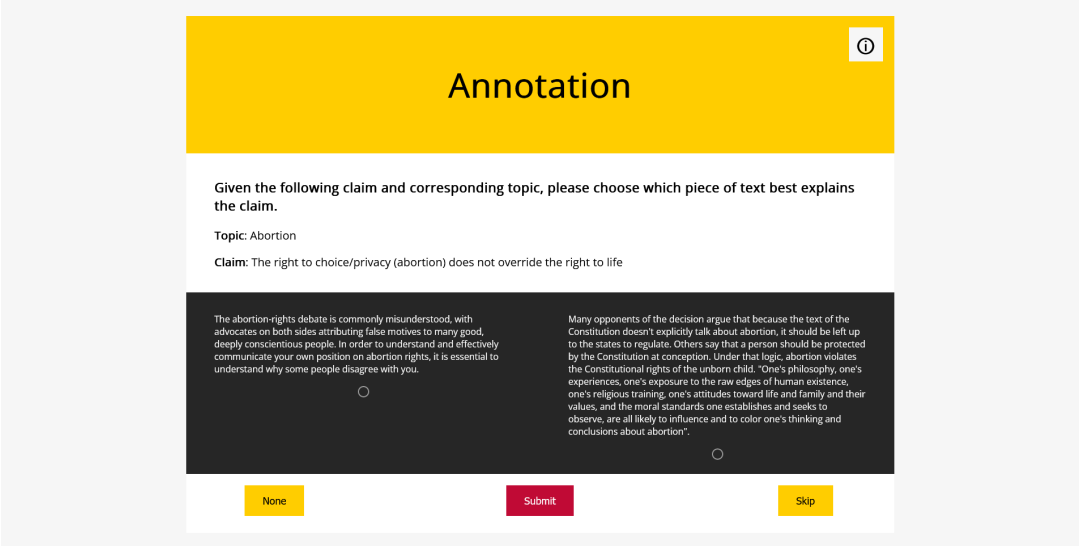


Figure 5.4: Screen capture of an annotation page containing a topic, claim, and two snippets to choose from.

will be given and the final results of the annotation itself will be discussed.

5.2.1 Descriptive statistics

In total, 138 annotations have been performed by 14 annotators, giving an average of around 10 annotations per annotator. The minimum number of annotations given was 1, which occurred only once, and the highest number was 31. The median lies at 6.5.

The 138 annotations cover a total of 47 items of which 44 received at least three annotations. The four choices were snippet, baseline, skip, and none. Snippet was chosen 59 times, baseline 64 times, skip 6 times, and none 9 times. In total, there were 140 items in the database. It was decided that at least 25 items had to be annotated at least three times. The system was designed in such a way that it would randomly pick an item from the database out of the first 25 items that had not yet been annotated by the current annotator and that had less than three annotations. If no snippets fulfilled these requirements, the first snippet in the database that did not yet have three annotations would be presented to the annotator. This ensures every snippet gets exactly three annotations.

However, it was found that three snippets had a total of four annotations, which should not have been possible given the constraints. The most likely explanation is that two people were performing the task at the same time while the first 25 items had already been finished. This would result in them getting the same item at the same time, which in both cases passed the criteria, because both had not yet submitted their choice.

5.2.2 Inter-annotator agreement

Before proceeding to the analysis on the performance of the pipeline, the inter-annotator agreement should first be established. This measure helps in understanding to what extent different annotators agree with each other. A high agreement among annotators gives more confidence in the correctness of the made annotations.

There are different approaches to calculating the inter-annotator agreement. Cohen's κ is an often used approach, but it can only be used to calculate the agreement between two annotators. In this study, multiple annotators are used that did not all annotate every single item, instead, every item has been annotated a fixed number of times. Therefore, Fleiss' κ is used, which calculates the agreement between multiple annotators [11, 27]. All items have to be annotated the same number of times, this is mostly the case, except for the earlier mentioned items that were annotated four times. As a result, the last annotation of these items has been removed, so they also have three annotations.

The value of Fleiss' κ ranges from -1 to 1 and is determined by calculating how the given annotations differ from chance. A score of exactly 1 means that their is full agreement on every annotation while a score of 0 means it is no different than chance. A score of -1 signals complete disagreement between the annotators.

The Fleiss' κ value for this study is 0.06. This is a poor score, meaning the annotators did not often agree. There may be multiple reasons for such a low score. It may be the case that the instructions were not clear or that the task was ambiguous. Furthermore, the content of the topic, claim, and explanations may have

influenced the choice made by an annotator. Although it was explicitly mentioned that this should not influence the choice, it may still have been hard at times to totally set it aside.

It may also be the case that there are outliers in the group of annotators. To further investigate this, Fleiss' κ will be calculated for multiple combinations of annotators, with the condition that there should always be at least 25 items with three annotations. Given this constraint, there are exactly 600 possible combinations of annotators, but none of these combinations have a sufficient inter-annotator agreement. The highest score is 0.17, which is still poor, which indicates that there is absolutely no consensus on the given annotations.

5.2.3 Performance

As the inter-annotator agreement is low, no good quantitative analysis can be performed with the data. Therefore, a qualitative assessment will be done on the snippets that have been gathered through the pipeline. This has given some insight on the different types of snippets that were extracted, and six different types have been defined. Using these categories, the results of the user study will be examined to get an idea of what choices have been made in which category and how the baseline may have affected the annotation.

Snippet assesment

Examining the snippets that have been annotated at least three times (44 in total) gave some interesting insights. Different types of snippets have been extracted and there are different levels at which a snippet can be evaluated. Only around eight snippets can be considered good explanations, as they provide new information that further explains the given claim. Other snippets might help somewhat in understanding the claim or topic, but can not be considered great explanations. The different categories that have been defined can be found in Table 5.1, together with the number of snippets belonging to that category.

Category	Number of snippets
Unrelated	3
Roughly related to topic	9
Opposing stance	4
General background	11
Same stance	9
Good explanation	8

Table 5.1: Different types of snippets found during the qualitative assessment including the number of snippets found per category.

In the remainder of this section, we will go over the different types of snippets found, including examples of the type of snippet. In the worst case, a snippet is **unrelated** and tells absolutely nothing about the topic or claim. Take the following example:

Topic: Driftnet ban
Claim: An effective driftnet ban must go beyond international waters into countries' Exclusive Economic Zones.
Snippet: Well our Gate1 16 day romantic Amalfi with lake region and Rome was fantastic. Firstly, our tour guide Remo was so good and we had so much fun with him. His knowledge of geography and history was incredible. I certainly would like to have him as a tour guide again on another Gate1 trip sometime. Our hotels were first class, the bus drivers were all very professional, our meals included were very good and all the sights we saw were very enjoyable. You can never get enough of Rome.

This snippet is totally unrelated to the claim and even to the broader topic. In the studied snippets, this happened three times. Apart from these, snippets generally covered at least the topic. Although, sometimes a snippet would not cover the given topic per se, but a related or broader topic. Take the following snippet that is **roughly related to the topic**:

Topic: Greece bailout
Claim: Bailing out Greece is a necessary evil
Snippet: With regards to Europe, Roubini predicted that Italy, and possibly a series of other eurozone countries (Portugal, Spain, Greece) might have to exit the eurozone if they did not implement "serious economic reforms". "[It] is not a foregone conclusion but, if Italy does not reform, an exit from EMU within 5 years is not totally unlikely. Indeed, like Argentina, Italy faces a growing competitiveness loss given an increasingly overvalued currency and the risk of falling exports and growing current account deficit.

This snippet is concerned with a topic related to the bailout of Greece, namely the economic situation of Italy and their position in the eurozone. However, it does not provide any extra information on the bailout of Greece or why it is a necessary evil. In a few instances a position is taken in the snippet, but it has an **opposing stance** to that of the claim. In these cases, either some background is given that argues for an opposing point, or a counter argument is presented. Take the following example:

Topic: Polygamy
Claim: Recognizing polygamy would cause a host of legal problems.
Snippet: Polyamory is a fact . People are living in group relationships today. The question is not whether they will continue on in those relationships. The question is whether we will grant to them the same basic recognition we grant to other adults: that love makes marriage, and that the right to marry is exactly that, a right.

The snippet does not necessarily counter the given claim, but it does give an argument with an opposing view. In the following example, the snippet provides a brief history of the Falkland Islands, which is some **general background** to the topic, but does not explain anything about the affect of the Nookta Sound convention on Britains rights to the islands:

Topic: Falkland Islands, return of
Claim: The Nookta Sound convention did not affect Britains rights to the islands as they already had an agreement with Spain where both nations rights were secured in 1771
Snippet: Controversy exists over the Falklands' discovery and subsequent colonisation by Europeans. At various times the islands have had French, British, Spanish, and Argentine settlements. Britain reasserted its rule in 1833, but Argentina maintains its claim to the islands. In April 1982 Argentine military forces invaded the islands. British administration was restored two months later at the end of the Falklands War.

The background given in this snippet is factual and does not take a stance. However, various times an argument has been extracted that holds the **same stance** as the claim, but gives a different perspective. In the following example, the snippet is also in favour of phasing out fossil fuel subsidies, but for different reasons than the claim:

Topic: Phasing out fossil fuel subsidies
Claim: Ending oil subsidies is good for air quality, human health.
Snippet: Supporters of oil subsidies contend that oil subsidies are necessary because clean energy is not yet viable and the economy remains dependent on oil. Yet, it is precisely these subsidies that make it impossible for renewable energy to compete and perpetuate the dependence on oil. Subsidies are, therefore, a self-fulfilling-prophecy in this regard.

The snippet gives extra reasons as to why fossil fuel subsidies should be phased out, but does not provide background on the claim. Lastly, some retrieved snippets do provide **good explanations** for the claim and contribute to a better understanding of the claim and topic. In the following example, the snippet exactly aligns with the claim and provides background information to back up the claim. This is a good example of what the system is expected to retrieve.

Topic: Gene patents
Claim: Gene patents do not incentivize innovation
Snippet: Staunch defenders of gene patents have also argued that the basic tenet of the patent system to require disclosure of the invention serves to promote follow-on research post initial gene discovery. 34 However, a rigorous probing of this question indicated that rather than promoting research, gene patents have an inhibitory effect on future knowledge production. Specifically, the study, conducted by Fiona Murray and Kenneth Huang at MIT, examined more than 1,000 gene discoveries and found that follow-on genetic researchers forego approximately one in ten research projects because of the causal impact of gene patents. 35 Moreover, this trend was found to be exacerbated in situations when patents are broad in scope, privately owned, or where the patented genes are closely linked to human disease, and especially cancer.

Although multiple snippets give explanations like this, it is not sufficient to conclude that the pipeline performs well. There are also some other general problems with the retrieved texts. One problem, that can be seen in the last example, is

that text from the internet may contain certain formatting that does not add to the actual content of the text. In the last example, these are the number 34 and 35 that seem to appear randomly in the text. A quick look at the web page where the snippet was found revealed that these are links that show a popup with extra information. The following example comes from a web page that is formatted as a quiz, containing different cards with questions and answers. This creates an incoherent snippet:

Topic: DC handgun ban
Claim: If the 2nd amendment was to protect an individual right it would have clearly expressed it
Snippet: Heller, case in which the U.S. Supreme Court on June 26, 2008, held (54) that the Second Amendment guarantees an individual right to possess firearms independent of service in a state militia and to use firearms for traditionally lawful purposes, including self-defense within the home. What is the significance of the Supreme Court's District of Columbia v Heller 2008 ruling quizlet? Heller, 554 U.S. 570 (2008), was a landmark case in which the Supreme Court of the United States held in a 5-4 decision that the Second Amendment to the United States Constitution applies to federal enclaves and protects an individual's right to possess a firearm for traditionally lawful purposes, such as self-defense ... How did District of Columbia v Heller impact states rights quizlet? Ruling: Yes. The Court held that the Second Amendment protects an individual right to possess a firearm unconnected with service in a militia, and to use that firearm for traditionally lawful purposes, such as self- defense within the home. Which of the following explains the outcome and significance of the District of Columbia vs Heller case?

Apart from the formatting, snippets may also be hard to understand because they are taken out of context. In some cases, the text refers to something mentioned earlier or the snippet starts at an odd point, although this seems to have improved compared to the version of the system where paragraphs were not taken into account.

User study results

Looking at the given annotations, the authors fully agreed in eleven instances. It is interesting to see that the agreement would only be on baseline or snippet, and never on none or skip. In eight instances, all three annotators chose a different option. The other 25 times the choice was two-to-one. Based on the rough classification presented in the previous section, the choices made in the user study will be examined. Mainly, it will be reviewed in what cases the pipeline snippet has been chosen compared to the baseline snippet. Although it may be expected that the snippets that according to the qualitative analysis offer a good explanation are chosen more than the other snippets, this totally depends on the baseline snippet to which it is compared.

First, the snippets that are deemed totally **unrelated** are inspected. These should be the most straightforwards, as the "None" and "Skip" option were added to prevent a participant from having to choose between two unrelated pieces of text. Therefore, these snippets should never have been chosen. Interestingly though, for one item, all three annotators agreed that the pipeline snippet was the best explanation, while it seems to be totally irrelevant to the claim and topic. Moreover, in this case the baseline snippet seems to give an okay explanation, even though it may

require some background knowledge.

Topic: Big government
Claim: Small government produces less healthy societies
Pipeline: Demands of other domestic programs, international conditions, and state of economic health of our Nation are only a few of the major influences upon the specific budget for space in a given fiscal year. Despite the highly variable nature of these influences, which produces a corresponding increasing uncertainty in projections of resource availability, it is important far planning purposes to look into the future and forecast the general nature of funding required to support decisions on content and pace of the program. Two basic questions arise.
Baseline: No more. When you compare the U.S. with Canada, Western Europe and Japan, the news is sobering. Our child-poverty and infant-mortality rates are the highest, our life expectancy is the lowest, our budget deficit as a share of gross domestic product (GDP) is the highest, and our 15-year-olds rank among the lowest on tests of math and science.

The pipeline snippet is about the budget for a space program and does not mention anything about a big or small government, while the baseline snippet compares the United States to other countries based on a few factors related to health. The United States is generally considered to have a small government and little social security, especially compared to the countries referred to in the baseline snippet. It may be that this information was not known by the participants, but even then, why would the explanation that covers a space program be chosen and not the "none" option? In another case where the pipeline snippet was deemed totally unrelated, the annotators made three different choices, none of them the pipeline snippet, while in the last case, the pipeline snippet was chosen once, and the baseline twice.

Next, there are the snippets that covered a related, broader, or more specific topic than the given topic, which we name a **roughly related topic**. These explanations can still not be considered sufficient. The choices of the annotators were mostly divided. In only one case did two out of three pick the pipeline snippet, while in the other eight cases the pipeline snippet was either chosen once or not at all. In the following instance, the pipeline snippet was chosen twice:

Topic: Democratic peace theory
Claim: Democracies mostly only engage in defensive wars with non-democracies.
Pipeline: As mentioned, V-Dem is only one of the leading approaches to measure democracy. And its electoral democracy index is only one main measure it provides alongside other, more comprehensive indices of democracy. Yet, using another approach or V-Dem index to measure democracy shows a similar development from a highly undemocratic world in the 18th and 19th century, to high democratic inequality in the earlier 20th century, and a much more democratic, and more equally democratic, world in recent decades. You can see so for yourself by exploring the four charts below, which use the Polity projects democracy index and V-Dems liberal democracy index. Taken together, the democratic political systems of many countries show that a world where people have much more say in how they are governed is possible.
Baseline: Democratic Party officials often trace its origins to the Democratic-

Republican Party, founded by Thomas Jefferson, James Madison and other influential opponents of the conservative Federalists in 1792. [21][59] That party died out before the modern Democratic Party was organized[citation needed]; the Jeffersonian party also inspired the Whigs and modern Republicans[citation needed]. Historians argue that the modern Democratic Party was first organized in the late 1820s with the election of Andrew Jackson. [13] It was predominately built by Martin Van Buren, who assembled a wide cadre of politicians in every state behind war hero Andrew Jackson of Tennessee, making it the world's oldest active political party.

In this case, the pipeline snippet does concern democracies, but does not mention anything about peace and war, but instead deals with an approach to measuring democracy. The third annotator marked this item as "None", which makes sense as the baseline snippet is about the Democratic Party in the United States.

In four cases, a snippet was found that provides a counter argument, or at least information that supports an argument with an **opposing stance**, to the given claim. In one case, all three annotators agreed that the pipeline snippet provided the best explanation. This is interesting, as the snippet is totally incoherent, due to the formatting of the website it was extracted from. This can be seen at the end of the previous section where the relevant snippet is quoted as an example of an incoherent snippet. However, as far as it can be understood, it does provide information about the 2nd amendment being an individual right. Although it seems to convey that the Supreme Court has decided that the 2nd amendment is indeed and individual right, which opposes the claim.

Another category of snippets that has been defined is snippets that give some **general background** information about the overall topic. Here, one pipeline snippet is found twice, but with a different baseline snippet for comparison. This is due to the selection of a snippet from each cluster based on the centroid, as sometimes, one snippet would be the closest to the centroid of two different clusters, while not belonging to the second cluster. In one case, all three annotators agreed on the pipeline snippet, while in the other case, two chose the pipeline and one chose the baseline. Interestingly, only one participant has scored both items, and it was this participant that made a different choice the second time.

Topic: Oil sands
Claim: Oil sands take more energy to extract, so emit more
Pipeline: For these reasons, lifting extra-heavy oil consumes copious amounts of energy, making it highly carbon-intensive to extract. The Carnegie Endowment rates Venezuelas Merey grade, which is the primary export blend, as one of the most carbon-intensive oil varieties produced globally, emitting 604 kilograms of greenhouse gases per barrel produced. Venezuelas Tia Juana and Hamaca grades produce even more greenhouse gas emissions to extract, with only Canadas oil sands ranked as more carbon-intensive. Heavy sour crude oil blends like Merey, which has an API gravity of 16 degrees and 2.45% sulfur content, are costly, complex and carbon-intensive to refine into high-grade low emission fuels.
Baseline 1: Crude bitumen production (mined and in situ) totaled about 2.8 million barrels per day (bbl/d) in 2017. Source: Alberta Energy Regulator (AER) ST 98, ST39 and ST53 reports.
Baseline 2: Technological advancements in the oil sands have helped create more

energy efficient practices and to decrease GHG emissions in the oil sands. One of the most important mechanisms used to achieve this is co-generation. This is a process where steam and electricity is produced simultaneously. By converting energy and by-product into electricity that would otherwise be waste, co-generation has contributed significantly to the 30% decrease in per barrel GHG emissions seen in the oil sands since 1990. Find out more about the work CanmetENERGY is doing to further reduce GHG emissions in the oil sands.

Although the pipeline snippet is not very straightforwards, it can be deduced that extracting oil sands produces high greenhouse gas emissions. It makes sense that the first baseline was not chosen as a good explanation by anyone. The second baseline snippet is more directly concerned with oil sands, however, it provides more of a counter argument to the given claim than an explanation.

Another observation made in the eleven pipeline snippets that have been categorized as being general background information, is that not once has the skip or none option been chosen. Four of the eleven times all participants agree, while for the remaining items they are divided. That the none option is never chosen makes sense, as the pipeline snippet already conforms to the minimum requirement, namely providing some background information. If the baseline snippet has a higher quality or is more specific, this one is chosen. So generally, there is no need to choose none. For the skip option though, it may have been the case that both pieces of text provide some general background, so this might have been chosen.

Apart from providing (background for) a counter argument, some snippets also provide an argument that has the **same stance** as the claim, but offers different reasons. In nine instances a snippet like this was found, and in only one instance did all three annotators agree. In four instances there is absolutely no agreement, while in the remaining four there is a two-to-one division. Take the following example:

Topic: Cellulosic ethanol
Claim: There is not enough land too grow sufficient cellulosic ethanol
Pipeline: Finally, ethanol must be delivered. A motivation to develop cellulosic ethanol is the high delivery cost of corn grain ethanol from the Midwest to the coasts, since ethanol can't be delivered cheaply through pipelines, but must be transported by truck, rail, or barge (Yacobucci 2003). The whole cellulosic ethanol enterprise falls apart if the energy returned is less than the energy invested or even one of the major stumbling blocks can't be overcome. If there isn't enough biomass, if the residues can't be stored without exploding or composting, if the oil to transport low-density residues to biorefineries or deliver the final product is too great, if no cheap enzymes or microbes are found to break down lignocellulose in wildly varying feedstocks, if the energy to clean up toxic byproducts is too expensive, or if organisms capable of tolerating high ethanol concentrations aren't found, if the barriers in Appendix A can't be overcome, then cellulosic fuels are not going to happen.
Baseline: Because of the difference in energy density, you need about 1.5 L of ethanol to replace a liter of gasoline. So the yearly requirement for ethanol would be about 780 billion L. A hectare of switchgrass can supply about 4700 L of ethanol a year, so the United States would need to devote roughly 170 million hectares (420 million acres) to it. That's an enormous quantity of land almost as much as the country now devotes to farming.

The pipeline snippet has the same stance towards the topic as the given claim, but it offers different reasons. On the other hand, the baseline snippet does not take an explicit stance, but does express that a lot of land is needed for ethanol, although it is not clear whether this also applies to cellulosic ethanol. It is no surprise then, that three different choices have been made: baseline, pipeline, and skip.

Lastly, the choices regarding the pipeline snippets that are deemed **good explanations** will be examined. Ideally, the pipeline snippet would always be chosen, but this of course depends on the quality of the baseline snippets. However, it is expected that for these items, none is never chosen, as the pipeline snippet already offers a good explanation. This is indeed the case, but there is little agreement among the annotators. In the eight instances, only once are the three choices the same, in which case the baseline is perceived better.

Topic: Free trade
Claim: Free markets and trade benefit the environment
Pipeline: In this way Mexico's environment would be enhanced by opening up trade. Furthermore, companies typically expect rising environmental standards in developing-country markets, so they tend to introduce state of the art technology initially. One GATT report published early last year cited cases in which firms gain a competitive edge by investing first in clean technologies. A Conflict of Visions If the environmental arguments against freer trade are so flawed, why are they nevertheless held so tenaciously? One reason is the intellectual framework of many environmental activists. Broadening trade runs against the environmental vision, particularly that of local self-reliance and a return to a simpler, low-technology world. Consider the argument of David Morris of the Institute for Local Self-Reliance: 'Most people believe that a global economy is the only path to a higher standard of living as well as the inevitable next step in economic evolution.
Baseline: 5. Free market economies regulate themselves naturally. Supply and demand principles govern a free market economy, which means the decisions that people make enable a process of self-regulation. If goods or services don't meet the ethical standards that consumers have for their transactions, then a choice to avoid those items will put pressure on the organization to make changes. Misleading people about the quality of an item or the availability of services causes circumstances that could force the company into bankruptcy or worse.

It seems weird that the baseline has been chosen by all participants, as the pipeline snippet specifically mentions how the environment benefits from a free market, while the baseline snippet is more general about how companies make ethical decisions in a free market. Apart from this one, the other seven items were two-to-one, with three times a majority for the pipeline snippet and four times for the baseline.

5.3 Discussion

In this section, a quick recap of the results of the user study will be given, after which the implications of these results are discussed. Limitations of the overall research as well as the annotation study are given and finally some recommendations for future

research are made.

The results indicate that the task of finding explanations that help in understanding claims is a difficult task on multiple levels. The low agreement between the different annotators shows that choosing which piece of text provides the best information is not trivial. The qualitative assessment of the retrieved snippets further confirmed that it was not always clear what a snippet exactly communicates and whether it is sufficient to help in understanding an argument. This was somewhat expected, as the process of building the pipeline revealed that certain techniques are not yet as advanced or widely available as expected.

Although the results show that the pipeline does not perform great, it can not be said that this approach does not work. This is because this research has been exploratory in two main ways. First of all, instead of finding facts or specific kinds of premises to augment an argument, in this study, the focus was on finding any kind of information that may help explain an argument, like assumptions and beliefs. Secondly, instead of restricting oneself to a specific kind of (structured) data, in this study, all available text on the internet was used. The results show that this approach does offer a way of finding information to help understand a claim, but the implementation is not sufficient yet. A more advanced implementation of the different components and the use of other methods and techniques may improve the performance.

This study provides a baseline for future research to build on top of. This has been a first step towards using unstructured natural language from web sources to find information that explains a claim, and can therefore be used as a guide and baseline to improve the task in future research.

The reliability of the results of the user study is impacted by multiple factors. First of all, only a limited number of people has participated in the study and for every snippet only three people have given their assessment. Secondly, these participants were chosen based on convenience sampling. Although no prior knowledge is needed for the task, choosing between explanations may be more difficult depending on the level of English of an annotator or the affiliation with a certain topic and claim. Lastly, the task may have been too vague and ambiguous, which may result in different interpretations of the task.

Although it was specifically the goal to find a wide variety of information to explain an argument, it may also have been a limitation during the building of the pipeline and the evaluation study. If a more specific definition of the kind of explanations that should be found was given, it might have helped in implementing the different components of the pipeline as well as in choosing the best explanation during the evaluation.

In future research, the different methods and techniques used in the pipeline could all individually be improved, as the current implementation is sometimes a bit shallow. The words and phrases that are extracted from the topic and claim and used to search articles could for instance be expanded using synonyms and other related words. To this end, language models can be used or knowledge bases containing this data. Further, it would be interesting to see how a technique like topic modeling can be used to make an initial grouping of the documents extracted through the web search. This may help the clustering in the last step as well. The extraction of snippets may be improved by using Rhetorical Structure Theory. This may greatly help in determining the boundaries of a snippet. Furthermore, the

structure of a text can be used to understand how certain parts of the text relate to other parts, so a more focused search to relevant snippets can be conducted. A possibility would be to find the given claim in a text and then through RST find how the different parts of the text relate to the part where the claim is found.

Chapter 6

Conclusion

This researched was set out to discover how natural language sources from the internet can be used to explain arguments, as is stated in the main research question. Two sub questions were created to help answering this question.

The first question is concerned with what is needed to extract relevant information from the internet to explain claims. To this end a pipeline is proposed consisting of five main tasks. Qualitative assessment of the different components during the development of the pipeline has shown that these five broad tasks may indeed aid in extracting relevant information, but that there is also more work needed on the different components to increase performance.

A structured and quantitative analysis of the pipeline is needed to determine the performance, which is the subject of the second sub question. A user study has been organized in which participants had to choose between two explanations for a given claim - topic pair. This study did not generate conclusive results, indicating that the task may have been too ambiguous.

Although the performance can not properly be confirmed using the results of the evaluation study, a qualitative assessment on the extracted snippets has been done to determine whether the pipeline has been successful. Overall, it can be said that a more advanced implementation of the components is needed, but that using the plethora of information found on the web is a promising approach in explaining claims.

This research has explored an alternative way to augment claims with background information in order to clarify the claim. Multiple methods and techniques have been combined to create a pipeline that can function as a baseline for subsequent research in this field.

Bibliography

[1] Patrick Abels et al. "Focusing knowledge-based graph argument mining via topic modeling". In: *arXiv preprint arXiv:2102.02086* (2021).

[2] Majid Hameed Ahmed et al. "Short Text Clustering Algorithms, Application and Challenges: A Survey". In: *Applied Sciences* 13.1 (2022), p. 342.

[3] Abeer ALDayel and Walid Magdy. "Stance detection on social media: State of the art and trends". In: *Information Processing & Management* 58.4 (2021), p. 102597.

[4] Fatima Alkhawaldeh, Tommy Yuan, and Dimitar Lubomirov Kazakov. "Warrant generation through deep learning". In: *Seventh International Conference on Natural Language Computing (NATL 2021), Proc. of.* AIRCC Publishing Corporation, 2021, pp. 53–75.

[5] Mehdi Allahyari et al. "A brief survey of text mining: Classification, clustering and extraction techniques". In: *arXiv preprint arXiv:1707.02919* (2017).

[6] Mehdi Allahyari et al. "Text summarization techniques: a brief survey". In: *arXiv preprint arXiv:1707.02268* (2017).

[7] Milad Alshomary et al. "Belief-based generation of argumentative claims". In: *arXiv preprint arXiv:2101.09765* (2021).

[8] Milad Alshomary et al. "Target inference in argument conclusion generation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 4334–4345.

[9] Rajaraman Anand and Ullman Jeffrey David. *Mining of massive datasets*. Cambridge University Press, 2011.

[10] Arvind Arasu et al. "Searching the web". In: *ACM Transactions on Internet Technology (TOIT)* 1.1 (2001), pp. 2–43.

[11] Ron Artstein. "Inter-annotator agreement". In: *Handbook of linguistic annotation* (2017), pp. 297–313.

[12] Jamilu Awwalu, Saleh El-Yakub Abdullahi, and Abraham Eseoghene Evwiekpaefe. "Parts of speech tagging: a review of techniques". In: *Fudma Journal of Sciences* 4.2 (2020), pp. 712–721.

[13] Maria Becker, Katharina Korfhage, and Anette Frank. "Implicit knowledge in argumentative texts: an annotated corpus". In: *arXiv preprint arXiv:1912.10161* (2019).

[14] Maria Becker, Siting Liang, and Anette Frank. "Reconstructing implicit knowledge with language models". In: *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. 2021, pp. 11–24.

[15] Maria Becker et al. "CO-NNECT: A Framework for Revealing Commonsense Knowledge Paths as Explicitations of Implicit Knowledge in Texts". In: *arXiv preprint arXiv:2105.03157* (2021).

[16] Maria Becker et al. "Enriching argumentative texts with implicit knowledge". In: *International Conference on Applications of Natural Language to Information Systems*. Springer, 2017, pp. 84–96.

[17] Emily M. Bender and Alexander Koller. "Climbing towards NLU: On meaning, form, and understanding in the age of data". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 5185–5198.

[18] Jamal Bentahar, Bernard Moulin, and Micheline Bélanger. "A taxonomy of argumentation models used for knowledge representation". In: *Artificial Intelligence Review* 33.3 (2010), pp. 211–259.

[19] Janek Bevendorff et al. "Elastic chatnoir: Search engine for the clueweb and the common crawl". In: *Advances in Information Retrieval: 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings 40*. Springer, 2018, pp. 820–824.

[20] Filip Boltužić and Jan Šnajder. "Fill the gap! analyzing implicit premises between claims from online debates". In: *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*. 2016, pp. 124–133.

[21] Teresa Botschen, Daniil Sorokin, and Iryna Gurevych. "Frame-and entity-based knowledge for common-sense argumentative reasoning". In: *Proceedings of the 5th Workshop on Argument Mining*. 2018, pp. 90–96.

[22] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine". In: *Computer Networks and ISDN Systems* 30.1-7 (1998), pp. 107–117.

[23] Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. "Building a discourse-tagged corpus in the framework of rhetorical structure theory". In: *Current directions in discourse and dialogue* 22 (2003), pp. 85–112.

[24] Tuhin Chakrabarty, Aadit Trivedi, and Smaranda Muresan. "Implicit Premise Generation with Discourse-aware Commonsense Knowledge Models". In: *arXiv preprint arXiv:2109.05358* (2021).

[25] Alebachew Chiche and Betselot Yitagesu. "Part of speech tagging: a systematic review of deep learning and machine learning approaches". In: *Journal of Big Data* 9.1 (2022), pp. 1–25.

[26] Debopam Das et al. "Constructing a lexicon of English discourse connectives". In: *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. 2018, pp. 360–365.

[27] Joseph L. Fleiss and Jacob Cohen. "The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability". In: *Educational and psychological measurement* 33.3 (1973), pp. 613–619.

[28] James B. Freeman. *Argument Structure:: Representation and Theory*. Vol. 18. Springer Science & Business Media, 2011.

[29] Michael Fromm, Evgeniy Faerman, and Thomas Seidl. "TACAM: topic and context aware argument mining". In: *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, 2019, pp. 99–106.

[30] Rob Grootendorst and Frans H. Van Eemeren. *A systematic theory of argumentation: The pragma-dialectical approach*. Cambridge University Press, 2004.

[31] Ivan Habernal et al. "The argument reasoning comprehension task: Identification and reconstruction of implicit warrants". In: *arXiv preprint arXiv:1708.01425* (2017).

[32] Wenjuan Han et al. "A survey of unsupervised dependency parsing". In: *arXiv preprint arXiv:2010.01535* (2020).

[33] Pengcheng He et al. "Deberta: Decoding-enhanced bert with disentangled attention". In: *arXiv preprint arXiv:2006.03654* (2020).

[34] Shengluan Hou, Shuhan Zhang, and Chaoqun Fei. "Rhetorical structure theory: A comprehensive review of theory, parsing methods and applications". In: *Expert Systems with Applications* 157 (2020), p. 113421.

[35] Ioana Hulpus et al. "Towards Explaining Natural Language Arguments with Background Knowledge." In: *PROFILES/SEMEX@ ISWC*. 2019, pp. 62–77.

[36] Wafaa S. El-Kassas et al. "Automatic text summarization: A comprehensive survey". In: *Expert Systems with Applications* 165 (2021), p. 113679.

[37] Md Abu Kausar, V. S. Dhaka, and Sanjeev Kumar Singh. "Web crawler: a review". In: *International Journal of Computer Applications* 63.2 (2013), pp. 31–36.

[38] Jonathan Kobbe, Ioana Hulpuș, and Heiner Stuckenschmidt. "Unsupervised stance detection for arguments from consequences". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 50–60.

[39] Jonathan Kobbe et al. "Exploiting background knowledge for argumentative relation classification". In: *2nd Conference on Language, Data and Knowledge (LDK 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[40] Trupti M. Kodinariya and Prashant R. Makwana. "Review on determining number of Cluster in K-Means Clustering". In: *International Journal* 1.6 (2013), pp. 90–95.

[41] Sandra Kübler, Ryan McDonald, and Joakim Nivre. "Dependency parsing". In: *Synthesis lectures on human language technologies* 1.1 (2009), pp. 1–127.

[42] Dilek Küçük and Fazli Can. "A tutorial on stance detection". In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022, pp. 1626–1628.

[43] Dilek Küçük and Fazli Can. "Stance detection: A survey". In: *ACM Computing Surveys (CSUR)* 53.1 (2020), pp. 1–37.

[44] Anne Lauscher et al. “Scientia Potentia Est–On the Role of Knowledge in Computational Argumentation”. In: *arXiv preprint arXiv:2107.00281* (2021).

[45] Marco Lippi and Paolo Torroni. “Argumentation mining: State of the art and emerging trends”. In: *ACM Transactions on Internet Technology (TOIT)* 16.2 (2016), pp. 1–25.

[46] Yang Liu and Mirella Lapata. “Text summarization with pretrained encoders”. In: *arXiv preprint arXiv:1908.08345* (2019).

[47] Artem A. Maksutov et al. “The Transformer Neural Network Architecture for Part-of-Speech Tagging”. In: *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*. IEEE, 2021, pp. 536–540.

[48] William C. Mann and Sandra A. Thompson. *Rhetorical structure theory: A theory of text organization*. University of Southern California, Information Sciences Institute Los Angeles, 1987.

[49] Marie-Catherine De Marneffe et al. “Universal dependencies”. In: *Computational linguistics* 47.2 (2021), pp. 255–308.

[50] Angel R. Martinez. “Part-of-speech tagging”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 4.1 (2012), pp. 107–113.

[51] Marie-Francine Moens. “Argumentation mining: Where are we now, where do we want to be and how do we get there?” In: *Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation*. 2013, pp. 1–6.

[52] Saif Mohammad et al. “Semeval-2016 task 6: Detecting stance in tweets”. In: *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*. 2016, pp. 31–41.

[53] Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. “Stance and sentiment in tweets”. In: *ACM Transactions on Internet Technology (TOIT)* 17.3 (2017), pp. 1–23.

[54] Arne Neumann. “Using and comparing Rhetorical Structure Theory parsers with rst-workbench”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 2021, pp. 1–6.

[55] Timothy Niven and Hung-Yu Kao. “Probing neural network comprehension of natural language arguments”. In: *arXiv preprint arXiv:1907.07355* (2019).

[56] Joakim Nivre et al. “Universal dependencies v1: A multilingual treebank collection”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*. 2016, pp. 1659–1666.

[57] Mahamed GH Omran, Andries P. Engelbrecht, and Ayed Salman. “An overview of clustering methods”. In: *Intelligent Data Analysis* 11.6 (2007), pp. 583–605.

[58] Andreas Peldszus and Manfred Stede. “An annotated corpus of argumentative microtexts”. In: *Argumentation and Reasoned Action: Proceedings of the 1st European Conference on Argumentation, Lisbon*. Vol. 2. 2015, pp. 801–815.

[59] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5485–5551.

[60] Pavithra Rajendran, Danushka Bollegala, and Simon Parsons. “Contextual stance classification of opinions: A step towards enthymeme reconstruction in online reviews”. In: *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*. 2016, pp. 31–39.

[61] Martin Reddy. *API Design for C*. Elsevier, 2011.

[62] Myrthe Reuver et al. “Is Stance Detection Topic-Independent and Cross-topic Generalizable?–A Reproduction Study”. In: *arXiv preprint arXiv:2110.07693* (2021).

[63] Ruty Rinott et al. “Show me your evidence-an automatic method for context dependent evidence detection”. In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 440–450.

[64] Lior Rokach and Oded Maimon. “Clustering Methods”. In: *Clustering methods*. (2005).

[65] Paul Schachter and Timothy Shopen. “Parts-of-speech systems”. In: *Language typology and syntactic description* 1 (1985), pp. 3–61.

[66] Keshav Singh et al. “Annotating Implicit Reasoning in Arguments with Causal Links”. In: *arXiv preprint arXiv:2110.13692* (2021).

[67] Keshav Singh et al. “Exploring Methodologies for Collecting High-Quality Implicit Reasoning in Arguments”. In: *Proceedings of the 8th Workshop on Argument Mining*. 2021, pp. 57–66.

[68] Keshav Singh et al. “Improving evidence detection by leveraging warrants”. In: *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*. 2019, pp. 57–62.

[69] Niko Solihin. “Search engine optimization: a survey of current best practices”. In: (2013).

[70] Manfred Stede, Tatjana Scheffler, and Amália Mendes. “Connective-lex: A web-based multilingual lexical resource for connectives”. In: *Discours.Revue de linguistique, psycholinguistique et informatique.A journal of linguistics, psycholinguistics and computational linguistics* 24 (2019).

[71] Manfred Stede and Jodi Schneider. “Argumentation mining”. In: *Synthesis Lectures on Human Language Technologies* 11.2 (2018), pp. 1–191.

[72] Milan Tofiloski, Julian Brooke, and Maite Taboada. “A syntactic and lexical-based discourse segmenter”. In: *Proceedings of the ACL-IJCNLP 2009 conference short papers*. 2009, pp. 77–80.

[73] Stephen E. Toulmin. *The uses of argument*. Cambridge university press, 2003.

[74] Atro Voutilainen. *Part-of-speech tagging*. Vol. 219. The Oxford handbook of computational linguistics, 2003.

[75] Steven Walfish. “A review of statistical outlier methods”. In: *Pharmaceutical Technology* 30.11 (2006), p. 82.

[76] Douglas Walton and Chris A. Reed. “Argumentation schemes and enthymemes”. In: *Synthese* 145.3 (2005), pp. 339–370.

[77] Douglas Walton, Christopher Reed, and Fabrizio Macagno. *Argumentation schemes*. Cambridge University Press, 2008.

[78] MeiShan Zhang. “A survey of syntactic-semantic parsing based on constituent and dependency structures”. In: *Science China Technological Sciences* 63.10 (2020), pp. 1898–1920.

[79] Bo Zhao. “Web scraping”. In: *Encyclopedia of big data* (2017), pp. 1–3.

Appendix A

Argument structure

Table A.1 provides an overview of the different terms used in the research field of argument mining to describe *argumentative units* (also referred to as *argument components* or *argumentative discourse units*). The first two structures from the left are the classical structure and the Toulmin [73] model, to which the structures used in other papers are related.

Classic (as used by Rajendran, Bollegala, and Parsons [60] and Walton, Reed, and Macagno [77])	Toulmin [73]	Hulpus et al. [35] and Alshomary et al. [8]	Habernal et al. [31], Singh et al. [66, 67], Botschen, Sorokin, and Gurevych [21], and Alkhawaldeh, Yuan, and Kazakov [4]	Rinott et al. [63]	Boltužić and Šnajder [20] and Chakrabarty, Trivedi, and Muresan [24]	Singh et al. [68]
Minor Premise	Data	Premise	Reason/Premise		Premise	
Major Premise	Warrant		Warrant	CDE[1]		Warrant
	Backing					Evidence
	Qualifier					
	Rebuttal					
Conclusion	Claim	Conclusion	Claim	Claim	Claim	Claim

Table A.1: Relation between different terms to define the structure of an argument.

Appendix B

Task description

Below, the task description can be found exactly as it was presented on the annotation website.

B.1 Introduction

First of all: thank you for participating in this study! My name is Nick van Bremen and I am following the Master's programme Business Informatics at Utrecht University. As final part of my study, I have to conduct research and write a thesis. My thesis is concerned with **understanding arguments** made in a discussion. More specifically, the goal is to research whether a system (from here on referred to as *the pipeline*) can be built that automatically finds relevant pieces of text (*snippets*) from the internet that may help people in understanding why a certain claim is made. This means the found snippets do not necessarily have to be factually correct, but may also express a believe or assumption that is implicit in the made claim.

In short: the pipeline takes as **input** a *claim*, a corresponding *topic*, and the *stance* of the claim towards the topic (the terms in italic are further explained below). The **output** of the pipeline are pieces of text of a few sentences, referred to as *snippets*, that may help in understanding why the claim given as input is made. To **evaluate** the performance of this pipeline, the gathered snippets have to be compared to some baseline. For that reason, this annotation study has been set up.

B.2 The task

As mentioned above, the goal is to compare the snippets gathered with the pipeline to a baseline in order to evaluate the performance of the system. A dataset consisting of snippets relating to a certain claim has been created and each snippet is paired with a baseline. For every claim - topic pair, multiple snippets have been gathered. So you may see the same claim - topic pair multiple times, but the snippets will always be different.

The task of the annotator is to, given a topic, corresponding claim, and two snippets (one found by the pipeline, one baseline), select which snippet best explains why the given claim is made. A snippet should at least cover the same overall subject as the topic and/or claim to be chosen. If it is impossible for you to make a decision, an item can be skipped by clicking the Skip button and no choice has to be made.

If both explanations are are totally irrelevant, and cover neither the claim or more general topic, it is possible to choose the option None.

In the ideal case, one snippet does not hold any relevant information while the other offers a perfect explanation. However, the snippets will probably mostly both contain information that may be deemed relevant. Therefore, the following points should be taken into account when making a choice:

- If snippet 1 gives information that is relevant to the claim, while snippet 2 gives information that is only relevant to the broader topic, choose the **more specific** one, in this case, snippet 1.

- If snippet 1 contains information that makes it clear **why** a certain claim would be made (e.g. by providing an assumption that is made, a believe that is held, or a piece of factual information that helps in understanding the claim and overall topic), and snippet 2 only holds information that doesn't necessarily explain the why, snippet 1 should be chosen.

- If both snippets offer different information on the question why someone would make a claim, choose the snippet that provides **the most new information** that cannot easily be inferred from the topic and the claim.

- If both snippets contain new or similar information, choose the **most coherent** snippet (e.g. the one that handles one specific piece of information and which is understandable from that piece of text alone).

- If both snippets are **too similar** to choose, Skip the item. If they are both **totally irrelevant** (e.g. not even cover the overall subject of the topic), choose the None option.

It is important to remember that it is **not necessary to have any previous knowledge** on the topic and the claim. The goal of the pipeline is to provide extra information on the topic and claim, which may or may not be factual. The task for the annotator is to choose the snippet that makes most sense to them and provides the best information, and not to choose the "correct" snippet. Furthermore, whether you agree or disagree with a certain snippet should not influence your choice.

The dataset contains approximately 140 items. Feel free to annotate as many as you like, it does not matter if you only do a few, or all of them.

B.3 Important terms

Claim: the conclusion of an argument that expresses a certain stance towards a topic
Topic: a subject of discussion, this can be a specific discussion or a broad theme
Stance: expresses whether a claim agrees or disagrees with the topic. The stance of a claim towards a topic is either *pro* or *con*
Snippet: piece of text of a few sentences that should provide extra information on the claim and topic

B.4 How the data is stored

To keep track of which items have been completed by which annotator, this system makes use of cookies. A randomly generated ID is used to identify your device, so that if you decide to come back later, the system will recognize you and make sure to give you items that you have not yet had. I would therefore like to ask you to not use different devices/browsers to annotate snippets, as it will otherwise not be possible to track who has annotated what, which might lead to you getting the same snippets. Apart from this random anonymous ID, no other information will be stored. If you decide to proceed, you accept the cookies.

Lastly, the text that is found on this page will always be available during the annotation process by clicking the information button on the top right of the screen. If you have any further questions, feel free to email me at j.a.vanbremen@students.uu.nl.

Appendix C

Planning

Phase 2 of the project will run from August 29th to January 23rd. Table C.1 gives an overview with the tasks to be completed per week.

Week	Dates	Tasks
1	29-08 / 02-09	Set up project, implement entity extraction
2	05-09 / 09-09	Start implementing web scraper
3	12-09 / 16-09	Finish implementing web scraper
4	19-09 / 23-09	Start implementing classifier for filtering opinionated articles
5	26-09 / 30-09	Finish classifier
6	03-10 / 07-10	Start implementing snippet extractor
7	10-10 / 14-10	Finish implementing snippet extractor
8	17-10 / 21-10	Start implementing stance detection
9	24-10 / 28-10	Finish implementing stance detection
10	31-10 / 04-11	Start implementing clustering algorithm
11	07-11 / 11-11	Finish clustering algorithm and ranking
12	14-11 / 18-11	Set up experiment and retrieve snippets
13	21-11 / 25-11	Set up annotation study for analysis
14	28-11 / 02-12	Perform annotation study
15	05-12 / 09-12	Round up annotation study and start quantitative and qualitative analysis
16	12-12 / 16-12	Finish analysis
17	19-12 / 23-12	Start writing or catch up on any delays
18	02-01 / 06-01	Write report
19	09-01 / 13-01	Write report
20	16-01 / 20-01	Finish up report

Table C.1: Weekly planning of Phase 2